3D Object Recognition and Relative Localization using a 3D sensor Embedded on a Mobile Robot

Gopi Krishna Erabati

Laboratoire d'Analyse et d'Architecture des Systèmes
(LAAS - CNRS) Toulouse, France

Under the supervision of
Dr. Frédéric Lerasle and
Dr. Jorge Francisco Madrigal Diaz

A Thesis Submitted for the Degree of
MSc in Computer Vision (MSCV)
Université de Bourgogne, France

· 2018 ·

Abstract

The technology in current research scenario is marching towards automation for higher productivity with accurate and precise product development. Vision and Robotics are domains which work to create autonomous systems and are the key technology in quest for mass productivity. The automation in an industry can be achieved by detecting interactive objects and estimating the pose to manipulate them. Therefore the object localization (i.e., pose) includes position and orientation of object, has profound significance. The application of object pose estimation varies from industry automation to entertainment industry and from health care to surveillance. The objective of pose estimation of objects is very significant in many cases, like in order for the robots to manipulate the objects, for accurate rendering of Augmented Reality (AR) among others.

This thesis tries to solve the issue of object pose estimation using 3D data of scene acquired from 3D sensors (e.g. Kinect, Orbec Astra Pro among others). The 3D data has an advantage of independence from object texture and invariance to illumination. The proposal is divided into two phases : An offline phase where the 3D model template of the object (for estimation of pose) is built using Iterative Closest Point (ICP) algorithm. And an online phase where the pose of the object is estimated by aligning the scene to the model using ICP, provided with an initial alignment using 3D descriptors (like Fast Point Feature Transform (FPFH)).The approach we develop is to be integrated on two different platforms : 1)Humanoid robot 'Pyrene' which has Orbec Astra Pro 3D sensor for data acquisition, and 2)Unmanned Aerial Vehicle (UAV) which has Intel Realsense Euclid on it. The datasets of objects (like electric drill, brick, a small cylinder, cake box) are acquired using Microsoft Kinect, Orbec Astra Pro and Intel RealSense Euclid sensors to test the performance of this technique. The objects which are used to test this approach are the ones which are used by robot. This technique is tested in two scenarios, firstly, when the object is on the table and secondly when the object is held in hand by a person. The range of objects from the sensor is 0.6 to 1.6m. This technique could handle occlusions of the object by hand (when we hold the object), as ICP can work even if partial object is visible in the scene.

Contents

List of Figures

List of Tables

Acknowledgments

No creation in this world is solo effort. Neither is this project. There are some people in this world, some of them so wonderful, that made this work become a thesis that you are holding in your hand. I would like to thank all of them and in particular:

I would like to express my sincere gratitude to my supervisor Dr. Frédéric Lerasle for his constant guidance, motivation and support throughout the course of this work, which helped me to make successful contributions in solving the proposed challenges.

I am deeply indebted to Dr. Fransisco Madrigal who constantly helped me to resolve the issues I face during the work. He always encouraged me to try new approaches in solving the problems. He helped me a lot when I had questions and troubles in the internship and gave me many helpful suggestions to make sure I am in right direction.

I am thankful to Dr. Olivier Stasse for providing me the 'Pyrene' humanoid robot and the 3D sensor to make acquisition of the datasets. I am thankful to Dr. Antonio Franchi and Dr. Anthony Mallet for lending me the Intel Euclid sensor for acquiring datasets and encouraging in the work.

I would like to thank these amazing people for taking care of me during this

work and provided me freedom to choose implementation strategies and planning of work. I am very delighted to work at LAAS-CNRS with some wonderful colleagues, for the fun-work environment to relieve the stress and stay active and motivated on the project.

I would like to express my sincere thanks to all my professors at University of Burgundy, for shaping me with the coursework and projects, to be ready for this research work. I would like to thank amazing friends who always stayed with me in the process of knowledge exchange and cultural exchange.

I would like thank my parents and family for their constant support and encouragement without which I would not be what I am today. Last but not least, I thank God for all His blessings and who continues to look after us despite our flaws!

Chapter 1

Introduction

Challenges are what make life interesting and overcoming them is what makes life meaningful

-Joshua J Marine

1.1 Background

The ultimate goal of technological advancement is to improve the quality of life of humans. From invention of wheel to more advanced technology now-a-days strive for the sophistication of human generations. In this context of technological advancement, Robotics played profoundly significant role improving sustainability and life quality [14].

Although not in the way what we know them today but robots (autonomous systems) are in the mind of mankind from long time in terms of autonomous artifacts created by humans. The word 'robot' and 'robotics' is quite recent. "Robot" comes from a Czech word 'robota' which means 'forced labour'. The term was first used in K. Capek's play R.U.R (Rossum's Universal Robots) in 1920 [59]. The advancement of technology over the last half-century allowed to start thinking about design and development of autonomous artifacts. This paved a way to develop robotics as a scientific field.

As technology advance, the robots become more autonomous and capable to

do several human activities. The characteristics of robots, i.e., precision, intelligence and energy levels makes them perfect for certain jobs that humans can't perform.

The development of automation in industries has seen a great change with the help of robotics since it paved the way for industrial revolution [20]. We are now able to build fully autonomous factories, but the robotics outside industry poses several problems that are arduous to solve. Robotics has its greater impact with its application ranging in the fields from military [55], factory [24], farmers [44], medicine [52] etc.

The eye [45] is the most important organ of human body. With the advent of computer vision, the robotic community has got its eyes. The vision based systems embedded to robots has helped the robotics community to go beyond the limitations. The computer vision systems helps robots to see and analyze the images to make decisions. The development of CMOS image sensors [12] played a crucial role in the development of computer vision systems. With the help of stereo vision and other technologies like time of flight, we are now able to capture the depth information of the scene which helps robots to detect objects in the 3D world.

A human can not only see and detect objects with his intelligence but also he can know the object's position and orientation with the help of neural processing in his brain. So, when robots want to interact with the world, they also need to recognize and localize objects in order to handle them. So, we have to develop technology which helps the robots interact with the world like a human does.

1.2 Statement and Motivation

This thesis deals with the problem of 3D object recognition and relative pose estimation using depth information for object-robot interaction. The combination of *position* and *orientation* is referred to as the pose of the object. This work on pose estimation is to be integrated on two different platforms: 1) Robot 'Pyrene', a humanoid robot for picking the objects like drill, brick (typically, which are laid on pavements), small cylindrical object and 2) Unmanned Aerial Vehicle (UAV) for picking the objects like brick (in shape of cuboid). This task brings challenges

such as, noise in depth information, occlusion of objects in the scene, message delay between the robot and a remote system due to communication lag among others.

Object recognition and pose estimation have many applications ranging from industrial automation, augmented reality, entertainment industry, health care and surveillance. The pose estimation can be used, for example, to manipulate an object or to move away from the object, to pick and place an object in a warehouse of an industry among others.

As a part of automation in industries, we use robots to serve in many tasks which deals with various objects in its environment. In an industry the arm type robots are common, which are used to manipulate the objects. So, the robot should be capable of recognizing the objects and estimating the pose of them with respect to its location in such a way that it can do a particular task such as pick and place, parts assembly, amongst others. We can see a manipulator stacking stones in Figure 1.1.

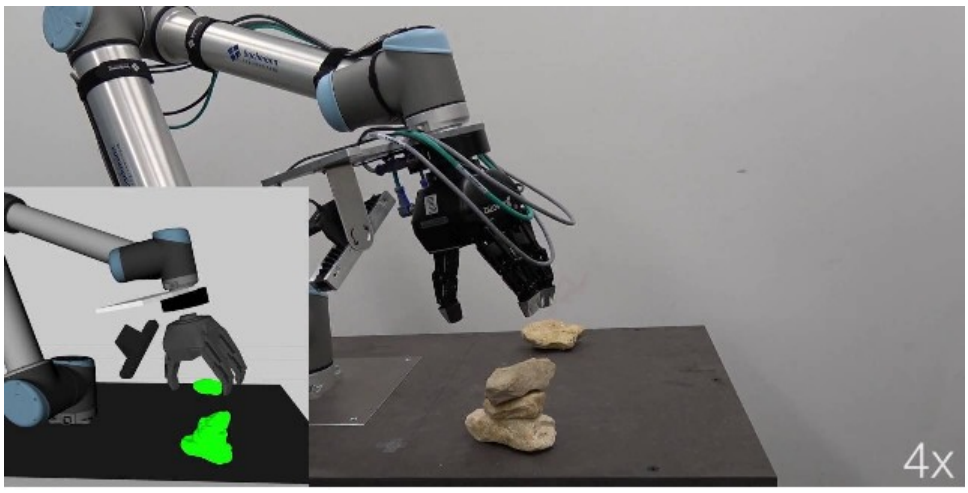


Figure 1.1: A robot manipulator stacking stones with pose planning.
Source : ETH Zurich Autonomous Systems Lab [22]

Pose estimation plays a crucial role in medical applications, like in robot assisted surgeries [17]. The laparoscopic experimental setup for robot assisted surgery is shown in Figure 1.2. Also, the human pose estimation plays significant role in surveillance applications.

This thesis does not deal about all the scenarios specified above, but it focuses

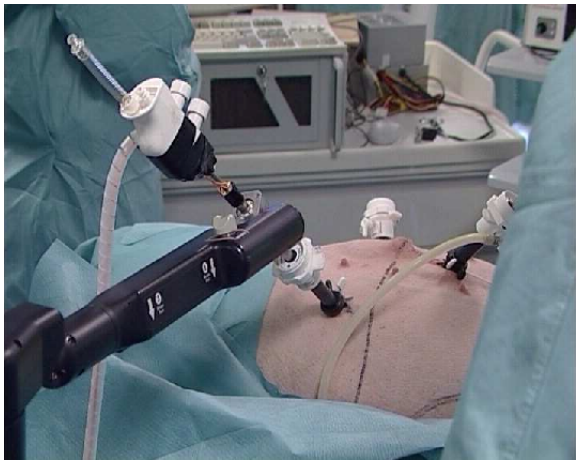


Figure 1.2: Robot Assisted Surgery

to object recognition and pose estimation using a humanoid robot that can manipulate recognized objects to do specific tasks. Here we work with two scenarios with one single object:

1. object is on the table

2. object is held by a person in front of the humanoid robot

We follow a visual-based approach using a RGB-D sensor, embedded in the robot, which provides 2D and 3D cues. The conventional RGB or the mono-color image is a 2D data and 3D data is fetched from the depth map. In computer vision, a depth map is a single channel image that contains the distances of the surfaces of observed objects from a view point. Here, we use pure 3D information to detect and estimate the pose of the objects in the scene as depth information enhances the accuracy of pose estimation. The depth- based algorithms have an advantage of independence from object texture and invariance to illumination. This approach is tested with Microsoft Kinect v1 and Orbec Astra Pro 3D sensors, whose specifications are described in Section 2.2.1.

In order to estimate the pose of an object we first need to create a 3D model of the target object in an off-line phase, and secondly pose estimation of the object can be calculated by aligning the current object point cloud to the model point cloud with initial alignment and refining the pose with Iterative Closest Point (ICP) technique in an on-line framework.

We are collaborating with other group in the lab who develop the path planning and control of humanoid that will interact with the objects. In the context of perception, we try to give them the best accurate pose of the object to grasp.

1.3 About LAAS

The Laboratory of Analysis and Architecture of Systems (LAAS)[1] is a CNRS[2] research unit linked with the Institute for Engineering and Systems Sciences (INSIS)[3] and the Institute of Information Sciences and their interactions (INS2I)[4].

It is located at Toulouse, France and is associated with University of Toulouse. LAAS research activities fall within the domain of Information Sciences and Technologies and address complex systems, and at different scales, to devise theories, methodologies and tools for modeling, designing and controlling them. The lab has strong relationships with the industry and works in a large number of collaborative projects international, national and regional industries of all scales.

The systems considered in the research of the lab are of different kinds such as integrated, distributed, embedded, mobile, autonomous and robotic, micro and nano, biological systems. They fall in various application domains such as aeronautics and space, telecommunications, transports, production, services, security and defense, energy management, health-care, environment and sustainable development. There are more than 400 researchers and PhD students, internship students, working in the lab supported by administrative staff.

The lab's scientific research is distributed into 8 main departments, in which Robotics is one of it.

Robotics. The Robotics department conducts research along several themes involving perception, control, decision-making, motion, communications and interaction between robot and its environment. It is subdivided into three groups working on different themes: 1)GEPETTO[5] - this team aims at analyzing and

[1] https://www.laas.fr/public/en

[2] The National Center for Scientific Research (CNRS), is a public organization under the responsibility of French Ministry of Education and Research http://www.cnrs.fr/index.html.

[3] http://www.cnrs.fr/en/institutes/insis-engineering-systems-sciences.html

[4] http://www.cnrs.fr/en/institutes/ins2i-information-sciences-technologies.html

[5] More information can be found at http://projects.laas.fr/gepetto/

generating the motion of anthropomorphic systems, 2)Robotics Action and Perception (RAP)[6] - this team's research mostly takes place within the fields of perception, sensor-based motion and sensor integration in robotics, 3)Robotics and InteractionS (RIS)[7] - this team develops projects on autonomous mobile machines that integrate perception, reasoning, learning, action and reaction capabilities.
Robotics Action and Perception(RAP). The research developed in this group is presented along 6 domains: visual perception of humans, perception on objects and environment, robot audition, sensor based motion, integrated sensors, signal processing. This thesis is carried out as a part of RAP group in the domain of perception on objects and on environment. We are collaborating with GEPETTO and RIS for the integration of pose estimation module on humanoid robot 'Pyrene' [46] and a UAV respectively.

1.4 Project Objectives

In order to achieve the specified problem statement, the following project objectives are noted.

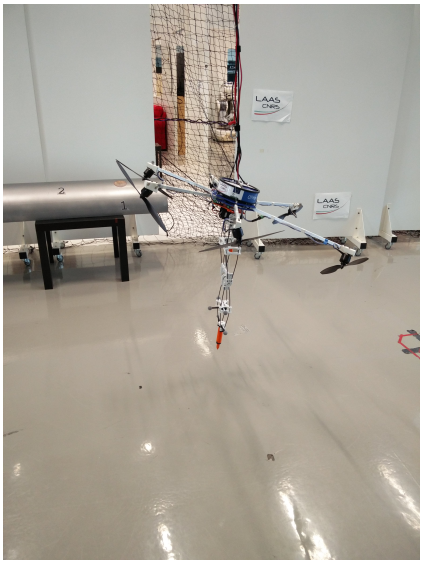
1. To study the state-of-art object localization techniques.
2. To generate 3D models of the objects.
3. To segment objects present on table and recognize them.
4. To estimate pose of object with the help of the 3D model and ICP.
5. To capture the ground truth data using Motion Capture [58] system and analyze the results quantitatively.
6. Integrating with robotic platforms.

The object recognition and pose estimation module is expected to be integrated with the humanoid robot 'Pyrene' (Figure 1.3a) of TALOS [46] generation from PAL Robotics and also with the Unmanned Aerial Vehicle (Figure 1.3b). The both robotic platform need a perception module for estimating the pose of the

[6]More information can be found at https://www.laas.fr/public/en/rap
[7]More information can be found at https://www.laas.fr/public/en/ris

object, to pick the objects and perform tasks with the objects (like pick the drill and make a hole to wood, pick the brick and stack them to build blocks with swarm of UAVs, among others).

