SAFE AND COLLABORATIVE NAVIGATION & INTERACTION WITH MOBILE MANIPULATORS IN DOMESTIC APPLIANCE TEST LABS

Presentata da: Wendwosen Bellete Bedada

Coordinatore Dottorato

Michele Monaci

Supervisore

Gianluca Palli

Esame finale anno 2022

ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

Abstract

Department Electrical, Electronic and Information Engineering - DEI

Doctor of Philosophy

Safe and Collaborative Navigation & Interaction with Mobile Manipulators in Domestic Appliance Test Labs

by Wendwosen Bellete BEDADA

Safe collaboration between a robot and human operator forms a critical requirement for deploying a robotic system into a manufacturing and testing environment. In this dissertation, the safety requirement for is developed and implemented for the navigation system of the mobile manipulators. A methodology for human-robot coexistence through a 3d scene analysis is also investigated. The proposed approach exploits the advance in computing capability by relying on graphic processing units (GPU's) for volumetric predictive human-robot contact checking. Apart from guaranteeing safety of operators, human-robot collaboration is also fundamental when cooperative activities are required, as in appliance test automation floor. To achieve this, a generalized hierarchical task controller scheme for collision avoidance is developed. This allows the robotic arm to safely approach and inspect the interior of the appliance without collision during the testing procedure. The unpredictable presence of the operators also forms dynamic obstacle that changes very fast, thereby requiring a quick reaction from the robot side. In this aspect, a GPU-accelarated distance field is computed to speed up reaction time to avoid collision between human operator and the robot. An automated appliance testing also involves robotized laundry loading and unloading during life cycle testing. This task involves Laundry detection, grasp pose estimation and manipulation in a container, inside the drum and during recovery grasping. A wrinkle and blob detection algorithms for grasp pose estimation are developed and grasp poses are calculated along the wrinkle and blobs to efficiently perform grasping task. By ranking the estimated laundry grasp poses according to a predefined cost function, the robotic arm attempt to grasp poses that are more comfortable from the robot kinematic side as well as collision free on the appliance side. This is achieved through appliance detection and full-model registration and collision free trajectory execution using online collision avoidance.

Acknowledgements

This journey is about to end and so many taught cross mind at this critical junction. It was a long journey, full of happy and sad moments. Sometimes unexpected things happen, but every single moment gave me something I could never have in any other way. I feel like I'm a better person than I was at the beginning. At this point, I would like to thank many people that helped me in various ways to reach this milestone.

First and foremost I am extremely grateful to my supervisor Prof. Gianluca Palli for his invaluable continuous support, patience and encouragement. I would also like to thank my colleagues at the Laboratory of Automation and Robotics, University of Bologna: Ismayil, Alessio, Kevin, Davide and Roberto. Those small talks during lunch breaks and coffee times were therapeutic and highlights of the day densely filled with work.

A special gratitude goes to my wife, Sefanit Abay. Thank you for carrying all the responsibility of raising our son. Your strength has kept me going even during the most dire days. We both endured the challenges of the past three years, apart from each other. I am eagerly looking forward to make up for these days. I also would like to thank my family for following up on me, share some of my worries and pray for me. I would also thank my friend Yilma Tadesse for his encouragement.

Joshua 1:5

Contents

List of Figures

List of Tables

Dedicated to: ***Sefanit & Yedidia***

Chapter 1

Introduction

1.1 Overview

There is a growing interest from the manufacturing companies side, to automate production line, to minimize cost and improve product quality and standard. In this regard, Robotic systems that perform automated tasks has risen, in recent years, as one of the main topics within Industry 4.0 and a relevant factor in firms adopting the main concepts of Smart Factory and Intelligent Manufacturing. Their successful development and implementation is now offering competitive advantage for the manufacturers in terms of efficiency and spans various application domains including manufacturing, assembly, warehouse automation and logistics.

Within the scope of this project, household appliance manufacturers have stepped up to deploy mobile manipulators to automate their product reliability test laboratories. This emerging interest targets Domestic appliances like washing machines which are designed and produced in mass scale to be introduced to the market. These manufacturers have to test all functional capabilities of all new products before shipping them to future customers. Today these tests require the appliance manufacturer to employ and train many people to perform these checks in time to be market effective. The company that proposed this project aims to automate this procedure. The project aim is to implement all the functionalities for the robot to carry out the same tasks that a human operator in a similar environment would be asked to perform. The said operator would follow the given tasks such as opening the lid of the washing machine, loading the to-be washed load, closing the lid of the washing machine, operating and selecting the program for the washing cycle, and activating the washing machine. Trivial tasks to the human operator such as walking to the washing machine without hitting any obstacles and seeing and understanding the information of the washing machine control panel, are not trivial to the robot. Moreover, the robotic system needs to be integrated with the tests and results databases to be informed about the test matrix and to report new data about the correct functionalities of the appliances under test. This dissertation focuses on the development of application for mobile manipulator in an industrial automatic laundry by addressing research challenges related to safety, real-time performance, manipulation and interaction in the test facility. The system object of this development is schematized in Fig 1.1.

To move in a dynamic environment such as the test laboratory, in which the presence of humans is not excluded, the robot controller needs to be implemented in a way that it will navigate to the requested workspace of the washing machine, interact with it safely, and then perceive and program the washing cycle as required. All this actions need to be executed ensuring the safety of the system with respect to humans as the primary goal. Therefore, the implementation of functionalities like safe-navigation, and safe-interaction are extremely important. Development of real-time

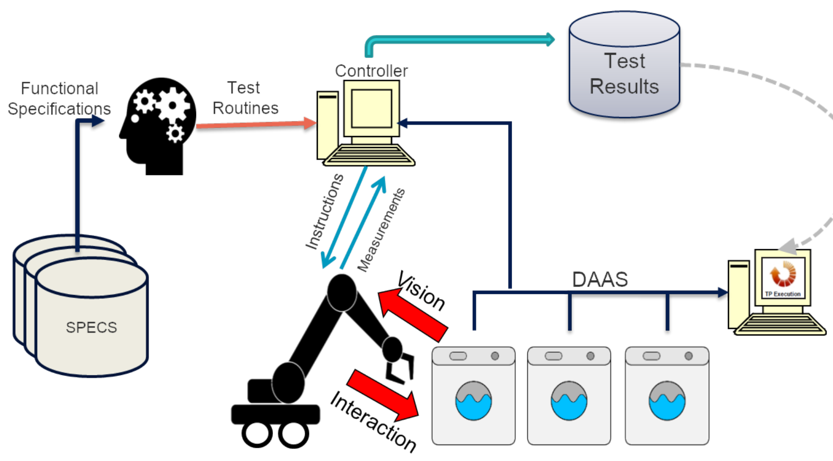


FIGURE 1.1: The overall system architecture.

software to detect collisions is also vital when working in dynamic environments.

The main issues investigated in this thesis and described in this report are:

- The general control framework for the mobile manipulators oriented to improve the mobility and the manipulation capabilities in restricted environments preserving safety;
- The localization of the washing machines in the laboratory environment;
- The interface between the robot and the waching machine, in terms of both interface with the internal data, the interpretation of the appliance console and the detection of its functional components by the vision system;
- The perception and manipulation of clothes to execute washing machine loading and unloading tasks.

This work was mainly developed using the ROS framework, due to the availability of powerful libraries for ROS. Within ROS, all the software was written in C++ or in Python. Matlab and the robotics toolbox are used to ease the controller testing and debugging phase, but all the controller components are then translated in C++. LabView is used to acquire data from the washing machine via DAAS interface since the company provided libraries for this software to interface with the washing machine control hardware.

1.2 Thesis Organization

Robots, nowadays are widely exploited to perform tasks commonly found in the manufacturing sectors that require speed or are dangerous and repetitive for humans. In this regard, home appliance manufacturing industry has shown a strong demand for rapid and efficient ways to test new products to meet the growing customers demand for high performance appliances and compete in the market. The application of mobile manipulators in domestic appliance test facility is useful to relieve human operators from time consuming tasks, speed up the tests and improve efficiency. However, alleviating the concern for safety in an environment shared by human operator and a robot plays essential part in the development of automated systems. The industrial robot safety standard given in (Shea, 2013) outlines safety requirements in a human-robot collaborative workspace. Speed and separation monitoring criterion aims at preventing physical impact from occurring by maintaining a static safe separation distance between the robot and any humans walking through the collaborative workspace. These safety standards usually has to be broken down into requirements on which robotic applications are built. The development of robotic systems that fulfills this safety requirement also involves integration of complex software and hardware subsystems in a reliable way. Even more, it is necessary to also tailor the development to fit in to the particular work environment where humans and the robot collaborate. Therefore, a safe robotic system considers the following aspects:

- Formulation of Safety requirement from the safety standard given in (Shea, 2013).
- Development of robotic Software and hardware systems to implement these requirements.
- Work environment and other practical considerations.

In Chapter 2, safety requirement definition and proposed implementation solutions are presented for the navigation system of the mobile manipulator. The proposed system aims at improving the safety of human operators that share the workspace with the robotic platform which is a common scenario of test laboratories. A deep learning algorithm is used for the human detection and pose estimation, while the integration between a conventional motion planning algorithm with a fast 3D collision checker has been implemented as a global planner plugin for the ROS navigation stack. With the twofold objective of improving safety and saving energy in the battery-powered mobile manipulator used in this project, the problem of minimizing the overall kinetic energy is addressed through a properly designed task priority controller, in which the manipulator inertia matrix is used to weight the joint speeds while satisfying multiple robotic tasks according to a hierarchy designed to interact with the appliances while preserving the safety of the human operators.

The manipulator mounted on a mobile base has to also insure safe movement in the presence of human operators and obstacles. However, the delay in updating the environment scene, computing a 3D distance map and generating control command plays crucial factor in utilizing real time collision avoidance for robotic applications. Most of the Algorithms and models for collision avoidance are based on the CPU computing power which are limited by the sequential execution nature. The dynamic environment representations, for instance formed using an octree data structure such as Octomap (Hornung et al., 2013), updates the scene hierarchically, therefore they are much slower than the sensor frame rate. If multiple sensor sources are involved in updating the scene, the scene updating process become even slower. This limitation has inspired the search for approximate models, for example those that exploit a coarse representations, thereby trading on the accuracy of the collision checking and distance computation. Even with approaches that approximate the environment and the robot, the close interaction remains critical, mainly due to the need for high resolution information (distance and contact queries) when the robot and the obstacle come in the near vicinity of each other. With regard to Self-collision avoidance, it is practically impossible to perform distance computation between full CAD models such as meshes in real time due to their complex data structure. Therefore, the links of the robot are approximated using primitive geometries for collision and distance queries. It is therefore apparent that collision checking and distance queries benefits from the massive parallelization available in GPUs. A high-speed reactive collision avoidance exploiting gpu's is detailed in this chapter 3. The following scientific problems are therefore addressed:

- Integration of Reactive collision avoidance with fast GPU-based distance field;
- A simplified robot model capable of exploiting the GPU computation is proposed;
- Practical demonstration of real-time reactive collision avoidance with live sensor data in a dynamic 3D scene.

A typical robotic application may include range of tasks including tracking of end-effector pose while also avoiding singular configurations, obstacle and self-collision. Even more, not all this tasks have equal importance or needed all the time. This requirement creates the need for a multiple task kinematic controller with a possibility of task prioritization.

In addition to safety and collision avoidance from the robot side, detection, segmentation, and full model registration of the appliance (including the interior) and it's various parts is also important during interaction which requires 6d model registration. The application of shape registration nowadays spans multiple areas of robotic assistant tasks. A practical problem usually emerges due to partial information acquired by range finders or 3D vision sensors because of occlusions. This problem becomes particularly relevant when the robot task objective is to interact and navigate around a large objects. In this regard, the shape alignment problem that takes into account large objects with very limited visibility and significant occlusions utilizing an octree data structure on the GPU is discussed in chapter 4. The proposed algorithm relies on the offline computed 3D model of the object and an initial estimation of it's pose using a deep learning technique to detect key features of the object, in order to improve the accuracy and the speed of the registration process. The final aligned pose is achieved by computing the iterative closest point algorithm on GPU utilizing octree, starting from the initial estimated pose. To highlight the application of the proposed method, autonomous robotic tasks requiring interaction with washing machine is discussed. Finally, the performance in terms of both speed and accuracy of the different implementations of the algorithm on the CPU and GPU, as well as with and without augmented octree neighbourhood search is provided.

