

Comparison and Evaluation of Common Hands with Self-designed Task-Specific Robotic Hands in context of Grasp Learning

A PROPOSAL PRESENTED
BY
ABHIJIT MAKHAL
TO
THE DEPARTMENT OF MECHANICAL ENGINEERING

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
ROBOTICS

IDAHO STATE UNIVERSITY
POCATELLO, IDAHO
NOVEMBER 2015

Comparison and Evaluation of Common Hands with Self-designed Task-Specific Robotic Hands in context of Grasp Learning

ABSTRACT

The main focus for any robotic pick and place task is the ability of grasp objects robustly with stability. In an industrial environment the task becomes naive as the environment is modified according to the benefits of the robot. The sensors receive the benefits of structured environment. On the other hand, as the robotic systems are rapidly implemented in assistive or home scenarios, the task of pick and place becomes highly complex. The robots have to perform in a human environment, where the world is designed for benefits of human. The robot have to adapt to that environment by planning with partial sensor information without colliding with human or other objects in the environment. The focus of the research is the theory and implementation of a grasping framework, where the common commercially available hands are compared and evaluated with the performance of self-designed robotic hands. The research in one hand deals with the complexity of designing robotic hands specific to a designated task in the context of tree topology. On the other hand the evaluation and comparison of the designed hand with commercially available robotic hands in a general framework of robotic grasping.

For assessment and evaluation of the hands, the grasping system generates stable grasp points by means of finger placement points on recognized objects by statistical learning methods. As the self-designed hands are task-specific, it can be easily assumed that the formulation of grasping points would be significantly different than the points calculated for common hands. After recognizing and calculating robust and stable grasping points, the robotic system would be able to execute the task of grasping and manipulate the object according to specified task. After the execution of the grasping task by both hands in an identical environment, the result of both hands will be assessed and evaluated on the basis of time-complexity, task execution, performance and stability.

Contents

1	INTRODUCTION	I
1.1	Recognizing objects	4
1.1.1	Generating object Dataset	5
1.2	Calculating Grasping Points	6
1.2.1	Hand Design for Grasping	7
1.2.2	Grasp Synthesis	8
1.2.3	Learn to Grasp	9
1.3	Motion Planning for Grasping	9
1.4	Design of Task-specific Robotic Hands	11
1.5	Simulation and Real Hardware	12
2	OBJECT RECOGNITION	13
2.1	2D v/s 3D	14
2.2	3D capturing devices	16
2.3	Processing in 3D	19
2.3.1	3D data representation	19
2.3.2	Normal Estimation	21
2.3.3	3D Filtering	22
2.3.4	3D Registration	24
2.3.5	segmentation	25
2.4	3D feature Extraction	25
2.5	Recognition Pipeline	31
3	GRASP SYNTHESIS	33
3.1	Grasp Taxonomy	34
3.2	Analytical Approach	35
3.2.1	Force Closure Graps	35

3.2.2	Task Compatibility	36
3.3	Empirical Approach	37
3.3.1	Systems based on human observation	37
3.3.2	Systems based on object observation	38
3.4	Wrench Space	39
3.5	Friction cones	39
3.6	Types of Grasps	40
3.7	Grasping System Design	40
3.8	Hand Posture Subspace	42
3.9	Grasp Planning using EigenGrasps	42
4	MOTION PLANNING	45
4.1	Moveit	48
4.1.1	Configuration	48
4.1.2	Robot Interface	49
4.1.3	Motion Planning	50
4.1.4	Motion Planning pipeline	52
4.1.5	OMPL	53
4.1.6	World Geometry Monitor	53
4.2	OpenRAVE	53
4.3	MIT-Drake	54
5	PRELIMINARY RESULTS	56
5.1	Generating database for objects	56
5.2	Recognizing Objects from Dataset	57
5.2.1	Using local Feature Descriptors	57
5.2.2	Using global Feature Descriptors	58
5.2.3	Object Recognition Kitchen	59
5.3	Grasp Synthesis	60
5.3.1	GraspIT	61
5.3.2	OpenRAVE	62
5.4	Robot Hardware	63
5.5	Simulation	64

6	PROPOSED RESEARCH	67
6.1	Recognizing object	68
6.1.1	Tasks	68
6.1.2	Methods	68
6.2	Grasping Point Calculation	69
6.2.1	Tasks	70
6.2.2	Methods	70
6.3	Planning Motion	71
6.3.1	Tasks	71
6.3.2	Methods	71
6.4	Design of robotic Hands	72
6.4.1	Task	72
6.4.2	Method	72
6.5	The robot in hardware and software	72
6.5.1	Tasks	73
6.5.2	Methods	73
6.6	Evaluation and Assessment of Hands	73
6.6.1	Tasks	74
6.6.2	Methods	74
6.7	Goal of the Research	74
6.8	Time Allotment	75
7	CONCLUSION	76
	APPENDIX A PUBLICATIONS	77
	REFERENCES	86

Listing of figures

1.1	Flowchart of designing Robotic Hands	11
2.1	Model Matching Failure in 2D images -1	15
2.2	Model Matching Failure in 2D images -2	15
2.3	3D capturing devices	17
2.4	Point Cloud Data example	18
2.5	Point Cloud Data example by Color	19
2.6	Point Cloud Data Structure	20
2.7	Volumetric representation using octree	20
2.8	Normal Estimation of a point cloud data	22
2.9	Point cloud registration	23
2.10	Point cloud Features	26
2.11	PFH point pairs	27
2.12	PFH frames	28
2.13	PFH histogram on a point cloud data	29
2.14	3D Recognition Pipeline	31
3.1	Cutkosky's Grasp Taxonomy	34
3.2	Friction cone concept	39
3.3	Grasp Planning Examples	41
3.4	Various commercially available hands	42
3.5	Eigengrasps for different hands	43
3.6	Stable Eigengrasps and the number of iterations taken	44
4.1	The MoveIt architecture	49
4.2	Planning adapters	51
4.3	Planning scene	52

4.4	MoveIT perception pipeline	54
4.5	Octomap generated by PR2 robot	54
5.1	Some objects from the local dataset	57
5.2	Recognition with FPFH descriptor	58
5.3	Recognition with VFH descriptor	59
5.4	Recognition with OUR-CVFH descriptor	59
5.5	Recognition with ORK	60
5.6	Graspit Framework with Barrett Hand	62
5.7	Eigen Grasp with GraspIT	62
5.8	Visualizing Grasp Quality with Graspit	63
5.9	Comparison of grasps on a object by different hands	63
5.10	Performing grasp with different hands in OpenRave	63
5.11	Robot hardware components	64
5.12	Objects imported in Gazebo Simulation	64
5.13	Our robotic module in Gazebo	65
5.14	Simulated robot in RViz	65

THIS IS BEEN DEDICATED TO MISTU AND DEBASHREE.

Acknowledgments

THANK YOU VERY MUCH, I would like to acknowledge the dedication and support provided by my advisor Dr.Alba Perez Gracia , my department of Mechanical Engineering and all the members of Measurement and Control Engineering Research laboratory.....

1

Introduction

The core mechanism for any robotic pick and place task for non-symmetric objects is the calculation for robust and feasible grasping points. The complexity of a grasping system for asymmetric objects in unstructured environment is highly dependent on the placement of fingers in the hand. i.e the kinematics of the hand. In the dissertation, a general grasping framework is been proposed where the commercially available robotic hands will be compared and evaluated with the self-designed robotic hands which are task-specific. For the evaluation process, the robotic system tries to recognize objects stored in a database and grasp them according to the calculation of grasping points based on the placement of robotic fingers for stable and suitable grasps. The accumulated grasps will be evaluated in

the larger context for picking, placing and manipulating the recognized objects.

The task can be divided into several modules where the combination of individual modules lead to the task of grasping the object and manipulating them according to the designated task. The modules can be defined as a) Recognizing the object, b) calculation of grasping points based on recognized features of the object, c) Evaluation and synthesis of the grasping points, d) Implementation of the grasping framework which leads the system to learn the suitable grasps for future run and less computation, e) Path planning for the robot to perform the grasps and f) assessment of the system for comparing grasps obtained from different strategies.

As the robotic systems are gradually moving towards the dimension where the grasping is not only limited to regular shaped objects, the complexity of calculating grasping points is increasing exponentially. The problem becomes severe as no general model exists for robotic hands. Slight variation in the placement of the fingers on the hand has huge impact on the complexity of the task. The hand structures vary depending upon the number of fingers, position of the fingers and specification. There are also several limitations which add extra constraints to the task of grasping. In the unstructured environment, the task of perceiving the object varies upon clutter, occlusion, viewpoint, lightning conditions and other barriers which constraints the task of generating a full model of the object based on limited sensor information which would lead to the recognition of object in 6D pose. Analyzing and calculating the grasping points of the object online and offline is computationally highly expensive. The task can be simplified by probabilistic methods which relies on artificial intelligence strategies. The calculated grasps can be stored in a database for later evaluation of grasping of similar-shaped objects. After calculation of the grasping points

the grasps should be synthesized based on their stability for the designated task of namely pick and place. For the task of manipulation of the objects, the robotic manipulator needs to find the inverse kinematics solution where the planned trajectory should be aware of the collision and constraints in the workspace environment.

The focus of the system is on the calculation of grasping points, evaluation and synthesis of the grasping points for feasibility and stability, and the approach to learn the grasping points for novel objects. For synthesizing the grasps on objects, two approaches are preferred. In the analytical approach, the grasps are based on force-closure properties and it is independent of the Task Wrench Space. The drawback of the methods is it is computationally expensive. Whereas in the empirical approach, the grasps are specified for designated tasks and grasping points are evaluated on the basis of perception of the object to be grasped. The task wrench space is based on evaluation of several grasps by iteration and choosing the best feasible grasping points for the task in hand. For deciding the best grasp pose on a recognized object for a specified task can be calculated by the means of statistical learning methods such as Support Vector Machine (SVM) or probabilistic methods such as Naïve Bayes, Hidden Markov Models or Recurrent Neural Networks. After calculating the feasible grasping points, the system should plan collision aware and constraint aware inverse kinematics solutions of the motions for the manipulator to place the end-effector in suitable pose for perform the grasps. As the environment is unstructured, there is a little possibility of performing same motion grasping different objects. The task of manipulation can be performed by searching algorithms or tree based searching algorithms such as Rapidly exploring Random Trees (RRT).

1.1 RECOGNIZING OBJECTS

Recognizing objects in the unstructured environment is one of the most challenging task. The performance of the object recognition system highly depends on several factors such as occlusion, lighting conditions, scale, reflection and many more which hinders the system to detect the proper features need to be detected for identifying a right object. These constraints may lead the system to high false positive values, where the system detects a completely different object or not detects at all.

Human perceive their environment in three dimension, whereas most robotic or computer systems are based on 2D images. The task is highly complex, as the robotic system have to find appropriate features provided by the pixels in the image. After that the system have to compare the sepecific part of the image which corresponds to an object to its previously stored image database. Although the human vision system works in the same way, but there is a difference in one added dimension. Colaborating another dimension to the task on one hand increases the accuracy and avoids several constraints mentioned above, but on the other hand it comes with the gift of complexity.

For recognizing an object in 3D, a system has to go through several processes which includes capturing the environment, processing the environment, detecting the features corresponding to the object, segmenting the object from the environment, detecting the 6D pose of the object. For the test environment, the system has to go through the same process again but with the added functionality of comparing the detected features upto a certain threshold which outputs an object as recognized by the system. The viewpoint is one of the most important constraint in a object recognition system as the system would have to detect an object where it is rotated or inverted. These would generate a huge variation in

the observable features in the training and test data. Most object recognition system fails to detect the right object in the variation of the viewpoint. So for training an object model the system have to consider the full 6d pose of the object(Translation in x,y and z direction and the rotation such as roll, pitch and yaw). The objects can be recognized in online and offline mode. In the offline mode, for testing and validation, the captured environment or the segmented object models can be compared in a later stage. For the mostly preferred on-line mode, the system have to perform the whole task of segmenting the object, extracting the features, comparing the features with the stored features have to be performed in real time and in less time consuming and robust manner.

1.1.1 GENERATING OBJECT DATASET

There are several variations of a single model. There exists more than thousand variations of a simple object such as a cup. The system should have to learn a model of the cup from several segmented object model corresponding to a cup. e.g The cylinder model with a handle can be considered as a cup model. So the system have to go through several model of a single object in the training phase for generating a specific category of a cup model. For creating an object model, the system has to go through several steps which includes segmenting the object model from the environment and registering the segmented views from different viewpoints to obtain a full model of an object which solves the problem of 6D pose information. The features collected in each view will be compared for registering the object scenes. The objects models can be stored in a database for detecting the object and recognizing them in the testing environment. By the process, a local database can be created with the objects in hand. For increasing the number of objects in the database, several

models can be borrowed from the other databases which are available to be borrowed. Such databases are namely YCB (Yale-California-Berkely), Amazon dataset, CMU, MIT, KIT are others.

1.2 CALCULATING GRASPING POINTS

When the system recognizes the object, the next task is to find the robust and stable grasping points based on the recognized features. Just like the generation of database for the object models for recognition, an approach can also be incorporated where the stable grasping points of an object can be stored in a database for further implementation. The other approach can be considered to calculate the grasping point on-the-fly where the system detects grasping point of an object in real time. For both offline or online grasping point calculation, we can refer to the database created for grasping points. For calculating the grasps the system has to consider several factors which includes the number of fingers in a hand, the position of fingertips on a hand, the placement of fingertips on the object, the 6D pose collected for the object from the recognition structure, the task wrench space for the grasping task and others.

With the recognized features the system calculates several grasps by which the object can be picked up. But as the final constraint is to calculate feasible and stable grasps, the system has to discard the grasps which does not implement feasibility. We also have to consider the factor of in-hand manipulation in the picking tasks, where the force-closed grasp provides the best solution among others. In a force-closed grasp the hand does not allow external motion. The role of no-slip conditions also have to be considered as for the task of pick and place, there is room of errors by unstable grasps where the object may slip from the hand or

may bring harm to the hand, the arm or the collaborating human in the environment.

1.2.1 HAND DESIGN FOR GRASPING

Robotic hands are kinematic chains with a tree topology consisting of several common joints that branch to a number of serial chains, each of them corresponding to a different end-effector. A typical example of a kinematic chain with a tree topology is a wristed, multi-fingered hand. The placement of the end-effectors of the hand plays the most vital role in the tasks of grasping objects in the context of stability and feasibility. Without a proper placement of fingers there would still be risk of slippage of object, improper grasping, harm to the object, hand or to the environment. Though the initial work would be performed on the Barrett hand, which is a reputed hand hardware in the robotics community, the final work will be performed on the self-designed hands. The assessment of the new designed hands will be based on the same task performed over and over again with the reputed hardware and self-designed hands and comparing their performance based on stability, calculation of feasible grasping points, time-complexity of the task, performance measure for some general tasks.

Hand design is based on tree topology structure and highly focused on the task in hand, which considers the task wrench space (TWS) of the task. Although it means that there is very little room for error and the task should be dealt with exact kinematic synthesis. One of the first steps for exact kinematic synthesis is to calculate the maximum number of positions that can be used, which define the workspace of the chain. In the case of tree topologies, we have to consider a task having the same number of positions for each of the multiple end-effectors; this means that we are dealing with a coordinated action of all those end-effectors, denoted as *simultaneous task*. We denote a tree topology of a hand as a kine-

matic chain as that of a chain having a set of common joints spanning several chains and ending in multiple end-effectors. The topology is modelled by Graph Theory. The kinematic chain is represented as rooted graph, with the root vertex being fixed with respect to a reference frame. In tree topologies, a vertex can be connected to several edges defining several branches; a tree topology will always have links that are ternary or above, which are identified in the graph as a vertex spanning several edges.

1.2.2 GRASP SYNTHESIS

Grasping synthesis needs to be performed when a contact is formed or broken. A grasping synthesis system should compute space of forces and torques that can be applied by the grasp where the grasps should be numerically evaluated. The evaluation of grasps can be performed by comparing grasps of one hand with one object, comparing grasps of many hands with one objects, comparing grasps of many hands across a task specified object set. Synthesis of grasps can be performed in two ways. The analytical approach determines the contact locations on the object and the hand configuration that satisfy task requirements through kinematic and dynamic formulations. On the other hand empirical approaches try to mimic human grasping to select a grasp that best conforms to task requirements and the target object geometry. The analytical approach is based on force-closure properties, but it is independent of tasks and computationally highly expensive. The empirical approach tries to find stable grasps in the lower-dimensional task space. In this approach the semantic, space, size of the object manipulated in the task is been provided. Many reputed grasp synthesis technologies such as GraspIT are based on the empirical approach.

1.2.3 LEARN TO GRASP

The evaluated grasps can be stored in a dataset for future references such as grasping other similar kind of objects. As calculating grasping points of an objects is a time-consuming and tedious task, selecting the most feasible and stable grasps brings other notions of complexity. The complexity can be highly reduced by implementing machine learning techniques such as Support Vector Machines (SVM), Hidden Markov Models (HMM), Bayesian Learning, Concurrent Neural Networks or Deep Learning Techniques. This techniques can allow the system to not only identifying the most stable grasping points, also they can empower the system to learn grasps which will be highly helpful in determining the grasping points for similar type of objects or novel objects.