(a) Humanoid Robot 'Pyrene' at LAAS



(b) UAV at LAAS

Figure 1.3: Robots at LAAS

1.5 Work Planning

The work is managed and planned effectively as shown in Figure 1.4.

1.6 Thesis Structure

This thesis is organized as follows: Chapter 2 presents the related work and framework. Chapter 3 introduces the methodology to achieve different tasks like clustering objects on table, 3D model generation and pose estimation. Chapter 4 shows the experimental results, analysis and discussion of results. Chapter 5 summarizes the key points in this work and proposes future work.

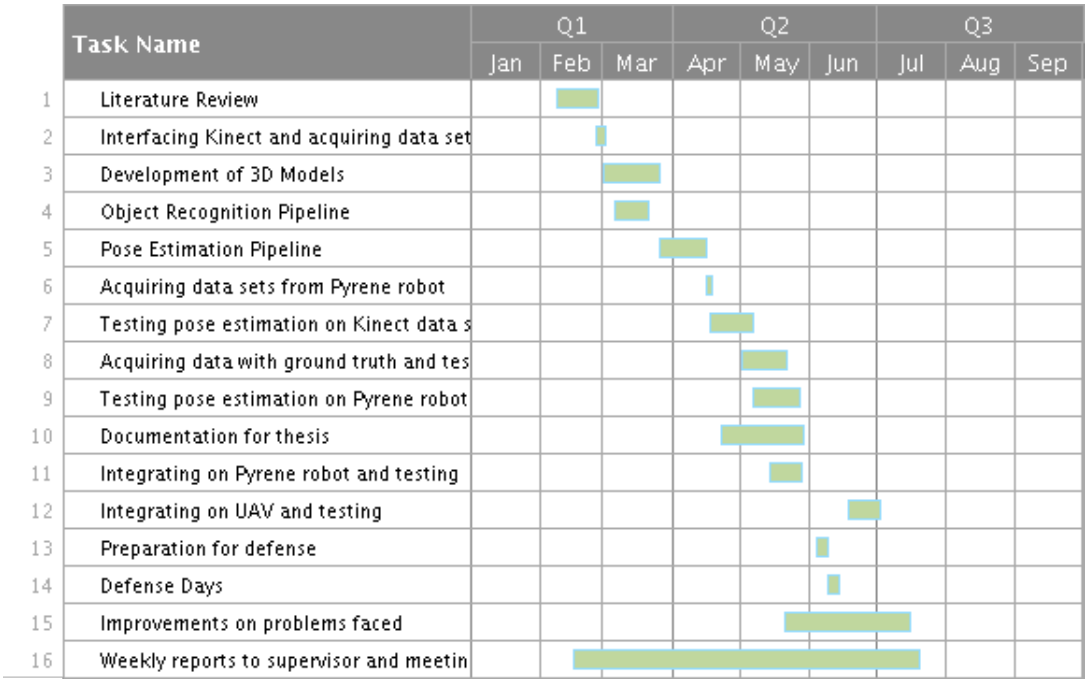


Figure 1.4: Gantt Chart of work

Chapter 2

Related Work and Framework

Necessity is the mother of Invention

-anonymous

2.1 Related Work

The topic of object recognition and pose estimation has been widely researched in the last decade, but there are many unresolved issues and challenges. The methods based on sparse feature [39] have shown good results for accurate pose estimation. Their popularity declined in the scenario of robotic applications because they rely on texture. In Hough-voting based methods, all pixels cast vote into a quantized prediction space. The cell with majority of votes is taken as the winner. Hough voting scheme for 2D object detection and pose estimation is used in [23]. An iterative latent class Hough-voting scheme is proposed by Tejani *et al.* [53] for object classification and pose estimation with RGB-D data as input.

Template based methods [29, 33] are also applied to estimate the pose. The template is scanned across the image and a distance metric is computed at each position to find the best match. A multi-modal template matching approach is proposed by Hinterstoisser *et al.* [30] which is able to detect textureless objects in highly cluttered scenes. A distance based approach is proposed by Lai *et al.* [36] for object classification and detection. Ruotao He *et al.* [28], proposed a template

matching based on LINEMOD for object detection and pose estimation.

With the advent of 3D acquisition systems (e.g Microsoft Kinect), there is a growing interest in 3D object recognition and pose estimation using depth data. Many advantages, like easier segmentation, robust pose estimation and new geometrical features has made the researchers to shift their focus from 2D approach to the analysis of RGB-D data. A method for object recognition in cluttered scenes using Point clouds has been developed by Papazov and Burschka [43].

With the development of Point Cloud Library (PCL) [50] - an open source library of algorithms for 3D point cloud processing - several algorithms for object recognition using 3D data have been proposed. Using a local Fast Point Feature Histogram (FPFH) descriptor [47] , Rusu *et al.* developed a global View Point Feature Histogram (VFH) descriptor [49]. A successor of both of the above descriptors, Cluster Viewpoint Feature Histogram (CVFH) [8] was developed to robustly handle more occlusions.

A lot of work has also been done in head pose estimation. An approach using key-frames with offline learning and online pose estimation using ICP is presented in [38]. Li *et al.* [37], proposed 3D head pose tracking using ICP with online face template reconstruction. Generalized Adaptive View based Appearance Model (GAVAM) [42] presents probabilistic framework which integrates all three pose estimation paradigms dynamic/motion based approaches, static user-independent, static user-dependent.

Pose estimation as a energy minimization with global hypothesis is presented in [40]. R. Brigier *et al.* [13], proposed symmetry aware evaluation of 3D object detection and pose estimation. A local bundle adjustment technique based on key frames and previous frames is proposed to avoid drift and jitter [54]. Object detection using multimodal point pair features and geometric edge extractor and pose estimation using local voting scheme, clustering and ICP is presented in [18]. W. Czajewski *et al.* [16], proposed a VFH based object detection and Camera Roll Histogram (CRH) descriptor based pose estimation with ICP as pose refinement. Single image object detection and pose estimation using deformable parts model classifier based on gPb contours with the help of superpixel and chordiogram matching is proposed by M. Zhu *et al.* [60]. Probabilistic classifier based on maximum likelihood, by defining Probability Density Function (PDF) for the

pose of the object, given correspondences between scene and model features is presented in [27].

With the recent trend and its success in the field of 2D image analysis, the deep convolutional neural networks emerges as a natural consequence in application to 3D object recognition. Due to the complexity of 3D information, approaches based on depth data could not tackle all the challenges although it has been used over its RGB counterpart. To jointly perform depth prediction and surface normal estimation, an adaptive multi-scale CNN architecture was proposed in [19]. The performance of deep learning methods is better than most of classical approaches in object detection and model alignment.

Although deep learning methods started to outperform the conventional techniques, they require very huge data sets and expensive processing units for computation. Even on powerful GPUs the training time requires several days for CNNs. The evaluation of different approaches should not be based exclusively on their pure performance but also on the effort put during the training phase and its cost. If the only goal is accuracy CNNs will definitely be the answer, but for cost-effective solutions, classical approaches should still be considered.

Considering the state-of-art techniques, we proceed further with the pose estimation using the depth data, due to the fact of it being invariance to illumination and independent from object texture. We use the classical ICP algorithm to estimate the pose of objects considering the fact that it does not require very huge dataset and expensive processing units like deep learning methods.

2.2 Framework

2.2.1 RGB-D Cameras

General

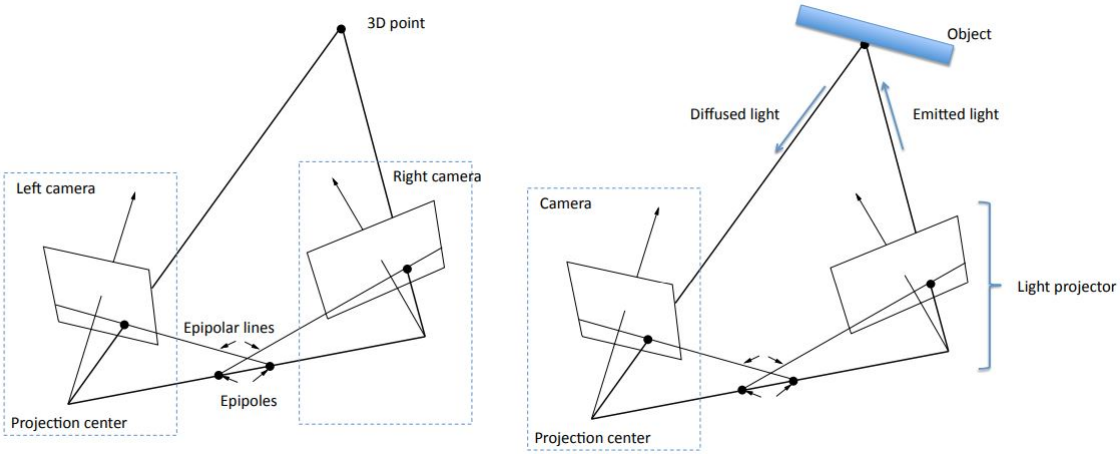
RGB-D cameras have been widely used in many computer vision and robotics applications such as pose estimation [57], 3D reconstruction [56], visual SLAM [26], amongst others. This camera provides two types of images.

1. A color image in RGB color space

2. A depth image, where every pixel corresponds to real world distance between surface of object in scene and the camera optical center.

Based on the technology used to measure the depth information, the RGB-D cameras can be divided into active and passive [25].

The passive RGB-D cameras contain two RGB cameras with known extrinsic parameters between them. Here, each camera grabs a frame, features in the two frames are matched and triangulation is applied to compute the depth. An illustration of passive RGB-D sensor is shown in Figure 2.1a.



(a) Passive stereo

(b) Active stereo

Figure 2.1: Illustration for the principle of passive and active stereo. Source [32]

Active RGB-D cameras usually have a light source to emit light, and to compute the depth information of the scene. Active cameras can be further classified as structured light and time of flight (ToF) cameras.

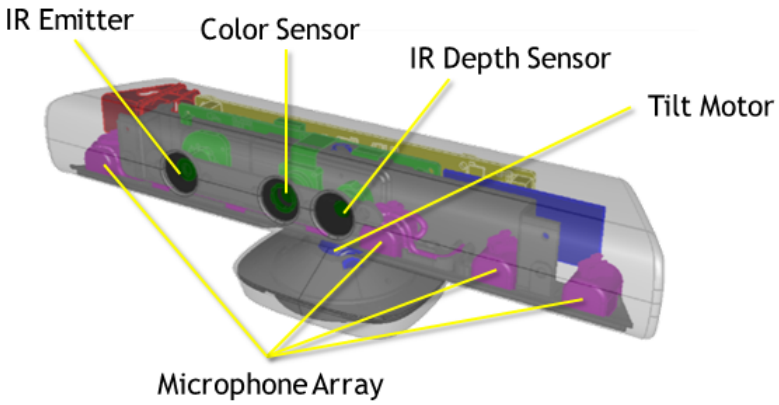
Microsoft Kinect v2 is an example of ToF camera that typically emits pulsed or modulated light and calculates the depth information by measuring the time delay between emitted and incident light after reflection from the object.

RGB-D camera with structured light use a projector to emit light of known pattern. Since, transformation between camera - projector is known a *priori*, camera observes the projected pattern and triangulates to calculate the depth as shown in Figure 2.1b. Microsoft Kinect v1 and Intel Realsense Euclid sensors are examples of structured light RGB-D cameras. As structured light camera projects light in Infrared (IR) range, the usage of these types of cameras is limited to indoor

environments. Kinect v1 is used for testing this approach before integrating with the robotic platforms.

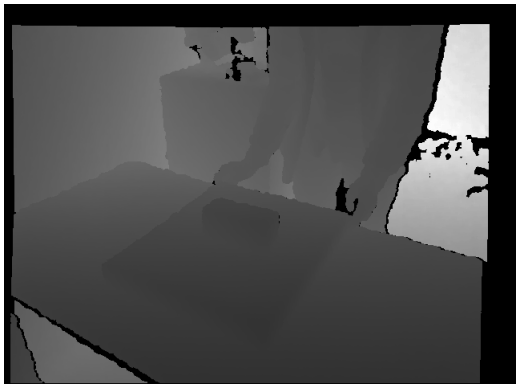
Microsoft Kinect v1

Kinect v1 has a Color CMOS camera that stores three channel information, an infrared (830nm) projector and Infrared CMOS camera, also it has an array of microphones and a tilt motor for sensor adjustment as shown in Figure 2.2a. The color camera has a resolution of 640 x 480 with 30fps and depth camera has a resolution of 320 x 240 with 60fps and 640 x 480 with 30fps. Its depth range varies from 0.5 to 3.5m.



(a) Camera Structure [41]

(b) RGB Image



(c) Depth Image

Figure 2.2: Upper Row : Structure of Kinect sensor. Lower Row : RGB and Depth image captured by RGB-D camera

Orbec Astra Pro

Astra Pro has a Color sensor, an IR projector and IR sensor and two microphones on either side of sensor as shown in Figure 2.3. The RGB camera has a resolution of 1280 x 720 with 30fps and depth camera has a resolution of 640 x 480 with 30fps. The range of depth sensor varies from 0.6 to 8m (optimal 0.6 - 5m).

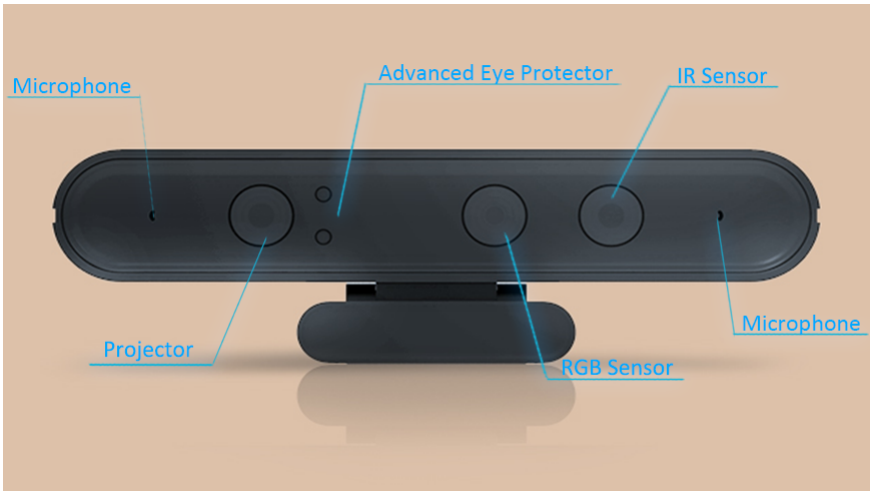


Figure 2.3: Structure of Orbec Astra Pro

Intel Euclid

Intel Euclid development kit is a full featured computer in the size of a candy bar as shown in Figure 2.4. It has an Intel Atom x7-8700 Quad-Core processor, 4GB memory 32 GB storage, Intel RealSense ZR300 camera and many other interesting features [34].