During the accelerated life cycle testing of an appliance, a robotic system that is capable of detecting and manipulating a laundry autonomously is required. Being deformable objects by their nature, the grasping of clothe like objects require a special treatment. Even more, the grasping is performed on a large set of clothes pilled up in a container for loading or in the drum for extraction, which poses the challenge of identifying stable grasp. In chapter 5, vision algorithm purely based on 3d point-clouds is applied to four laundry handling tasks, namely the cloth bin picking, drum picking, washing machine door recovery picking and floor recovery picking. This leads toward a complete task set for robotized laundry. The core of the approach is based on a wrinkledness measure for the initial identification of good graspable areas in clothes. Additional refinement steps are implemented to fit spline curves on the major wrinkles to estimate possible grasping frames. The approach is validated experimentally performing the full insertion and extraction operations using the computed grasping poses. A 7-DoFs robotic arm and a task-priority based controller allowing for real-time collisions avoidance with the appliance and the robot itself are utilized in the experiments.

Chapter 2

Robotic System for Domestic Appliance Test Automation

2.1 Introduction

Home appliance manufacturing industry has shown a strong demand for rapid and efficient ways to test new products to meet the growing customers demand for high performance appliances and compete in the market. The application of mobile manipulators in domestic appliance test facility is useful to relieve human operators from time consuming tasks, speed up the tests and improve efficiency.

Accelerated Life Test (ALT) is a commonly used technique to acquire reliability information quickly (Escobar and Meeker, 2006). In particular, appliance manufacturing industry conducts ALT on the small fraction of appliances statistically sampled from the production line and are supervised by a human operators. The test is carried out simultaneously on these appliances with the aim of testing their performances through the simulation of the whole life by the accelerated cycles conducted in reduced time. The utilization of robotic system comes in hand in these types of industrial setup where a considerably long and continuous test is required for product verification.

However, alleviating the concern for safety in an environment shared by human operator and a robot plays essential part in the development of automated systems. The industrial robot safety standard given in (Shea, 2013) outlines safety requirements in a human-robot collaborative workspace. Speed and separation monitoring criterion aims at preventing physical impact from occurring by maintaining a static safe separation distance between the robot and any humans walking through the collaborative workspace. The safeguards for speed and separation monitoring criterion in a collaborative scenario includes, among other thing, a human detection system and a reduced speed mode for the robot depending on the separation distance when human operator is detected in a close vicinity to the robot (Marvel, 2013). The development of robotic systems that fulfills this safety requirement involves integration of complex software and hardware subsystems in a reliable way. Even more, it is necessary to tailor the development to fit in to the particular work environment where humans and the robot collaborate.

It is also reasonable to consider mobile manipulators for these test labs to fully take advantage of the above mentioned points due to the extended workspace they offer while also minimizing the cost associated to the test as sensors and the end-effector are mounted on a mobile agent and can be shared for all appliances (Choi, Jin, Shin, et al., 2008; Hamner et al., 2010). In fact, deployment of mobile manipulators in manufacturing and testing floor has been investigated extensively in the

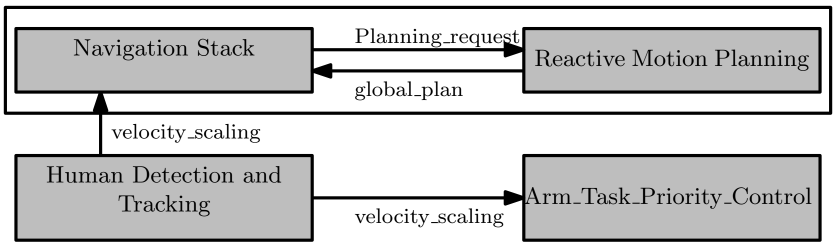


FIGURE 2.1: Proposed safety system.

past. Important works include flexible test platform for household appliance (Cesetti et al., 2010), clinical laboratories (Choi, Jin, Shin, et al., 2008) and assembly tasks (Hamner et al., 2010).

This work presents safety requirements associated with human-robot collaboration in a domestic appliance test lab and proposes a safety oriented robotic system with energy efficient arm controller. This work utilizes off-the-shelf (OTS) mobile manipulators, the TIAGO robot from PAL Robotics, equipped with redundant arm and RGB-D camera to be deployed in the life test laboratory of washing machines. A deep learning approach is adopted to detect the humans in the scene, providing to the robot controller their pose estimation. Moreover, a global planner plugin for the ROS navigation stack has been implemented through the integration between a conventional motion planning algorithm and a Graphic Processing Unit (GPU) based fast 3D collision checker. To improve both the safety of the system and to save energy in the battery-powered mobile manipulator, the overall kinetic energy of the robot is minimized by the task priority controller through the use of the manipulator inertia matrix to weight the joint speeds. The task priority controller ensures also to satisfy multiple robotic tasks to perform proper interaction with the appliances while preserving the safety of the human operators. Simulations and preliminary experiments have been carried out to evaluate the system capabilities and develop all the required functionalities, and the results are reported in this chapter. Work on this project has been presented in (Bedada et al., 2020) and the following key contributions are reported:

- A complete description of the system and safety requirement.
- A collaborative navigation strategy blending standard navigation stack, gpu-based 3d collision checking and the safety requirement.
- Development of a full simulation model for evaluating robot-appliance interaction and various aspect related navigation and size of the passage (corridor) and appliance operation.
- Development of a full simulation model for evaluating robot-appliance interaction and various aspect related navigation and size of the passage (corridor) and appliance operation.
- Experimental validation of the system, to evaluate its effectiveness in a different scenarios that arise in the test lab.

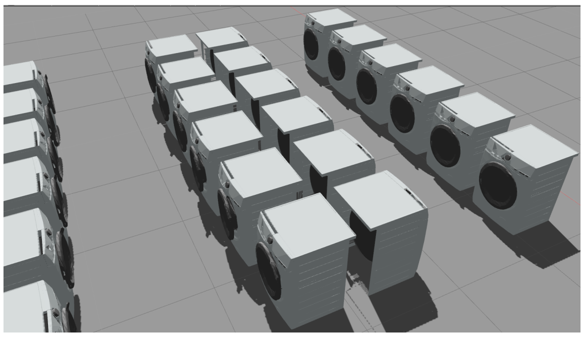


FIGURE 2.2: Test Lab. layout in Gazebo Simulation Environment.

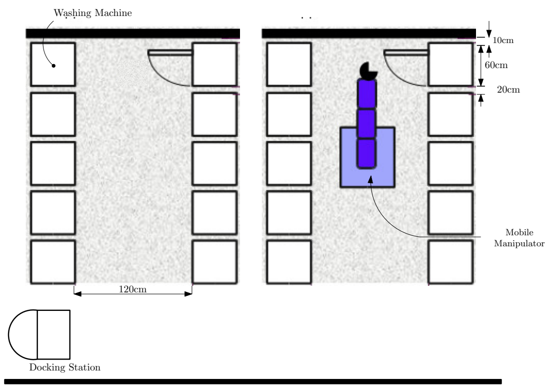


FIGURE 2.3: Lab. layout dimensions.

2.2 System Description and Safety Requirement

Apart from detecting obstacles, the robotic system should be able to distinguish the presence of human operators in the environment. This allows the robot the enforcement of a safe velocity during the workspace sharing with the human operators. The overall system structure shown in Fig. 2.1 considers safety in a distributed manner in such a way that both the robotic arm and the mobile base fulfills these requirements.

The navigation system must achieve a 3D collision checking during the motion planning for navigation in a partially structured indoor environment and manage unpredictable changes due to the presence of human operators and unknown obstacles. Moreover, the navigation system should also be able to predict future collisions and demonstrate reactive motion planning capabilities. After reaching the goal, the system should be capable of detecting the pose of the washing machine as a whole and the different parts of the appliance such as knobs, buttons and door handle in particular. Finally, the system should also manipulate and physically interact with the products while avoiding collision and unsafe maneuvers. The list of tests that the robot has to set or perform on the appliance must be loaded to the robot in real-time through a wireless interface from external system. Depending on the size of the product (small, medium and large), the system should flexibly reconfigure itself to interact with each appliance.

For the purpose of testing the proposed system, a TIAGO mobile manipulator (Pages, Marchionni, and Ferro, 2016) has been used. However, different commercial robotic solution will be proposed in the future for deployment of the developed application in the appliance test lab. TIAGO is completely based on the Robot Operating System (ROS), that facilitates access to hardware and hides the complexities

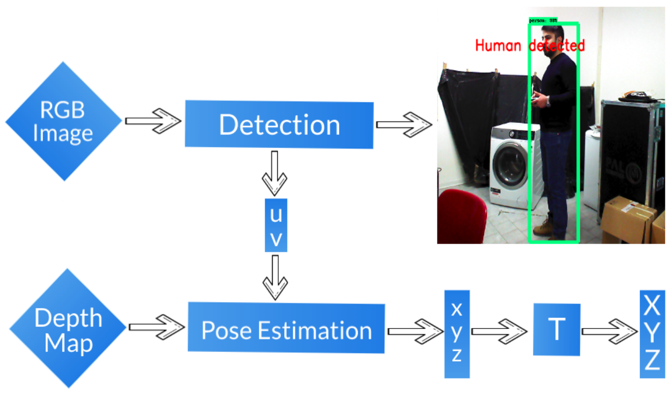


FIGURE 2.4: Human Detection and Pose Estimation Procedure.

of transferring data between components (Quigley et al., 2009).

The test lab contains large number of appliances that are arranged in a row with pair of rows facing each other and separated by a gap that allow appliance transportation as indicated in Fig. 2.2. This should be considered as an additional constraint for motion planning of the arm and the base of the robot.

2.3 Human Detection and Pose Estimation

Almost all robotic systems relies on a robust perception system to demonstrate autonomous and intelligent behaviors. Perception algorithms have been dominated mainly by computer vision algorithms that involves usage of descriptive features for object detection and recognition which usually involved human operator in the past. The emergence of Deep Learning (DL), however, introduced the concept of end-to-end learning where feature selection and extraction as well as classification is performed in one big learning network (O'Mahony et al., 2019).

We have applied human detection and estimation of the human pose in order to provide the required safety and to eliminate the risk of any damage to human body in the working space of the robot. To implement human detection, Single Shot Detector (SSD300) (Liu et al., 2016), which has been adopted to MobileNetV2 (Sandler et al., 2018) neural network, was used in this step. The neural architecture has previously been trained on COCO dataset (Lin et al., 2014a).

The human pose with respect to the robot mobile base must be estimated for safety reasons. After detection step, the corresponding points on the image frame $[u, v]^T$ given by detection algorithm, are used as input to the pose estimation step, which produces $[x, y, z]^T$ coordinates of the corresponding points with respect to the camera optical frame as output. For that goal, the depth map of the scene is obtained using the RGB-D camera of the robot. Data encoded on the depth map represents the distance of the arbitrary point with respect to camera optical frame. After human detection and pose estimation in the robot workspace, the estimated values of the pose are used to implement safety measures for a shared worskpace.