1.3 MOTION PLANNING FOR GRASPING

When the objects are recognized in the environment for grasping and the grasping points are been evaluated, now it is the role of the arm of the robot to reach a certain position where the hand can execute the grasps. For manipulating the arm to a such position the inverse kinematics should be calculated in real time. As the environment is unstructured, there cannot exist a single solution for the arm to reach the desired position. The trajectory can be varied based on the placement of the object, obstacles in the path of the arm, the human involvement in the environment or the occlusion of the robot arm in the environment. So the trajectory defined by the motion planned by the system should be collision aware and constraint aware.

For collision aware inverse kinematics solution, the system has to consider the obstacles present in the environment such as table corner , robot bodies or the other objects which

are not meant to be grasped. The system has to maintain a collision map of the environment in real time where the inclusion or exclusion of an object in the environment can be detected by constant updates. Also the system has to filter out the own body of the robot which can occlude the environment. The system has to perform several types of goals such as pose goal, joint goal or cartesian point trajectories. For a simple task such as picking or placing an object in the environment, the system has to perform defined trajectories which may be a single type of goal or a combination of all of them. The system have to find constraint aware inverse kinematics solutions, where a special type of motion should be defined as constraint. As for example, for picking and placing a glass of water, the system should constraint the arm roll rotation, where the arm should not rotate in such a way where the water in glass will be spilled out. But for pouring the water in a different container we have to allow such motion.

To generate collision aware and constraint aware inverse kinematics solutions, search based algorithms or tree-searching has been proved to be best solutions. The motion planning task can be performed with the help of libraries such as OMPL (Open Motion Planning Library) which is a collection of motion planning algorithms. We can utilize search based motion planners such as RRT (Rapidly exploring Random Trees) or SBPL(Search based motion planners),Gradient based optimization techniques such as CHOMP(Covariant Hamilton Optimization for motion planning), or semi-constraint trajectory planners such as Descartes planner.

1.4 DESIGN OF TASK-SPECIFIC ROBOTIC HANDS

For the design of task-specific robotic hands, first the task-in-hand need to be specified in the terms of motion. For that purpose we are utilizing the motion capturing system. The motion is simplified in terms of characteristic points which define the task as solvability points. For this process we have utilized various methods such as PCA(Principal Component Analysis), K-Mean algorithm and EM(Expectation Minimization) algorithm. After the task is simplified we choose the topology of the hand which would be able to perform the task. In this context, we are defining the robotic hands as kinematic chains with a tree topology consisting of several common joints that branch in different serial chains, each of them corresponding to a different end-effectors. The process of formulating the topology in terms of solvability has been shown in [MA12]. After the topology has been selected for the specified task, the screw axis for each joint and link length can be simplified by Artreeks in the means of Genetic Algorithm. The hand can be designed with the specified screw axis and link lengths.

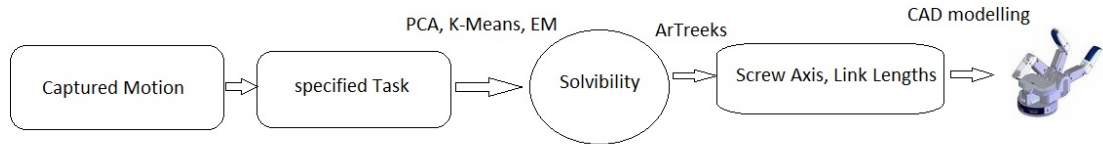


Figure 1.1: Flowchart of designing Robotic Hands: From capturing motions to designing robotic hand specified to the task

1.5 SIMULATION AND REAL HARDWARE

By the recent simulation software environment, the real hardware situation can be perfectly simulated. We are utilizing such softwares namely Gazebo and OpenRave. In gazebo the robot model or object model can be imported in URDF (Unified Robot Description Format) where the exact measurement of the model can be loaded by CAD information. In OpenRave the objects are modeled in DAE format, which is also loaded by CAD models. The robotic system comprises of an UR-5 arm and barrett hand. The environment is perceived by a Kinect camera. The hand model can be altered based on developed arm in later stages for assessment in real hardware and simulation. The same exact measurement of the components is imported in Gazebo simulation environment and can be evaluated by trying to grasp different object models which are also presented in URDF format. The simulated arm, hand and perception devices work in same way such as the real hardware. In a ROS-based environment, the nodes of the components publishes the same topic such as the real hardware. After the grasping framework is been evaluated in the simulation environment, the entire grasping system would be evaluated and assessed in the real hardware.

Our vision for robotics depends on perception...

GARY BRADSKI

2

Object Recognition

The robots are accumulating themselves in home environment. When the term "robot" appears, the first image comes to our mind is huge arms working in the industry and factories stealing jobs of human. They are working day and night precisely where repetition and accuracy is the main weapon. But the situation is changing rapidly. Robots are not only limited to industries only. Scientists around the world is trying to move them in our human environment as a social devices helping the elders, helping children even in personal services. While in the industrial environment, the task of the robot is quite easy, as the environment is solely managed for the robot. The environment is modeled such as where the sensors working for the robot does not have large constraints. The sensors are dedicated to

detect only the simple objects in the production. On the other hand, in the unstructured environment, such as our home, the environment is modeled for the benefit of the human. The robots and their sensors are modeled in such a way where the robot have to acquire the human environment. The environment is changing constantly. There is no guaranty that an object would still be in a same place where it was half and hour ago. So the robot have to detect and recognize the object every time while the robot tries to find the object.

2.1 2D v/s 3D

Humans perceive their environment as 3D and perceives the world on basis of what they see. Looking over an object we can decide the object's distance from us. The preliminary results and progress over 2D community suggests that for robots the world can be perceived in same way by image recognition. While for few years, there was a substantial belief that by only perceiving the world in 2D, we can achieve the same results of perception as human. But this erroneous theory ignores the facts of simple constraints such as lighting conditions, scale, occlusions and various others. In fig 2.1 and 2.2 we can see how framing the perception problem in this way can lead to failures in capturing the true semantic meaning of the world.

There are mainly two reasons for preferring 3D over 2D. The first one is monocular computer vision applications are flustered by both fundamental deficiencies in the current generation of camera devices and limitations in the datastream itself. The former will most likely be addressed in time, as technology progresses and better camera sensors are developed. An example of such a deficiency is shown in Figure 2.1, where due to the low dynamic



Figure 2.1: Model matching failure in underexposed 2D images. None of the features extracted from the model (left) can be successfully matched onto the object of interest (right)(image from <https://pointclouds.org/>).



Figure 2.2: An example of a good model match using features extracted from 2D images (left), where a beer bottle template is successfully identified in an image. However, zooming out from the image, we observe that the bottle of beer is in fact another picture stitched to a completely different 3D object (in this case a mug). The semantics of the objects in this case are thus completely wrong (image from <https://pointclouds.org/>).

range of the camera sensor, the right part of the image is completely underexposed. This makes it very hard for 2D image processing applications to recover the necessary information for recognizing objects in such scenes. On the other hand computer vision applications make use of 2D camera images mostly, which are inherently capturing only a projection of the 3D world. Figure 2.2 attempts to capture this problem by showing two examples of matching a certain object template in 2D camera images. While the system apparently matched the model template (a beer bottle in this case) successfully in the left part of the image, after we zoom out, we observe that the bottle of beer was in fact another picture in itself, stitched to a completely different 3D geometric surface (the body of a mug in this case). This is a clear example which shows that the semantics of a particular solution obtained only using 2D image processing can be lost if the geometry of the object is not incorporated in the reasoning process. [RBRBo9]

2.2 3D CAPTURING DEVICES

Three dimensional (3D) point clouds provide very important cues to analyze objects or environments. They are, for instance, heavily used in topographical mapping, where an airplane or a satellite passes over an area and takes several snapshots with a laser range finder. [ESSo5] used laser range finders to build height map of scanned area. [JWS04] tried to develop such kind of distance sensor in the field of mobile robotics, where the robot tries to find path, avoid obstacles from the 3D data. [PAHo3] used point clouds to reconstruct historic monuments. One of the most important properties for mobile robots is their ability to process information from the surrounding environment. There has been a lot of work done in the area of laser range finders and scanners for perceiving 3D. [JCCEo1] discusses the

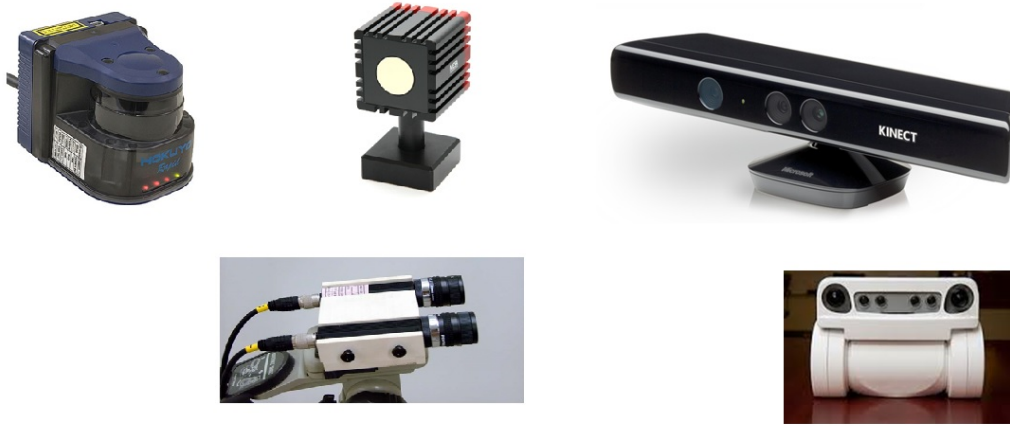


Figure 2.3: 3D capturing devices: (from the left)laser range finder, Time-of-flight cameras, Kinect Camera, Stereo Camera, PR2 head with two stereo camera pair.

used of radial basis function to reduce the number of point clouds in a point cloud dataset. Same type of work has been done in [CBX95] where algorithms are presented to reconstruct objects from a point cloud dataset. In order to perform its tasks the robot needs to be able to perceive its environment. In navigation, for instance, the robot has to recognize certain features that can be used as landmarks. During the DARPA Grand Challenge and Urban Challenge, laser range finders have been used extensively on board of the different autonomous vehicles to build a 3D map of the immediate environment. In order to grasp an object, the robot needs first to find the object in the scene. To execute this task, 3D object detection and classification have to be performed. Recognizing objects in 3D is a completely different problem. [Joh97a] presented work where the 3D descriptors are presented in terms of spin images for object recognition. There are several types of 3D capturing devices available. The use of cheap devices such as Microsoft kinect has revolutionized the 3D capturing system for RGBD images. Figure 2.3 presents some of the devices used for cap-

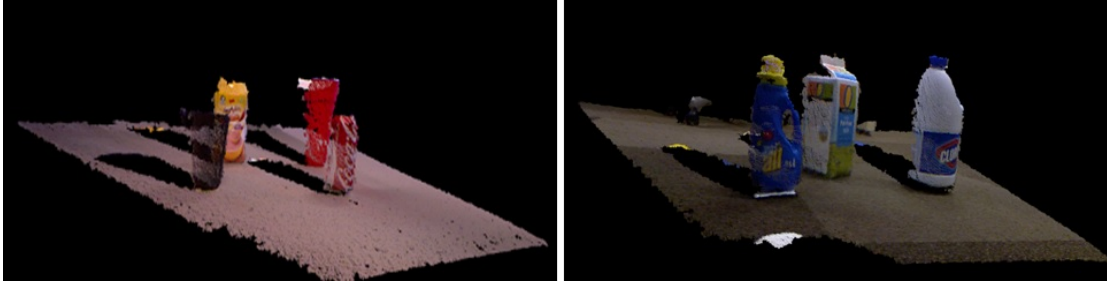


Figure 2.4: An example of point cloud data with some example object, cup, glass, gerberbox,cokecan,milk,colorax

turing the environment in 3D. The Time-of-flight cameras or Laser Measurement systems (LMS) or LIDAR systems sends rays of light(laser) or sound(sonar) in the world, which will reflect and return back to sensor. Knowing the speed with which a ray propagates, and using precise circuitry to measure the exact time when the ray was emitted and when the signal returned, the distance d can be estimated as (simplified):

$$d = \frac{ct}{2}$$

where c is the speed of ray and t is time for signal emitted and coming back to the sensor. In contrast, triangulation techniques usually estimate distances using the following equation (simplified):

$$d = \left\| \frac{ft}{x_1 - x_2} \right\|$$

where f represents the focal distance of both sensors, T the distance between the sensors, and x_1 and x_2 are the corresponding points in the two sensors. Though many different triangulation systems exist, the most popular system used in mobile robotics applications is the stereo camera. The sensor utilized here is a kinect sensor. An example of 3D point cloud has been presented in Figure 2.6. Dense 3D point clouds can be processed and meaningful

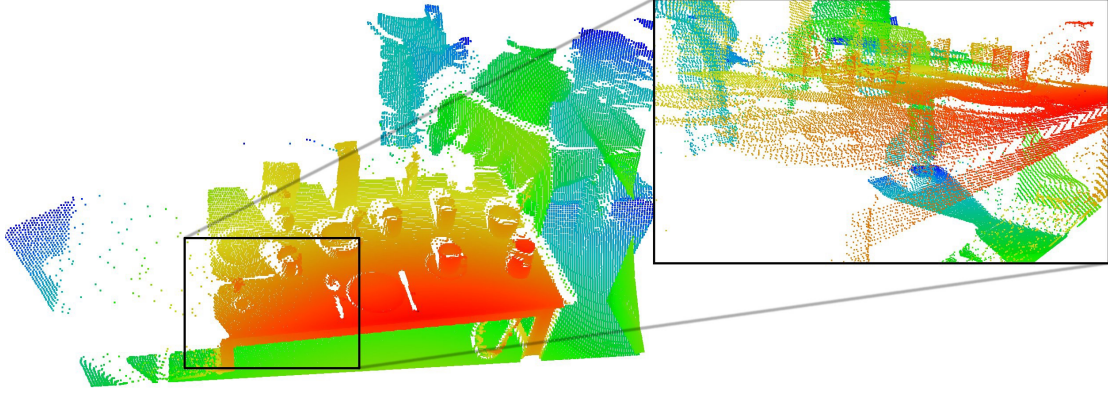


Figure 2.5: Point Cloud Dataset showed in distance (red is close, blue is far away) spectrum(image from <https://pointclouds.org/>).

data can be extracted by Point Cloud Library(PCL). [RBRBo9]

2.3 PROCESSING IN 3D

An important aspect when dealing with point cloud representations, is that they are able to store much more information than just the 3D positions of points as acquired from the input sensing device. During the acquisition process of a point cloud P , the distances from the sensor to the surfaces in the world can be saved as properties for each resultant 3D point $p_i \in P$. The representation of pointcloud based on their distances is presented in fig 2.5 where the close objects are shown in red and far objects are shown in blue.

2.3.1 3D DATA REPRESENTATION

Giving the fact that all the above examples need point cloud representations which hold multiple properties per point, the definition of a point $p_i = \{x_i, y_i, z_i\}$ changes to that of $p_i = \{f_1, f_2, f_3, \dots, f_n\}$, where f_i denotes the a feature value in a given space (color, class level, geometry etc.) thus changing the concept of 3D to nD. From these requirements, we can

$$\begin{bmatrix} x_1 & y_1 & z_1 & r_1 & g_1 & b_1 & l_1 & d_1 & \dots \\ x_2 & y_2 & z_2 & r_2 & g_2 & b_2 & l_2 & d_2 & \dots \\ & & & & & & & & \dots \\ x_n & y_n & z_n & r_n & g_n & b_n & l_n & d_n & \dots \end{bmatrix}$$

Figure 2.6: Point cloud data shown in terms of x,y,z,r,g,b,d

deduce that an appropriate I/O data storage format for a point cloud P , would be to save each point with all its attribute values on a new line in a file, and thus have a file with n lines for the n total number of points in P . A fictitious example is shown in the Fig. 2.6, where x_i, y_i, z_i represent the 3D coordinates, r_i, g_i, b_i are color associated with each point, l_i is the class level, and d_i is the distance from the surface. To obtain the geometry around a query point p_q , most geometric processing steps need to discover a collection of neighboring points p^k , that represents the underlying scanned surface through sampling approximations. A solution is to use spatial decomposition techniques such as kd-tree or octree, and partition the point cloud data into P chunks. Though by implementation most spatial decomposition techniques can represent as a volumetric representation of P as shown in fig 2.7. Here all

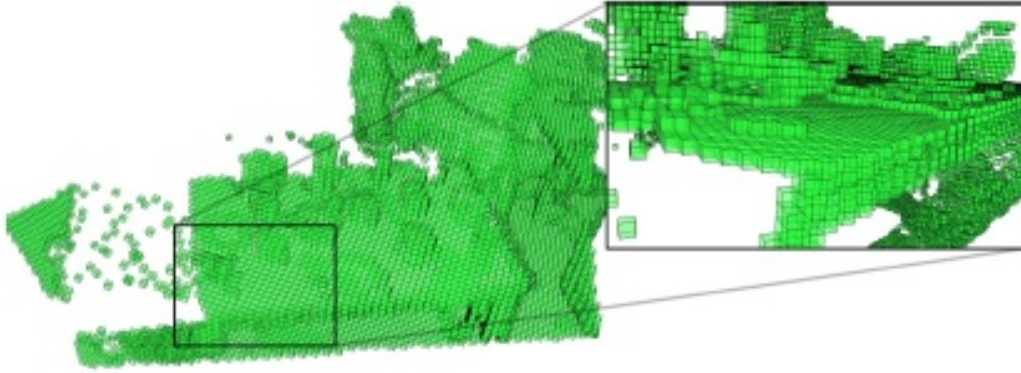


Figure 2.7: Volumetric representation using an octree structure, with a leaf size of 1.5cm (image from <https://pointclouds.org/>)

the points are enclosed in boxes with different widths namely "voxels".