2.2.2 Open Source Libraries

The project is developed in C++ and we used the following open source libraries to realize the project objectives :

OpenCV

Open source Computer Vision (OpenCV) [35] is a library for computer vision and machine learning algorithms. It was developed to provide common infrastructure

Figure 2.4: Structure of Intel Euclid. Source [34]

for computer vision applications and accelerate the development of machine perception methods. It has numerous algorithms, which include a set of both classic and state-of-art computer vision and machine learning algorithms. These algorithms can be used to track moving objects, detect and recognize faces, produce 3D point clouds from stereo camera etc. We used OpenCV 2.4.8 in our project to read and manipulate the RGB and depth images provided by the RGB-D sensor, in such a way we can create the point cloud with the help of intrinsic parameters of RGB-D cameras.

PCL

The Point Cloud Library [50] (PCL) is a standalone, large scale, open project for 2D/3D image and point cloud processing. It consists of numerous state-of-art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. It is cross-platform and successfully compiled and deployed on Linux, MacOS, Windows and Android/iOS. PCL is split into a series of smaller code libraries, that can be compiled separately. We used PCL 1.7 in our project. We used PCL to do all the pre-processing and main processing tasks with the point clouds we produce.

ROS

The Robot Operating System [6] (ROS) is a flexible framework for writing robot applications. It consists of libraries, conventions and tools to simplify the task of creating complex robotic behavior. ROS was built from the ground up to encourage collaborative robotics software development. And it's all open source. We used ROS Indigo as a part of our project and is used to recover data from all the evaluated cameras. This is because all the robotic platforms, such as the humanoid robot 'Pyrene', works fully on ROS.

2.3 Conclusion

The related work of pose estimation is studied and presented, and the framework required for our approach with respect to different RGB-D cameras and open source libraries is stated. We further develop the methodology of our approach in next chapter.

Chapter 3

Methodology

Those who wish to succeed must ask the right preliminary questions

-Aristotle

The objective of the system is to detect 3D objects and estimate their pose. This information will be used by humanoid robot, that will grasp the objects. The entire point cloud processing is performed with the help of PCL [50] and 2D image processing is performed with OpenCV [35] library. A general diagram of the proposed pose estimation system is shown in Figure 3.1

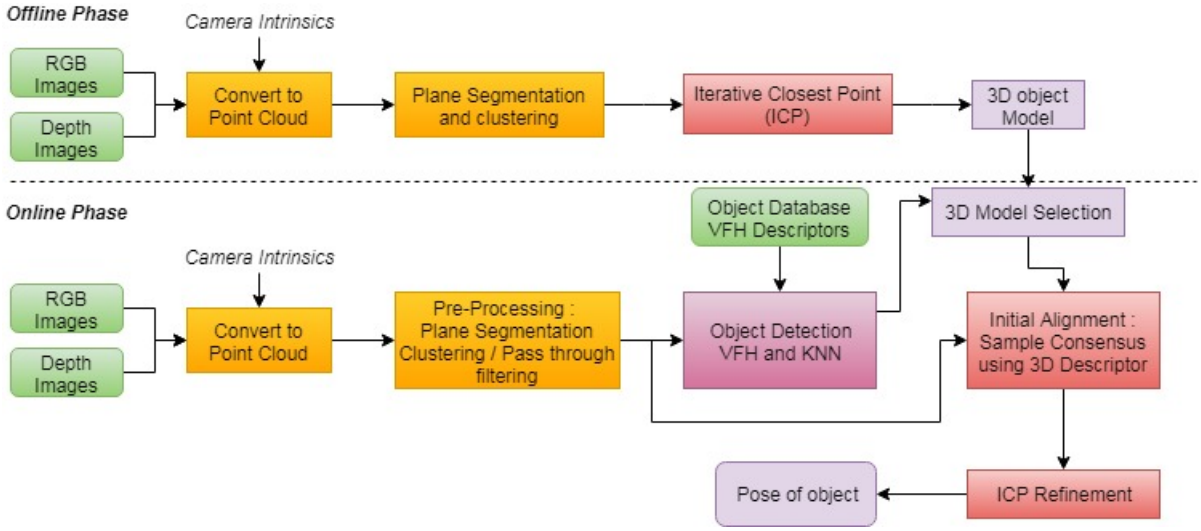


Figure 3.1: Overview of the proposed vision system

An RGB-D image captured by a sensor is firstly converted into a point cloud through the camera intrinsic parameters. Then, the point cloud is filtered and segmented into clusters. Below is a quick overview of the two steps of our method:

First, in an offline phase, we acquire a set of RGB-D images of the object in different views. Then, we convert them to point clouds and perform an initial filtering to remove those points that have a depth greater than a set threshold. We place the objects on a rotating platform (a plane) to reconstruct their 3D model. We can estimate the plane and segment the object on the plane. By this way we could segment it from the scene and acquire the point cloud of object in different viewpoints by rotating the platform and performing the steps as described above. As we get point clouds of object in different viewpoints, we apply Iterative Closest Point (ICP) over all the point clouds to register them and then we form a 3D representation of the object, as shown by the red block in offline phase in Figure 3.1, which is later used in an online pose estimation phase . The point clouds of the object in different views can also be used to create a part of dataset for object detection. We compute View Point Feature Histogram (VFH) [49] descriptor for the point clouds of the object and represent the VFH descriptors as a kd-tree for efficient search with K-Nearest Neighbors (k-NN), in online phase for object detection.

Second, in an online phase, we acquire the RGB-D image of the scene to detect and estimate the pose of the object. We convert the RGB-D image into a point cloud and then we apply a pre-processing step that suppress flat areas, i.e. the table, and segment the objects by clustering them. The segmented clusters on the planar surface are described with View Point Feature Histogram (VFH). The VFH descriptor of the segmented clusters are compared against the descriptors of the target object using k-Nearest Neighbors. The clusters are classified according to number of matches and the labels of the corresponding object is assigned, we choose the 3D model of that particular object and we proceed to estimate the pose. We apply an initial alignment between the object and the model using sample consensus and a 3D descriptor like FPFH. Finally, we refine the alignment of the object and model using Iterative Closest Point (ICP) algorithm to estimate accurate pose of the object.

3.1 3D Model Reconstruction - Offline phase

We reconstruct the 3D model of the object by acquiring the depth frames of the object (placed on a plane) in different view angles, converting them to point clouds followed by clustering of object to segment it from plane. Then, we apply ICP to register all the frames and reconstruct the 3D model of the object.

3.1.1 Conversion of RGB-D to point Cloud

We use ROS [6] with OpenNI package [5] to launch the RGB-D sensor (i.e., Kinect). The sensor node[1] publishes various topics (data) such as the RGB image, depth image and point clouds. As subscribing to point cloud's topic is expensive with respect to data transfer rate, we subscribe to RGB and depth image topics which has a better data transfer rate than point cloud topic.

We need to convert the RGB and depth images acquired from the sensor to the point clouds. The depth image is registered with the RGB image by the sensor. Therefore, if we know the intrinsic parameters of the RGB camera, we can convert depth and RGB images to RGB point cloud.

The depth image is of 16-bit length, i.e., unsigned integer type with single channel. Thus, the values range from 0 to 65535. The depth is given in units of sensor, i.e., millimeters.

Then, the 3D point (X, Y, Z) is calculated as follows:

$$Z = z/S \tag{3.1}$$

$$X = (u - p_x)Z/f_x \tag{3.2}$$

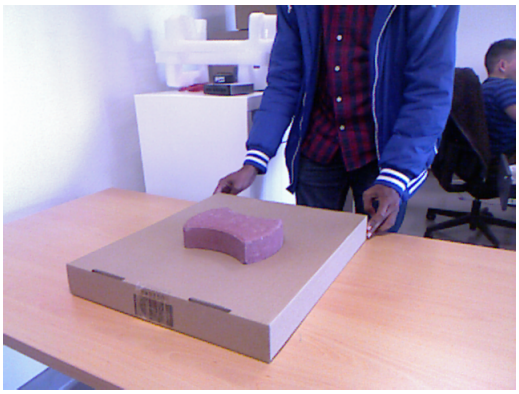
$$Y = (v - p_y)Z/f_y \tag{3.3}$$

where z is the depth at pixel location (u, v). S is 1000, as we need to convert millimeter to meters for processing using PCL (PCL, default is in meters). (p_x,p_y) are the center of the camera and (f_x, f_y) are focal length of camera in X and Y directions respectively.

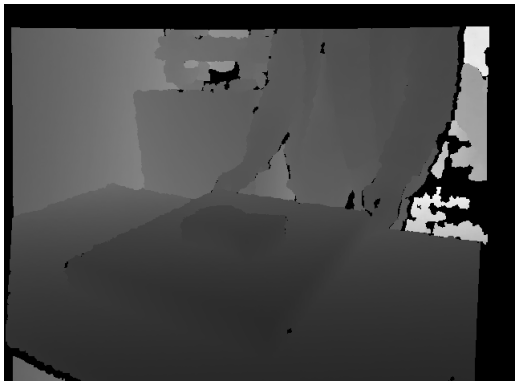
From the above equations we can get the 3D point (X, Y, Z) in world coordinates and we can use the color information from RGB image at pixel location

[1] In ROS, node is a process that performs computation

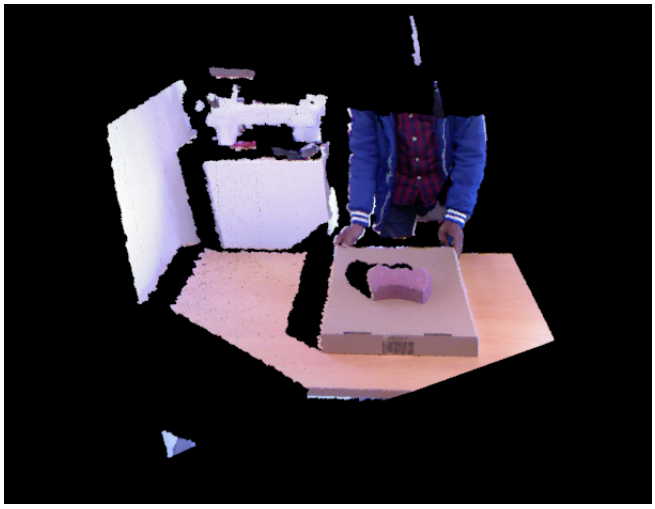
(u, v) to colorize that particular point in the cloud. An example of point cloud is shown in Figure 3.2.



(a) RGB image



(b) Depth image



(c) Point cloud from RGB and depth images

Figure 3.2: An example of point cloud

3.1.2 Plane Segmentation and Clustering

To reconstruct the 3D model, we have kept the objects on a rotating plane to observe the object from all points of view. In order to form the 3D model of the object, it is indeed necessary to remove the irrelevant points from the scene trying to keep only the points corresponding to the object. This is done by the plane segmentation and clustering module. The diagram of this module is as shown in Figure 3.3

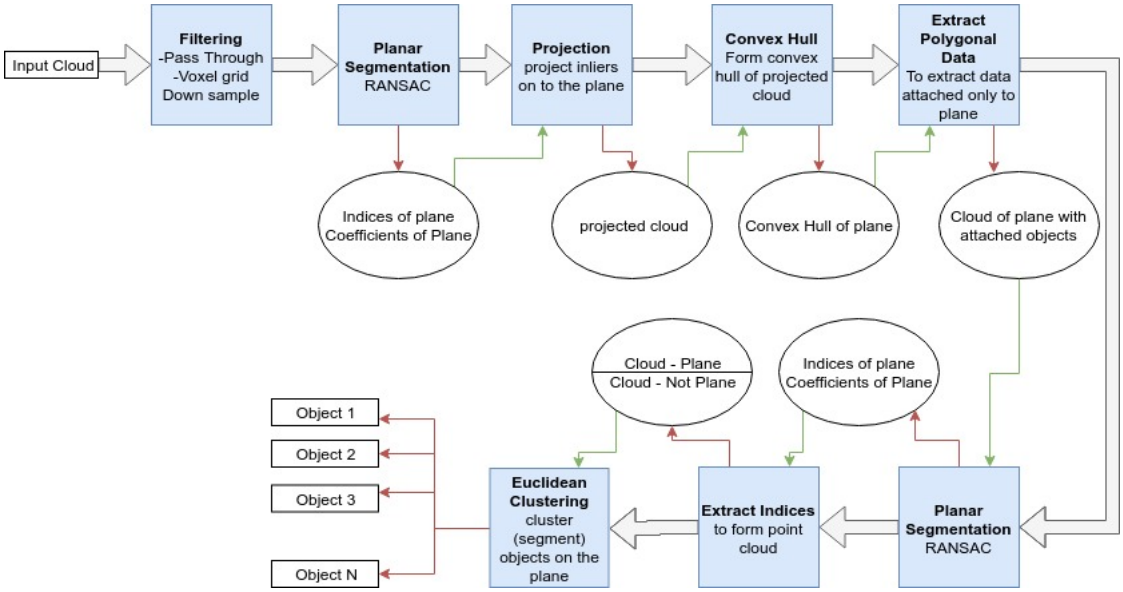


Figure 3.3: Diagram of segmenting objects on a plane

Here, the idea is to form a plane model using RANSAC (Random Sample Consensus) [21]. Once we have a plane model, we can find objects standing on the plane by computing the convex hull of planar points and extruding this outline along the plane normal to segment the plane and objects attached to it from the remaining points. PCL's *ExtractPolygonalPrismData* is a class just intended for this purpose. Now, we just have the plane and objects standing on it. Then we want to segment each object standing on plane.

As we have the plane model already built, we can use the inlier points to remove the plane from the point cloud, so the remaining points in the cloud are just those of the objects. We clusterize the points using the Euclidean clustering technique [1] so each isolated cluster is an object candidate. The process is briefly described as below:

Filtering. First, the point cloud of the scene is filtered by a pass through filtering module[2] and down sample the point cloud, to speed up the processing.