2.4 Navigation Module

Navigation capability of mobile robots plays crucial role in the success of an autonomous mission. Different variations of software and hardware have been used in the navigation systems with motion planning, Localization and Control forming the basic structure. Potential field based navigation of robots have gained prominence in the papers of early 90's (Khatib, 1986; Schneider and Wildermuth, 2003), (Dozier et al., 1998). However, due to trap in local minima and oscillation in the presence of obstacles and narrow passages, potential field methods are exposed to be ineffective in these situations (Koren and Borenstein, 1991). To overcome this, the dynamic window approach to navigation (Fox, Burgard, and Thrun, 1997) is proposed that considers periodically only a short time interval in to the future when computing the next steering command to avoid the complexity of the general motion planning problem. Admissible (translational and rotational) velocities within the dynamic window that maximize an objective function is selected. The objective function includes a measure of progress towards a goal location, the forward velocity of the robot, and the distance to the next obstacle on the trajectory (Brock and Khatib, 1999; Fox, Burgard, and Thrun, 1997). This approach that overlays global planner with the dynamic window approach have been implemented on interactive tour-guide robots for museums (Burgard et al., 1998; Thrun et al., 1999) with good success. In (Marder-Eppstein et al., 2010), a navigation algorithm that consider three dimensional obstacle data in an efficient way is proposed. This work employs three dimensional voxel grid to encode the robot's knowledge about the environment as free, occupied or unknown.

Collision detection and path monitoring consumes up to 90 percent of computation time during robot motion planning task (Pan, Lauterbach, and Manocha, 2010a). This implies that fast collision detection plays central role in enforcing reactive behavior that interleaves planning, continuous motion validation and execution. The advent of fast computing capability has improved the development fast collision checking and re-planning algorithms that allows to realize robust navigation system for mobile robots. Computations related to collision detection and proximity queries on GPU can be more advantageous because of their multi-thread capability. A comparison of mesh based collision checkers with CPU implementation (Pan, Chitta, and Manocha, 2012a) and voxel based approach on GPU which discretize the robot and the environment is given in (Hermann et al., 2013). The voxel based implementation of collision detection on GPUs has shown improved results compared to the mesh based CPU implementation in the case of multiple queries of collision checking.

In this work, we propose a new approach that utilizes GPU-based collision checker given in (Hermann et al., 2014a) for motion planning in the framework of ROS navigation stack. A sampling based motion planner and a GPU-based state validity checking implementation forms the global planner plugin. A sampling based motion planning implementation in Open Motion Planning Library (OMPL) (Sucan, Moll, and Kavraki, 2012) is utilized as the global planner. The main difference between the proposed approach and the standard navigation stack is that we abandoned the use of 2D-costmap for environment representation. Instead, a 3D voxelmap created from depth sensors is utilized to facilitate a more reliable global planning that considers a three dimensional scene.

A local planner that follows the global plan is combined with an independent path monitoring implementation utilizing a swept volume of the robot used to detect future collisions and leveraging on the computational capabilities of modern GPUs. The architecture of the proposed GPU-Based navigation algorithm is reported in Fig. 2.5. During the execution of a plan, the collision checker monitors the global plan until the goal is reached. This is achieved by creating a voxelized swept volume of the robot along the planned path and detecting colliding voxels

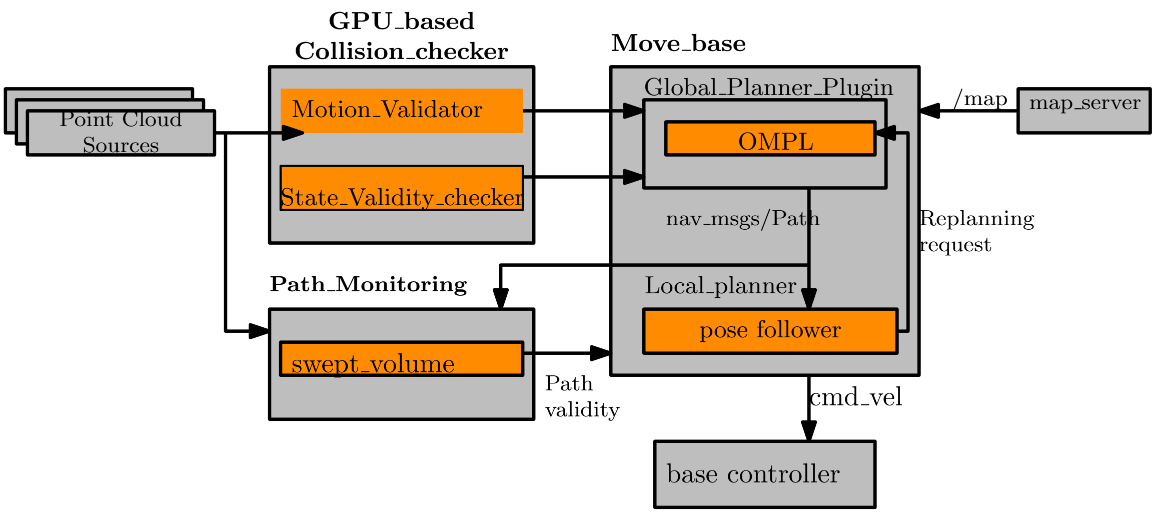


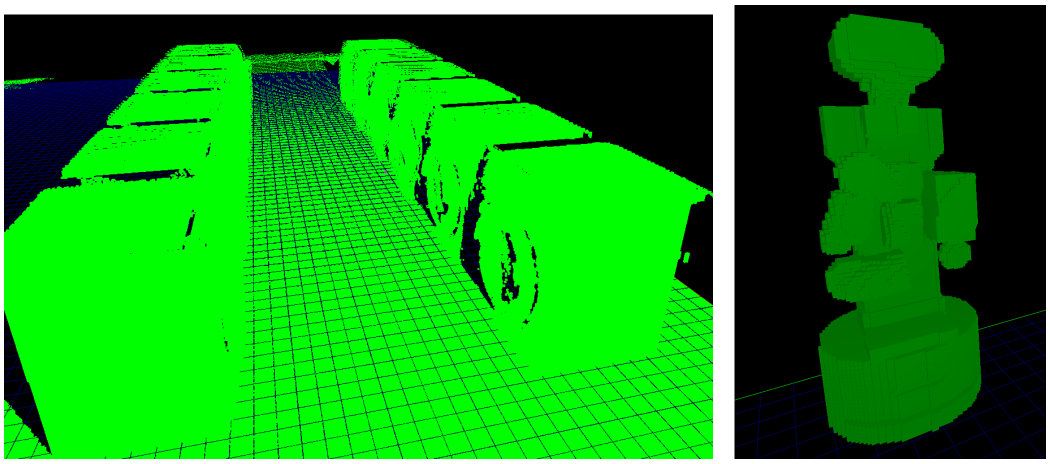
FIGURE 2.5: GPU-Based Navigation Stack.

in the environment map. The swept volume is generated by keeping the results of path planning in the Voxel-Map to create a virtual corridor for the robot. The generation of the voxelized representation of the robot is performed offline and inserted into the map. The core idea of sampling based motion planning (SBMP) is the approximation of the connectivity of the sampled search space with a graph structure. The variations in the different types of SBMP originates from how the sampling of the planning space is performed to create the vertices of the graph and the strategies employed to generate the edges from it. The sampled states and connection between them forms the vertices and edges of the graph respectively. It obvious, however that the graph should constructed only from the valid vertices and edges which denote the valid states and path segments connecting them. Validating the sampled states and the connection among them is where the fast collision checking comes into play.

OMPL facilitate the integration with external collision checking algorithms through two abstract classes named *StateValidityChecker* and *MotionValidator*. The former allows the planner to evaluate the validity of states while the later validates motions between two specified states, effectively creating valid vertices and segments.

The *StateValidityChecker* class implements a routine that takes sampled state space and return the validity information. The path segment between the valid portion of the sampled state space is also validated to form the edges through a routine in the *MotionValidator* class of OMPL. The GPU-based collision checking is used to implement state validity checker for OMPL using a swept volume technique given in (Hermann et al., 2013) in which the virtual model of the robot in Fig. 2.6(b) is inserted into the robot map at every pose we are checking for validity. The validity of the states is verified by comparing it to the corresponding occupancy of the voxels in the environment map. If the same voxel location in the robot and environment map is occupied, the particular state associated with the robot pose carrying the colliding voxels will be invalid. In case of motion validity checker, After valid states are sampled, a linearly interpolated path segment between two valid state creates the swept volume by inserting the discrete model of the robot along this path segment.

The environment map carries a discrete representation about the occupancy of the scene in a probabilistic way. To create this environment map, a transformed and filtered pointcloud from the robot's onboard RGB-D camera is inserted and updated on the GPU memory for fast collision detection. To perform the transformation of the



(a) Environment map in GPU voxels visualizer. (b) TIAGO robot voxel representation.

FIGURE 2.6: Voxel-based representation of the robot and working environment.

pointcloud from camera frame of the robot to world frame, an Adaptive Monte Carlo Localization (AMCL) algorithm is used to localize the robot thereby transforming the points to the world frame in real-time. The environment map shown in Fig. 2.6(a) contains voxelized representation of two rows of washing machine partially seen from the TIAGO on-board camera.

As highlighted above, collision checking and monitoring are performed virtually using the environment scene and the voxelized robot model. The mesh model of the robot is rasterized into a binary 3D voxel grid to create a binary voxel model of the robot using an offline software given in (Nooruddin and Turk, 2003a). This offline generated voxel representation of the robot will be inserted into a separate robot map according to the joint state and odometry information of the actual robot. By inserting the voxelized shape for each corresponding mesh surface on the robot, a full discrete representation of the robot is added into a robot map with same dimension as the environment map. The resolution of discretization of the mesh surfaces of the robot affect the memory usage and collision checking performance and should be selected carefully. The voxelized shape of the TIAGO robot given in Fig. 2.6(b) is used throughout this chapter.

The motion planning developed in this work takes into account the presence of human to enable the robot to operate differently from the basic obstacle avoidance problem to ensure the safety of humans and enforce collaborative behaviour during the interaction. To this end, a virtual cylindrical volume is inserted in to the map in real-time to allow the motion planner to plan safe path. During execution, the velocity of the robot is varied depending on the distance between the robot and the detected human operator in the environment as 1) Full Operational, 2) Human-Aware and 3) Danger mode. In the scenario where either human is not in the worskpace or the distance between the robot and human operator is greater than certain predefined threshold, the robot operates in the Full Operational mode exploiting its full velocity as generated by the local planner. In the Human-Aware operation mode, however, the robot is at a distance that can injure the human operator, if they come in contact. Thus, the robot operates in a reduced velocity mode proportional to the distance. Finally, the Danger mode sets both linear and rotational velocity to zero because of the overlapping of the human safety zone and the robot's footprint.

•

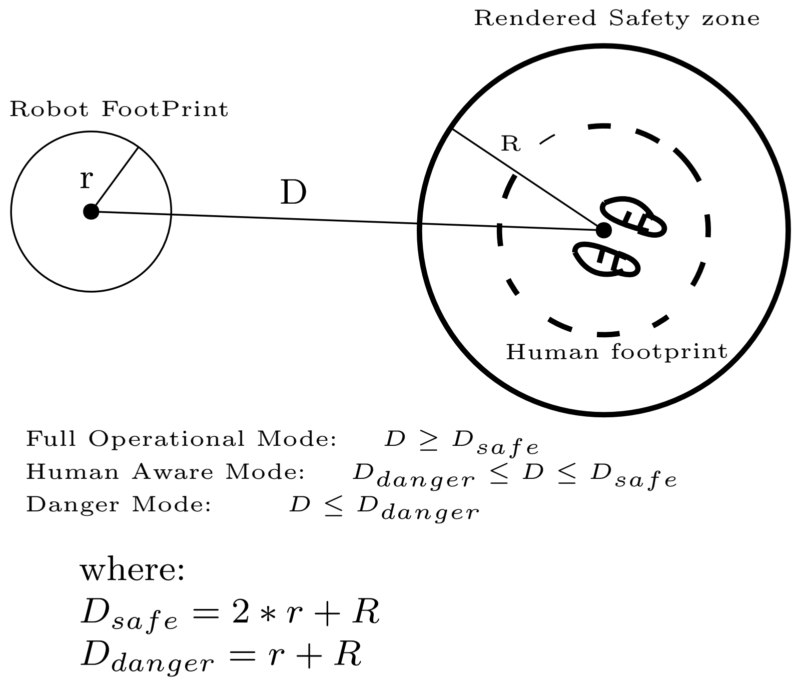

Rendered Safety zone
Robot FootPrint
r
D
R
Human footprint
Full Operational Mode: $D \geq D_{safe}$
Human Aware Mode: $D_{danger} \leq D \leq D_{safe}$
Danger Mode: $D \leq D_{danger}$
where:
$D_{safe} = 2 * r + R$
$D_{danger} = r + R$

FIGURE 2.7: Safety requirement description.