2.3.2 NORMAL ESTIMATION

Surface normals are important properties of a geometric surface, and are heavily used in many areas such as computer graphics applications, to apply the correct light sources that generate shadings and other visual effects. Given a geometric surface, it's usually trivial to infer the direction of the normal at a certain point on the surface as the vector perpendicular to the surface in that point. However, since the point cloud datasets that we acquire represent a set of point samples on the real surface, there are two possibilities:

1. obtain the underlying surface from the acquired point cloud dataset, using surface meshing techniques, and then compute the surface normals from the mesh;
2. use approximations to infer the surface normals from the point cloud dataset directly.

The solution for estimating the surface normal can be reduced to an analysis of the eigenvectors and eigenvalues (or PCA – Principal Component Analysis) of a covariance matrix created from the nearest neighbors of the query point. More specifically, for each point p_i , we can assemble the covariance matrix C as follows:

$$C = \frac{1}{k} \cdot \sum_{i=1}^k (p_i - \vec{p}) \cdot (p_i - \vec{p})^T, C \cdot \vec{V}_j = \lambda_j \cdot \vec{V}_j, j \in \{0, 1, 2\}$$

where k is the number of point neighbors considered in the neighborhood of p_i , \vec{p} represents the 3D centroid of the nearest neighbors, λ_j is the j -th eigenvalue of the covariance

matrix and \vec{v}_j is the j -th eigenvector. An estimation of surface normal of a point cloud data has been presented in Fig. 2.8.

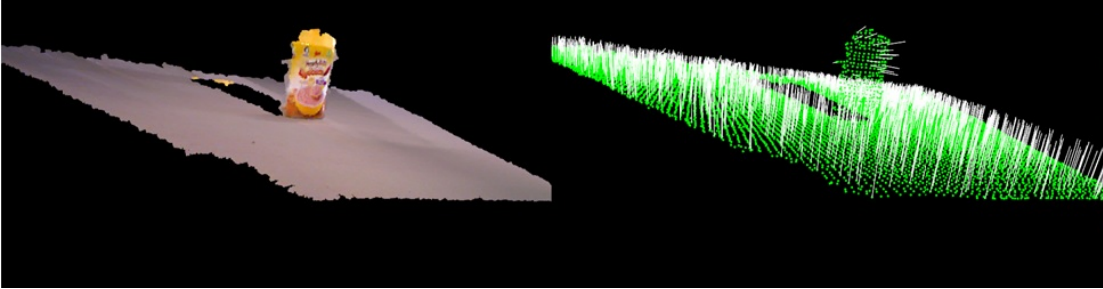


Figure 2.8: Point cloud data of a gerber box(left), the estimation of normals (right)

2.3.3 3D FILTERING

Point clouds data can be filtered in several ways. In the process of acquisition of 3D point cloud data, there may be several noise due to sensor inefficiency. Otherwise the point cloud becomes dense, where more data is present than we need. So processing the unneeded data makes the system unstable or increases the time needed for processing. In a voxel grid filter, the cloud is divided in multiple cube-shaped regions with the desired resolution. Then, all points inside every voxel are processed so only one remains. The simplest way would be to randomly select one of them, but a more accurate approach would be to compute the centroid, which is the point whose coordinates are the mean values of all the points that belong to the voxel. In a passthrough filter, certain points will be removed from a point cloud dataset whose values do not fall within a user-provided certain range. As for example, filtering point cloud data based on Y value will remove all the points situated on the table. In a conditional removal filtering, all the points will be filtered based on a certain condition provided by the user. In a statistical outlier removal filtering process, for every point, the

mean distance to its K neighbors is computed. Then, if we assume that the result is a normal (gaussian) distribution with a mean μ and a standard deviation σ , we can deem it safe to remove all points with mean distances that fall out of the global mean plus deviation. It runs a statistical analysis of the distances between neighboring points, and trims all which are not considered "normal". So the points outside of the region, will be removed. All of the filtering process have been shown in the Fig.

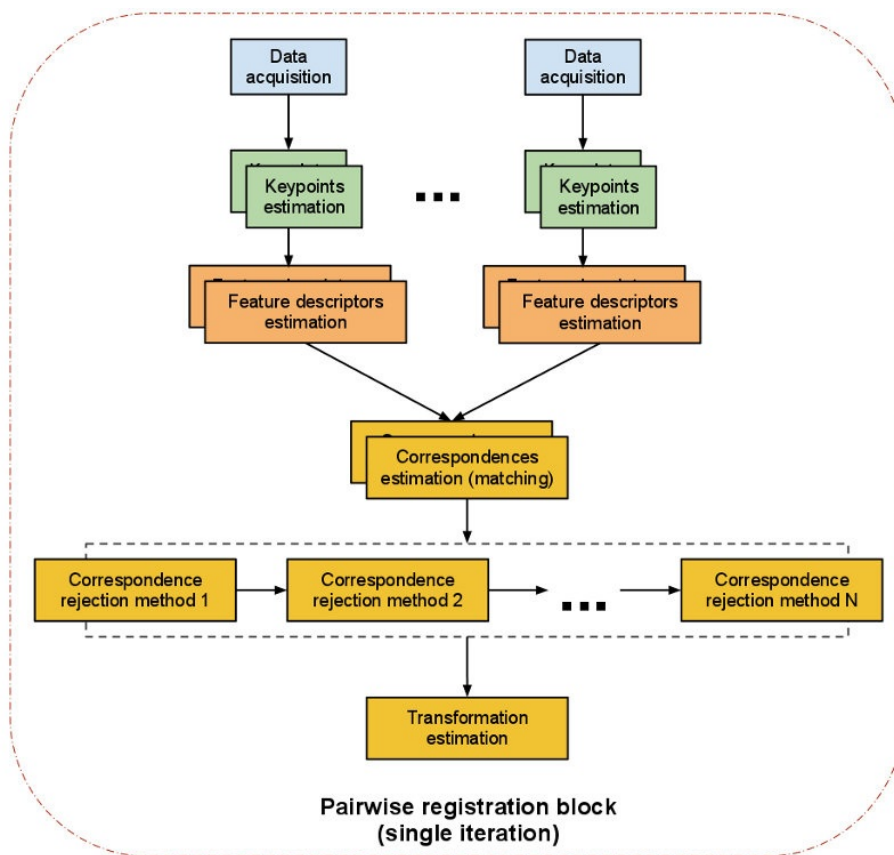


Figure 2.9: Flowchart showing point cloud registration by iteration (image from <https://pointclouds.org/>)

2.3.4 3D REGISTRATION

Registration is the technique of aligning two point clouds. The registration algorithm finds a set of correspondences between two point clouds. There should be an area in the scene which is captured by both point clouds. A linear transformation is then computed, which outputs a matrix that contains a rotation and a translation. The operations will be performed one point cloud get in place with respect to the other, with intersecting areas overlapping. The main task is to minimize the transformations between them. So the sensor should be moved in a steady interval. There are mainly two types of 3D registration.

1. ICP registration: ICP stands for Iterative Closest Point. It is an algorithm that will find the best transformation that minimizes the distance from the source point cloud to the target one. The problem is that it will do it by associating every point of the source cloud to its "twin" in the target cloud in a linear way, so it can be considered a brute force method. If the clouds are too big, the algorithm will take its time to finish, so we have to downsample clouds first.
2. Feature-based registration: the algorithm finds a set of keypoints in each cloud, computes a local descriptor for each, and then performs a search to see if the clouds have keypoints in common. If at least 3 correspondences are found, a transformation can be computed. For accurate results, several correspondences must be found. This method is faster than ICP, because matching is only done for the keypoints, not the whole cloud.

2.3.5 SEGMENTATION

Segmentation is the process of dividing the cloud data into segments or clusters, where each clusters represent meaningful object in the dataset. Segmentation can be performed based on points, normals or even textures. Segmentation task can be performed for extracting objects sitting on the table. We can detect the table and even the objects on the table. There are several segmentation technique namely Euclidean, Conditional, Region Growing, Color-based, Min-Cut etc.

2.4 3D FEATURE EXTRACTION

As point feature representations go, surface normals and curvature estimates are somewhat basic in their representations of the geometry around a specific point. Though extremely fast and easy to compute, they cannot capture too much detail, as they approximate the geometry of a point's k-neighborhood with only a few values. As a direct consequence, most scenes will contain many points with the same or very similar feature values, thus reducing their informative characteristics. For a feature to be optimal, it must meet the following criteria:

1. It must be robust to transformations: rigid transformations (the ones that do not change the distance between points) like translations and rotations must not affect the feature. Even if we play with the cloud a bit beforehand, there should be no difference.
2. It must be robust to noise: measurement errors that cause noise should not change the feature estimation much.

3. It must be resolution invariant: if sampled with different density (like after performing downsampling), the result must be identical or similar.

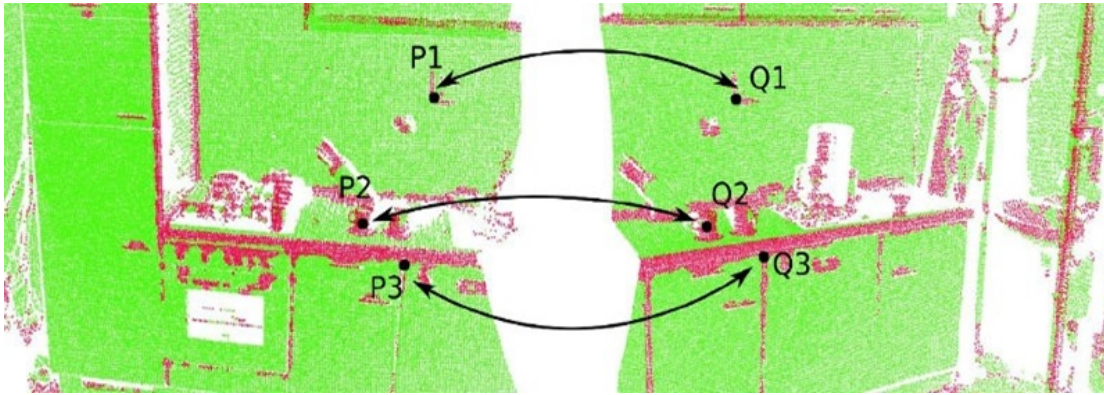


Figure 2.10: Finding Correspondences between point features of two clouds (image from <https://pointclouds.org/>)

There are many 3D descriptors implemented into PCL [RC11]. Each one has its own method for computing unique values for a point. Some use the difference between the angles of the normals of the point and its neighbors, for example. Others use the distances between the points. Because of this, some are inherently better or worse for certain purposes. A given descriptor may be scale invariant, and another one may be better with occlusions and partial views of objects.

After calculating the necessary values, an additional step is performed to reduce the descriptor size: the result is binned into an histogram [FTS10]. To do this, the value range of each variable that makes up the descriptor is divided into n subdivisions, and the number of occurrences in each one is counted. For example, a descriptor that computes a single variable, that ranges from 1 to 100. We choose to create 10 bins for it, so the first bin would gather all occurrences between 1 and 10, the second from 11 to 20, and so on. We look at the value of the variable for the first point-neighbor pair, and it is 27, so we increment the value

of the third bin by 1. We keep doing this until we get a final histogram for that keypoint. The bin size must be carefully chosen depending on how descriptive that variable is. The variables do not have to share the same number of bins, and also the bins do not have to be of the same size; if for example most values from the previous example fell in the 50-100 range then it would be sensible to have more bins of smaller size in that range.

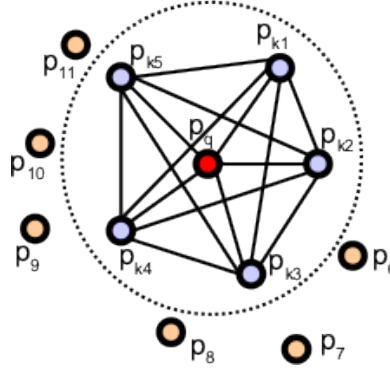


Figure 2.11: Point pairs established when computing the PFH for a point (image from <https://pointclouds.org/>)

The goal of the Point Feature Histogram (PFH) [RBRBo8] formulation is to encode a point's k -neighborhood geometrical properties by generalizing the mean curvature around the point using a multi-dimensional histogram of values. This highly dimensional hyperspace provides an informative signature for the feature representation, is invariant to the 6D pose of the underlying surface, and copes very well with different sampling densities or noise levels present in the neighborhood.

A Point Feature Histogram representation is based on the relationships between the points in the k -neighborhood and their estimated surface normals. Simply put, it attempts to capture as best as possible the sampled surface variations by taking into account all the interactions between the directions of the estimated normals. The resultant hyperspace is thus dependent on the quality of the surface normal estimations at each point.

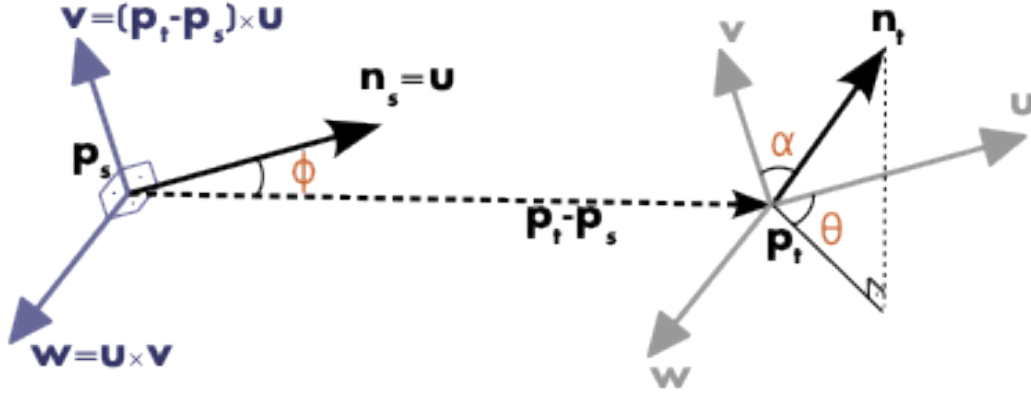


Figure 2.12: Fixed coordinate frame and angular features computed for one of the pairs (image from <https://pointclouds.org/>)

Fig 2.11 presents an influence region diagram of the PFH computation for a query point (p_q), marked with red and placed in the middle of a circle (sphere in 3D) with radius r , and all its k neighbors (points with distances smaller than the radius r) are fully interconnected in a mesh. The final PFH descriptor is computed as a histogram of relationships between all pairs of points in the neighborhood, and thus has a computational complexity of $O(k^2)$.