Planar Segmentation. We use RANSAC to find a plane, it is a randomized algorithm for robust model fitting. The basic operations here are :

- Select sample set

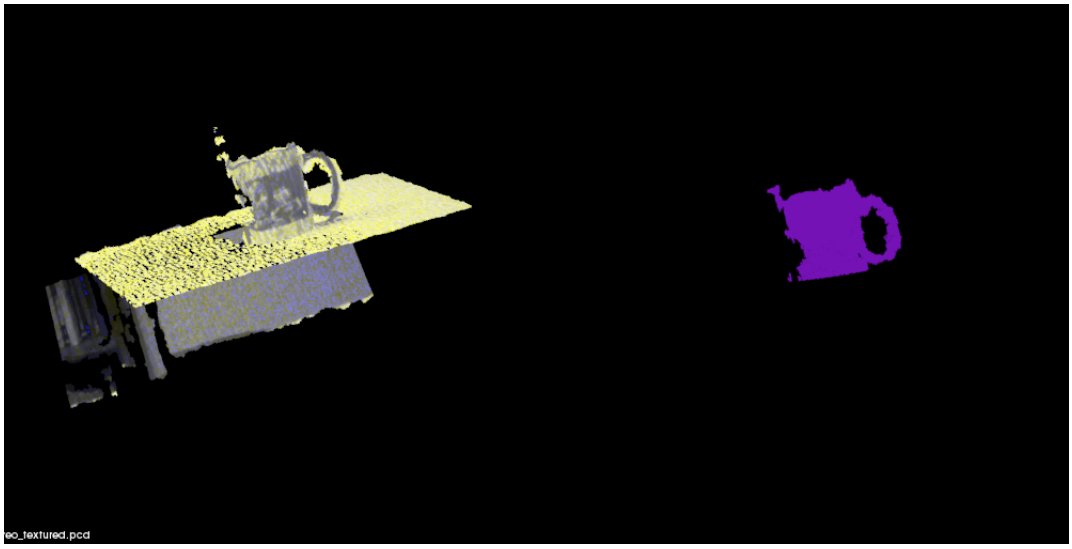
[2]Filtering based on spatial coordinates to remove the point which are above the set threshold in X,Y and Z directions

- Compute Model
- Compute and count inliers
- Repeat until sufficiently confident

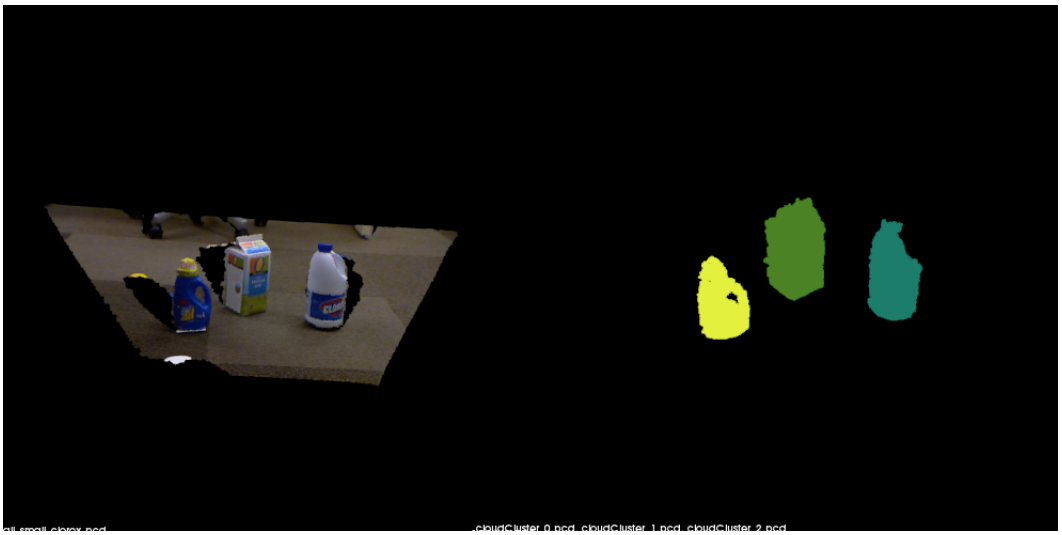
There are several models that can be used with the RANSAC method and PCL supports models such as plane(with constraints such as orientation), Sphere, Cone, Cylinder, Line, Circle etc. After the plane model has been fitted we retrieve best set of inliers and the corresponding plane model coefficients.
Projection. As we fit the model, some inliers may not be on the plane. Therefore, we project all the inliers on to the plane, with the help of parametric model of plane, to form a convex hull which is used for further segmentation. Here, we get a projected cloud representing the plane.
Convex Hull. We can now form a convex hull polygon of the plane, which is used in segmenting polygonal data from the point cloud of scene.
Extract Polygonal Data. It uses a set of point indices that represent a planar model - and together with a given height - it generates a 3D polygonal prism. Thus, points associated with the 3D polygonal prism can be extracted from the scene point cloud.
Extract Indices. RANSAC-based plane segmentation gives a set of inliers, we can extract their indices to get a cloud with the points of the plane and thereby we can subtract not plane points. So, now we finally get point cloud with only objects to further segment them.
Euclidean Clustering. The remaining points are then segmented with the help of a cluster extraction algorithm in an Euclidean sense. The basic idea is to use a region growing approach that cannot "grow"/connect points with a high distance, therefore it merges local dense areas and splits clusters. Finally, we have segmented the objects into different clusters as shown in Figure 3.4.

3.1.3 3D Model - Registration using ICP

Registration consists in the alignment of multiple frames (of point clouds) of same target in different points of view. The goal is to find the relative pose (position and orientation) between all the views in a global coordinate frame, such that



(a) Cup over a flat table



(b) 3 Objects over a flat table

Figure 3.4: Two examples of Object Segmentation on a Plane (Left) Point cloud (Right) Segmented point cloud with target objects

overlapping areas between point clouds match as well as possible. This procedure will align individual point clouds and fuse them into a single point cloud.

We do pairwise registration of 3D point clouds, i.e., aligning two point clouds with overlap by estimating the transformation between both.

The problem of registration tries to find the correspondences between source and target point and estimate a transformation. The transformation when applied

on source point cloud, aligns all pairs of corresponding points with target point cloud. Here, the corresponding points are usually not known and needed to be determined.

Typically, the registration consists as follows :

1. *Selection:* Sampling of point clouds to speed up the process.
2. *Matching:* Estimating the correspondences between points in subsampled point clouds.
3. *Rejection:* Filtering the false correspondences.
4. *Alignment:* Assigning an error metric and minimizing it to find the optimal transformation.

The ICP algorithm was developed by Besl and Mckay [11] as an iterative registration method. There, the closest points in Cartesian space are considered correspondences of each others. ICP searches the closest point correspondences (*matching*) and align the found point pairs (*alignment*). These two steps are repeated until convergence thereby iteratively refining the alignment between source and target point cloud. If there is perfect overlap[3], the alignment converges to global minimum, i.e., optimal alignment. But the main drawback is that the algorithm may get caught in local minima if there is very less overlap or if initial alignment is bad. In order not to get caught in local minima, we have to suppress the false correspondences (*rejection*) as they affect negatively the registration results.

Selection - Sampling Point Clouds

Depending on the sensor and resolution, the point clouds can be quite large. Consequently, registering large point clouds is considerably more computationally expensive than with less number of points. As the data is unnecessarily detailed for the task of registration, we perform a sampling of point clouds before registration, saving computation time. PCL implements several sampling methods such as: sampling in index space, uniform sampling and sampling in space of

[3]The overlap that has majority of corresponding pairs between source and target point cloud

local surface normals. We used uniform sampling as it preserves the geometry of the object.

Matching - Correspondence Estimation

The task of correspondence estimation try to match the points from the source point cloud to their correspondence in target point cloud. The goal is to find ideal correspondences.

The basic way to search for nearest neighbors is to search through all target points for each source point. But this is computationally expensive, thus, various data structures for quick searches have been proposed, such as kd-trees and octrees.

Correspondence Estimation based on Projection. It computes correspondences by projecting the source point cloud into the target point cloud using the camera intrinsic and extrinsic parameters.

Correspondence Estimation using Normal Shooting. It computes correspondences between the clouds that have the minimum normal distance. We use it in our approach taking the advantage of normals to correctly estimate the correspondences.

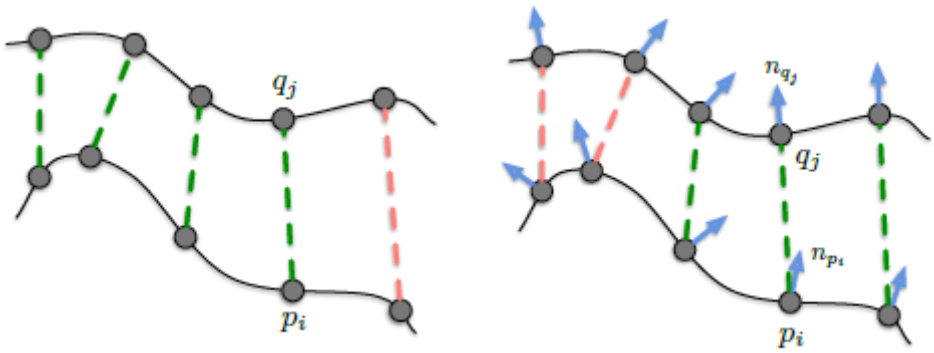
Rejection - Filtering Correspondences

Invalid correspondences can negatively affect the registration results, therefore correspondence rejection is a vital step in the registration pipeline. It filters the point pairs, matched in the correspondence estimation stage, in order to facilitate the transformation estimation and improve convergence towards global minimum.

Rejection based on distance threshold. We can filter out correspondences between pair of point clouds by specifying a maximum distance between corresponding points such that pairs with a larger point-to-point distance are filtered out as shown in Figure 3.5a.

Rejection based on median distance. Instead of a fixed threshold like the previous rejector, we can compute it as the median of all point-to-point distances in the input set of correspondences.

Rejection based on normal compatibility. This rejector rejects corresponding pair of points based on normal information. It will reject the pairs with inconsistent



(a) Distance threshold rejector (b) Normal compatibility rejector

Figure 3.5: Different Correspondence rejectors. Good correspondence pairs (green) and bad correspondence pairs (red). Source [31].

normals, i.e., the angle between their normals is larger than a given threshold as shown in Figure 3.5b.

There exists other rejectors (such as rejection based on surface boundaries, duplicate matches among others) which are not relevant to the problem.

Alignment - Error Metric and Transformation Estimation

The goal of this step is to estimate the transformation from source to target point cloud using the estimated correspondences. Here, we define an error metric, that is to be minimized, to optimize the transformation estimation. The transformation estimation T is composed of rotation R and a translation t. We would use homogeneous coordinates for this task.

There are two main error metrics to be minimized that have been considered: *point-to-point* (Eq. 3.4) and *point-to-plane* (Eq. 3.5).

$$E_{point-to-point}(T) = \sum_{k=1}^{N} ||Tp_k - q_k||^2 \tag{3.4}$$

$$E_{point-to-plane}(T) = \sum_{k=1}^{N} ((Tp_k - q_k).n_{q_k})^2 \tag{3.5}$$

where (p_k, q_k) is the k-th of the N pair correspondences from the source cloud to the target cloud.

(a) Point to point error

(b) Point to plane error

Figure 3.6: Error metrics. Source [31].

Point-to-point error metric. The default transformation estimation of ICP in PCL is based on Singular Value Decomposition (SVD), which uses the point-to-point error metric (Eq. 3.4). The point-to-point error metric is shown in Figure 3.6a [10].

Point-to-plane error metric. It was introduced by Chen and Medioni [15] (Eq. 3.5) as shown in Figure 3.6b, which is more stable and converge faster than its counter parts. It uses the distance between the source point p_k and plane described by the target point q_k and its local surface normal n_{q_k}. It uses Levenberg-Marquardt for the estimation instead of SVD.

Termination Criteria

Iterative registration algorithms require to set the termination criteria. The more classic criteria are:

Maximum number of iterations. We can set the maximum number of iterations and exceeding this means the optimizer diverged. This parameter has to be tuned depending on complexity of registration.

Transformation epsilon. This parameter defines the minimum transformation between two consecutive iterations of ICP that must be achieved to consider convergence.

Euclidean distance epsilon. It stops when the euclidean distance between two corresponding point pairs after alignment is less than this value.

From the above pipeline of selection, matching, rejection and alignment we can align two points.

We have used uniform sampling to down sample the original point cloud. The

correspondence estimation is done using normal shooting and rejection is based on the threshold of angles between surface normals of the corresponding points. The transformation estimation is based on point-to-plane transformation estimation.

The pipeline used for registration is as shown in Figure 3.7.

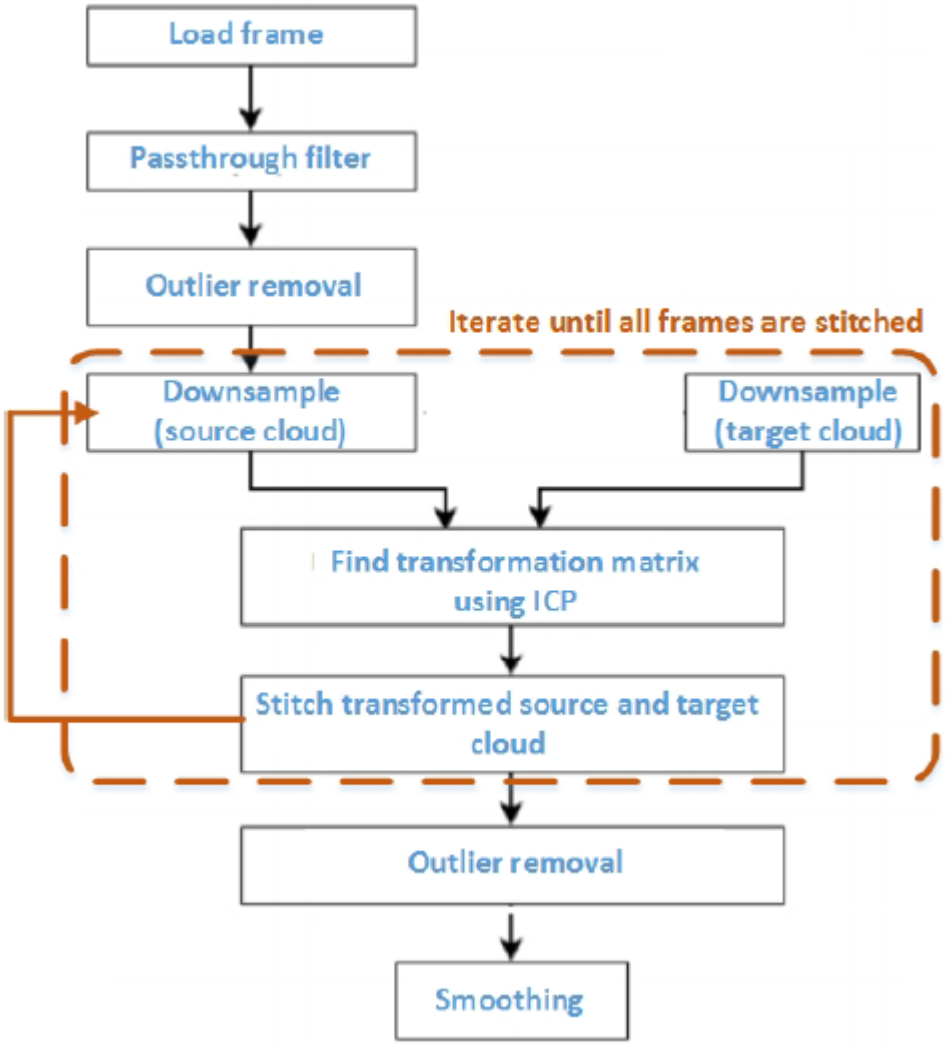


Figure 3.7: Registration pipeline

Statistical Outlier removal is used to remove the outliers based on standard deviation of point neighborhood [51]. Smoothing of the final cloud is done using Moving least square algorithm [9].

An incremental approach is employed in order to register a sequence of point clouds. This works as follows: we first register frame 1 (source cloud) and frame

2 (target cloud). The result (source cloud) is then registered with frame 3 (target cloud), and so on. Below illustrates this approach as shown in Figure 3.8.