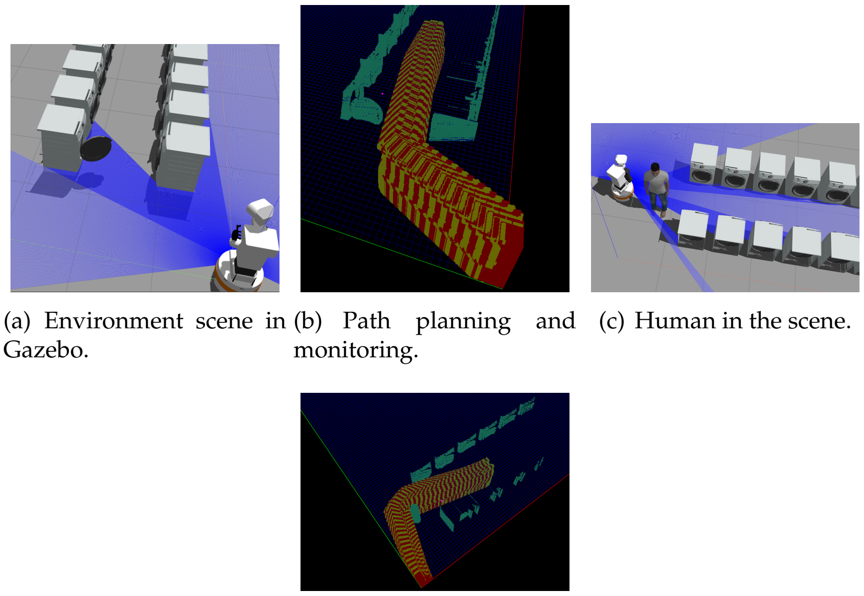
The robot safety features are implemented at both the global and local planner level inside the software architecture of the GPU-Based navigation stack. At the global planner level, the presence of human operator is accommodated by inserting and updating a virtual safety zone during motion planning. The planner considers a cylindrical volume larger than the footprint of the human operator as a safety region. This imposes additional planning constraints that insure no part of the 3D robot shape violates the safety zone around the human. The local planner is responsible to generate desired velocity commands to the robot base and thus, has a direct influences on how the robot executes the global plan. Therefore, safety related behaviours of the robot are implemented in the local planner as follow:

$$\begin{cases} \text{Full Operational Mode} & \text{if} \quad \left\|x - x_{\text{safety}}\right\| > D_{\text{safe}} \\ \text{Human-Aware Mode} & \text{if} \quad D_{\text{danger}} < \quad \left\|x - x_{\text{safety}}\right\| < D_{\text{safe}} \\ \text{Danger Mode} & \text{if} \quad \left\|x - x_{\text{safety}}\right\| < D_{\text{danger}} \end{cases} \tag{2.4.0.1}$$

where the x is the mobile robot's pose, x_{safety} is the center of the virtual safety volume around the human, D_{safety} is the minimum distance between the robot and the human before the local planner switches to the Human-Aware mode and D_{danger} the minimum safe distance between the human and robot.

2.5 Task Priority Control Framework

Task priority control enable the execution of several robotic tasks, such as singularity avoidance, joint limit avoidance, collision avoidance, joint speed limitation other than conventional end-effector position control, in a hierachical order, i.e. lower priority tasks do not influence on the behavior of higher priority ones (Simetti and Casalino, 2016a).



(a) Environment scene in Gazebo. (b) Path planning and monitoring. (c) Human in the scene.

(d) Cylindrical safety zone around the human operator.

FIGURE 2.8: GPU-based planning and monitoring: path planning with washing machine door opened and human-aware safety zone rendered in GPU memory.

Let's consider a robotic system with n Degrees of Freedom (DoFs), being $\mathbf{q} \in \mathbb{R}^n$ the vector of configuration variables, which control input is the desired value of configuration variable time derivative $\dot{\mathbf{q}}$, and a hierarchical set n_t of tasks where $\mathbf{x}_i \in \mathbb{R}^{m_i}$, $i = 1, \cdots, n_t$ is the vector of the i-th task variables, each of those can be represented as a function of $\mathbf{q}$, i.e $\mathbf{x}_i = f_i(\mathbf{q})$ and m_i is the dimension of the i-th task. The time derivative of the i-th task variables can be written as

$$\dot{\mathbf{x}}_i = \frac{df_i(\mathbf{q})}{dt} = \mathbf{J}_i\dot{\mathbf{q}} \tag{2.5.0.1}$$

where $\mathbf{J}_i \in \mathbb{R}^{m_i \times n}$ is the i-th task Jacobian. The solution of eq. (2.5.0.1) is the minimum norm solution of the minimization problem

$$\min_{\dot{\mathbf{q}}} ||\dot{\bar{\mathbf{x}}}_i - \mathbf{J}_i\dot{\mathbf{q}}||^2 \tag{2.5.0.2}$$

where $\dot{\bar{\mathbf{x}}}$ is a reference velocity vector of the i-th task. This solution is given by the generalized inverse matrix $\mathbf{J}_i^{\#}$ as:

$$\dot{\mathbf{q}} = \mathbf{J}_i^{\#}\dot{\mathbf{x}}_i + (\mathbf{I} - \mathbf{J}_i^{\#}\mathbf{J}_i)\dot{\mathbf{q}}_0 \tag{2.5.0.3}$$

where $(\mathbf{I} - \mathbf{J}_i^{\#}\mathbf{J}_i)$ is the projection matrix with image space equal to the null space of $\mathbf{J}_i$, $\dot{\mathbf{q}}_0 \in \mathbb{R}^n$ is an arbitrary vector of null-space joint velocities. The i-th task can be then controlled by a simple a proportional controller

$$\dot{\bar{\mathbf{x}}}_i = \mathbf{K}_i(\mathbf{x}_{di} - \mathbf{x}_i) \tag{2.5.0.4}$$

where $\mathbf{x}_{di}$ is the desired value of the i-th task variables, $\mathbf{K}_i \in \mathbb{R}^{m_i \times m_i}$ is the symmetric and positive definite task gain matrix. Adding the vector $\dot{\mathbf{q}}_0$ to the solution, as it is clear in eq. (2.5.0.3), allows the generation of internal motions in the kinematic chain without affecting the goal of the i-th task, i.e. reaching its vector space $\mathbf{x}_i$. Hence, eq. (2.5.0.3) can be used again to find $\dot{\mathbf{q}}_0$ the solution of the $(i-1)$-th task having lower priority. This concept will be applied recursively on all the tasks of the task framework, so lower priority tasks do not influence on the behavior of higher priority tasks. However, the generalized inverse matrix used to solve eq. (2.5.0.1) creates a discontinuity when a singularity of the matrix $\mathbf{J}_i$ is encountered. Another type of discontinuity appears during the activation and the deactivation of inequality control objectives of a given task. Inequality control objective are tasks requiring to constraints of the type $\mathbf{x}_i \leq \mathbf{x}_{Mi}$ or $\mathbf{x}_i \geq \mathbf{x}_{mi}$. When $\mathbf{x}_i$ needs to stay within an interval, two separate inequality objectives can be used to represent the problem. A relevant example of inequality task is the joint limiter. An inequality task becomes active only when its control objective is going to be violated. On the other hand, even though conventional equality control tasks, such as end-effector pose control, can be always considered active, the activation and deactivation feature enables to switch among different control objectives according to the task objectives. To deal with task activation and deactivation, a diagonal activation matrix $\mathbf{A}_i \in \mathbb{R}^{m_i \times m_i}$ can be considered for the i-th task. The j-th element on the diagonal of $\mathbf{A}_i$, namely $a_i(j)$, is given by:

$$a_i(j) = \begin{cases} 1 & \mathbf{x}_i \geq \mathbf{x}_{Mi} + b, \\ s_i(\mathbf{x}_i) & \mathbf{x}_{Mi} \leq \mathbf{x}_i \leq \mathbf{x}_{Mi} + b, \\ 0 & \mathbf{x}_i \leq \mathbf{x}_{Mi}. \end{cases}$$

where $s_i(x)$ is a sigmoid function given by

$$s_i(\mathbf{x}_i) = \frac{1}{2}(\cos(\frac{(\mathbf{x}_i - \mathbf{x}_{Mi})\pi}{b}) + 1) \tag{2.5.0.5}$$

where x_{Mi} is threshold and b is the buffer of the sigmoid. Thus, the original minimization problem (2.5.0.2) is replaced by the following

$$\min_{\dot{\mathbf{q}}} [||\mathbf{A}_i(\dot{\mathbf{x}}_i - \mathbf{J}_i\dot{\mathbf{q}})||^2 + ||\mathbf{J}_i\dot{\mathbf{q}}||^2_{\mathbf{A}_i(\mathbf{I}-\mathbf{A}_i)} + ||\mathbf{V}_i^T\dot{\mathbf{q}}||^2_{\mathbf{P}_i}] \tag{2.5.0.6}$$

where $\mathbf{V}_i^T$ is the right orthonormal matrix of the SVD decomposition of $\mathbf{J}_i^T\mathbf{A}_i\mathbf{J}_i = U_i\Sigma_i\mathbf{V}_i^T$ and $\mathbf{P}_i$ is a diagonal regularization matrix where each element $p_{(i,i)}$ is a bell-shaped function of the corresponding singular value of $\mathbf{J}_i$, or zero if the correspondent singular value do not exist and the notation $\|\cdot\|_{\mathbf{P}}$ indicates the weighted norm, i.e. $\|\dot{\mathbf{q}}\|^2_{\mathbf{P}_i} = \dot{\mathbf{q}}^T\mathbf{P}_i\dot{\mathbf{q}}$.