To compute the relative difference between two points p_i and p_j and their associated normals n_i and n_j , we define a fixed coordinate frame at one of the points as referenced in Fig 2.11.

$$u = n_s \quad (2.1)$$

$$v = u \times \frac{(p_t - p_s)}{\|p_t - p_s\|_2} \quad (2.2)$$

$$w = u \times v \quad (2.3)$$

Using the uvw frame, the difference between the two normals n_s and n_t can be expressed as a

set of angular features as follows:

$$\alpha = v.n_t \quad (2.4)$$

$$\varphi = u.\frac{(p_t - p_s)}{d} \quad (2.5)$$

$$\vartheta = \arctan(w.n_t, u.n_t) \quad (2.6)$$

where d is the Euclidean distance between the two points p_s and p_t , $d = \|p_s - p_t\|_2$. The quadruplet $\langle \alpha, \varphi, \vartheta, d \rangle$ is computed for each pair of two points in k -neighborhood, therefore reducing the 12 values (xyz and normal information) of the two points and their normals to 4. There are two types of feature descriptors. Local feature descriptors are computed

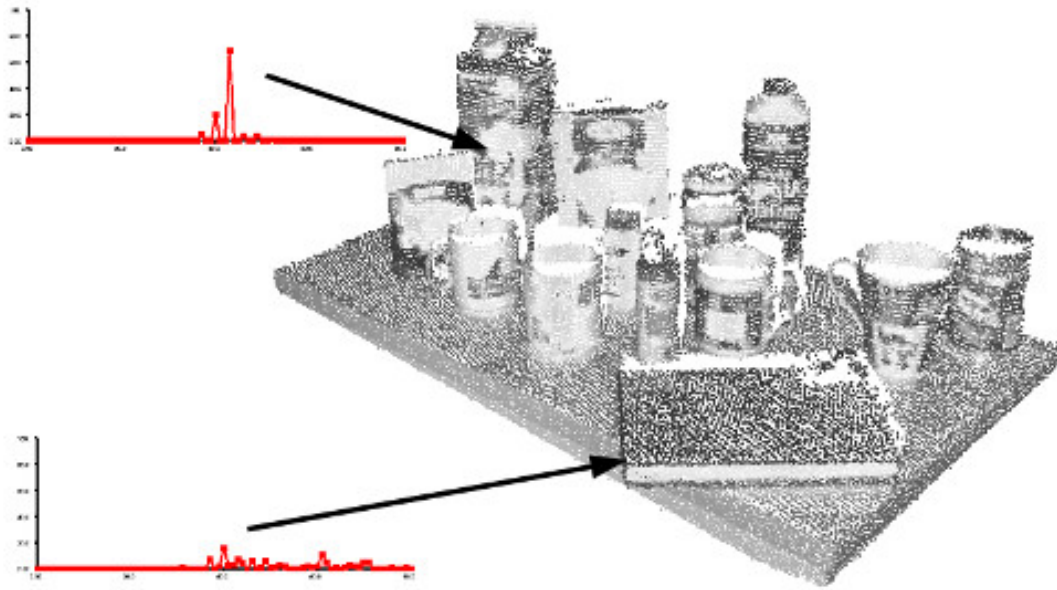


Figure 2.13: PFH estimation on a point cloud data (image from <https://pointclouds.org/>)

for individual points that we give as input. The concept of object is not valid here as the matching is done for all the points in the point cloud data. Some of the local descriptors

<i>Name</i>	<i>type</i>	<i>size</i>	<i>Custom Point Type</i>
<i>PFH(PointFeatureHistogram)</i>	local	125	Yes
<i>FPFH(FastPointFeatureHistogram)</i>	local	32	Yes
<i>RSD(Radius – basedSurfaceDescriptor)</i>	local	289	Yes
<i>3DSC(3DShapeContext)</i>	local	1980	Yes
<i>USC(UniqueShapeContext)</i>	local	1960	Yes
<i>SHOT(SignaturesofHistogramsofOrientations)</i>	local	352	Yes
<i>SpinImages</i>	local	153	No
<i>RIFT(Rotation – InvariantFeatureTransform)</i>	local	32	No
<i>NARF(NormalAlignedRadialFeature)</i>	local	36	Yes
<i>RoPS(RotationalProjectionStatistics)</i>	local	135	No
<i>VFH(ViewPointFeatureHistogram)</i>	global	308	Yes
<i>CVFH(ClusteredviewpointFeatureHistogram)</i>	global	308	Yes
<i>OUR – CVFH(Oriented, UniqueandRepetableCVFH)</i>	global	308	Yes
<i>ESF(EnsembleofShapeFunctions)</i>	global	640	Yes
<i>GFPFH(GlobalFastPointFeatureHistogram)</i>	global	16	Yes
<i>GRSD(GlobalRadius – basedsurfaceDescriptor)</i>	global	21	Yes

Table 2.1: Local and Global Point Cloud Feature Descriptors

are FPFH(Fast Point Feature Histogram) [RBRBo9], RSD(Radius Surface based descriptor) [Mario, MPBB11a], 3DSC (3D Shape Context) [FHK⁺04, SBP02], USC(Unique Shape Context) [TSDS10a], SHOT(Signature of Histogram for Orientation) [FT11, TSDS10b], SI(Spin Images) [Joh97b, JH99], RIFT(Rotation Invariant Feature Transform) [LSP05], NARF(Normal Aligned Radial Feature) obtained by spherical projection and planar projection, RoPS(Rotational Projection statistics feature) [GSB⁺11] On the other hand global features correspond to the notion of object. So the matching is not done on the whole point cloud. A separate segmentation task is performed to obtain the object from the dataset and the feature extraction is performed on the detected object. Some of the global descriptors are VFH(Viewpoint Feature Histogram) [RBRH10], CVFH(Clustered Viewpoint Feature Histogram) [AAB11], OUR-CVFH(Oriented, Unique and Repetable Clustered Viewpoint Feature Histogram) [ATRV12],

ESF(Ensemble of Shape Function) [WV11], GFPFH(Global Fast Point Feature Histogram) [Rus09], GRSD(Global Radius based surface Descriptor) [MPR⁺10, MPBB11b, KMP⁺11]. A few local and global feature descriptors and their corresponding features are presented in Table 2.1.

2.5 RECOGNITION PIPELINE

Ideally, a 3D object recognition system should be able to grab clouds from the device, pre-process them, compute descriptors, compare them with the ones stored in our object database, and output all matches with their position and orientation in the scene, in real time. Several components must then be implemented to perform these sequential steps, each one taking as input the output of the previous. This pipeline will be different depending on what type of descriptor we are using: local or global. The local and global object recognition pipeline is shown in Fig. 2.14

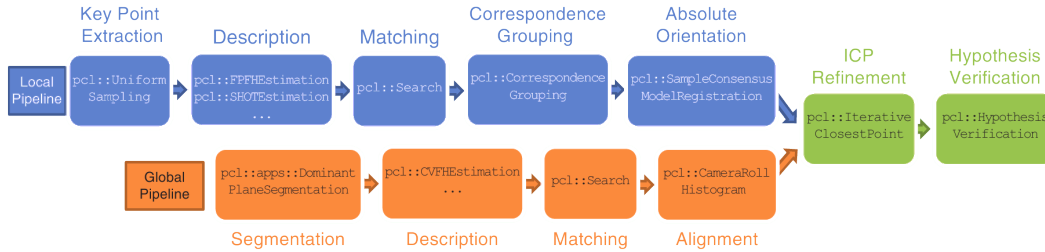


Figure 2.14: 3D recognition pipeline for local and global descriptors(image from <https://pointclouds.org/>)

Various snapshots are taken from different viewpoints. After this, the desired descriptors must be computed for every snapshot of each object, and saved to disk. If global descriptors are being used, a Camera Roll Histogram (CRH) should be included in order to retrieve the full 6 DoF pose, as many descriptors are invariant to the camera roll angle, which would

limit the pose estimation to 5 DoF. Finally, ground truth information about the camera position and orientation will make it possible to compare it against the result given back by the recognition system.

*The human hand is a very complex piece of equipment
with amazing capabilities*

Peter k. Allen

3

Grasp Synthesis

Robotic Grasping is one of the most needed qualities for manipulating objects. The robots to be placed in home or unstructured environment such as assistive services, healthcare, household applications, they should have the capability of grasping objects. In the recent years, the area of robotic grasping has gained significant attention and improvement. But still there is no grasping algorithm available that can be considered as an "ideal" grasping algorithm. The main cause behind that is the vast quantity of objects availability and the uncertainty in the environment. Also it is quite complex to present a grasping algorithm without the previous knowledge about the shape and pose of the object. The problem becomes more complex as the generated sensor data is noisy and incomplete.

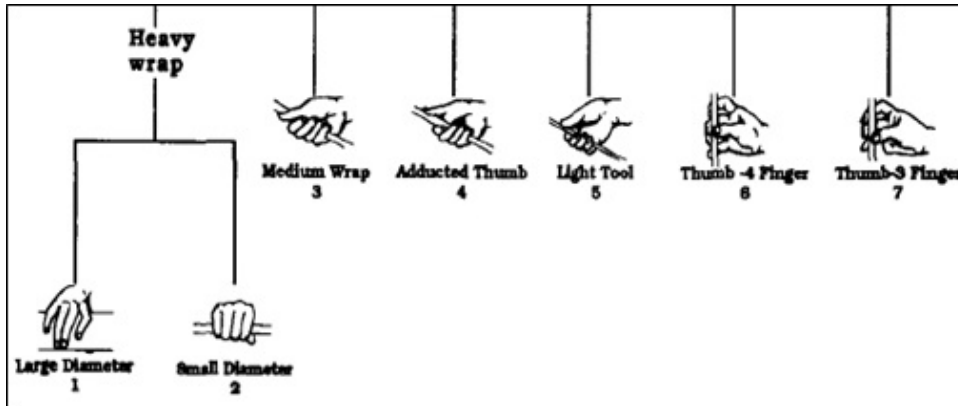


Figure 3.1: Cutkosky's Grasp Taxonomy

3.1 GRASP TAXONOMY

As most of the robotic hands try to mimic the human hand, understanding the grasp system of human hand is necessary. In the earlier work of [MR.31], a human grasp taxonomy is presented. The taxonomy is a systematic arrangement of the space of human grasps, and the organization of the taxonomy reveals some of the factors influencing grasp choice. Grasps can be placed on a continuum according to object size and power requirements. The taxonomy shows how task requirements (forces and motions) and object geometry combine to dictate grasp choice. The taxonomy is based on observations of single-handed operations by machinists working with metal parts and hand tools. The machinists were observed and interviewed and their grasp choices were recorded as they worked. In addition, their perceptions of tactile sensitivity, grasp strength, and dexterity were recorded. The Grasp Taxonomy is presented in Fig. 3.1.

For a successful grasping algorithm, the system needs to successfully locate the object to be grasped in a cluttered environment. Then it should be segmented from the background and the pose should be estimated. Then the suitable and stable grasping region should be

located on the object where the hand should be placed for grasping without slipping or any deformation of the object.

Researchers around the world put various approaches for robotic grasping. The significant number of approaches based on grasping mechanism and interaction between the object and hand are discussed in [ABoo]. Also the role of hand design and control in the context of robotic grasping has been discussed in [EAG93]. [ASAS] has reviewed and divided the grasping approaches in mainly two aspects: analytical and empirical.

3.2 ANALYTICAL APPROACH

Analytical approach determines the contact locations on the object and the hand configuration that satisfy task requirements through kinematic and dynamic formulations. The approach can be assessed from two perspectives 1) Force-closure properties and 2) Task-compatibility.

3.2.1 FORCE CLOSURE GRASPS

Based on the object model a lot of approaches have been proposed. [JP93] has proposed a method where each point in a plane face was parameterized linearly with two parameters. The work also been extended to the formation of linear conditions for three and four finger force closure grasps. [Liu99] has generated an algorithm where force closure grasp can be achieved by n fingers. In their work, $n-1$ fingers were fixed in a position, so they could not generate force-closure grasp. A search have been performed for the location on the object face for n -th finger using a linear parameterization technique, by which force-closure grasp could have been achieved. Another method proposed by [DDoo], where the position of

force closure grasp for all fingers were found based on an initial random grasp. These type of methods consider objects as basic shapes such as boxes and here the selection face of grasping point is not been considered.

[DDo1] also analyzed the force closure properties with 7 frictionless contact. Discretization of the grasped object was performed so a large number of contact wrenches could have been found. [SEK09] has shown that the wrenches which allow the association of any three non-aligned contact points, could form a basis of the wrench space. By this formulation, force-closure condition can be achieved which can work with general objects. There are also several works have been done where the comparison of different grasps have been performed and the best suitable grasp is been chosen based on force-closure properties. One such criterion is presented in [BM94], where extraction of optimal two or three fingered grasps have been performed on 2D objects. Also the three finger grasp on 3D polyhedral objects have been achieved. Force-closure properties is computationally expensive as searching for an optimal grasp in the solution space needs heavy computational efforts. Using the predefined procedure, [CB03] generated some random grasps. [ATM99] have also address the issue with a set of rules that defined prior to grasping.

3.2.2 TASK COMPATIBILITY

Grasping is the main reason for manipulation of objects. So the grasping system need to be aware of the task-in-hand. There are several approaches have been performed where the intended task is been considered. In the work of [Chi98], an index have been considered where task compatibility is measured based on match of the optimal direction of the manipulator and the actual direction of the movement required for the task to be per-

formed. A measure to quantify the grasp quality in the context of the task is been presented in [ZL98]. To tackle the issue, [Pol97] presented an algorithm that could model a task wrench space(TWS) with a unit square. In later work, [CBo3] came up with the idea of object wrench space(OWS), which can describe the TWS in the context of OWS. It is also been shown that the computational complexity can be marginally decreased by modelling the OWS with a 6D ellipsoid.

3.3 EMPIRICAL APPROACH

In the empirical approaches, the system tries to mimic human grasping to select a grasp that best conforms to task requirements and the target object geometry. On the basis of computational complexity, empirical approach outperforms analytical approaches. The approach also can be assessed from two perspectives 1) Systems based on human observation and 2) Systems based on object observation.

3.3.1 SYSTEMS BASED ON HUMAN OBSERVATION

These type of methods are classified based on some policy learning or learning by demonstration where a human collaborator tries to teach the robot how to grasp. Later the robot tries to mimic that. The process of learning based on who, what and how to emulate the operator is been described in [ABo8]. In [So4, FK05] a seclized grasping framework is been proposed where the human collaborator and the robot stands in front of a table with objects to be manipulated. First the human shows the robot how to manipulated the object. Then the robot tries to eulated it by means of Hidden Markov Model(HMM). For the purpose magnetic trackers are been used. A method for grasping using vision and audio

has been proposed in [MHo6]. In the demonstration phase, the robot tries to track the hand of the operator stereoscopically. In later work in [MHo8], an advanced method for grasping has been proposed by Self-Organizing maps(SOM) and Q-learning approach. Vision based approach has also been presented in [JR08], where the system is constructed by three parts: grasp classification, measurement of the hand position relative to the object and grasp strategy for the robot to perform grasping. Developing grasping strategy using the concept of mirror neurons have been presented in [EO02]. Another approach for grasping has also been presented based on neural network in [FK05].

3.3.2 SYSTEMS BASED ON OBJECT OBSERVATION

This grasping approach considers object affordances, properties of the object and produces an algorithm that is generalized to find grasping points on any object. The strategy of using support vector machines(SVM) to generate mapping between the shape of the object, parameters of the grasp and quality of the grasp is been presented in [RP04]. The utilization of affordance learning strategy is been presented in [MS08]. A shape matching algorithm for grasping has been presented in [YLo7] considering the availability of 3D model of the object. [Sax06] proposed a grasping algorithm that finds a 2D point on an object by using support Vector Machines. Later in [Sax09, ASNo8] used supervised learning methods to detect grasping points using image features on novel objects. [YJS11] proposed a rectangle region detection technique using SVM-rank algorithm to detect robotic grasps. Deep Learning methods for detecting grasps is presented in [ILS13].

3.4 WRENCH SPACE

In a 3D space, a wrench is a vector composed of force and torque. The space of wrenches that may need to be applied during a task is the task wrench space. The space of wrenches that can be applied by a grasp is the grasp wrench space. A possible grasp quality measure can be the fraction of TWS and GWS.

$$\text{ContactWrenches } W_{(i,j)} = \begin{vmatrix} f_{(i,j)} \\ \lambda(d \times f_{(i,j)}) \end{vmatrix}$$

where λ is the Torque scale factor and $\lambda = \frac{I}{d_{max}}$

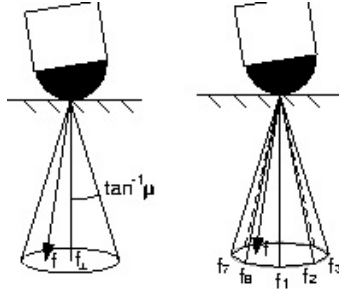


Figure 3.2: Friction cone concept.

3.5 FRICTION CONES

Friction at a contact point allows forces in directions other than the contact normal. We have to estimate friction cone as convex sum of a force vector on the boundary assuming a unit normal vector.

$$f \approx \sum_{j=1}^m \alpha_j f_j$$

3.6 TYPES OF GRASPS

Several types of grasps can be performed. They are mainly:

1. Force-closure Grasp: These type of grasp completely restrain the objects.
2. Torque-closure Grasp: In these type of grasp the fingers completely restrain any external force.
3. Equilibrium: The contact forces can balance the object weight and external forces and the origin is contained within grasp wrench space.
4. Manipulable Grasp: A manipulable grasp can impart arbitrary velocities on the object without breaking contact.

3.7 GRASPING SYSTEM DESIGN

A robotic grasping system should be based on the following properties:

1. Hand Design: There are mainly two levels of hand design. In the lower level, the number of fingers, kinematic structure should be considered. On the higher level, the mechanism design, motors and materials should be considered.
2. Hand Control Algorithms: In the high-level of control the system have to find an appropriate posture for a given task, while in the lower level, the capability of executing the desired posture.
3. The hand should be collaborated with the sensors such as tactile, vision, range sensors etc.