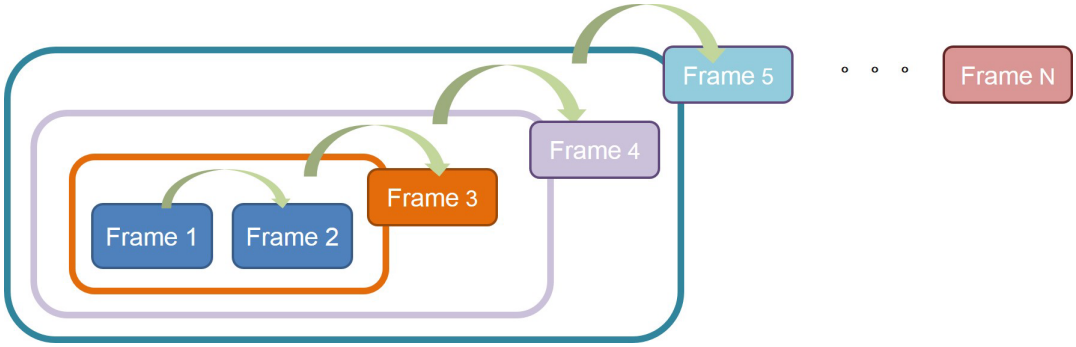


Figure 3.8: Incremental Registration N Frames

We had applied the process as described in Section 3.1.3 to the datasets we acquired using Kinect sensor. The results of registration are as shown in Figure 3.9.

3.2 Pose Estimation - Online Phase

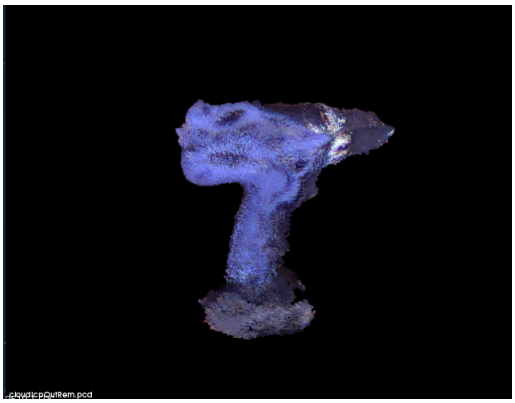
In order to estimate the pose of the object in the online phase, the RGB and depth image of the scene is acquired using 3D sensor (e.g., Kinect, Astra Pro). The RGB and depth images are converted to point clouds as described in section 3.1.1. If the object is placed on a plane surface (i.e., a table), we do the plane detection and object clustering as described in section 3.1.2. Then the object point cloud is aligned with the respective model using initial alignment and the pose is refined by final alignment using ICP.

3.2.1 Object Recognition

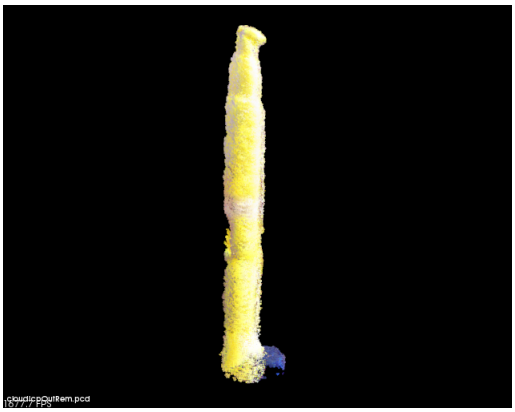
The existing object recognition methods can be broadly classified into : global feature-based methods and local feature-based methods. Each of them is different in the object representation.

Local feature descriptors are calculated based on geometrical features extracted from the individual keypoints found on the object. It does not require

(a) Brick



(b) Drill



(c) Cylinder object

Figure 3.9: 3D Registration of brick, drill and cylinder

prior segmentation but it involves time-consuming keypoint detection, description and comparison.

On the other hand, global feature descriptors are based on the geometry of entire object. As it defines the object, it requires prior segmentation of object from the scene. They have the ability to represent an entire object with a single feature vector, the recognition process is faster and robust to noise, which is important for near real-time applications. Hence, in this application we use a global

recognition method based on Viewpoint Feature Histogram (VFH) descriptor.

Viewpoint Feature Histogram (VFH)

The VFH [49] is based on the local Fast Point Feature Histogram (FPFH) [47] which is described in next section. It consists of viewing direction component and the extended FPFH component. The idea is to extend the FPFH feature descriptor that is to be estimated for the entire object cluster, and to compute additional statistics between the viewpoint direction and the normals estimated at each point. To do this, they used the key idea of mixing the viewpoint direction directly into the relative normal angle calculation in the FPFH.

The viewpoint component is computed by collecting a histogram of the angles that the viewpoint direction makes with each normal. The second component measures the relative pan, tilt and yaw angles as in FPFH descriptors, but now measured between the viewpoint direction at the central point and each of the normals on the surface.

The new feature with both components is called viewpoint Feature Histogram. An example is, here Figure 3.10, we can see two components:

1. The viewpoint component.

2. The extended FPFH component.

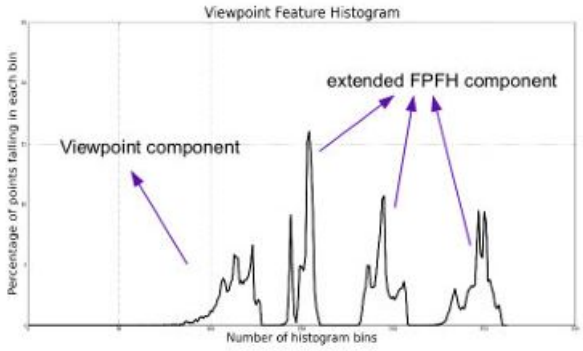


Figure 3.10: Viewpoint Feature Histogram

The default VFH implementation uses 45 bins for each of the three angles of a FPFH component. Another 45 bins for the distance between viewpoint and

centroid and 128 bins for the viewpoint component, which results in a 308 bins histogram. Since the bins are normalized using the total number of points in the cloud, VFH descriptor is sale invariant.

We firstly build the training set by acquiring the point clouds of objects from different viewpoints. We compute the VFH feature descriptor for every point cloud and convert it to Fast Library for Approximate Nearest Neighbors (FLANN) format. As kd-tree is one of the efficient search structures, we form a kd-tree of the training data and store if for further testing.

In testing phase, the target object is clustered and its VFH descriptor is computed, matching is performed using k-Nearest Neighbor search from the FLANN [3] library with a Chi-Square metric using the kd-tree built in the training phase.

3.2.2 Initial Alignment

Due to its greedy nature, ICP algorithm requires a reliable initial alignment to avoid converging to local minima. Therefore, we apply an initial alignment between the scene and model point clouds before refining the alignment with ICP.

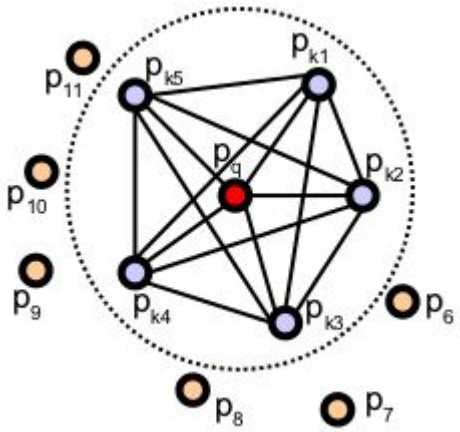
We use SAmple Consensus Initial Alignment (SAC-IA) which tries to maintain the same geometric relations between correspondences without having to try all combinations, it is limited to set of correspondences. Here we use Fast Point Feature Histogram (FPFH) descriptor as a similarity metric for the correspondence pairs. FPFH is a simplified version of Point Feature Histogram (PFH) to reduce the computational complexity.

Point Feature Histogram (PFH)

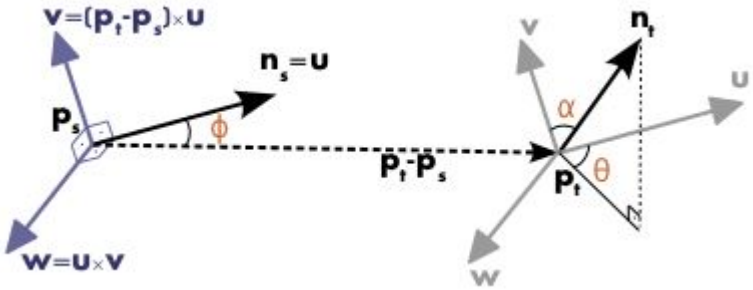
The PFH is a descriptor that encode a point's k-neighborhood geometrical properties by the mean curvature around the point using multi-dimensional histogram of values. It is invariant to pose and noise in neighborhood.

The PFH is based on the points in the k-neighborhood and their estimated surface normals. The quality of surface normal estimations at each point is very significant for this approach. The estimated surface normals helps to capture the surface variations in the neighborhood.

The Figure 3.11a shows an influence region diagram of PFH computation for a query point (p_q), and its k neighbors within the radius r. The PFH computation depends on all pairs of points in the neighborhood, thus resulting in computational complexity of $O(k^2)$.



(a) Influence region for query point p_q



(b) The local reference frame and the three angles , and computed by FPFH at each point pair.

Figure 3.11: Point Feature Histogram Computation. Source [47].

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 Darboux frame at one of the points as shown in Figure 3.11b.

$$
\begin{aligned}
\mathsf{u} &= \boldsymbol{n}_s \\
\mathsf{v} &= \mathsf{u} \times \frac{(\boldsymbol{p}_t - \boldsymbol{p}_s)}{\|\boldsymbol{p}_t - \boldsymbol{p}_s\|_2} \\
\mathsf{w} &= \mathsf{u} \times \mathsf{v}
\end{aligned}
\tag{3.6}
$$

Using the uvw frame as in Figure 3.11b, the difference between the two normals n_s and n_t can be expressed as a set of angular features as follows:

$$
\begin{aligned}
\alpha &= \mathsf{v} \cdot \boldsymbol{n}_t \\
\phi &= \mathsf{u} \cdot \frac{(\boldsymbol{p}_t - \boldsymbol{p}_s)}{d} \\
\theta &= \arctan(\mathsf{w} \cdot \boldsymbol{n}_t, \mathsf{u} \cdot \boldsymbol{n}_t)
\end{aligned}
\tag{3.7}
$$

where d is the Euclidean distance between the two points $\boldsymbol{p}_s$ and $\boldsymbol{p}_t$, $d = \|\boldsymbol{p}_t - \boldsymbol{p}_s\|_2$. The quadruplet $\langle \alpha, \phi, \theta, d \rangle$ 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.

Then, for a query point, we can create the final PFH representation by setting of all quadruplets in the bins of a single histogram. This process divides the range of each feature into b subparts counting, in each subinterval, the number of occurrences. Since 75% of the features are measures of the angles between the normals, we can easily normalize their values to the same interval in the trigonometric circle. An example of binning is to divide, into the same number of equal parts, the interval of each features. This creates a fully correlated histogram with b^4. Hence, a bin increments the corresponds having some values for its features.

Fast Point Feature Histogram (FPFH)

For real time applications, the computation of FPH in dense point neighborhoods can represent one of the major bottlenecks as the computational complexity of FPH for a given point cloud P with n points is $O(nk^2)$, where k is the number of neighbors for each point p in P.

FPFH is a simplified form of PFH, that reduces the computational complexity to $O(nk)$.

There are two steps in this process:

1. A simplified Point Feature Histogram (SPFH) is created using PFH as described in Section 3.2.2 for each query point, by computing angles between itself and its neighbors.

2. The final histogram (FPFH) is computed by re-determining the neighbors for each point and using neighboring SPFH values to weigh the final histogram as in Eq 3.8.

$$FPFH(\boldsymbol{p}_q) = SPFH(\boldsymbol{p}_q) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\omega_k}\cdot SPFH(\boldsymbol{p}_k) \tag{3.8}$$

   where the weight ω_k represents a distance between the query point p_q and a neighbor point p_k , thus scoring the (p_q, p_k) pair. To understand the importance of this weighting scheme, the Figure 3.12 presents the influence region diagram for a k-neighborhood set centered at p_q.

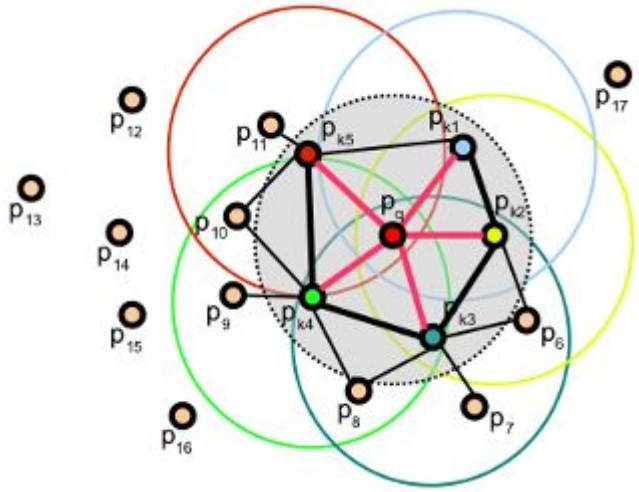


Figure 3.12: FPFH influence region for query point p_q

For a given point, the FPFH computation first estimates its SPFH values (between itself and its neighbors). This is repeated for all the points, followed by re-weighting of SPFH values of the point using SPFH values of its neighbors.

The pairs between a point and its neighbors is illustrated using red lines in the diagram. The additional lines with weighting scheme are shown in black and pairs counted twice are shown with thicker lines in the Figure 3.12.

SCA-IA

This approach is more detailed in [48] and it considers all possible matching pairs for initial alignment. But in SCA-IA, the matching pairs are sampled and we follow following scheme:

1. We select p sample points from source cloud with distance between any pair greater than a distance threshold.

2. For each of points selected in previous step, we find a list of points in target cloud whose histograms are similar to source cloud sample points histogram. Out of all matched histogram, we select one randomly and form correspondence.

3. Thus, we define an error metric to compute quality of transformation and compute the rigid transformation defined by sample points and their correspondences.

These three steps are repeated until convergence, and the transformation that resulted is used to roughly align the source to target cloud.

3.2.3 Final Alignment - ICP

With the help of initial alignment described in section 3.2.2, the scene and model point clouds are aligned coarsely which are needed to be aligned finely using ICP.

ICP requires an initial alignment due to its greedy nature, this is provided by SCA-IA. Now, we can perform ICP on the initially aligned model and scene point cloud to get the accurate pose of the object in scene.

As described in section 3.1.3, we register (align) the pair scene-model point clouds. The alignment procedure comprises of four steps:
Selection. We apply uniform sampling on source (model) and target (scene) point clouds to reduce the computational complexity of ICP.

Matching. As we need to estimate the correspondence pairs between source and target clouds, we use *correspondence estimation by normal shooting*, to find matching pairs between two clouds.
Rejection. As the clouds do not have full overlap, there exists some false matching pairs from correspondence estimation phase, which are needed to be rejected for accurate alignment. We use a combination of several rejectors. *Rejection based on distance*, *rejection based on normal compatibility* are two among them.
Alignment. We computed the genuine corresponding pairs by rejecting the false pairs. We have tried using the *point-to-plane error metric* and the *point-to-point error metric*. The *point-to-plane error metric* is more accurate than the *point-to-point error metric* but it crashes during the run time when the majority of normals are occluded. We looked into this problem and found that the program crashes as the program provides *NaN* (Not a Number) values to the kd-tree search. This is resulting because interiorly, the transformation matrix is computed and it results in *NaN*, which makes the points to *NaN* after applying the transformation. This is a bug in PCL program that needed to be corrected. So, we further use the *point-to-point error metric* for transformation estimation.