The solution of (2.5.0.2) can be then written as

$$\begin{aligned} \dot{\mathbf{q}} = & (\mathbf{J}_i^T\mathbf{A}_i\mathbf{J}_i + \mathbf{V}_i^T\mathbf{P}_i\mathbf{V}_i)^{\#}\mathbf{J}_i^T\mathbf{A}_i\mathbf{A}_i\dot{\mathbf{x}}_i \\ & + (\mathbf{I} - (\mathbf{J}_i^T\mathbf{A}_i\mathbf{J}_i + \mathbf{V}_i^T\mathbf{P}_i\mathbf{V}_i)^{\#}\mathbf{J}_i^T\mathbf{A}_i\mathbf{A}_i\mathbf{J}_i)\dot{\mathbf{q}}_0 \end{aligned} \tag{2.5.0.7}$$

To maximize also safety and power saving capabilities by means of the proposed task priority controller, the previously described control framework is further extended by including the robot inertia matrix $\mathbf{M}$ in the computation of the generalized pseudo-inverse. Hence, the kinetic energy enters directly in the minimization problem, thus reducing the motion of the robot DoFs associated to larger inertia and leveraging more on the lighter ones, thus reducing power consumption and improving battery life. By considering this different type of discontinuities, the extension of any priority levels with the initialization $\dot{\mathbf{q}}_0 = \mathbf{0}$, $\mathbf{Q}_0 = \mathbf{I}$, for $k = 1, \cdots, n_t$, to any priority levels is expressed in the following equations

$$\begin{aligned}\mathbf{W}_k &= \mathbf{J}_k\mathbf{Q}_{k-1}(\mathbf{J}_k\mathbf{Q}_{k-1})^{\#,\mathbf{A}_k,\mathbf{Q}_{k-1},\mathbf{M}}\\ \mathbf{Q}_k &= \mathbf{Q}_{k-1}(\mathbf{I}-(\mathbf{J}_k\mathbf{Q}_{k-1})^{\#,\mathbf{A}_k,\mathbf{I},\mathbf{M}}\mathbf{J}_k\mathbf{Q}_{k-1})\\ \mathbf{T}_k &= (\mathbf{I}-\mathbf{Q}_{k-1}(\mathbf{J}_k\mathbf{Q}_{k-1})^{\#,\mathbf{A}_k,\mathbf{I},\mathbf{M}}\mathbf{W}_k\mathbf{J}_k)\\ \dot{\mathbf{q}}_k &= \mathbf{T}_k\dot{\mathbf{q}}_{k-1}+\mathbf{Q}_{k-1}(\mathbf{J}_k\mathbf{Q}_{k-1})^{\#,\mathbf{A}_k,.,\mathbf{M}}\mathbf{W}_k\dot{\mathbf{x}}_k\end{aligned} \tag{2.5.0.8}$$

$$\mathbf{x}^{\#,\mathbf{A},\mathbf{q},\mathbf{M}} \triangleq (\mathbf{M}^{-1^T}\mathbf{x}^T\mathbf{A}\mathbf{x}+\eta\mathbf{M}^{-1}(\mathbf{I}-\mathbf{Q})^T(\mathbf{I}-\mathbf{Q})+\mathbf{V}^T\mathbf{P}\mathbf{V})^{\#}\mathbf{M}^{-1^T}\mathbf{x}^T\mathbf{A}\mathbf{A} \tag{2.5.0.9}$$

where $\mathbf{V}$ is the right orthonormal matrix of the SVD decomposition of $\mathbf{x}^T\mathbf{A}\mathbf{x}+\eta(\mathbf{I}-\mathbf{Q})^T(\mathbf{I}-\mathbf{Q})$ and η is a suitable damping coefficient.

In our application, we considered four different tasks: the end-effector position controller task, the joint velocity minimizer task, the joint limiter task and the singularity avoidance task. To implement the singularity avoidance task, we used the concept of the manipulability index proposed in (Yoshikawa, 1985). The idea is therefore to consider as control variable the smallest singular value of the robot Jacobian. Hence, by satisfying the last value, which is the smallest singular values, we are also guarantee the other singular values (Sverdrup-Thygeson et al., 2017). The joint limiter task is added in the hierarchy to ensure the safety of the robot. However, the joint velocity limiter task is added in order to reach the goal of the whole task priority while minimizing the joint speeds so the energy spent by the robot. Since TIAGO is battery powered, the joint velocity minimizer task, together with the inertia-weighted generalized pseudoinverse, increases also its battery life by reducing the overall system energy consumption.

2.6 Simulations and Experiments

2.6.1 Planning with GPU-based 3D Collision Checking

The modified navigation stack is tested in an environment with cluttered three dimensional environment scene. The test is conducted by creating an environment that has objects not detected by the laser such as open washing machine door and a suspended bar between boxes which can lead to the failure of navigation systems based on 2D obstacle rendering. For a simple comparison, two sampling based planners, namely RRT-star (Karaman and Frazzoli, 2011) and LBKPIECE1 (Bohlin and Kavraki, 2000), are chosen for the evaluation of the proposed GPU-based collision checking. Although they produced approximately identical path as shown in Fig. 2.8(b), planning time has improved when using control based motion planner such as LBKPIECE1. The detail of time during planning is shown with breakdown for state motion validation in Tab. 2.1.

The case of successful path monitoring is also demonstrated in Fig. 2.8(a) and Fig. 2.8(b) where the robot is able to plan and monitor it's path through a corridor with an open door washing machine. The path monitoring is able to detect the particular robot pose in the plan that is in collision with the environment ahead of the actual collision through the swept volume. This effectively prune out poses of the plan that are in collision at anytime during the execution. This, in turn, results in the capability to initialize re-planning from a more suitable pose which avoids the chance that the robot get stuck in case a new obstacle in encountered along the way. This approach has also been demonstrated on Tiago robot with in the lab, by creating unstructured

TABLE 2.1: comparison of RRT-star and LBKPIECE when applying GPU collision checker.

Pose and Motion validation time	LBKPIECE1	RRT-star
Total no. of Poses Inserted	674	44
Pose insertion time (ms)	3.54021	3.82593
Pose collision time (ms)	0.383763	0.463711
Poses collision	0.383763	0.463711
Total no. of motion Inserted	165	295
motion collision time (ms)	15.4182	145.363
Total Planning time in Sec.	5.188	25.48

environment requiring a whole body collision checking for the path planning task as demonstrated in this video[1].

2.6.2 Human-Aware Safety Region Rendering

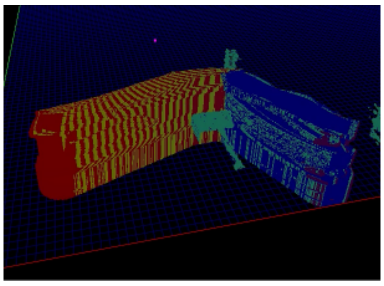
To demonstrate the proposed safety approach, the human detection and pose estimation algorithm is combined with the human-aware safety region rendering in GPU to ensure safety and collaborative behaviour. Even from partial view of the human operator, the related cylindrical safety zone can be inserted into the environment map and used during planning as shown in Fig. 2.8(d). This 3D-safety requirement ensures all parts of the robot(i.e. arm and mobile base) stay outside the region that collide with the safety zone during motion planning. The velocity of the robot during each execution cycle satisfies the safety requirement as shown in Fig. 2.10(b). The robot only reduces it's full operational velocity when it is in the human-aware mode and completely stops when the robot enters danger zone. Moreover, the validity of the remaining path segment is checked on every cycle to avoid collision and trigger re-planning ahead. The comparison of the velocity command with and without the safety constraint is shown in Fig. 2.10(a) and Fig. 2.10(b). It can be seen that the safety constraint reduced the peak velocities only in the region where human operator is in the vicinity. This allows full operational velocity when human operators are not in the scene or the distance between the operator and the robot is greater than D_{safe}.

2.6.3 Task priority control of the TIAGO arm

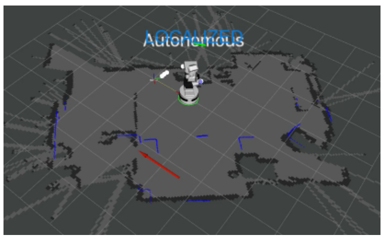
We used */arm_controller/command* interface to control the TIAGo arm in a task priority hierarchy. During our experiments, we considered 4 different tasks in the following order of priority: joint limiter task, end-effector position task, singularity avoidance task and velocity joint limiter task. After setting the different task parameters, the goal for TIAGo end-effector position control to reach the following pose (translation (x,y,z): [0.857 0.070 0.825], orientation (x,y,z): [3.033 0.079 -0.759]) is passed. In 2.6.3, we show the effects of the velocity minimizer task during the motion of TIAGo arm between from its initial position to the end-effector goal position. However, in 2.6.3, we present the influence of using the inertia-weighted generalized pseudoinverse on the kinetic energy of the TIAGo arm.

[1]https://drive.google.com/file/d/13rYR5F276MD4fuYXhRTD0U5TH384f8V2/view?usp=sharing

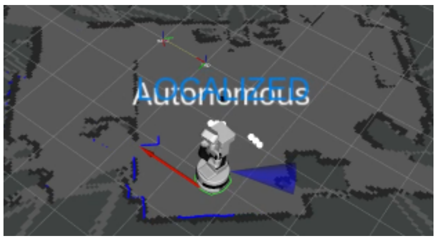
(a) The planning Environment.



(b) Final path with Swept volume.

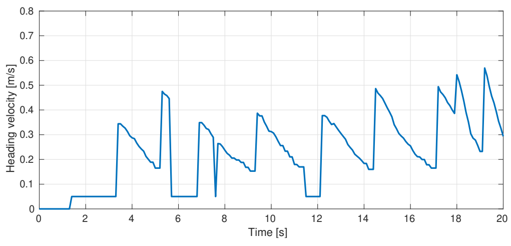


(c) Initial pose of the robot in the map.

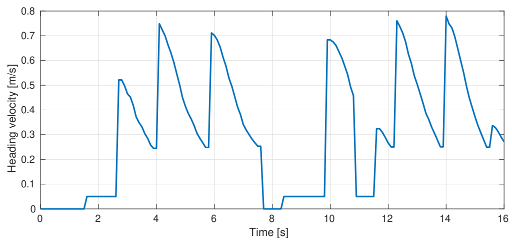


(d) Final pose of the robot in the map.

FIGURE 2.9: Modified navigation stack. In (a) The planning environment containing 3d obstacle(an extended bar undetectable by the laser scan) (b) Shows the final path planned with full body collision checking. (c) and (d) indicate the initial and final poses on a static map.



(a) Velocity command with safety constraints.



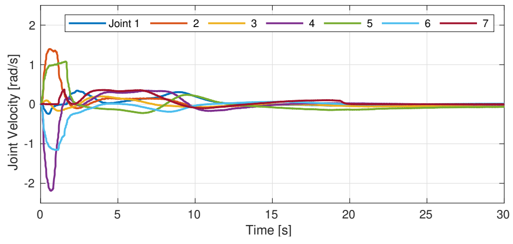
(b) Velocity command without safety constraints.

FIGURE 2.10: Linear velocity Plot with and without Safety Requirement.

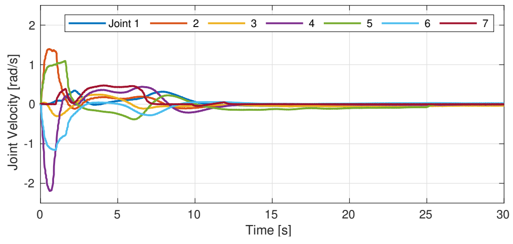
Effects of the Velocity Minimizer Tasks

Figure 2.11 compares the joint velocities of TIAGo arm for two different types of task priority control where the velocity minimizer task is present and absent. Below, we can see that adding the joint velocity minimizer task reduces the joint velocities: the joint velocities of different joints is lower when the velocity minimizer task is present 2.11(a) compared to when it is absent 2.11(b). This conclusion is clearly seen in Table 2.2 and 2.3. Table 2.2 shows how the root mean square (RMS) velocities of all the joints reduced when joint velocity minimizer task is added compared to the joint velocities without the minimizer task. Table 2.3 also indicate that the time required to reach a steady state velocities lower than 0.02 rad/s. A reduced joint velocities implies less energy consumption, which improves the battery life of the TIAGo robot.

Figure 2.13 shows different steps of the robot motion starting from its initial position and reaching the end-effector goal.



(a) Joint velocities with velocity minimizer task.



(b) Joint velocity without velocity minimizer task.

FIGURE 2.11: Effects of the velocity minimizer task during the TIAGo arm motion.

Joint	1	2	3	4	5	6	7
With	0.080	0.169	0.05	0.240	0.174	0.160	0.100
Without	0.082	0.172	0.065	0.243	0.180	0.165	0.114

TABLE 2.2: Effect on the velocity minimizer on the RMS value of joint velocities.

Effects of the Inertia Matrix in Task Priority Control

Figure 2.12 compares the kinetic energy of TIAGo's arm during the motion task with and without inertia-weighted generalized pseudoinverse. The task priority hierarchy considered the 4 different tasks that are already introduced and the robot gripper should reach the latter defined goal. It is clear that the robot kinetic energy is lower when the inertia-weighted pseudoinverse is used (see Figure 2.12). Though, this solution allows to save energy while guaranteeing.