4. The system should have a priori knowledge of the objects such as shape, semantics and tasks to be performed

Most of the commercially available hands try to mimic the human hand which is a highly complex system. But it is very illusive to try to replicate the human hand, as in the whole lifetime we perform the task of grasping and learn strategies to grasp objects. Also the sensor system of the human is highly complex to be mimicked, as no sensor available which the can replicate the human sensors such as eyes for visual sensing and skin as tactile sensing. A grasping system should able to compute space of forces and torques that can be applied

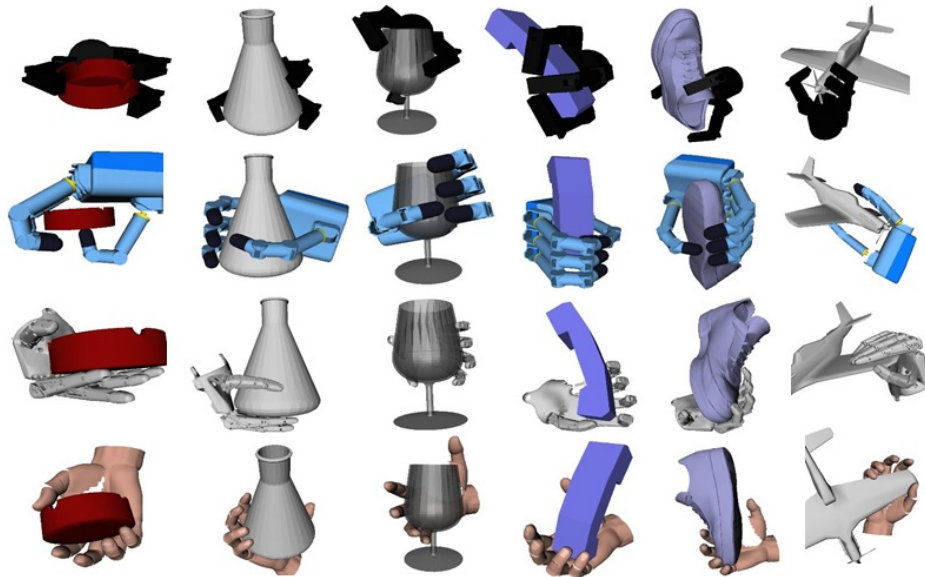


Figure 3.3: Grasp Planning Examples.

by the grasp. The system should numerically evaluate grasps by comparing grasps of one hand with one object, comparing grasps of many hands with one object, comparing grasps of many hands with one object and comparing grasps of many hands, across a task specified object set.

3.8 HAND POSTURE SUBSPACE

A grasp can be considered a point in a high-dimensional hand configuration space. To find a grasp we need to search this grasp space which is highly expensive and intractable. Low-dimensional subspaces can approximate most of the variance needed for common grasping tasks.



Figure 3.4: Various commercially available hands.

3.9 GRASP PLANNING USING EIGENGRASPS

Based on the work of [MR₃₁], another classical experiment for human taxonomy is performed in [MSS98] where human test subjects asked to grasp imaginary objects. The finger poses recorded with data-glove with the set of 59 typical household objects e.g. small and large ones, simple shape, more complex shape, tools etc. The motion dynamics were recorded, but analysis was performed for static grasps only. Statistical analysis was performed to search for common patterns. Principal component analysis reveals highly significant correlation of finger movements, called grasp synergies. It was shown that only 2 principal components could validate 85% of the whole space.

Eigengrasps can be visualized as the generalization of grasp taxonomy. It performs on a lower dimensionality basis for grasping. Eigengrasp generally derived from human user studies and mapped to robotic hands. For detecting a good grasp, we have to search the

whole space. The searching is generally done by simulated annealing. Energy function formulation attempts to bring pre-specified contact locations on the palm in contact with the object. Simulated annealing search is performed over 8 variables where 6 for wrist position and orientation and the other 2 for eigengrasp amplitudes. For eigengrasps, we need the fi-





















Eigengrasp 1			Eigengrasp 2		
Description	min	max	Description	min	max
Prox. joints flexion			Dist. joints flexion		
Spread angle opening			Finger flexion		
Prox. joints flexion Finger abduction			Dist. joints flexion Thumb flexion		
Thumb flexion MCP flexion Index abduction			Thumb flexion MCP extension PIP flexion		
Thumb rotation Thumb flexion MCP flexion Index abduction			Thumb flexion MCP extension PIP flexion		

Figure 3.5: Eigengrasps for different hands.

nal posture for stable form-closure grasps. The perfect grasp postures can not be found in EG space, but the result can come very close with simple heuristics. The form-closure tests can be performed in multi-threaded environment which can take advantage of multi-core architecture.

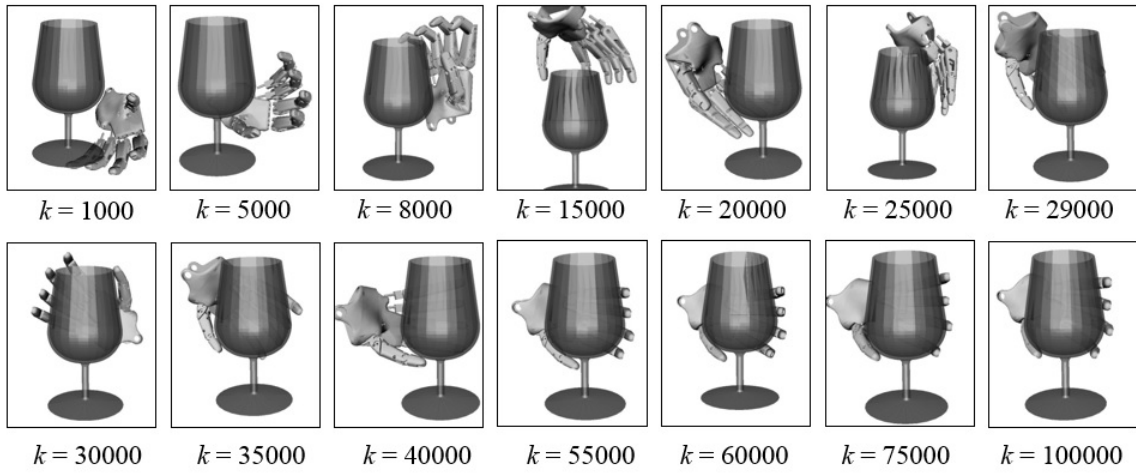


Figure 3.6: Stable Eigengrasps and the number of iterations taken .

Manipulation is the process of using one's hands to rearrange one's environment.

Matthew T. Mason

4

Motion Planning

As the system progresses in its task of picking and placing the object, after recognizing the object and calculating grasping points for the object, now its the task for motion planners to deliver the hand to the designated pose where the grasping task starts. Motion planning for a robotic arm is one of the most explored area in the robotics community. Still there exists no general framework for manipulating a robotic arm with constraints. The task of planning motion for an arm becomes exponentially complex, as the Degree of Freedom(DOF) increases.

Planning motion for a robot in home or unstructured environment is computationally difficult than the robots in industrial environment, as the industrial robots have to perform

same task over and over again and there is very little scope of constraints. Also the sensors are less prone to errors as they are implemented in a robot environment, whereas the sensors for domestic robots are not meant for human environment. They have to plan with the partial sensor information. While planning for a robotic arm in home environment, we have to consider the factors of collision and constraint. As the environment is meant for the human, there is a high probability for collision in the environment. Also we have to include the constraints as for example we don't want the robot to spill liquids from a glass while manipulating the glass filled with liquid.

From factory robots to robots in home, for many useful tasks, manipulation planning is an essential capability. Though manipulation seems like a standalone capability, it is based on few building blocks such as, perception of the environment, generating a world model, reasoning the world model, generating a joint-space path for executing the task, converting path to trajectory and execute. The block of reasoning is necessary as the system has to differentiate the objects to be manipulated and the static objects and verifying the capability of control system to move the robot to the desired pose. There is a severe role of controllers and planners: while controllers work locally, planners work globally. [Ber11]

Most successful methods for control theory follow the concept of minimizing the function in the neighborhood of robot's configuration space by gradient descent. There are several controllers that have been implemented based on the idea. Some of them are used for collision-avoidance [SK05], placing the end-effector in the task space [SK87], or for two-legged robots [SNO2]. In the [SK05], a concept of null-space projection has been introduced which prioritizes the constraints on a null-space and tries to adapt the lower-level constraints. When the prioritization problem is solved, it is still not sufficient to solve the

problem, as the gradient descent algorithm tends to find the local minima, not the global one. [Bel57] has shown how a global solution can be found in building control policies through dynamic programming. But such approaches do not validate in high-dimensional configuration space, which most robotic manipulator generates. Another popular approach has been presented in [AS97, DBCo4, MHVo8] for learning policies by demonstration.

Control methods are not sufficient to provide a practical solution to the problem of solving global planning for manipulators. On the other hand, sampling based planners [LKoo, LEKO96] have provided better solutions for high-DOF manipulators. Sampling-based manipulation planners are designed to explore the space of solutions efficiently, without the exhaustive computation required for dynamic programming and without being trapped by local minima like gradient-descent controllers [LKoo]. The global planning ability is highly important as the most trivial problem of collision-avoidance can be affected by a local minimum by gradient descent. Sampling based planners can work in state space, but they are most useful in c-space. The c-space paths can be converted to robot trajectories. [JBG85] Higher level reasoning is needed for commanding the planner. The commands can be obtained from an user, but for fully autonomous robots, the reasoning commands should drive the robot. Scientists and artificial intelligence researchers are developing higher level reasoning methods for several years. The most relevant higher level reasoning techniques for manipulating planning are STRIPS. [FN71, Nil80, MGT04] The STRIPS framework describes the world in terms of instances, predicates, and operators. The instances are the set of distinct objects in the world. The predicates are the properties of and relations between these instances. The operators are actions that change the state of

the world. An operator can be executed if its set of preconditions is valid and the execution of the operator produces a set of changes to the world. This type of planners are suitable for producing higher-level tasks for the robots. As for a pick and place operation, a robot needs to execute a set of motions, this type of planners can be helpful. But still the motion is difficult to express in propositional or first-order logic. For our task, it would be beneficial for the reasoning system to decide which object to grasp and the motion planner to do the rest.

4.1 MOVEIT

MoveIT is motion planning interface that incorporates various techniques such as Inverse Kinematics, perception, control and planning in a single framework for planning and executing complex trajectories for robots with higher DOF. The main architecture of MoveIT has been presented in fig. 4.1

4.1.1 CONFIGURATION

MoveIT [SC] requires robot description and some parameter configurations. It relies on URDF(Unified robot Description Format) for robot link and joint description, joint transmission values. Also the SRDF(Semantic Robot Description Format) needs to be provided as the SRDF maintains a tree structure for collision matrix and it is utilized for the self-filtering and self-collision for the robot. Moveit also relies on several configurations including joint limits, kinematics, motion planning, perception and other information.

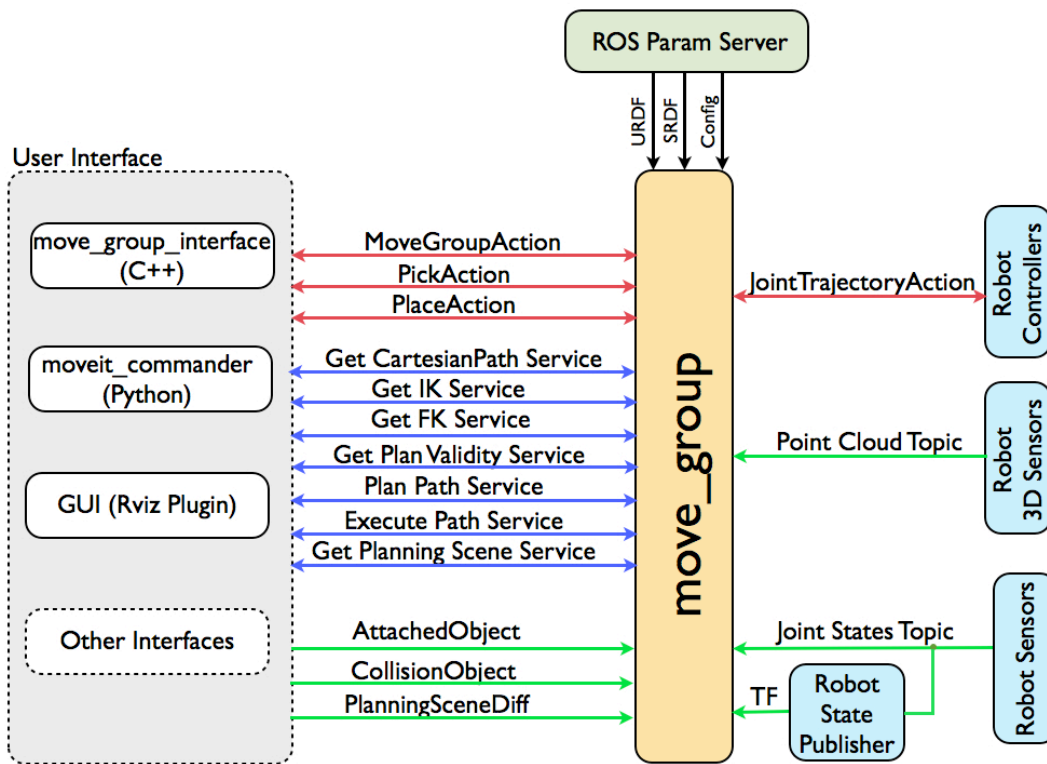


Figure 4.1: The MoveIt architecture

4.1.2 ROBOT INTERFACE

For planning the robot needs various state information: The joint state information provides the current state of the joints. The transform information is also needed as ROS-architecture based system keeps track of every joint and links by a library called TF. MoveIT generates the transform information from the library. For manipulating a robot either in real time or in simulation, the package publishes its command through several action interfaces. MoveIT needs information for all the action topics. The system also needs to keep track of the planning scene by which it can characterize itself, the objects need to be manip-

ulated, obstacles and the world environment.

4.1.3 MOTION PLANNING

The motion plans are executed by the motion planning requests. There are mainly three types of motion plan request. They are:

1. Pose : The final position of the end-effector is been provided as input.
2. Joint Space: The final joint positions of the arm are provided to the robot.
3. Cartesian space: The robot have to move through several positions which generates the cartesian space trajectory.

When the motion plans are executed the collision checking are automatically performed including the self-collision of the robot. When the robot picks up an object, the object is automatically added to the robot model and deleted from the world environment. The object is been considered as a part of the robot model. When the robot later places the object in the environment, the object is automatically deleted from the robot model and added to the world environment. There are several constraints that can be posed on the motion or trajectory of the robot.

1. Position Constraints: Constraints can be imposed where the position of a link to lie within a region of space.
2. Orientation Constraints: Constraints can be imposed where the orientation of a link can lie within a range of roll, pitch and yaw.

3. Visibility Constraints: Constraints can be imposed where a point on a link can be restricted to lie within the visibility cone of a particular sensor.
4. Joint Constraints: Constraints can be imposed where a joint is restricted within specified values.
5. User-specified constraints: Constraints can be generated based on the need of the application.

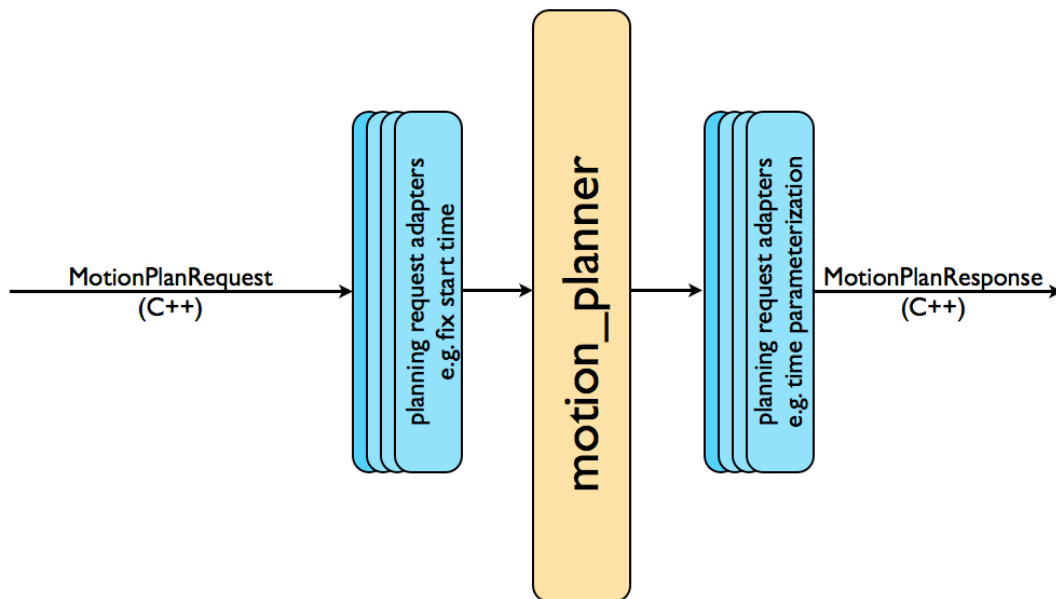


Figure 4.2: Planning adapters

4.1.4 MOTION PLANNING PIPELINE

The result of a plan generated by MoveIT is a desired trajectory with desired maximum velocity and acceleration at joint level. The motion planning pipeline collaborates motion planners with motion planning adapters. The role of the motion plan adapters are preprocessing the postprocessing the trajectories generated by MoveIT motion planners. In the preprocessing step, the adapters check if the start state is in collision or not. The post processing step performs the time parameterization of the final trajectory. The motion plan adapters are shown in fig 4.2.