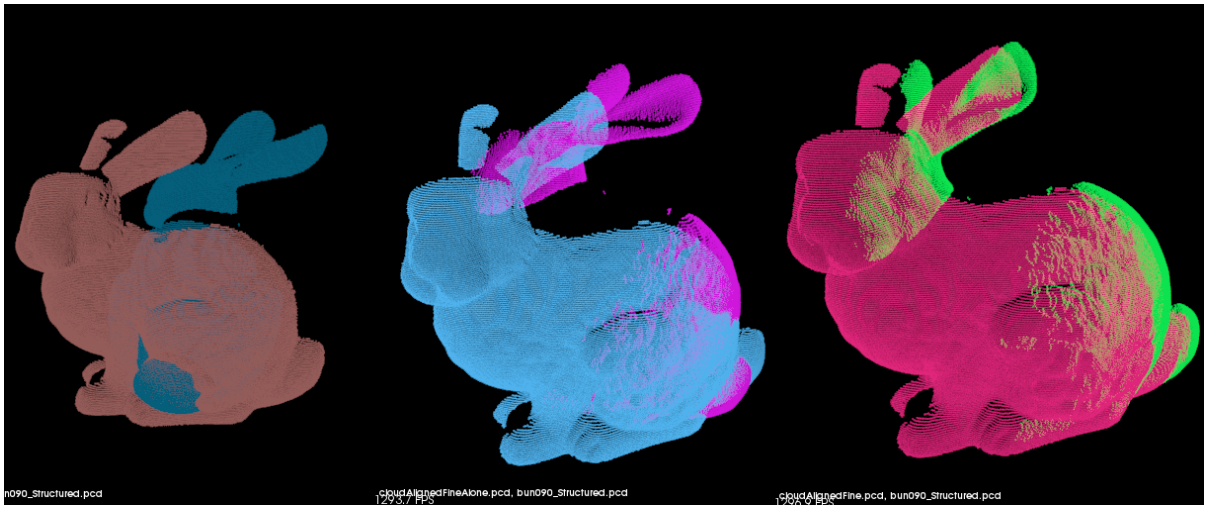
The above four steps are repeated until convergence which is imposed by termination criteria such as *maximum number of iterations*, *maximum transformation epsilon* etc.

We have tested this pose estimation approach based on SCA-IA on *Stanford Bunny Dataset* [7]. The results are as shown in Figure 3.13

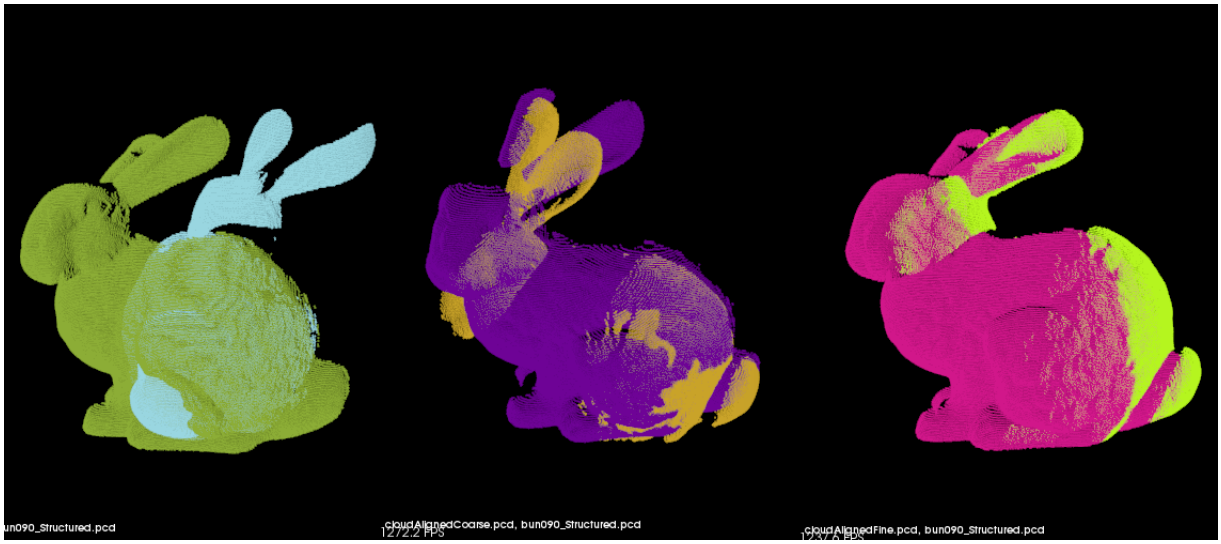
We can see the effect of SAC-IA on the final result in Figure 3.13a. Without the SAC-IA, the ICP did not converge accurately to align the two point clouds which are initially 90 degrees apart. We can see the significance of fine alignment step using ICP in Figure 3.13b, which resulted in a more accurate alignment of the two point clouds of bunny.

3.3 Conclusion

An approach for pose estimation is presented. The pose is estimated with the alignment of 3D model to target object. The model is generated in an offline phase with ICP. The model is initially aligned with target object with SCA-IA



(a) Two partial models of bunny before alignment (left); ICP on partial models without SAC-IA (middle); ICP on partial models with SAC-IA (right)



(b) Two partial models of bunny before alignment (left); result after applying SAC-IA (middle); result after applying SAC-IA + ICP (right)

Figure 3.13: Pose Estimation with SCA-IA and ICP

algorithm and further refined by ICP.

Chapter 4

Results and Discussion

One picture is worth more than ten thousand words

-anonymous

The process of initial alignment described in Section 3.2.2 is computationally expensive (Table 4.1) , we follow a strategy described below to estimate the pose of the objects in real time.

In order to align the model and object, we perform the SCA-IA method for the first frame, this provides a good initial guess of object pose which drastically reduce the computational time. The next frames only need a fine alignment, by ICP, considering the previous estimation. We place our model cloud at a predefined position and orientation. We estimate the pose of the object in first frame using SCA-IA and ICP then we align the model with the object in scene and leave the model at the aligned place. In the next frame,we use the previous estimation as the initial guess. This is based on the assumption that - the movement of objects between two consecutive frames is small. With this approach, we are not always required to compute the SCA-IA for every frame, thereby reducing the computational complexity. However, we can recover the rigid body transformation between the initial model and the model aligned to previous frame using Singular Value Decomposition (SVD) and one-to-one corresponding pairs of the model.

4.1 3D Object Pose Definition

This thesis tries to solve the problem of pose estimation of objects. Therefore it would be helpful to define the pose of an object.

In order to define a pose of an object in 3D space we need to define 6 variables, as objects have 6 degrees of freedom. The first 3 variables, referred as *x, y, z*, define the object location in 3D space. However this is not enough, since the object might be arbitrarily rotated. Therefore we need another 3 variables to measure the relative rotation with regard to the three principal axes. In Euler angles, the rotation around the x axis is called *roll*, around the y axis is called *pitch* and around the z axis is called *yaw*.

4.2 Datasets

The humanoid 'Pyrene' is needed to grasp some objects, like an electric drill, a brick and a cylinder like object as shown in Figure 4.2. We acquired our own datasets of these objects in such a way that we can apply a reconstruction method that generates 3D models of each object. Those models are used later to test pose estimation in different scenarios.

3D Reconstruction. We put the object on a rotating platform and rotate the object at 6 degrees/sec to acquire the RGB and depth images of object with the RGB-D sensor.

Evaluation of pose estimation. The object can be present on a table or held in hand by an operator in front of humanoid for grasping. We acquired two types of datasets to test. One, a dataset which had object on a table with changing poses. Second, a dataset in which object is held in hand by a person and changing the pose of object to estimate the pose.

Ground-Truth. For the quantitative evaluation of pose estimation, we have used a commercial Motion capture system (Mo-cap) to recover the ground truth data of the pose of the objects. Mo-cap is a process of digital recording of object's or people's movements. An example is shown in Figure 4.1. It is widely used in entertainment industry (such as film making, gaming), sports, military, medical, validation of computer vision and robotics among others. There are two types of mo-cap system : 1)optical and 2)non-optical. We used optical mo-cap system

to recover ground-truth data with passive markers. Optical mo-cap system uses many number of cameras (2 to 100s) to capture the object from different angles (track the markers), then to determine the pose of the object. Passive markers are covered with a reflective material to reflect the light source. We need at least three markers to be visible for mo-cap system to recover the three degree orientation of the object. Stereometric techniques correlate common tracking points on the tracked objects in each image and use this information along with knowledge concerning the relationship between each of the images and camera parameters to calculate position of object.

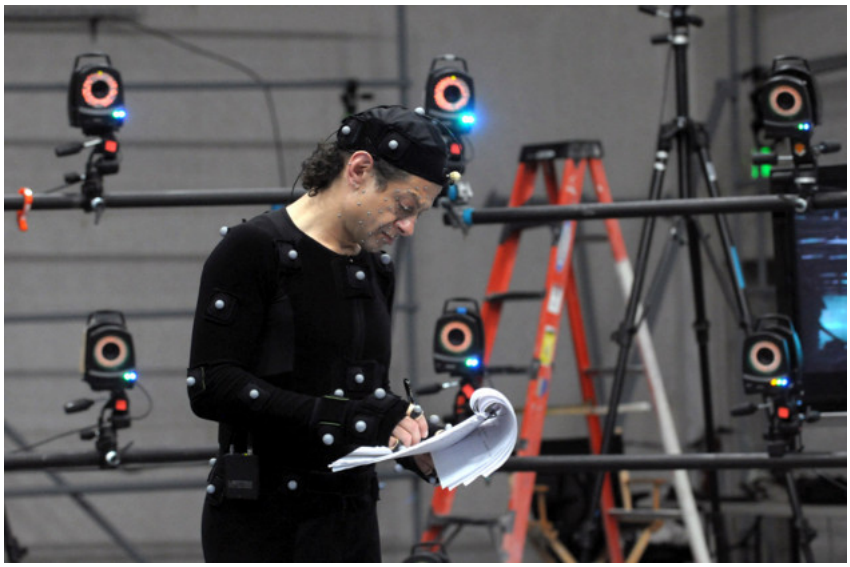
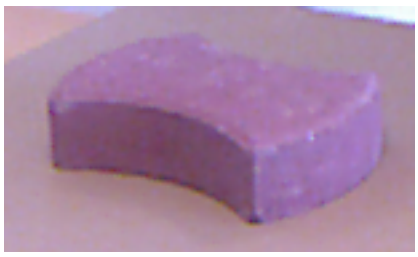


Figure 4.1: Mo-cap system. Source [4]

4.3 Qualitative Evaluations

As mentioned in the Section 1.4, this project of pose estimation is oriented as a module that is used over two different robotic platforms, each with a different sensor: 1)Humanoid 'Pyrene', which has an embedded Orbec Astra Pro 3D sensor and 2) UAV which has a Intel Realsense Euclid sensor on board. We have also tested this approach on Microsoft Kinect v1, for initial testing before integrating on the robotic platforms.



(a) Brick

(b) Cylinder

(c) Drill

Figure 4.2: Objects for pose estimation

4.3.1 Results of pose estimation of objects acquired using Microsoft Kinect v1

The pose of three different objects (brick, drill and yellow cylinder), acquired using Microsoft Kinect sensor, is estimated as detailed in Section 3.2, by aligning the respective object models created in Section 3.1 to the objects in the scene.

We have tested our pose estimation approach on two scenarios, viz., 1) the object is placed on a table (a plane) and 2) the object is held in hand by a person.

Object - Brick

The pose estimation of the brick with partial occlusion (from a hand) is shown in Figure 4.3. We can see the red color point cloud depicting the 3D model of the brick which is aligned with the point cloud of brick in the scene. ICP results in a proper alignment of the model - and the object in scene, we can recover the final transformation from the ICP which is the pose estimation of the object.

We had observe during rapid movements of the object in the scene that the alignment of the model and the object in scene for pose estimation - takes more

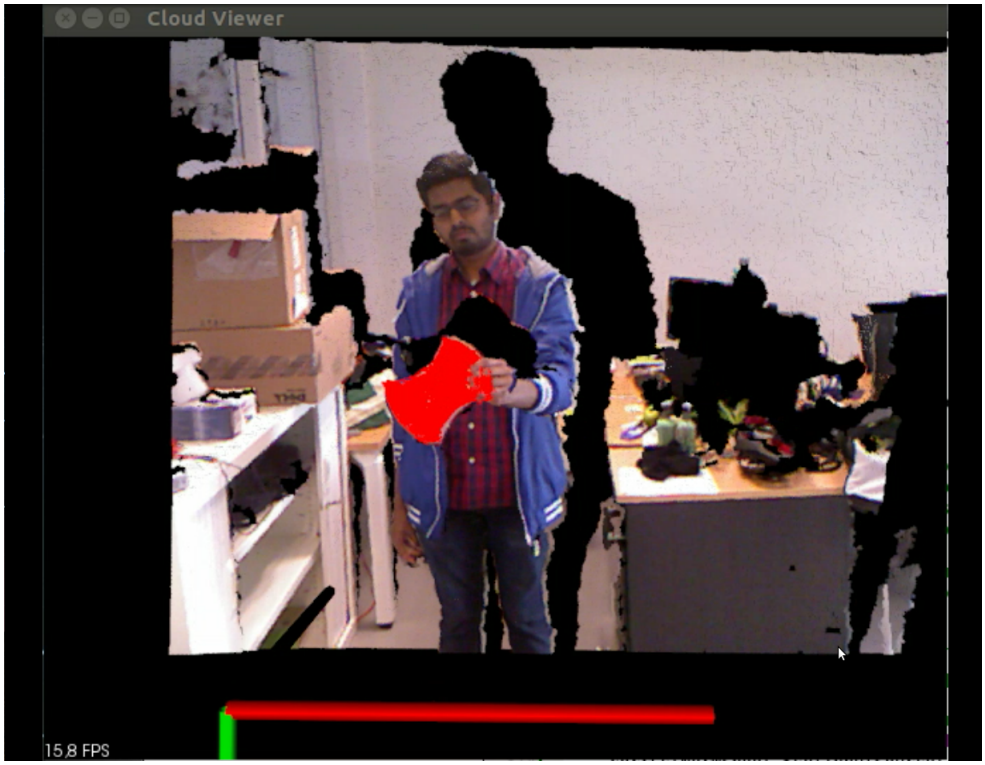


Figure 4.3: Pose Estimation of brick with hand occlusion

time than usual as the object needed to be initially aligned by SCA-IA.

During the major occlusion of the brick (only small part of brick is visible) the ICP fails to align the model as shown in Figure 4.4. We can also see in the Figure 4.4 that the brick in the scene and the model is realigned after the major occlusion.

A short video of pose estimation of brick (held by hand) is provided as supplemental material at this link[1].

A short demonstration of pose estimation of brick on the table is provided as a supplemental material at this link[2].

Object - Yellow Cylinder

The proper alignment between the model and object results in a accurate pose estimation as shown in Figure 4.5.