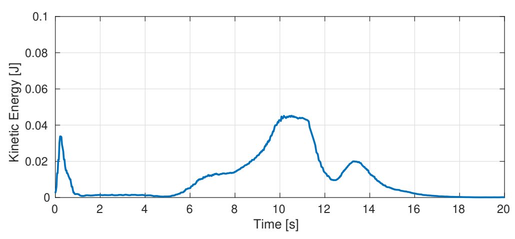
2.6.4 Interaction With The Appliance

All the proposed algorithms presented in this chapter were evaluated using the Tiago robot platform which is available in our lab. However, the final robotic system to be deployed in the test labs should be an industry grade robot. Therefore, all the evaluations related to the interaction with the washing machine e.g. opening appliance door, operating knobs or inspecting the interior of the drum are performed on a simulation platform developed for this purpose using rbkairos mobile base with ur10 arm from robotnik and universal robot respectively. The detail of the choice of this platform and other considered candidates are presented in chapter 5 together with the final procured hardware.

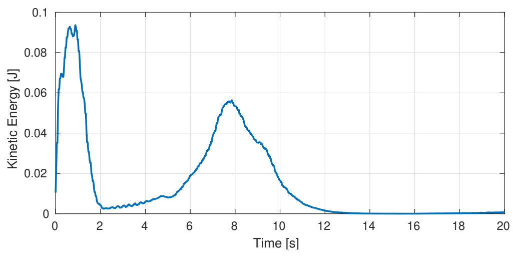
By utilizing the simulation model, the movement of the robotic arm inside the washing machine has been studied. From this, the size of the wrist of the ur10 arm has been identified to be restrictive to safely maneuver the drum, and to improve

Joint	1	2	3	4	5	6	7
With	10.50	12.71	10.90	12.62	25.30	13.83	12.09
Without	12.17	14.34	13.39	15.07	34.45	18.62	19.73

TABLE 2.3: Time required for each joint to reach a velocity less than 0.02 rad/s.



(a) Robot kinetic energy with inertia-weighted pseudoinverse.



(b) Robot kinetic energy without inertia-weighted pseudoinverse.

FIGURE 2.12: Kinetic energy of the TIAGo arm during the motion task with and without inertia-weighted generalized pseudoinverse.

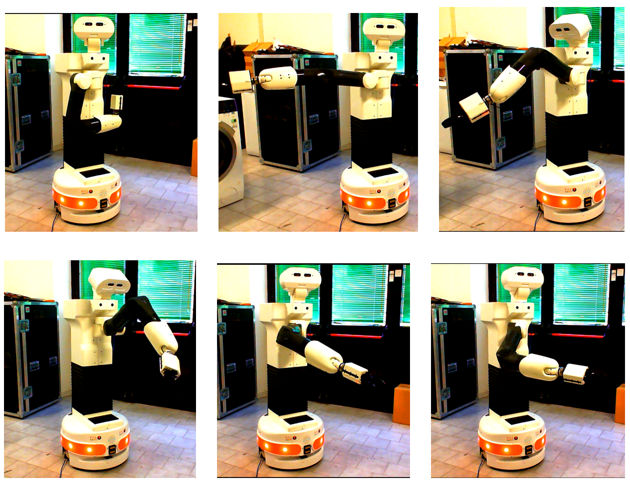


FIGURE 2.13: Different steps of the motion of TIAGo arm.

this, a mechanical extender is added to the simulation with an RG6 industrial gripper attached to it. This configuration allows the robotic arm to reach all points in the drum as shown in this simulation video[2].

2.7 Conclusion and Future work

This work has presented the results on collaborative navigation system by overlaying the conventional navigation stack with fast collision checking, human detection and tracking pipeline and safety constraint for the test laboratory environment. The overall system is shown to operate in a safe way in the vicinity of human operator while also optimizing energy usage of the arm.

The experimental and simulation tests are conducted for all the subsystems considering a scenario similar to the actual test laboratory. The result of the experiment showed significant success in navigating through narrow corridors while maintaining correct detection of appliances and estimating their pose both from close and far range.

[2]https://drive.google.com/file/d/1rdbS4MYSvqJ9yniJ1430QXvVUFd7WNj3/view?usp=sharing

Chapter 3

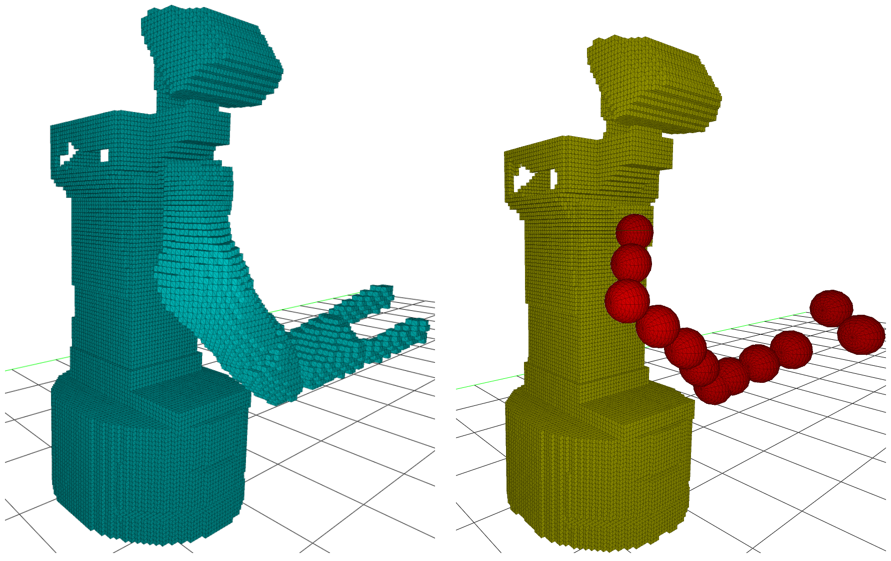
Realtime Collision avoidance

3.1 Introduction

Real time collision avoidance are shown to be relevant for robots in unknown dynamic environments, for example in unmanned aerial vehicles and mobile manipulation applications. They are based on reactive motion control which allow them to be employed within real time feedback loop and are suitable for applications with unstructured workspaces. As the paradigm of human robot interaction (HRI) shifts from complete autonomy to collaboration, Real-time reactive collision avoidance should accommodate the presence of humans in the vicinity of robots without posing danger to their safety. If human operators are considered in the robot environment, the safety requirements are even more stringent, since human movements can be fast and difficult to predict. In this regard, the speed at which the environment scene is updated, minimum obstacle distance and control commands are computed play crucial factor in utilizing real time collision avoidance in the context of HRI.

The perception pipelines dedicated to capturing the scene from live sensors and feed the robot control algorithms with the important information about obstacles usually consume significant (up to 90%) computational power (Pan, Lauterbach, and Manocha, 2010b). To minimize the latency during collision checking, the environment model is usually restricted to 2.5D and the robot model undergoes significant simplification in their representation (Kaldestad et al., 2014; Di Lillo et al., 2018). However, the advance in computing capability, particularly the emergence of high performance GPU's has accelerated the computation time thereby allowing a complete 3D representation of the environment and robots. In this aspect, GPU-Voxels is an open source library for CUDA based Graphics Processing Units (GPU) which allows massively parallel computation of collision checking in 3D environments (Juelg et al., 2017) and offers constant runtime regardless of the occupancy density in the environment. It is also computationally inefficient and unnecessary to compute minimum obstacle distance to every point on the robot body, therefore requiring a simplified but correct robot model approximations. In addition to the requirement for computational efficiency, real time collision avoidance relies on an algorithm that generate control directions that avoids obstacles. Task priority control is commonly used in this scenario since it allows to activate and deactivate tasks with higher priority, e.g. collision avoidance, only when they are really needed.

In this work, a generalized obstacle avoidance technique exploiting GPU voxels to compute 3D Euclidean Distance Transform (EDT) for obstacle and self-collision avoidance is proposed. The EDT contains the nearest obstacle information of all environment voxels which can be enquired to give the closest obstacles of any number points on the robot links. In this work, the parallelism offered by the GPU is exploited by removing the typical hierarchical representation of e.g. Octomap and directly performing distance computation on a body tight spheres as shown in



(a) Tiago robot voxel model.

(b) Obstacle links and bounding sphere approximation of the manipulator.

FIGURE 3.1: Representation of the Tiago robot in the scene: (a) links are offline voxelized and inserted according to the joint state of the robot; (b) the arm of the robot is approximated with spherical volumes, the body highlighted in yellow is used as an obstacle for self-collision avoidance.

Fig 3.4(b). Therefore, instead of relying on hierarchical representations, the novelty in the proposed algorithm is that it only relies on simplified robot shape representation by means of spheres that tightly approximate the robot links shape together with high resolution distance field of the environment in the range of 1 cm, thereby allowing collision free movement even in a close vicinity of the obstacle. Task priority control is then exploited to implement the collision avoidance task. The proposed approach is evaluated on a mobile manipulator, the Tiago robot from PAL Robotics, showing that all obstacles and self collision are avoided within one single framework in real time and in presence of dynamic obstacles, while the robot simultaneously performs end-effector pose tracking. A comparison with related algorithms that depend on CPU computed distance fields is also presented to highlight the time performance as well as accuracy of the GPU distance field. The key contributions of in this chapter therefore are:

1. Integration of Reactive collision avoidance with fast GPU-based distance field;
2. A simplified robot model (OBB-Aligned minimal spherical bounding volumes) capable of exploiting the GPU computation is proposed;
3. A novel approach for self collision avoidance that processes a voxelized model of each link on a GPU to compute minimum self collision distance;
4. A flexible task priority controller suitable for dynamic task transition and able to handle multiple tasks including obstacle and self-collision avoidance in one framework;

And finally, Practical demonstration of real-time reactive collision avoidance with live sensor data in a dynamic 3D scene contributing to practical usage of the algorithm. The remainder of the chapter is organized as follows. In Section 3.2, a literature review about collision avoidance and distance field computation is presented. The general framework of the proposed approach is reported in the context task-priority control using GPU-computed exact EDT together with an algorithm that unifies them in section 3.3. The summary of the mathematical background for task priority control and task oriented regularization is also presented in this section. Section 3.4 discusses experimental results both for obstacle and self-collision avoidance, together with a comparison with similar approaches reported in literature. Finally Section 3.5 presents conclusion and proposes a future extension.

3.2 Literature Review

Robot collision avoidance task has been approached as an offline or online problem. In the offline case, the goal is to solve a motion planning problem that is free of collision (Karaman and Frazzoli, 2011), (Karaman et al., 2011), (Moll, Sucan, and Kavraki, 2015), (Koenig and Likhachev, 2002) whereas online collision avoidance is characterized by a reactive motion and it is susceptible to local minima. When the robot operates in a dynamic environment it has to react very fast to obstacles putting a real time requirement. A reactive motion planning approach is proposed for collision avoidance in (Juelg et al., 2017) using exact 3D EDT that enable online motion re-planning. In (Simoni et al., 2018) a Task-Priority controller which supports set-based tasks is implemented for obstacle avoidance of an underwater vehicle using spherical collision objects generated from multi resolution Octomap for the obstacle representation. However, the current Octomap implementation only relies on Central Processing Unit (CPU) and it is therefore expected that it yields poor real time performance (Aalerud, Dybedal, and Hovland, 2018). A real time self collision avoidance based on Euclidean distance calculation between bounding spheres rigidly attached to robot links using robot kinematics is reported in (Bosscher and Hedman, 2011; Lei et al., 2020). The work in (Pan et al., 2013) by Jia Pan et al presents a real time dynamic AABB tree for fast culling as well as direct usage of an octree for obstacle data structure representation. However, octrees are known to slow down the update from incoming pointcloud, therefore limiting the performance. A different approach based only on depth image as opposed to point clouds to compute minimum obstacle distance is reported in (Flacco et al., 2012). A recent work based on Potential field method for real-time collision avoidance that exploits minimum distance computation between geometric primitives is reported in (Safeea, Neto, and Bearee, 2019). In this work four wearable inertial sensors are utilized per person to determine the pose of the geometric shapes approximating the shape of the person (Safeea and Neto, 2019), thereby providing human-like reflexes. It should be noted that the approach proposed in (Safeea, Neto, and Bearee, 2019) is more suited to an industrial scenario where one person with a wearable device interact with the robot. Another recent work in (Di Lillo et al., 2018) addresses obstacle avoidance within task priority framework for a selected control points by computing the minimum obstacle distance at 30 Hz from depth image by exhaustively scanning the neighbourhood of the control points until it finds the nearest obstacle pixel.