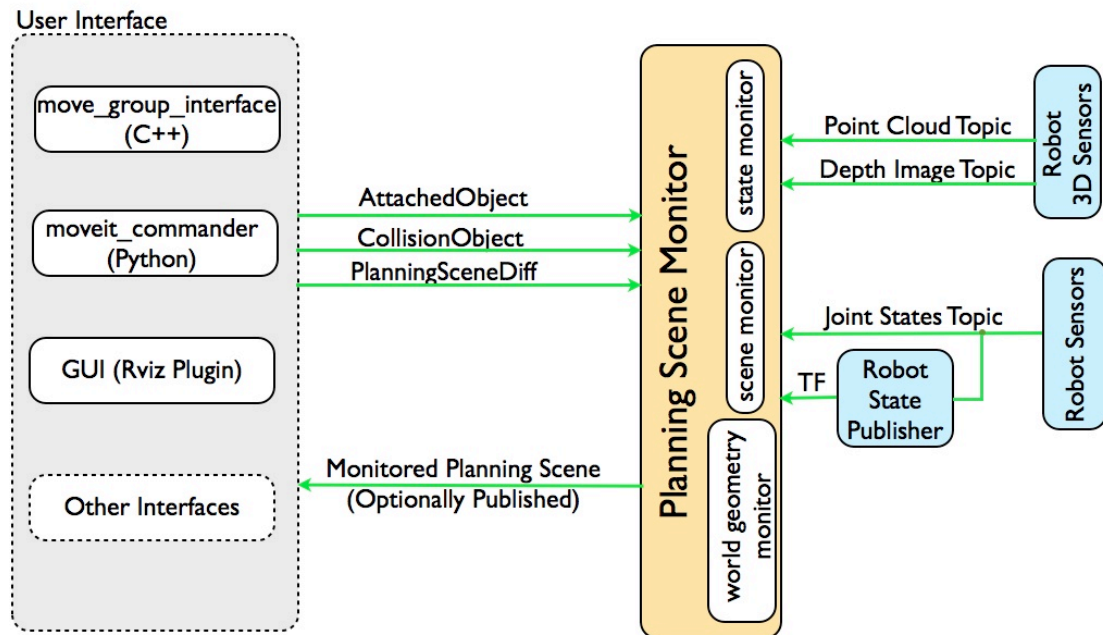


Figure 4.3: Planning scene

4.1.5 OMPL

OMPL stands for Open Motion Planning Library [SMK12]. OMPL implements randomized motion plans in MoveIT. OMPL is generated for general motion planning. MoveIT utilizes OPML as the main motion planners. There are several motion planners included in the motion planning for MoveIT. The planners receives information from the world as planning scene, robot state information and sensor information.

4.1.6 WORLD GEOMERTY MONITOR

The World Geomerty monitor utilizes information from sensor or user interface to create a model of the robot. The world is a 3D representation of the environment and been imported to the planning scene. The 3D representation is handled by occupancy map monitor and it relies on the 3D sensor and images sensor of the robot. It creates an octomap [HWB⁺₁₃] from the 3D sensor of the robot and continuously updates the map for real time collision avoidance. Fig 4.4 shows the perception architecture of MoveIT. The moveit architecture is been shown on a PR2 robot in fig. 4.5. In the left the robot plans path avoiding the obstacle. In the middle and right the robot creates octomap of the environment.

4.2 OPENRAVE

OpenRAVE [Diaio] provides an environment for testing, developing, and deploying motion planning algorithms in real-world robotics applications. The main focus is on simulation and analysis of kinematic and geometric information related to motion planning.

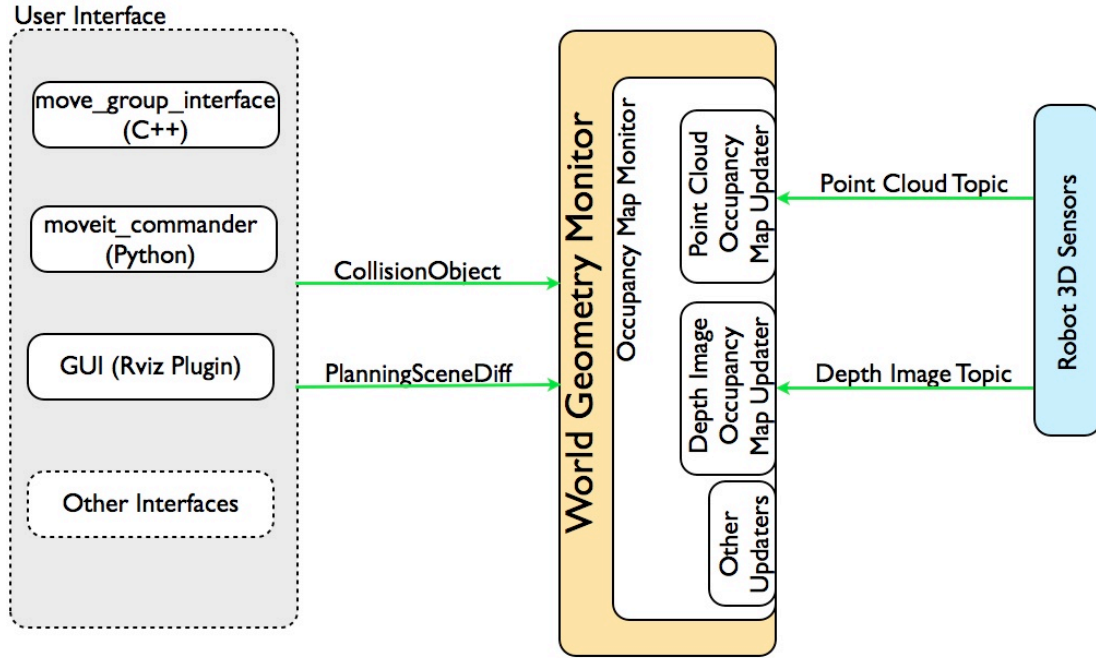


Figure 4.4: MoveIT perception pipeline

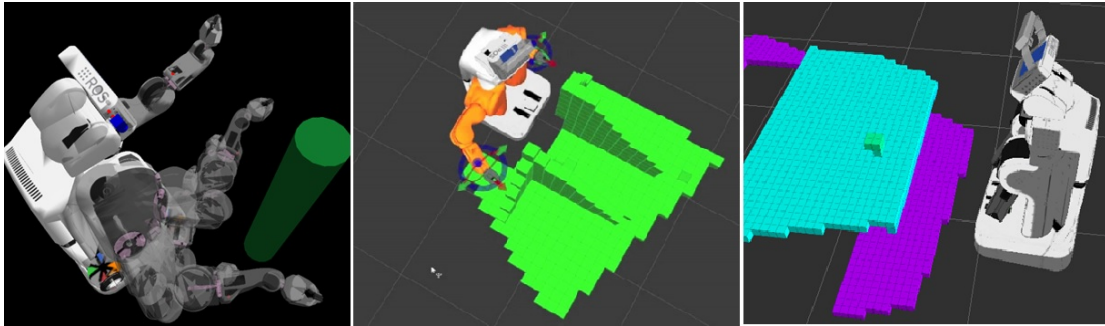


Figure 4.5: The moveit architecture on PR2 robot. (left) The robot plans path avoiding the obstacle, (middle) The robot creates an octomap of the cluttered environment, (right) The robot creates an octomap of the table environment

4.3 MIT-DRAKE

Drake [Ted14] is a toolbox maintained by the Robot Locomotion Group at the MIT Computer Science and Artificial Intelligence Lab (CSAIL). It is a collection of tools for

analyzing the dynamics of our robots and building control systems for them in MATLAB and C++, with a heavy emphasis on optimization-based design/analysis.

Insanity: Doing the same thing over and over again and expecting different results.

Albert Einstein

5

Preliminary Results

Several experiments are been performed for capturing object snapshots and registering them to create 6D pose model. Several object models are also been collected from various open-source databases to expand our local model set.

5.1 GENERATING DATABASE FOR OBJECTS

For the object Recognition part we have acquired several models of daily-life objects by taking snapshots by different viewpoints to create our own database. Most of the objects are simple shape-based object such as band-aid box, coffeecup, cigarette packet, wine glasses, pringle boxes etc. A list of objects models captured are shown in fig. 5.1 The other object



Figure 5.1: Object snapshots captured by Kinect Device

models are borrowed from several open-source databases. A list of number of objects borrowed from other databases is presented in Table 5.1

5.2 RECOGNIZING OBJECTS FROM DATASET

Several experiments have been performed with local and global feature descriptors for recognizing objects in cluttered environment. Some of the algorithms failed completely, while some algorithm shown better performance.

5.2.1 USING LOCAL FEATURE DESCRIPTORS

Experiments are performed for the process of recognizing objects based on local feature descriptors. While with FPFH, the object is recognized for only a partial view, neither the

<i>Name of Database</i>	No. of objects borrowed
<i>Amazon</i>	32
<i>CMU₁KEA</i>	124
<i>Pacman</i>	26
<i>TUM(Technical University Munich)</i>	54
<i>TUW(Technical University Wien)</i>	33
<i>Universita_{CA}Foscari Venezia</i>	86
<i>Willow Garage Dataset</i>	46
<i>Berkley Dataset</i>	132
<i>KIT</i>	55
<i>PCL dataset</i>	45
<i>Washington dataset</i>	86
<i>YCB(Yale – Columbia – Berkley) dataset</i>	74

Table 5.1: Number of object models borrowed from other datasets

algorithm works on scattered environment, nor the algo succeeded in recognizing the object with the change in viewpoint. Recognition process with FPFH is shown in Fig. 5.2

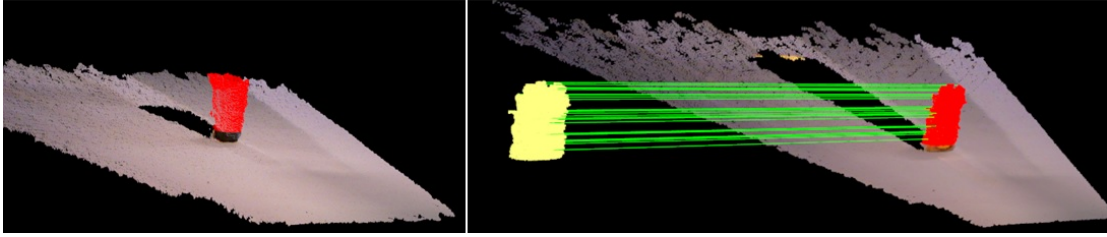


Figure 5.2: Recognition experiment with FPFH descriptors)

5.2.2 USING GLOBAL FEATURE DESCRIPTORS

Using global feature descriptors such as VFH, we obtained promising results, but the training process is time consuming and the object model does not detect proper results if the viewpoint in the test set belongs to a region in-between the stored viewpoints. The system then tries to average the two viewpoints, and the result becomes poorer. Example of recog-

nitition by VFH descriptor is shown in Fig. 5.3 Another promising results are obtained by

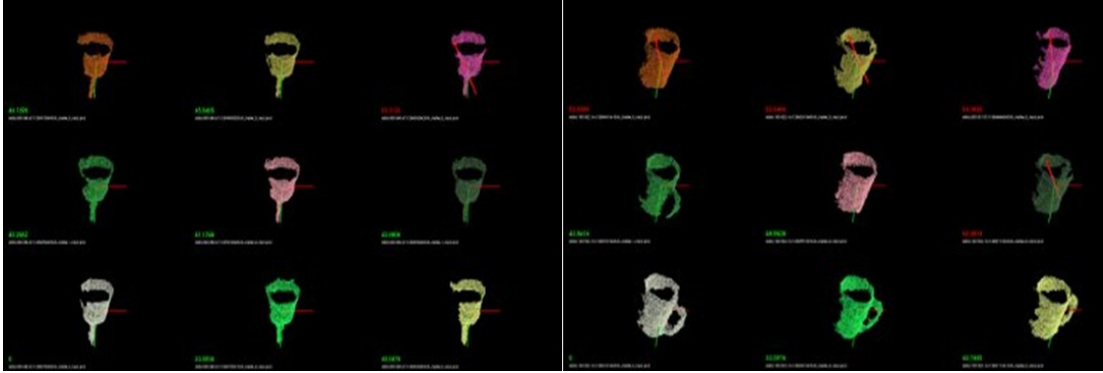


Figure 5.3: Recognition experiment with VFH descriptors)

the utilization of OUR-CVFH. The result is also simplified as the training model generation by the algorithm is quite simple and not so time consuming. Result of using OUR-CVFH has been shown in Fig. 5.4

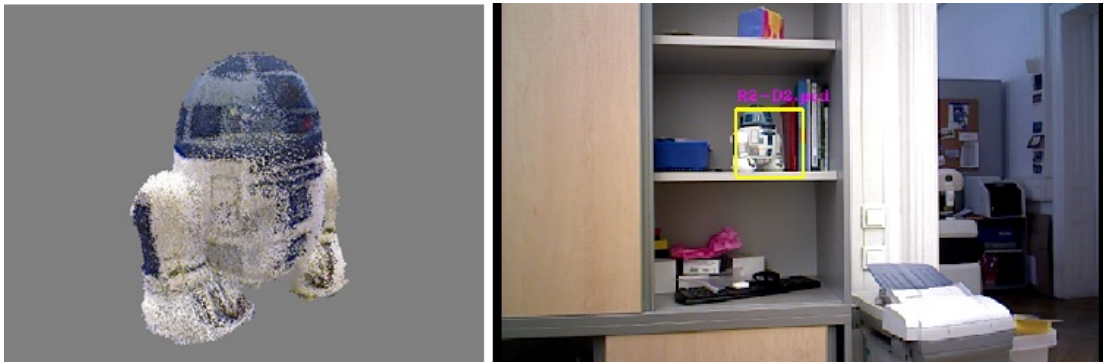


Figure 5.4: Recognition experiment with OUR-CVFH descriptors)

5.2.3 OBJECT RECOGNITION KITCHEN

The Object Recognition Kitchen (ORK) [WG] is a project started at Willow Garage for object recognition. ORK hosts mainly four types of object recognition scheme.

1. LineMOD: The object recognition algorithm works by 2D and 3D and works on rigid and lambertian objects.
2. tabletop: The object recognition algorithm works by 3D and works on rigid, lambertian, rotationally symmetric objects.
3. TOD: The object recognition algorithm works by 2D and 3D and works on rigid, textured and lambertian objects.
4. Transparent Objects: The object recognition algorithm works by 2D and 3D and works on rigid and transparent objects.

Detecting a cup and the tabletop by Object Recognition Kitchen is been shown in fig. 5.5

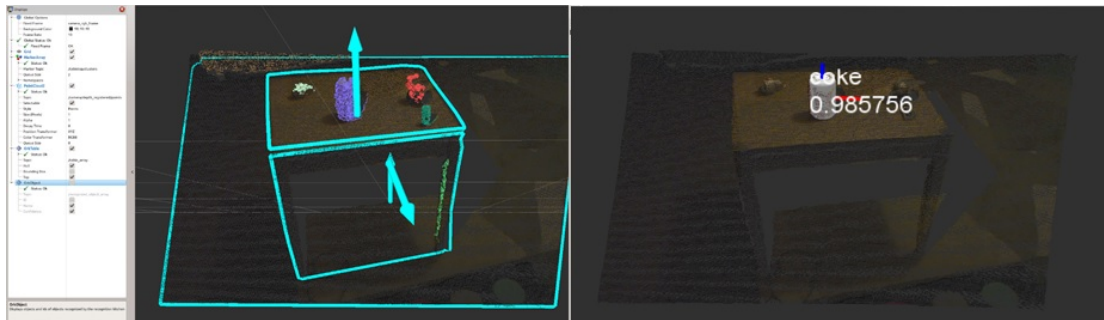


Figure 5.5: Recognition experiment with Object Recognition Kitchen)

5.3 GRASP SYNTHESIS

For synthesis of grasps to find stable and feasible grasps we are utilizing two opensource grasping framework: GraspIt and Openrave

5.3.1 GRASPIT

GraspIT is a tool for grasping research. It features a huge library of hands and objects, an intuitive user interface, visualization of grasp wrench space, quality measures to evaluate grasp, dynamic simulation and grasp planning capabilities. The main components of GraspIT are:

1. World Construction: capability of reading object models, link models, kinematics and assembling robots
2. User Interface: capability of viewing 3D scene, changing hand pose, auto-grip, manually moving joints and interfacing with matlab.
3. Contact determination: Capability of detecting collisions, adjusting contact to object surface, finding contact area and adding friction cones
4. Grasp Analysis: Capability of computing grasp wrench space, using metrics on space and grasp force optimization
5. Wrench Space visualization: Capability of creating projections of Grasp Wrench Space
6. Rigid Body Dynamics: Capability of computing object motions
7. Grasp Planner: Capability of generating and testing grasps
8. Control Algorithms: PD controllers

For including a novel hand in the grasping framework we have to include the hand kinematics in Denavit-Hartenberg parameters which can specify the transforms between links, and can handle coupled joints. We also need the CAD model file for 3D link geometries.