[1] https://www.youtube.com/watch?v=HoIW_qzToWU&feature=youtu.be
[2] https://www.youtube.com/watch?v=tJQIqIBMq-4&feature=youtu.be

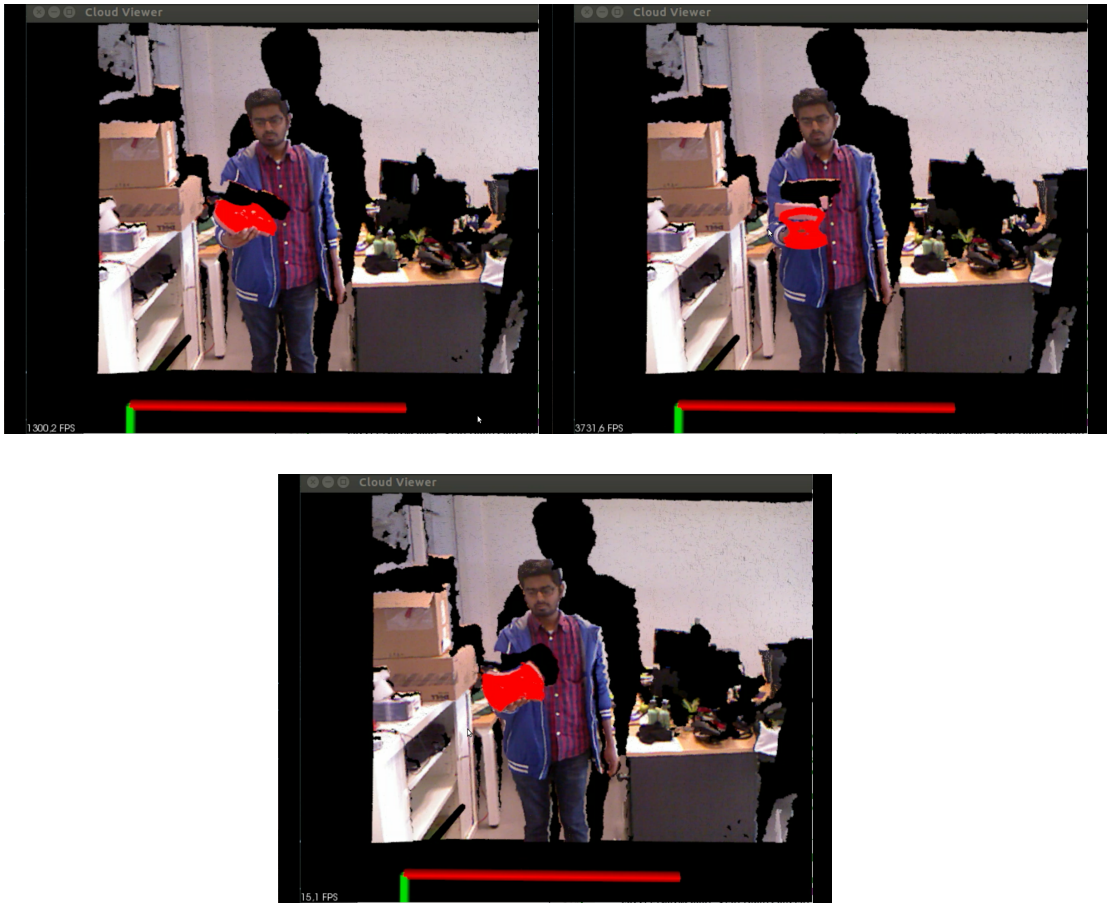


Figure 4.4: Pose Estimation of Brick with ICP before major occlusion (top - left) , during major occlusion (top - right) and after major occlusion (bottom).

Figure 4.5: Pose Estimation of yellow cylinder

In spite of been occluded with both hands, we can see a correct alignment of the object with the model in Figure 4.6. In the case of the brick, when there are rapid movements of the object, the alignment takes more time than usual as it needs to undergo the process of initial alignment. We also observed that the pose estimation of the yellow cylinder is bad when the object is occluded in majority.

Figure 4.6: Pose estimation of yellow cylinder with partial occlusion

A short demonstration of the pose estimation of the yellow cylinder is provided as a supplemental material at this link[3].

Object - Drill

The alignment of the drill on a plane and its model is shown in Figure 4.7. The pose is estimated when the drill is held in hand as seen in Figure 4.8.

We observed some problems with the alignment of drill with the model when this is viewed with major occlusion (only some part of drill is viewed in sideways) as shown in Figure 4.9.

A short demonstration of pose estimation of drill is provided as a supplemental material at this link[4].

[3] https://www.youtube.com/watch?v=OPh9xTyCC4c&feature=youtu.be and https://www.youtube.com/watch?v=L1omC3XFNWA&feature=youtu.be

[4] https://youtu.be/VY4g3-xFZSM and https://www.youtube.com/watch?v=ZPO-CSAq5tI&feature=youtu.be

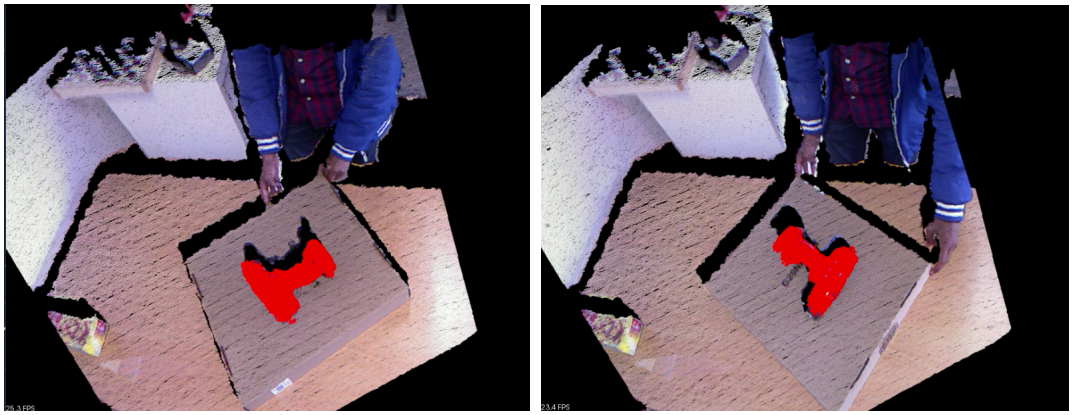


Figure 4.7: Pose estimation of a drill placed on a plane

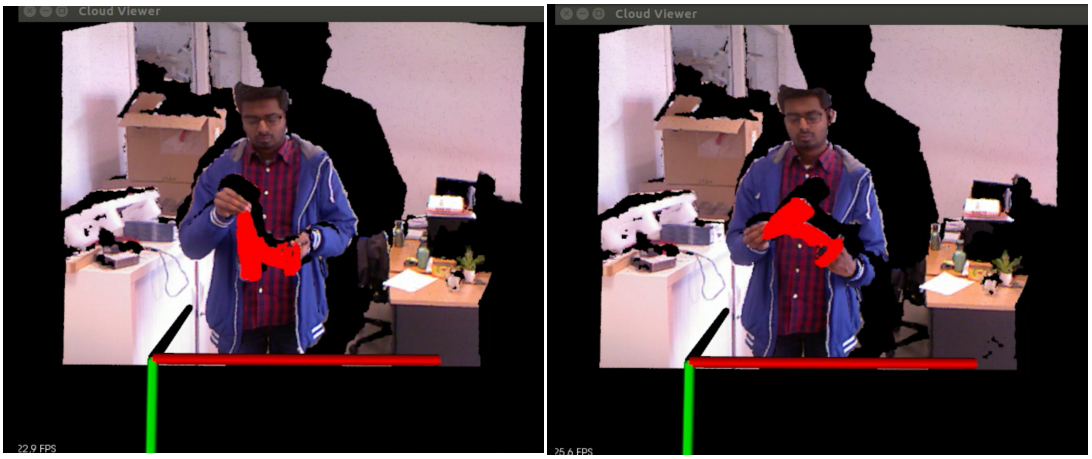


Figure 4.8: Pose estimation of a drill held in hand

4.3.2 Results of pose estimation of objects acquired using Orbec Astra Pro

The Orbec Astra Pro 3D sensor is embedded on the robot Pyrene [46], which is connected to a remote computer on WiFi network. Due to this type of connection, there is communication latency between robot and computer to recover the ROS topics related to point clouds. Instead, we tried to recover the depth image and we convert it into a point cloud using intrinsic parameters. Due to latency, the RGB and depth images are not synchronized, so we could not use the color information in the creating of the point clouds.

Figure 4.9: Improper alignment of drill with the model during major occlusion

Object - Brick

The pose estimation of a brick placed on a plane is shown in Figure 4.10 and brick held in hand is shown in Figure 4.11. We observed that, if the brick is partially viewed the model would not properly align with the brick, which leads to wrong pose estimation.

A short demonstration of pose estimation of brick is provided as a supplemental material at this link[5].

Object - Yellow Cylinder

The pose estimation of yellow cylinder is shown in Figure 4.12. When the yellow cylinder is occluded in majority the model could not align with the object as seen in this video[6].

[5] https://www.youtube.com/watch?v=8ptkc-3vFcw&feature=youtu.be and https://www.youtube.com/watch?v=R8IOfDtlKPA&feature=youtu.be

[6] https://www.youtube.com/watch?v=HcS91xX4Naw&feature=youtu.be

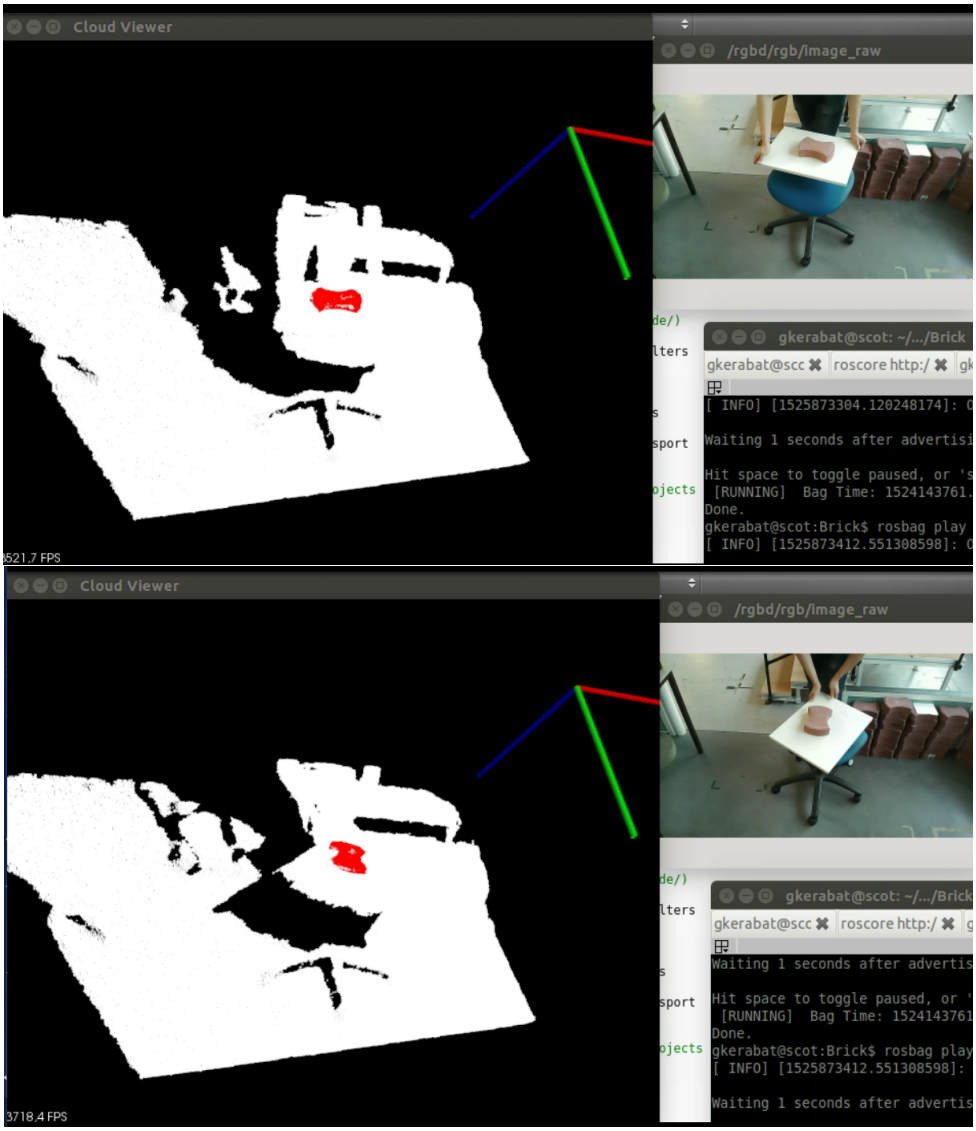


Figure 4.10: Pose estimation of brick on plane

Object - Drill

The pose estimation of drill held in hand is shown in Figure 4.13. A short demonstration of pose estimation can be seen at this link[7].

[7] https://www.youtube.com/watch?v=Hhuj3Cwn7hs&feature=youtu.be

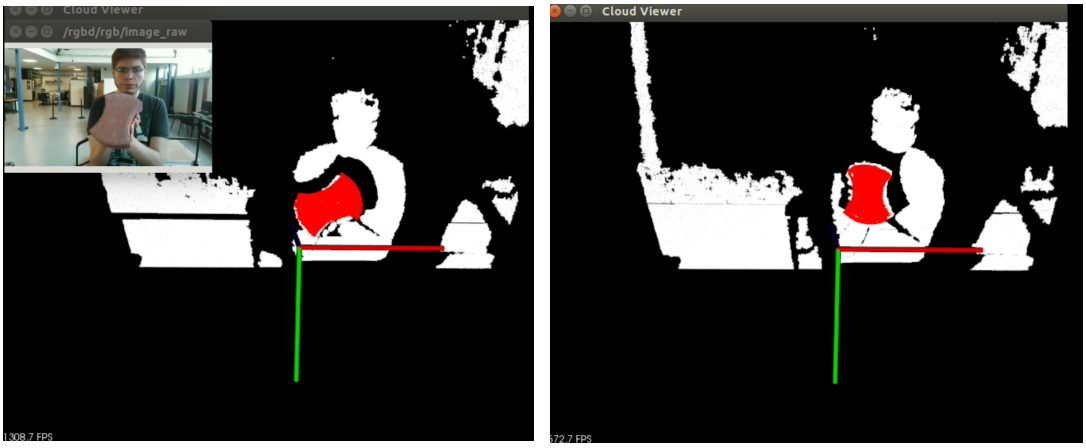


Figure 4.11: Pose estimation of brick held in hand. We use only the depth information to estimate the pose. A color image of the scene is depicted at the top left.