Apart from collision checking and minimum distance computation, online motion control also relies on a minimization problem with different task constraints to select optimal control inputs in real time. Motion control strategies with online collision avoidance are usually handled by exploiting the task priority framework in which null-space projection matrices are employed (Slotine and Siciliano, 1991), (Simetti and Casalino, 2016b) and (Mansard, Khatib, and Kheddar, 2009). This approach applies the null space of higher priority task to achieve lower priority tasks. For this, a null space projector matrix is derived using pseudo-inverse. To facilitate task insertion or removal, an activation matrix of dimension equal to the dimension given task is used. The problem with this approach is that the use of activation matrix resulted in what is commonly known as a practical discontinuity in which a mathematically continuous rapidly varying property of the control becomes discontinuity when implemented on actual robot. The work in (Simetti and Casalino, 2016b) proposed a task oriented regularization to tackle the problem of practical discontinuity i.e. it handles any general number of equality and inequality tasks while allowing task activation and deactivation without practical discontinuity.

3.3 General Framework for Collision Avoidance

In the proposed collision avoidance algorithm, the environment and the robot are described inside two separate but equal discretized regular grids of cubic voxels, with grid resolution equals to the vovel size. At every time step, both the robot and the environment maps are updated by defining the occupied voxels in their respective grids and the distance transform of the environment map is computed in real time. From this distance map, we can efficiently extract obstacle distance for few key points on the robot to activate collision avoidance tasks. The selection of these key points, also known as control points, is usually located at the center of the spheres that approximate the robotic links.

With this general approach in mind, there are challenges associated to exploiting GPU for the collision avoidance algorithm. The first, perhaps the starting point for our work is to separate various components of the collision avoidance algorithm based on whether they are suitable for parallelization or not. The collision avoidance algorithm based on a task-priority controller (Alg. 2) is implemented to run on a CPU due to their nature, while the EDT (Alg. 1) benefits from the massive parellization available in GPU. Secondly, both parts of the algorithm exchange information which may lead to bandwidth bottleneck in copying data from CPU to GPU and vice versa and therefore minimal data copying procedure is intended. Here, 3D pointcloud is down-sampled before it is copied to GPU memory. After computing the EDT, copying back the entire Distance map to CPU is avoided by simplifying the robot model so that only minimum obstacle distance to selected points suffice for the collision avoidance algorithm. Lastly, as each voxels are associated to a memory in the GPU, large map size will result in a significant memory consumption. As our development targets only online collision avoidance, a map size that accommodate the immediate workspace of the robot is initialized thereby leveraging memory usage.

To efficiently exploit the redundancy in the manipulator, the obstacle avoidance task is activated only for those parts of the robot whose distance to the closest obstacle is less than the radius of the spheres used to enclose, i.e. discretize, the robot structure. The discontinuity due to task activation and deactivation is mitigated by exploiting the task priority framework and the task oriented regularization, thereby imposing smooth behavior in the joint control velocity.

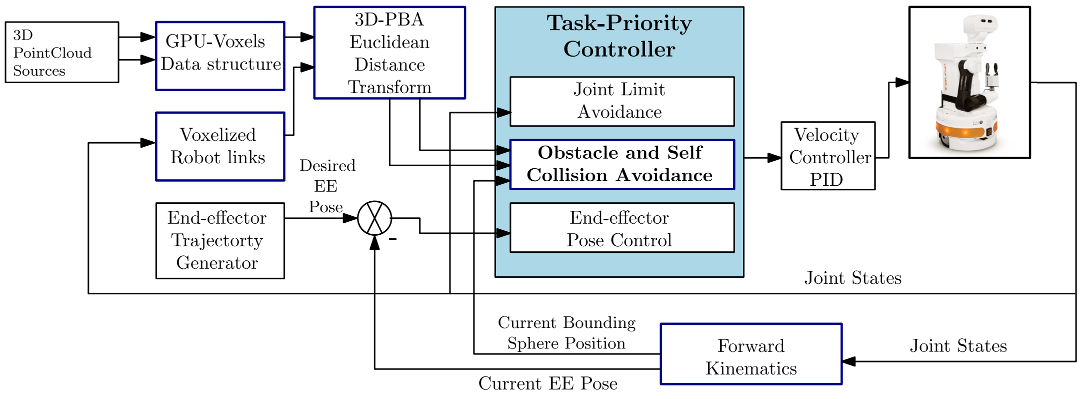

3D PointCloud Sources
GPU-Voxels Data structure
3D-PBA Euclidean Distance Transform
Task-Priority Controller
Joint Limit Avoidance
Obstacle and Self Collision Avoidance
End-effector Pose Control
Voxelized Robot links
Desired EE Pose
End-effector Trajectorty Generator
Velocity Controller PID
Joint States
Current Bounding Sphere Position
Forward Kinematics
Joint States
Current EE Pose

FIGURE 3.2: The Collision Avoidance control scheme implemented on the Tiago Robot testing platform. The main components of the proposed system are embedded in the blocks marked with blue.

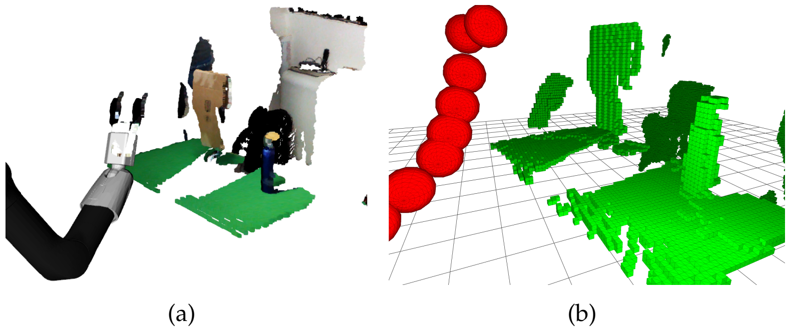
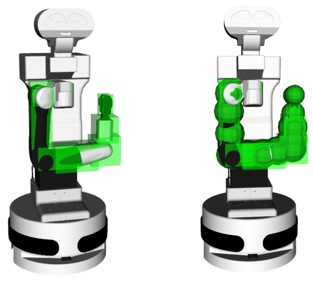


FIGURE 3.3: Raw point cloud processed to remove noise and robot part from the scene: Raw point cloud visualization in Rviz containing points coming from the robot gripper (left) and GPU voxels representation (right).

3.3.1 Obstacle Representation

In our proposed approach the environment (obstacles) is represented as a GPU voxels occupancy grid (Hermann et al., 2014b) in a probabilistic way. To create this environment map, the acquired point clouds are raw-copied into GPU memory to perform a statistical outlier filtering based on their Euclidean distance to their neighbors and followed by a transformation to a fixed coordinate. After transformation, the coordinates of each point are discretized to determine the according Voxel, whose occupancy status is updated as a Bayesian process. During insertion of points into the environment map corresponding Voxels' meanings in the robot map are reviewed if they are occupied by the robot model to prevent the insertion of points originating from robot parts, see Fig 3.3. The impact of occlusions can also be tackled using multiple source of point cloud, without affecting the speed of the algorithm.

For the self-collision avoidance task, the mesh model of the robot is rasterized into a binary 3D voxel grid to create a binary voxel model of the robot using an offline software tools such as in (Nooruddin and Turk, 2003b). This offline generated voxel representation of the robot will be inserted into a a map according to the joint state information of the actual robot to represent obstacles for the self-collision avoidance task as shown in Fig. 3.1(a) and Fig 3.1(b). By inserting the voxelized shape of each corresponding robot link, a full discrete representation of the robot is added into a robot map with same dimension as the environment map.



(a) OBBs defined over robot links.

(b) Spheres defined over the OBBs.

FIGURE 3.4: Bounding spheres computed using OBB for collision avoidance.

3.3.2 Robot Representation

The robot model is critical when performing minimum distance computation as complex models such as triangle meshes consume considerable amount of time while an over simplified models might lead to collision. To speedup distance query, various robot model approximation schemes has been proposed. The predominant approach utilizes simple primitive shapes in a hierarchical manner (Greenspan and Burtnyk, 1996; Steinbach et al., 2006; Simoni et al., 2018). An alternative approach to the hierarchical representation is to utilize complex geometric primitives such as cylinders, boxes and capsules (Pan et al., 2013; Safeea, Neto, and Bearee, 2019; Safeea and Neto, 2019). These geometries minimize the number models involved (one of these shapes can in-close an entire robot link). However, additional computations are required to determine points of minimal distance on both the models and the environment side in addition to multiple distance queries. Even more, these computations are not tailored toward parallel operation.

In this work, we propose a collision model composed of minimal number of spherical volumes with different radii across different links that discretize each link with low approximation error and without incurring additional computation on both CPU and GPU side. Here, spheres are merely used for enclosing the robot geometry and we do not rely on the hierarchy of spheres for distance query as opposed to the approaches reported in (Greenspan and Burtnyk, 1996; Steinbach et al., 2006; Simoni et al., 2018). To generate this set of bounding spheres, first an oriented bounding box (OBB) of each link in the kinematic chain is computed. Each OBB is represented by a center point, an orientation matrix and three half-edge lengths and are computed only once and in offline, therefore no overhead is added in run-time. The smallest diagonal dimension on the face of the OBB will be assigned as the diameter of the spheres and are placed along the longest dimension of the box. For an OBB with dimensions d_1, d_2, d_3 and $d_1 \leq d_2 \leq d_3$, spheres of diameter $\sqrt{d_1^2 + d_2^2}$ where will be utilized. This guarantees inscription of the OBB within the sequence of $\lceil \frac{d_3}{\sqrt{d_1^2+d_2^2}} + 1 \rceil$ spheres, thereby allowing finite minimal number on each link. The distance queries can easily be compared to the radius of each sphere for collision detection thanks to their rotational invariant property.

3.3.3 Exact EDT Computation

Given a 3D voxel grid of $\mathbb{G} = n \times n \times n$ voxels, the EDT problem is to determine the closest occupied voxel for each voxel in the grid. This problem is closely related to the Voronoi diagram computation, i.e. The EDT of a binary grid can be thought of as a discretized version of the Voronoi diagram whose Voronoi sites are the occupied voxels of the grid.

The parallel banding algorithm (PBA) proposed in (Cao et al., 2010) computes exact EDT on GPU by computing partial Voronoi which involves three phases where each of them are parallelized using bands to increase the number of threads (Alg. 1). For a 3D voxel grid $\mathbb{G}$ of size n, Consider a slice $\mathcal{I}_k$ at $z = k$ where k ranges from $k = 0$ to $n - 1$. This slice is basically a 2D binary image of $n \times n$ size. Algorithm 1 is a high level pseudo code summarizing the three steps as follow:

Step-1 (BandSweep (Line 3-4)):

Calculates $S_{i,j,k}$ i.e, the nearest Voronoi site, among all sites in slice k and row j, of the voxel (i, j, k). To efficiently employ threads, the slice $\mathcal{I}_k$ is divided into m1 vertical bands of equal size, and use one thread to handle one row in each band, performing the left-right sweeps followed by across band propagation. $n * m_1$ threads are utilized per slice in this step.