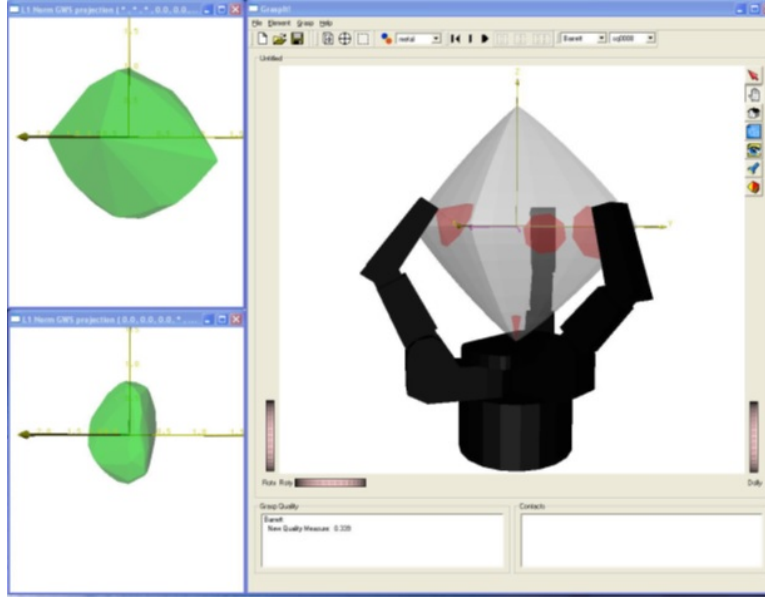


Figure 5.6: Graspit Framework with Barrett Hand)

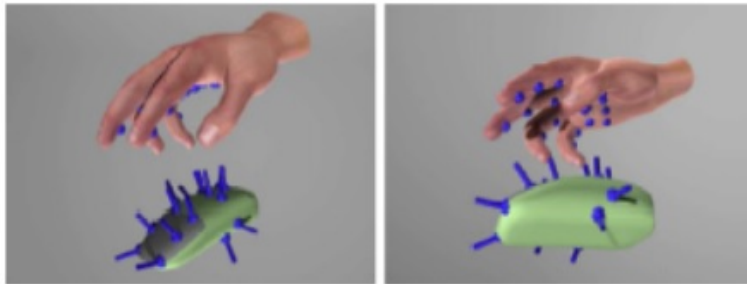


Figure 5.7: Eigen Grasp with GraspIT)

5.3.2 OPENRAVE

The OpenRave framework also been explored for computing grasps as shown in Fig. 5.10.

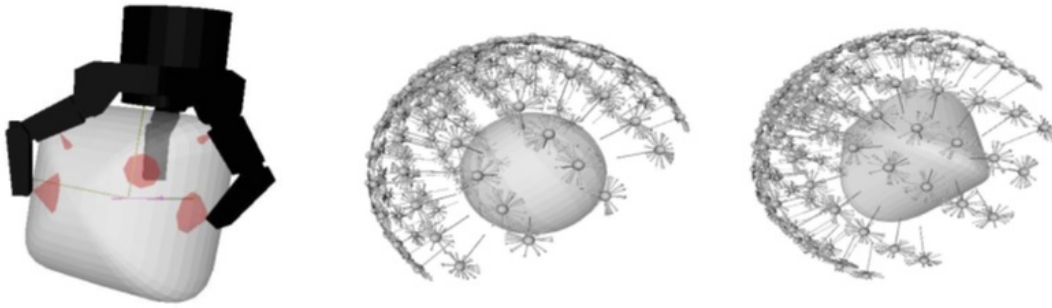


Figure 5.8: Visualizing Grasp Quality with Graspit)

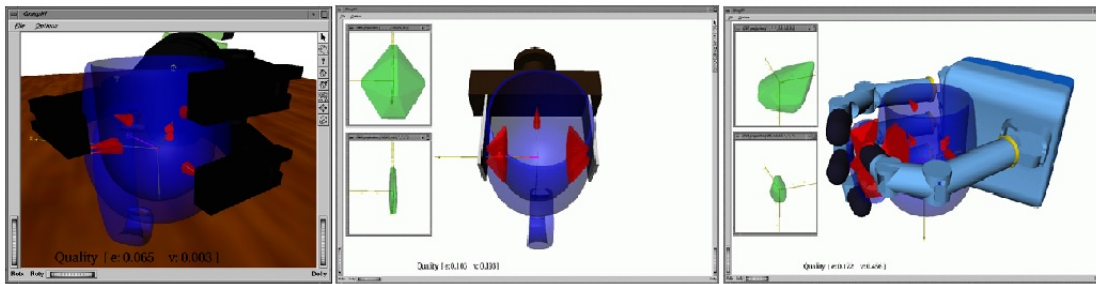


Figure 5.9: Comparison of grasps on a object by different hands: Barrett hand(left), Paraller Gripper(middle), DLR hand(right))

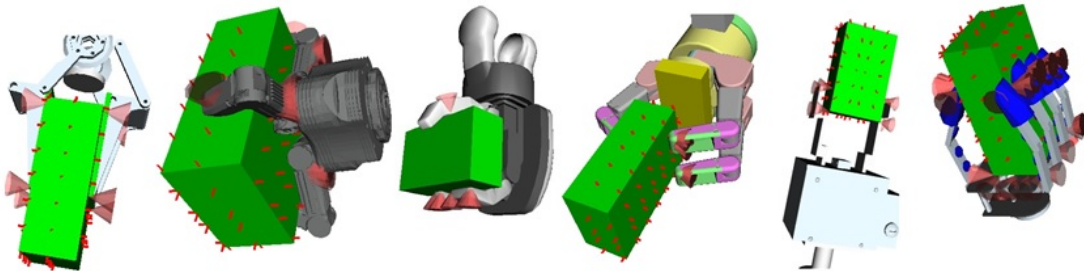


Figure 5.10: Performing grasp with different hands in OpenRave)

5.4 ROBOT HARDWARE

Our robot hardware consists of mainly three components. The robot hand is a Barrett hand. The sensor is a Microsoft Kinect and the arm is an UR5 robotic arm. The compo-

nents are shown in Fig. 5.11



Figure 5.11: Robot hardware components: (left)Microsoft Kinect, (middle) Barrett Hand, (right) UR5 arm

5.5 SIMULATION

We are utilizing Gazebo as our primary simulation software. Gazebo [KH04] offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. It is an well-designed simulator where it is possible to rapidly test algorithms, design robots, and perform regression testing using realistic scenarios. To represent a robot in Gazebo, we have to create an URDF of the robot which accumulates all the exact dimensions of the links and joints of the real robot.

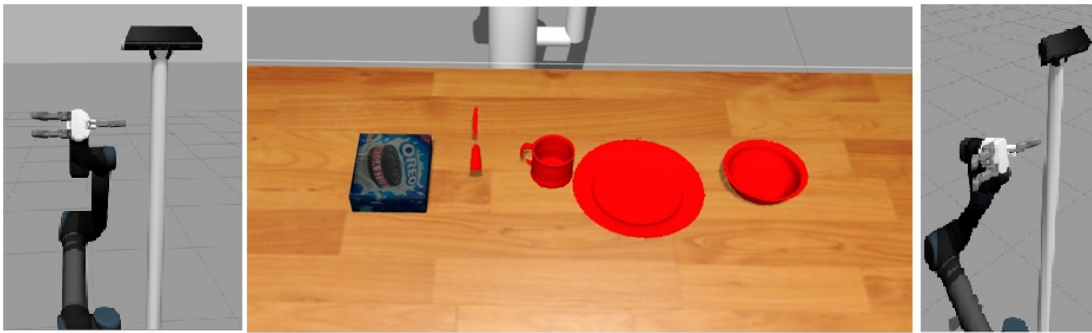


Figure 5.12: (left/right) The robotic module in simulation, (middle) The table and various objects imported in Gazebo

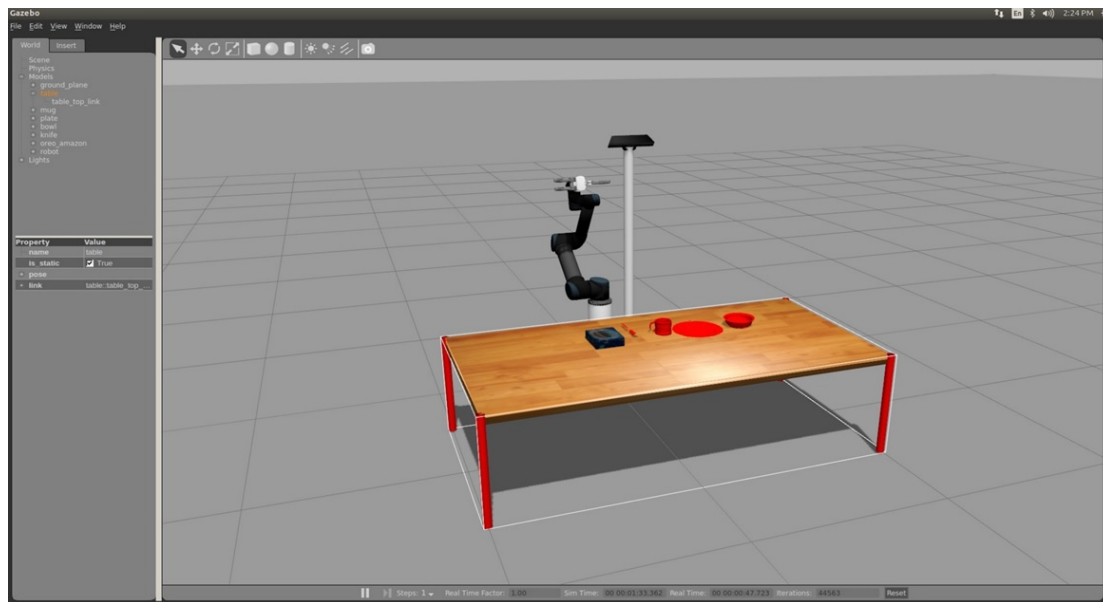


Figure 5.13: Our robotic module in Gazebo

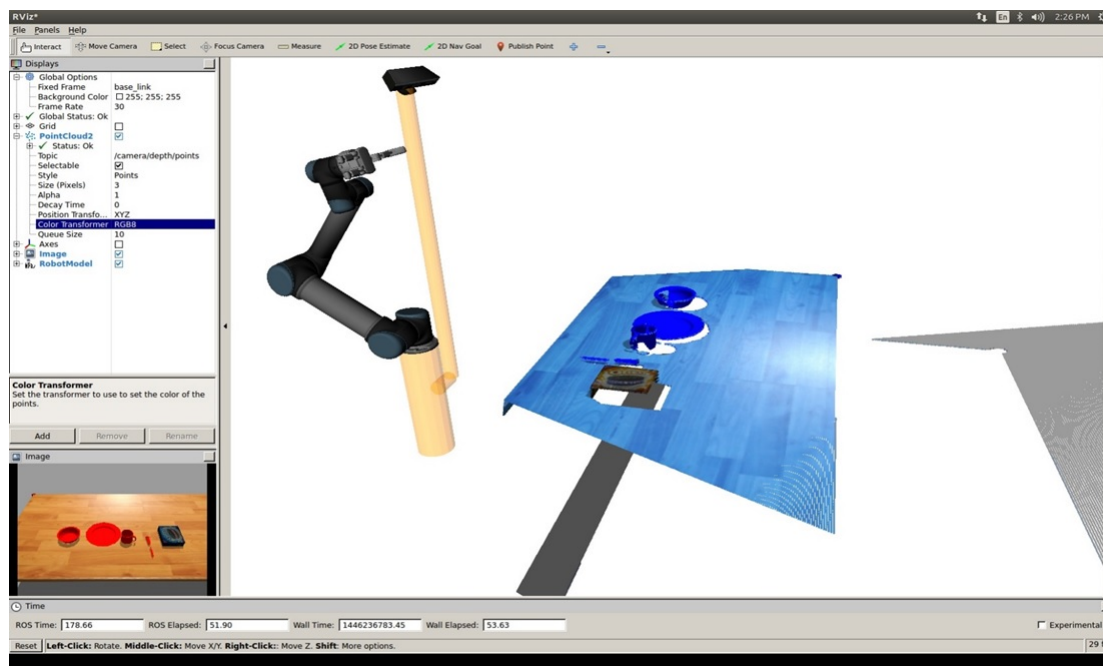


Figure 5.14: Simulated robot in RViz. The sensor streaming is also presented in robot's viewpoint

We have created a robotic module of our robot hardware in Gazebo by the means of URDF. The simulated robot matches to the real robot hardware in terms of visualization and dimensions. Also the simulated controllers mimics the controller of the real robotic arm, hand and sensor devices. The simulated robot is presented in Fig. 5.13. Some objects from our object database are imported in Gazebo simulation environment. The objects in gazebo are shown in Fig. 5.12. RVIZ is tool belonging to ROS-framework for visualization of real time robot and sensor streaming. Our robot is visualized in RViz in fig. 5.14

*Research is to see what everybody else has seen, and to
think what nobody else has thought*

Albert Szent-Gyorgyi

6

Proposed Research

The goal of the research is to create an environment in which grasping capabilities of different robotic hands can be tested and assessed. In order to do so, the theory and algorithm for a robotic system that can grasp objects has to be developed. For the task, the robotic system have to compare the commercially available robotic hands with the self-designed robotic hands in the context of grasping capabilities. On the basis of finger placement in the hands, the system need to calculate grasping points on the objects and manipulate the objects based on the in-hand-task. For manipulating the object, the robotic system needs to find the designated object in a cluttered environment, find stable and robust grasping points and plan motions for picking up the object. The main focus of the research is to

evaluate and compare feasible grasping points in the context of slippage conditions and stability for both hands on the objects in the dataset in identical environment.

6.1 RECOGNIZING OBJECT

For a successful grasping framework it is needed for the system to robustly detect and recognize objects in the cluttered environment. For detecting the particular object, the object's characteristics should be stored in a database which can be considered as features. Comparing the stored features with the segmented test environment, we can detect the object with 6D pose. Various 3D object recognition pipeline will be evaluated and the most robust algorithm will be chosen for the purpose.

6.1.1 TASKS

- Creating a huge database of household objects
- Calculating 3D features of the objects
- Recognizing objects based on the 3D features
- Identify 6D pose information of recognized object

6.1.2 METHODS

- Constructing object models by rotating table and registering snapshots towards a full model
- Borrow objects from various open-source database

- Implementation of Point cloud library for detecting 3D feature descriptors of the objects
- Recognize objects in cluttered environment with the utilization of point cloud library features (global)
- Implementation of Object Recognition Kitchen(ORK) library for recognition of objects
- Implementation of OUR-CVFH (Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram) for recognizing objects
- Implementation of CRH(Camera Roll Histogram) for detecting 6D pose of the object

6.2 GRASPING POINT CALCULATION

After recognizing or detecting the object in the environment, the robotic system will try to find stable and robust grasping points to grasp the object. To be stable, the grasp have to be force closure. Several methods have been implemented for grasping objects with parallel plate grippers, where a rectangle pose on the object have been evaluated. As our system accumulates the barrett hand as the main grasping hand, calculating the grasping points faces severe complexity. Later other self-designed hands will be evaluated based on the algorithm generated by the research.

Calculating grasping point is highly complex as the work can be formulated as a search space in lower dimension. So the process of determining the stable grasp from a set of grasps adds extra complexity to the task. The searching of stable grasp in a set of grasp can

be solved by the implication of learning techniques such as HMM, Bayesian Network, Convolution Neural Network or Deep Learning Methods. The machine learning approaches will be implemented and evaluated based on grasp stability and time complexity.

6.2.1 TASKS

- Calculating Grasping points from the recognized feature descriptors of the objects.
- Defining stability conditions based on the object and the hand
- Provide force-closure grasps on the object
- Analyze the grasp based on Task Wrench Space and Grasp Wrench Space

6.2.2 METHODS

- Exploration of grasping framework such as GraspIT, OpenGrasp and OpenRAVE
- Calculating grasping points based on the suitability of the grasping framework
- importing self-designed hands in the grasping framework
- Implementation of Learning algorithms such as HMM, Bayesian Network, Convolutional Neural Network or Deep learning methods for calculating feasible grasping points
- Enable the system to learn grasp for similar type of objects

6.3 PLANNING MOTION

After the objects been recognized and the grasping points are evaluated, the robotic system should able to plan motions for grasp the object, pick it up, and place according to the task in hand. Several motion planning algorithms are present for planning motions for high-DOF robotic arms. The motion planning algorithms will be evaluated and the best solution would be implemented in the robotic system.

6.3.1 TASKS

- Plan collision-aware and constraint-aware inverse kinematic solutions for robotic arm to reach towards grasping points
- Explore various motion planning algorithms for planning motions
- Define pick-and-place tasks for various objects with constraints

6.3.2 METHODS

- Explore MoveIT framework for planning motions
- Explore OpenRave framework for planning motions
- Explore MIT-Drake framework for planning motions
- Explore sampling based motion planners such as OMPL(Open Motion Planning Library), SPBL (Search Based Motion Planners), CHOMP(Co-variant Hamilton Optimization for motion planning)

- Explore RRT-based (Rapidly Exploring Random Trees) motion planners for planning motions.

6.4 DESIGN OF ROBOTIC HANDS

New robotic hands will be designed based on specified task. The design task consists of capturing motions, converting motions into solvable tasks, defining hand parameters for design and designing hand assembling new hands.

6.4.1 TASK

- Design and assemble new task-specific robotic hands

6.4.2 METHOD

- Capturing motion data
- Synthesizing motion data into task specific solvability points.
- Explore hand parameters by the solvability and hand topology by Artreeks
- Design hands CAD models based on Artreeks solutions

6.5 THE ROBOT IN HARDWARE AND SOFTWARE

The grasping framework as a combination of capabilities of recognizing objects, calculating grasping points, planning and executing motions are implemented in both hardware and simulation. As the simulation system is defined by the identical programs and measurements of the real robot, it is beneficial to evaluate the system in simulation before implementing in real hardware.