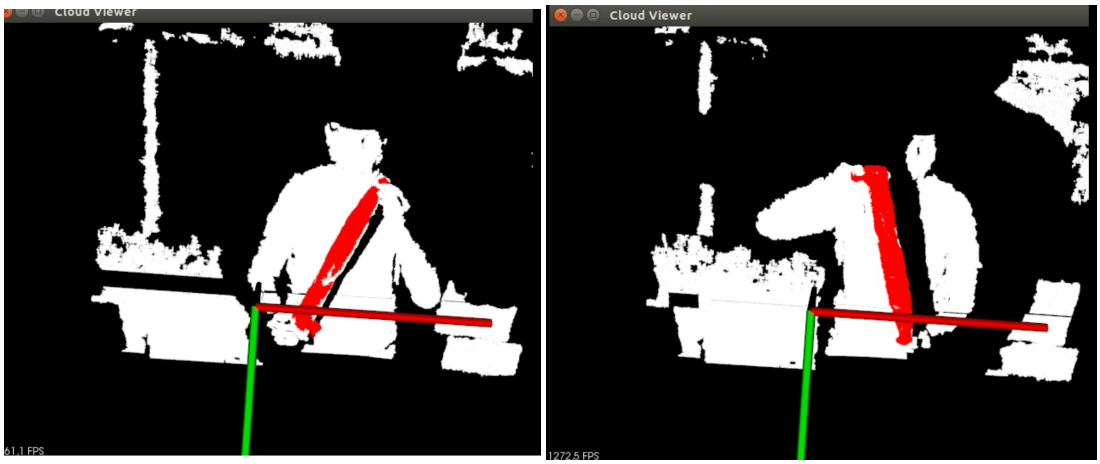


Figure 4.12: Pose estimation of yellow cylinder held in hand

4.4 Quantitative Evaluations

The average time taken by the various modules in the approach is as shown in Table 4.1. The first frame need to have initial alignment, this takes around 1.3 sec to process and followed by approximately 2 frames per second (fps) processing the rest of frames.

For the quantitative evaluation of pose estimation, we have used a commercial Motion capture system (Mo-cap) [4] to recover the ground truth data of the pose of the objects. In order to record the ground truth data, we place markers on

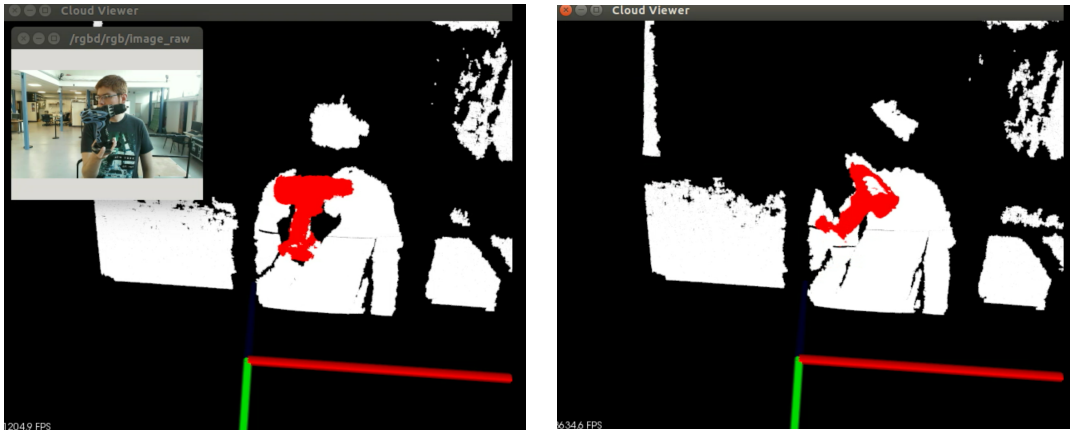


Figure 4.13: Pose estimation of drill held in hand

Table 4.1: Average time taken by each module

Module	Time (in ms)
Convert to Point Cloud	10
Table Segmentation	200
SCA-IA	800
ICP	350
Total (with SCA-IA)	1360
Total (without SCA-IA)	560

the object. We create a rigid body for the markers placed on the object in the Mo-cap system. We need to make sure that the markers placed on the object does not satisfy the rule of symmetry and at least 3 markers should be visible to the Mo-cap cameras at any point of time to recover the pose of the data.

The Mo-cap system provides the pose of the object in its coordinate reference system. Our algorithm estimates the pose in 3D sensor coordinate system, therefore, we need to find the transformation from sensor to Mo-cap frame of reference in order to compare the estimated pose with the ground truth data.

We use the 3D model of the object to find the transformation between sensor and Mo-cap coordinate system. First, we transform the 3D model of object by the estimated pose of object, to align the model to the object in the scene. Second, we also transform 3D model of object by the pose from the Mo-cap system, to align the model to object in scene. Then, we have two 3D models of objects with different orientation, one with respect to 3D sensor coordinate

system and other with respect to Mo-cap coordinate system. We estimate the rigid body transformation between the two orientation of same model by using singular value decomposition (SVD) and one-to-one correspondences between the models to find the transformation between sensor and Mo-cap coordinate system. We can compute the product (of rotation matrix) of the obtained transformation and Mo-cap pose estimation to get the ground truth pose in 3D sensor coordinate system to calculate the error between estimated and ground pose.

We report the errors in roll, pitch and yaw angles [2] of the pose estimated using our approach and the ground truth in Table 4.2. The errors are a bit higher than what we visually see in qualitative evaluations (Section 4.3), because of the fact that the landmarks are not placed exactly on the object and also due to ambiguity caused by symmetry of objects.

Table 4.2: Root Mean Square (RMS) error of orientation angles of different objects

Object	(Error in degrees)		
	Roll	Pitch	Yaw
Brick	12.09	3.12	11.89
Drill	7.93	7.94	5.42
Yellow Cylinder	16.11	13.84	7.64

4.5 Conclusion

We evaluated the approach both qualitatively and quantitatively. The results were presented. This approach works well for pose estimation in most of the cases except when the object suffers major occlusion. The important scenario considered is when objects are placed on table. Even when multiple objects are placed on table, this approach is able to estimate pose of specified object with the help of ICP.

Chapter 5

Conclusion

5.1 Conclusion

Using a RGB-D camera, an approach for pose estimation is proposed. This project is aimed to develop a pose estimation module, that is to be integrated on two robotic platforms: 1)Humanoid Robot 'Pyrene' and 2)UAV, with two different depth sensors on board. This brings us the challenges of communication latency, occlusion and noise in the data. The humanoid robot is intended to manipulate specific objects such as drill, brick and a small cylindrical object and UAV is intended to manipulate brick in the shape of cuboid.

This project solves the pose estimation of object with a two phase approach: 1)Offline phase - 3D model of the object is generated using ICP and 2)Online phase - model is aligned with the scene using an initial guess from SCA-IA algorithm, then further refined using ICP. Our approach uses the depth data taking the advantage of invariance to illumination and independence from object texture. We have tested this approach, both qualitatively and quantitatively using Microsoft Kinect v1, before the integration on the robotic platforms. The results are presented in Chapter 4. Our approach works well in case of partial occlusion of the object, but it is not robust in case of major occlusions of the object. We could achieve a near real-time system with 2 fps for pose estimation, given the model as a *priori*.

5.2 Future Work

The offline phase for 3D model generation can be improved to work online to simultaneously learn the model and estimate its pose, with a probabilistic pose prediction of object. We can detect the pose of multiple objects at the same time further refining this approach. We could improve the segmentation of the object and scene using the color information for more robust alignment of model and scene. We can improve the object recognition pipeline for more robust object recognition. We can add more rejectors (e.g., rejection of self-occluded normals) to reject the false matching pairs to increase the speed and robustness of the system.

Bibliography

[1] Euclidean cluster extraction. http://www.pointclouds.org/documentation/tutorials/cluster_extraction.php.

[2] Euler angles. https://en.wikipedia.org/wiki/Euler_angles.

[3] Fast approximate nearest neighbor search (flann). https://www.cs.ubc.ca/research/flann/.

[4] Motion capture (mo-cap). https://en.wikipedia.org/wiki/Motion_capture.

[5] Openni ros package. http://wiki.ros.org/openni_launch.

[6] Robot operating system. www.ros.org.

[7] The stanford 3d scanning repository. http://graphics.stanford.edu/data/3Dscanrep/.

[8] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 585–592, Nov 2011.

[9] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, Jan 2003.

[10] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, Sept 1987.

[11] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.

[12] M. Bigas, E. Cabruja, J. Forest, and J. Salvi. Review of cmos image sensors. *Microelectronics Journal*, 37(5):433 – 451, 2006.

[13] R. Brgier, F. Devernay, L. Leyrit, and J. L. Crowley. Symmetry aware evaluation of 3d object detection and pose estimation in scenes of many parts in bulk. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2209–2218, Oct 2017.

[14] G. Bugmann, M. Siegel, and R. Burcin. A role for robotics in sustainable development? In *AFRICON, 2011*, pages 1–4, Sept 2011.

[15] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 2724–2729 vol.3, Apr 1991.

[16] Witold Czajewski and Krzysztof Koomyjec. 3d object detection and recognition for robotic grasping based on rgb-d images and global features. *Foundations of Computing and Decision Sciences*, 42(3):219 – 237, 2017.

[17] Christophe Doignon, Florent Nageotte, Benjamin Maurin, and Alexandre Krupa. Pose estimation and feature tracking for robot assisted surgery with medical imaging. 2007.

[18] B. Drost and S. Ilic. 3d object detection and localization using multimodal point pair features. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*, pages 9–16, Oct 2012.

[19] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *CoRR*, abs/1411.4734, 2014.

[20] G. C. Fernandez, S. M. Gutierrez, E. S. Ruiz, F. M. Perez, and M. C. Gil. Robotics, the new industrial revolution. *IEEE Technology and Society Magazine*, 31(2):51–58, Summer 2012.

[21] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[22] F. Furrer, M. Wermelinger, H. Yoshida, F. Gramazio, M. Kohler, R. Siegwart, and M. Hutter. Autonomous robotic stone stacking with online next best object target pose planning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2350–2356, May 2017.

[23] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2188–2202, Nov 2011.

[24] E. Garcia, M. A. Jimenez, P. G. De Santos, and M. Armada. The evolution of robotics research. *IEEE Robotics Automation Magazine*, 14(1):90–103, March 2007.

[25] Frederic Garcia, Djamila Aouada, Hashim Kemal Abdella, Thomas Solignac, Bruno Mirbach, and Björn Ottersten. Depth enhancement by fusion for passive and active sensing. In Andrea Fusiello, Vittorio Murino, and Rita Cucchiara, editors, *Computer Vision – ECCV 2012. Workshops and Demonstrations*, pages 506–515, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[26] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1524–1531, May 2014.

[27] Jonathan M. Huntley Harshana G. Dantanarayana. Object recognition in 3d point clouds with maximum likelihood estimation. *Proc.SPIE*, 9530:9530 – 9530 – 7, 2015.

[28] Ruotao He, Juan Rojas, and Yisheng Guan. A 3d object detection and pose estimation pipeline using RGB-D images. *CoRR*, abs/1703.03940, 2017.

[29] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient response maps for real-time detection of textureless

objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, May 2012.

[30] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *2011 International Conference on Computer Vision*, pages 858–865, Nov 2011.

[31] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke. Registration with the point cloud library: A modular framework for aligning in 3-d. *IEEE Robotics Automation Magazine*, 22(4):110–124, Dec 2015.

[32] Radu Horaud. A short tutorial on three-dimensional cameras. http://perception.inrialpes.fr/~Horaud/Courses/pdf/Horaud_3Dcameras_tutorial.pdf, 2013.

[33] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, Sep 1993.

[34] Intel. Intel euclid development kit. https://www.euclidcommunity.intel.com/#about, 2018.

[35] Itseez Intel Corporation, Willow Garage. Open source computer vision library. https://opencv.org/, 2018.

[36] K. Lai, L. Bo, X. Ren, and D. Fox. Sparse distance learning for object recognition combining rgb and depth information. In *2011 IEEE International Conference on Robotics and Automation*, pages 4007–4013, May 2011.

[37] S. Li, K. N. Ngan, R. Paramesran, and L. Sheng. Real-time head pose tracking with online face template reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1922–1928, Sept 2016.

[38] JORGE FRANCISCO MADRIGAL DIAZ, Frédéric Lerasle, and André Monin. 3D Head Pose Estimation enhanced through SURF-based Key-Frames. In *WACV 2018*, 2018 IEEE Winter Conference on Applications of

Computer Vision (WACV), page 9p., Reno, Nevada, United States, March 2018.

[39] M. Martinez, A. Collet, and S. S. Srinivasa. Moped: A scalable and low latency object recognition and pose estimation system. In *2010 IEEE International Conference on Robotics and Automation*, pages 2043–2049, May 2010.

[40] F. Michel, A. Kirillov, E. Brachmann, A. Krull, S. Gumhold, B. Savchynskyy, and C. Rother. Global hypothesis generation for 6d object pose estimation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 115–124, July 2017.

[41] Microsoft. Microsoft kinect xbox 360. https://msdn.microsoft.com/en-us/library/jj131033.aspx, 2018.

[42] L. P. Morency, J. Whitehill, and J. Movellan. Generalized adaptive view-based appearance model: Integrated framework for monocular head pose estimation. In *2008 8th IEEE International Conference on Automatic Face Gesture Recognition*, pages 1–8, Sept 2008.

[43] Chavdar Papazov and Darius Burschka. An efficient ransac for 3d object recognition in noisy and occluded scenes. In Ron Kimmel, Reinhard Klette, and Akihiro Sugimoto, editors, *Computer Vision – ACCV 2010*, pages 135–148, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[44] H. Pota, R. Eaton, J. Katupitiya, and S. D. Pathirana. Agricultural robotics: A streamlined approach to realization of autonomous farming. In *2007 International Conference on Industrial and Information Systems*, pages 85–90, Aug 2007.

[45] L. Prives. An eye for detail. *IEEE Women in Engineering Magazine*, 3(2):19–20, Dec 2009.

[46] PAL Robotics. Talos : High performance humanoid robot. https://www.pal-robotics.com/en/products/talos/.

[47] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, May 2009.

[48] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3391, Sept 2008.

[49] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162, Oct 2010.

[50] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, May 2011.

[51] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robot. Auton. Syst.*, 56(11):927–941, November 2008.

[52] R. H. Taylor. A perspective on medical robotics. *Proceedings of the IEEE*, 94(9):1652–1664, Sept 2006.

[53] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim. Latent-class hough forests for 3d object detection and pose estimation. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 462–477, Cham, 2014. Springer International Publishing.

[54] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1385–1391, Oct 2004.

[55] D. Voth. A new generation of military robots. *IEEE Intelligent Systems*, 19(4):2–3, Jul 2004.

[56] K. Wang, G. Zhang, and H. Bao. Robust 3d reconstruction with an rgb-d camera. *IEEE Transactions on Image Processing*, 23(11):4893–4906, Nov 2014.

[57] Xiaoqin Wang, Y. Ahmet ekerciolu, and T. Drummond. A real-time distributed relative pose estimation algorithm for rgb-d camera equipped visual sensor networks. In *2013 Seventh International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–7, Oct 2013.

[58] Wikipedia contributors. Motion capture — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Motion_capture&oldid=828154525, 2018. [Online; accessed 11-May-2018].

[59] Wikipedia contributors. Robot — Wikipedia, the free encyclopedia, 2018. [Online; accessed 10-May-2018].

[60] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis. Single image 3d object detection and pose estimation for grasping. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3936–3943, May 2014.