6.5.1 TASKS

- Construct the robotic module
- Import the robot in simulation
- Import the objects in database towards simulation
- Set up the environment for pick and place task

6.5.2 METHODS

- Assembling all the robotic components such as Kinect, Barrett Hand and UR5 arm to construct the robotic module
- Create a coupler for connecting the UR5 link with the barrett hand link
- Construction of an URDF(unified robot description format) of the robot
- Importing the URDF in Gazebo simulation
- Create the pick and place environment in Gazebo simulation
- Develop ROS packages for the robot to manipulate the robot in simulation and real time

6.6 EVALUATION AND ASSESSMENT OF HANDS

When the framework and algorithms provide sufficient results, the self-designed robotic hands would be tested in the place of barrett hand. The evaluation would be based on task-specification, stability, robustness and time complexity.

6.6.1 TASKS

- Reconstruction of the robot with the self-designed hands
- Evaluate and compare results with the result obtained by barrett hand

6.6.2 METHODS

- Removal of barrett hand and assemble of self-design hand
- Construction of new urdf of the hand to be imported in simulation

6.7 GOAL OF THE RESEARCH

Summarizing the main goal of the research:

1. An object recognition framework that utilizes 3D point cloud for feature extraction will be implemented among other techniques based on robustness and time-complexity.
2. A grasping framework that can calculate stable and robust grasps on the recognized object utilizing learning techniques.
3. A motion planning framework that can plan collision aware and constraint-aware trajectories for the robot to reach grasping pose and manipulate the object.
4. Design of task-specific robotic hands
5. Utilization of the grasping framework for comparing the commercially available robotic hands with self-designed robotic hands.

6.8 TIME ALLOTMENT

<i>Tasks</i>	Duration(in months)
<i>Explore methods for Object Recognition</i>	3
<i>Design of new robotic hands (in parallel)</i>	6
<i>Explore Grasp point selection algorithms</i>	3
<i>Explore Learning Techniques</i>	3
<i>Explore Motion Planning Techniques</i>	3
<i>Evaluate Grasping of different hands</i>	3
<i>Total</i>	15

7

Conclusion

A general grasping framework has been proposed where the robotic system would recognize the objects based on RGBD features, calculate robust and stable grasping points based on the recognized features by statistical learning methods, executes the task of grasping based on the calculated grasping points. At the initial stage, the task would be performed commercially available robotic hands, which will be replaced by self-made robotic modules for evaluation.



Publications

1. Abhijit Makhal, Alba-Perez-Gracia “Task Data Association for Robot Design Problem” Submitted in International Conference on Robotics and Automation.
2. Abhijit Makhal, Alba-Perez-Gracia “Solvable Multi-fingered Hands for Exact Kinematic Synthesis” Advances in Robot Kinematics, 2014, pp 139-147
3. Abhijit Makhal, Manish Raj, Karan Singh, Rajan Singh, Pavan Chakraborty and G C Nandi. Article: ATOM- A Low-Cost Mobile Manipulation Platform for Research and Testing. International Journal of Applied Information Systems 4(10):1-7, December 2012. Published by Foundation of Computer Science, New York, USA.

4. A.Makhal, M.Raj, K.Singh, P.Chakraborty and G.C.Nandi “Path Planning through Maze Routing for a Mobile Robot with Nonholonomic Constraints” (Accepted in The 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2012), Korea)
5. A.Makhal, A.Sar, S.K.Samanta, “Development of a Mobile Robot for Image Acquisition through Teleoperation” in Proceedings of 2nd International Conference on Computational Vision and Robotics (ICCV ‘11), pp. 74-77, Bhubaneswar, INDIA, August 2011
6. A.Sar, A.Makhal, S.K.Samanta, “Formation of an Objective Function using Vector Quantization Technique from Multi-variable Regression” in Proceedings of 1st International Conference on Soft-Computing and Engineering Applications (SEA‘11), pp. 18-21, Kolkata, INDIA, September, 2011
7. S.Das, A.Makhal, “A logical Topology to find Alternate Routes in WDM lightwave Network” Published by Springer Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST) Series , volume 0084, January 2-4,2012.
8. Saumitra Krishna, Sachin Kansal, Abhijit Makhal, Pavan Chakraborty and G.C. Nandi “Systematic study of binocular depth finding using two web cameras” accepted in 3rd International Conference on Computer and Communication Technology (ICCCT-2012) proceedings to be published by the IEEE.
9. Sachin Kansal, Abhijit Makhal, Dr.Pavan Chakraborty, Prof.G.C.Nandi “Tracking of a Target Person Using Face Recognition by Surveillance Robot” (Under review)

References

- [AAB11] D. Gossow S. Gedikli R. Rusu M. Vincze A. Aldoma, N. Blodow and G. Bradski. Cad-model recognition and 6 dof pose estimation using 3d cues. *Proc. ICCV 2011, 3D Representation and Recognition*, pages 585–592, 2011.
- [AB00] V. Kumar. A. Bicchi. Robotics grasping and contact: A review. *Proc. IEEE Int. Conf. Robotics and Automation*, 2000.
- [AB08] R. Dillmann S. Schaal A. Billard, S. Calinon. Survey: Robot programming by demonstration. *Handbook of Robotics*, 2008.
- [AS97] C. G. Atkeson and S. Schaal. Learning tasks from a single demonstration. *Proc. IEEE International Conference on Robotics and Automation*, pages 1706–1712, 1997.
- [ASNo8] J. Driemeyer A. Saxena and A. Ng. Robotic grasping of novel objects using vision. *International Journal of Robotic Research*, 27(2):157, 2008.
- [ASAS] P. Bidaud A. Sahbani, S. El-Khoury. An overview of 3d grasp synthesis algorithms. *Proc of Robotics and Autonomous Systems*, 2011, publisher=RAS.
- [ATM99] P. K. Allen A. T. Miller. Examples of 3d grasp quality computations. *Proc. in IEEE International Conference on Robotics and Automation*, pages 1240–1246, 1999.
- [ATRV12] Aitor Aldoma, Federico Tombari, RaduBogdan Rusu, and Markus Vincze. Our-cvfh – oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation. *Pattern Recognition*, pages 113–122, 2012.
- [Bel57] R. Bellman. Dynamic programming. *Princeton, NJ*, 1957.
- [Ber11] Dmitry Berenson. Constrained manipulation planning. *PhD Thesis*, 2011.

- [BM94] J. Canny B. Mirtich. Easily computable optimum grasps in 2d and 3d. *Proc. in IEEE International Conference on Robotics and Automation*, 1:739–747, 1994.
- [CB03] G. Hirzinger C. Borst, M. Fischer. Grasping the dice by dicing the grasp. *Proc. in IEEE International Conference on Robotics and Automation*, 3:3692–3697, 2003.
- [CBX95] F. Bernardini C. Bajaj and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3d scans. *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 109–118, 1995.
- [Chi98] S. Chiu. Task compatibility of manipulator postures. *International Journal of Robotics Research*, 7(5):13–21, 1998.
- [DBCo4] C. G. Atkeson D. Bentivegna and G. Cheng. Learning tasks from observation and practice. *Robotics and Autonomous Systems*, 47:163–179, 2004.
- [DD00] S. Wang D. Ding, Y. H. Liu. Computing 3d optimal form closure grasps. *Proc. in IEEE International Conference on Robotics and Automation*, 4:3573–3578, 2000.
- [DD01] S. Wang D. Ding, Y. H. Liu. On computing immobilizing grasps of 3-d curved objects. *Proc. in IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 11–16, 2001.
- [Dia10] Rosen Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.
- [EAG93] K. Warwick E. Al-Gallaf, A. Allen. A survey of multi-fingered robotic hands: Issues and grasping achievements. 1993.
- [EO02] M. A. Arbib E. Oztop. Schema design and implementation of the grasp related mirror neuron system. *Biological Cybernetics*, 87(2):116–140, 2002.
- [ESS05] H. Maas E. Schwalbe and F. Seidel. 3d building model generation from airborne laser scanner data using 2d gis data and orthogonal point cloud projections. *Proceedings of the International Society for Photogrammetry and Remote Sensing*, 3:12–14, 2005.

- [FHK⁺04] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bulow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. May 2004.
- [FK05] S. Saito M. Nakajima F. Kyota, T. Watabe. Detection and evaluation of grasping positions for autonomous agents. *Proc. in International Conference on Cyberworlds*, pages 453–460, 2005.
- [FN71] R. Fikes and N. Nilsson. Strips: A new approach to the application of theorem proving. *Artificial Intelligence Journal*, 2:189–208, 1971.
- [FT11] Luigi Di Stefano Federico Tombari, Samuele Salti. A combined texture -shape descriptor for enhanced 3d feature matching. *Proc. 18th IEEE International Conference on Image Processing*, 2011.
- [FTS10] S. Salti F. Tombari and L. Di Stefano. Unique signatures of histograms for local surface description. *Proc. 11th European Conf. Computer Vision*, pages 356–369, 2010.
- [GSB⁺11] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Rotational projection statistics for 3d local surface description and object recognition. *International Journal of Computer Vision*, 105(1):63–86, 2011.
- [HWB⁺13] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. Software available at <http://octomap.github.com>.
- [ILS13] H. Lee I. Lenz and A. Saxena. Deep learning for detecting robotic grasps. *ICLR*, 2013.
- [JBG85] S. Dubowsky J. Bobrow and J. Gibson. Time-optimal control of robotic manipulators along specified paths. *International Journal of Robotics Research*, 4:3–17, 1985.
- [JCCE01] J. B. Cherrie T. J. Mitchell W. R. Fright B. C. McCallum J. C. Carr, R. K. Beatson and T. R. Evans. Reconstruction and representation of 3d objects with radial basis

- functions. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 67–76, 2001.
- [JH99] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(5):433–449, May 1999.
- [Joh97a] A. Johnson. Spin-images: A representation for 3d surface matching. *PhD Thesis*, 1997.
- [Joh97b] Andrew Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.
- [JP93] J. D. Boissonnat J. P. Merlet J. Ponce, S. Sullivan. On characterizing and computing three and four finger force closure grasps of polyhedral objects. *Proc of IEEE International Conference on Robotics and Automation*, 2:821–827, 1993.
- [JR08] D. Kragic J. Romero, H. Kjellstrom. Human-to-robot mapping of grasps. *Proc. in IEEE International Conference on Robotics and Automation*, pages 9–15, 2008.
- [JWS04] G. Gruener J. Weingarten and R. Siegwart. A state-of-the-art 3d sensor for robot navigation. *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2160, 2004.
- [KH04] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154 vol.3, Sept 2004.
- [KMP⁺11] Asako Kanezaki, Zoltan-Csaba Marton, Dejan Pangercic, Tatsuya Harada, Yasuo Kuniyoshi, and Michael Beetz. Voxelized Shape and Color Histograms for RGB-D. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception and Object Search in the Real World*, San Francisco, CA, USA, September, 25–30 2011.

- [LEKO96] J. C. Latombe L. E. Kavraki, P. Svestka and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):586–580, 1996.
- [Liu99] Y. H. Liu. Qualitative test and force optimization of 3-d frictional form closure grasps using linear programming. *IEEE Transactions on Robotics and Automation*, 1(15), 1999.
- [LK00] S. LaValle and J. Kuffner. Rapidly-exploring random trees: Progress and prospects. in *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2000.
- [LSP05] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using local affine regions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1265–1278, August 2005.
- [MA12] M. Perez Makhal A. Solvable multi-fingered hands for exact kinematic synthesis. In *Proceedings of Advances of Robot Kinematics*, pages 139–147, 2012.
- [Mar10] D.; Blodow N.; Kleinhellefort J.; Beetz M. Marton, Zoltan-Csaba; Pangercic. General 3d modelling of novel objects from a single view. *Proc IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3700–3705, 2010.
- [MGT04] D. Nau M. Ghallab and P. Traverso. Automated planning: Theory and practice. *San Francisco, CA*; 2004.
- [MHo6] J. Zhang M. Hueser, T. Baier. Learning of demonstrated grasping skills by stereoscopic tracking of human hand configuration. *Proc. in IEEE International Conference on Robotics and Automation*, pages 2795–2800, 2006.
- [MHo8] J. Zhang M. Hueser. visual and contact-free imitation learning of demonstrated grasping skills with adaptive environment modelling. *Proc. in IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 17–24, 2008.
- [MHV08] M. Gienger C. Goerick M. Howard, S. Klanke and S. Vijayakumar. Learning potential-based policies from constrained motion. *IEEE-RAS International Conference on Humanoid Robots*, 2008.

- [MPBB11a] Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow, and Michael Beetz. Combined 2d-3d categorization and classification for multimodal perception systems. *Int. J. Rob. Res.*, 30(11):1378–1402, September 2011.
- [MPBB11b] Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow, and Michael Beetz. Combined 2d-3d categorization and classification for multimodal perception systems. *Int. J. Rob. Res.*, 30(11):1378–1402, September 2011.
- [MPR⁺10] Zoltan-Csaba Marton, D. Pangercic, R.B. Rusu, A. Holzbach, and M. Beetz. Hierarchical object geometric categorization and appearance classification for mobile manipulation. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 365–370, Dec 2010.
- [MR31] RD Howe MR.Cutkosky. Human grasp choice and robotic grasp analysis. *Dexterous Robot Hands*, (1), 5–31.
- [MS08] M. Zillich B. Schiele M. Stark, P. Lies. Functional object class detection based on learned affordance cues. *Proc. in International Conference on Computer Vision Systems*, pages 435–444, 2008.
- [MSS98] Martha Flanders Marco Santello and John F. Soechting. Patterns of hand motion during grasping and the influence of sensory guidance. *Journal of Neuroscience*, 23(18), 1998.
- [Nil80] N. Nilsson. Principles of artificial intelligence. *Wellsboro, PA*, 1980.
- [PAH03] A. Troccoli B. Smith M. Leordeanu P. Allen, I. Stamos and Y. Hsu. 3d modeling of historic sites using range and image data. *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, 1:145–150, 2003.
- [Pol97] N. S. Pollard. Parallel algorithms for synthesis of whole hand grasps. *Proc. in IEEE International Conference on Robotics and Automation*, 1:373–378, 1997.
- [RBRBo8] Z. C. Marton R. B. Rusu, N. Blodow and M. Beetz. Aligning point cloud views using persistent feature histograms. *Proc 21st IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3384–4491, 2008.

- [RBRBo9] N. Blodow R. B. Rusu and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pages 1848–1853, 2009.
- [RBRH10] R. Thibaux R. B. Rusu, G. Bradski and J. Hsu. Fast 3-d recognition and pose using the viewpoint feature histogram. *Proc. 23rd IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 2155–2162, 2010.
- [RC11] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 1–4, 2011.
- [RPO4] P. Allen T. Jebara R. Pelossof, A. Miller. An svm learning approach to robotic grasping. *Proc. in IEEE International Conference on Robotics and Automation*, 4:3512–3518, 2004.
- [Rus09] A.; Beetz M.; Bradski G Rusu, R.B.; Holzbach. Detecting and segmenting objects for mobile manipulation. *Computer Vision Workshops*, pages 47–54, 2009.
- [So4] D. Kragic S, Ekvall. Interactive grasp learning based on human demonstration. *Proc. in IEEE International Conference on Robotics and Automation*, 4:3519–3524, 2004.
- [Saxo6] A. Saxena. Robotic grasping of novel objects. *NIPS*, 2006.
- [Saxo9] A. Saxena. Monocular depth perception and robotic grasping of novel objects. *PhD thesis*, 2009.
- [SBPO2] J. Malik S. Belongie and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Machine Intell.*, 24(4):509–522, 2002.
- [SC] Ioan A. Sucan and Sachin Chitta.
- [SEK09] A. Sahbani. S. El-Khoury. On computing robust n-finger force closure grasps of 3d objects. *Proc. in IEEE International Conference on Robotics and Automation*, pages 2480–2486, 2009.
- [SK87] S. Sentis and O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Transactions on Robotics and Automation*, 3(1):43–53, 1987.

- [SK05] S. Sentis and O. Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2:505–518, 2005.
- [SMK12] Ioan A. Sucan, Mark Moll, and Lydia E. Kavraki. The open motion planning library. *IEEE Robotics and Automation magazine*, 19(4), 2012.
- [SNo2] T. Sugihara and Y. Nakamura. Whole-body cooperative balancing of humanoid robot using cog jacobian. *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.
- [Ted14] Russ Tedrake. Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems, 2014.
- [TSDS10a] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique shape context for 3d data description. In *Proceedings of the ACM Workshop on 3D Object Retrieval*, 3DOR '10, pages 57–62, New York, NY, USA, 2010. ACM.
- [TSDS10b] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. *Lecture Notes in Computer Science*, 6313:356–369, 2010.
- [WG] ROS community Willow Garage.
- [WV11] W. Wohlkinger and M. Vincze. Ensemble of shape functions for 3- d object classification. *Proc. IEEE Int. Conf. Robotics and Biomimetics*, pages 2987–2992, 2011.
- [YJS11] S. Moseson Y. Jiang and A. Saxena. Efficient grasping from rgb-d images: Learning using a new rectangle representation. *Proc. in IEEE International Conference on Robotics and Automation*, pages 3304–3310, 2011.
- [YLo7] N. Pollard. Y. Li, J. L. Fu. Data driven grasp synthesis using shape matching and task based pruning. *IEEE Transactions on Visualization and Computer Graphics*, 4(13):732–747, 2007.
- [ZL98] S.S. Sastry Z. Li. Task oriented optimal grasping by multi-fingered robot hands. *International Journal of Robotics Research*, 4(1), 1998.