Step-2 (ComputeProximateSite (line 6)):

i.e, $\mathcal{P}_i$. Let $S_i = S_{i,j,k} \mid S_{i,j,k} \neq \emptyset, j = 0, 1, 2, ..., n - 1$ be the collection of closest sites for all voxels in column i of slice k. The sequential implementation to determine $\mathcal{P}_i$ is to sweep sites in S_i from topmost to bottommost, while maintaining a stack of sites that are potentially proximate sites, i.e; sites whose voronoi region intersects with column i. For every new site c, we evaluate whether the site at the top of the stack is dominated by c and the site a at the second top position in the stack. If so, b is popped out of the stack, and c is pushed onto the stack, while the sweeping continue until the column i is completed by returning the stack containing $\mathcal{P}_i$. To parallelize this sequential operation, each column S_i is partitioned horizontally into m_2 bands, thereby employing $n * m_2$ gpu threads per slice to perform the above computation.

Step-3 (QueryNearestSite (line 7))

This step utilizes $\mathcal{P}_i$ from step 2 to compute the closest site for each voxel V in column i of slice k top-down by checking two consecutive sites a and b in $\mathcal{P}_i$ in increasing y coordinate. If a is closer, then a is the closest site to V, if not a will be removed and the process continues with b and the next site. This is also performed in m_3 horizontal bands; therefore $n * m_3$ threads deployed per slice.

Step-4 (Extention to 3D (line 8-11))

At the end step-3, the voxel map contains a stack of 2D Voronoi diagrams. To extend this to 3D, step-2 and step-3 are repeated for all columns in the direction of the stack (line 8-11).

3.3.4 Obstacle Avoidance Task

The obstacle avoidance control is formulated as an m dimensional kinematic constraint associated to m bounding spheres distributed along the links of the robot.

Algorithm 1: Parallel_Banding_EDT

```
   Input: $\mathbb{G} : [0..n-1] \times [0..n-1] \times [0..n-1] \to \{0,1\}, m_1, m_2, m_3$
   Output: $EDT(\mathbb{G})$
1  for $k = 0$ to $n-1$ do
2      $\mathcal{I}_k \leftarrow$ Slice($\mathbb{G}, z = k$)
3      $S'_{i,j,k} \leftarrow$ BandSweep($\mathcal{I}_k, band = m_1$)
4      $S_{i,j,k} \leftarrow$ PropagateAcrossBand($S'_{i,j,k}$)
5      for $i \in [0..n-1]$ do
6          $\mathcal{P}_i \leftarrow$ ComputeProximateSite($S_i, band = m_2$)
7          $EDT_{\mathrm{i,j,.}} \leftarrow$ QueryNearestSite($\mathcal{P}_i, band = m_3$)
8  for $(i, j) \in [0..n-1] \times [0..n-1]$ do
9      for $k = 0$ to $n-1$ do
10         $\mathcal{P}_{i,j,.} \leftarrow$ ComputeProximateSite($S_k, band = m_2$)
11         $EDT_{\mathrm{i,j,k}} \leftarrow$ QueryNearestSite($\mathcal{P}_{i,j,.}, m_3$)
```

Consider, a collision avoidance task vector $X_c \in \mathbb{R}^m$, with i-th element x_{c_i} corresponds to a scalar task associated to i-th sphere on the robot link and defined as the distance between it's center $\mathbf{C_i}$ and it's nearest obstacle voxel coordinate $\mathbf{O_i}$:

$$x_{c_i} = \|\mathbf{O_i} - \mathbf{C_i}\|_2 \tag{3.3.4.1}$$

To implement this, we keep track of the coordinates of voxels at the center of the spheres described in section 3.3.2 (line 10) and their closest obstacle coordinate (line 11) of Alg. 2. The goal of collision avoidance control is to keep the task variable x_{c_i} bigger than the radius of the i-th bounding sphere, x_{M_i}.

$$x_{c_i} \geq x_{M_i} \tag{3.3.4.2}$$

The task Jacobian J_i, for the i-th scalar task x_{c_i} defines the direction that pushes the particular bounding sphere away from the nearest obstacle. This Jacobian is given by single row matrix derived by projecting position Jacobian J_{c_i} at $\mathbf{C_i}$ in the vector direction connecting $\mathbf{C_i}$ and $\mathbf{O_i}$.

$$J_i = \left(-\frac{\mathbf{O_i} - \mathbf{C_i}}{x_{ci}}\right)^T J_{c_i}(q) \tag{3.3.4.3}$$

At every control cycle the task constraint (eq. 3.3.4.1) and task Jacobian (eq. 3.3.4.3) are updated in Alg. 2, (lines 14-17). The linear velocity in the opposite direction w.r.t the closest obstacle at $\mathbf{C_i}$ is related to the joint velocity through the differential kinematic equation $\dot{x}_i = J_i \dot{q}$. Subsequently the suitable reference velocity for collision avoidance can be defined to be proportional to the difference between current task variable x_{c_i} and any $x^* > x_{M_i}$. Therefore, the velocity that drives x_{c_i} toward its corresponding objective x^* is

$$\dot{\bar{x}}_{c_i} = \kappa(x^* - x_{c_i}) \tag{3.3.4.4}$$

To generalize the above scalar representation of the collision avoidance task in to an m-dimensional task, we make use of the following notation: $X_c \in \mathbb{R}^m$ is the task variable vector, $\dot{\bar{X}}_c \in \mathbb{R}^m$ is a vector of suitable reference rate for the task vector X_c and $J_{CA} \in \mathbb{R}^{m \times n}$ is the obstacle avoidance task Jacobian. They can be described using the scalar quantities shown in eqs. (3.3.4.1) - (3.3.4.4) as follows:

$$\begin{aligned} X_c &= \left[x_{c_1}\ x_{c_2}\ \cdots\ x_{c_i}\ \cdots\ x_{c_m}\right]^T; \\ \dot{X}_c &= \left[\dot{x}_{c_1}\ \dot{x}_{c_2}\ \cdots\ \dot{x}_{c_i}\ \cdots\ \dot{x}_{c_m}\right]^T \end{aligned} \tag{3.3.4.5}$$

and the collision avoidance task Jacobian is given by

$$J_{CA} = \left[J_1\ J_2\ \cdots\ J_i\ \cdots\ J_m\right]^T \tag{3.3.4.6}$$

The main call to the collision avoidance starts in Alg. 2 by initializing the maps and loading bounding sphere centers in link frame (line 1). At every control cycle, new point cloud and the voxelized robot links are updated and inserted into their corresponding voxel map while the position of the sphere centers are computed using forward kinematics (line 2-8). This is followed by a call to a gpu kernel that implements Alg. 1 to compute the EDT of the maps (line 9). Note that, the EDT transform is stored and updated in GPU memory while only the minimum distance obstacle coordinate to the sphere centers $\{\mathbf{C_1} \ldots \mathbf{C_m}\}$ are copied back to the CPU to update the Jacobian (eq. 3.3.4.3) and control (eq. 3.3.4.4) by calling Alg. 3 in line 12 of Alg. 2.

3.3.5 Self-Collision Avoidance

Self-collision avoidance of an arbitrary link utilizes EDT of a set of prior links to determine the closest points. In general, the obstacle of a kinematic chain in self-collision sense is composed of all other links in the robot body $L_1, L_2, \ldots, L_i$ that are not in the allowed collision matrix (ACM) and it is given as:

$$O_L = \{L_1, L_2, \ldots, L_i\} \tag{3.3.5.1}$$

All links in the set O_L are voxelized offline, transformed and inserted into voxel grid according to their corresponding joint state to compute the EDT for self-collision avoidance as shown in Fig. 3.5. For our particular robotic platform, i.e Tiago robot, links shown in yellow voxels in Fig. 3.1(b) form the set of all possible self collision objects for the arm kinematic chain indicated by the red spherical balls.

Once the closest points are computed, for instance, o_{s_i} for the a task sphere of c_i, a self-collision avoidance task constrains the relative motion of the two points in the direction of the line connecting them. In a similar way to obstacle avoidance, the i-th scalar self-collision avoidance task x_{s_i} and It's corresponding Jacobian J_{s_i} are given as:

$$x_{s_i} = \|\mathbf{O_{s_i}} - \mathbf{C_i}\|_2 \tag{3.3.5.2}$$

and

$$J_{s_i} = \left(-\frac{\mathbf{O_{s_i}} - \mathbf{C_i}}{x_{ci}}\right)^T J_{c_i}(q) \tag{3.3.5.3}$$

3.3.6 Task Priority Controller Formulation

For a general robotic system with n-DoF, in a given configuration, $\mathbf{q} = [q_1\ q_2\ \cdots\ q_n]^T$, the forward kinematics of a particular task $\mathbf{x} \in \mathbb{R}^m$ can be expressed as a function of joint configuration:

$$\mathbf{x(t)} = \mathbf{x(q(t))} \tag{3.3.6.1}$$

Algorithm 2: Obstacle and Self_Collision_Avoidance

```
Input:     Bounding Sphere Centers in Link frame $\{\mathbf{C_1} \dots \mathbf{C_m}\}$
           Bounding Sphere Radius $\{x_{M_1} \dots x_{M_m}\}$
1:  Initialisation: GPU_Voxels Self Collision Obstacle Map $\mathbb{M}_\mathbb{S}$ and Environment
    Obstacle Map $\mathbb{M}_\mathbb{I}$
2:  while True do
3:      update Joint_States $\mathbf{q} = \{q_1, q_2, \dots q_n\}$
4:      $\{\mathbb{M}_\mathbb{S}, \mathbb{M}_\mathbb{I}\} \leftarrow$ ClearMap
5:      $O_{L_k} \leftarrow$ UpdateLinks($\mathbf{q}$)
6:      $\mathbb{M}_\mathbb{S} \leftarrow$ Insert($O_{L_k}$)
7:      $\mathbb{M}_\mathbb{I} \leftarrow$ Insert(Point_Cloud)
8:      $\{\mathbf{C_1} \dots \mathbf{C_m}\}_{\text{global\_frame}} \leftarrow$ ForwardKinematics($\mathbf{q}$)
9:      $\{\mathbb{M}_\mathbb{S}, \mathbb{M}_\mathbb{I}\} \leftarrow$ PBA_EDT($\mathbb{M}_\mathbb{S}, \mathbb{M}_\mathbb{I}, m_1, m_2, m_3$)
10:     $\{\mathbf{O_{s_i}} \dots \mathbf{O_{s_m}}; \mathbf{O_i} \dots \mathbf{O_m}\}_{\text{global\_frame}} \leftarrow$
          ExtractClosestObstaclePose($\mathbb{M}_\mathbb{S}, \mathbb{M}_\mathbb{I}, \{\mathbf{C_1} \dots \mathbf{C_m}\}$)
11:     $X_c, \dot{X}_c, J_{CA}, J_{SA}, A \leftarrow$
          Update_Col_Jac($\mathbf{C_1} \dots \mathbf{C_m}; \mathbf{O_{s_i}} \dots \mathbf{O_{s_m}}; \mathbf{O_i} \dots \mathbf{O_m}$)
12:     $\dot{q}_p \leftarrow$ UpdatePriorityLevel($A, X_c, \dot{X}_c, J_{CA}, J_{SA}, \dot{q}_{p-1}$)
13: end while
```

For such task variable, we also assume the existence of Jacobian relationship between task space velocity $\dot{x}$ and the joint velocity vector $\dot{\mathbf{q}}$ as

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{3.3.6.2}$$

where $J(\mathbf{q}) \in \mathbb{R}^{m \times n}$ is the Jacobian matrix. Given a reference task space velocity vector $\dot{\bar{x}}$, joint velocity vector $\dot{\mathbf{q}}$ that satisfies $\dot{\bar{x}}$ in the least-square sense, can be computed using Pseudo inverse as

$$\min_{\dot{\mathbf{q}}} \left\|\dot{\bar{x}} - J\dot{\mathbf{q}}\right\|^2 \implies \dot{\mathbf{q}} = (J^T J)^{\#} J^T \dot{\bar{x}} \tag{3.3.6.3}$$

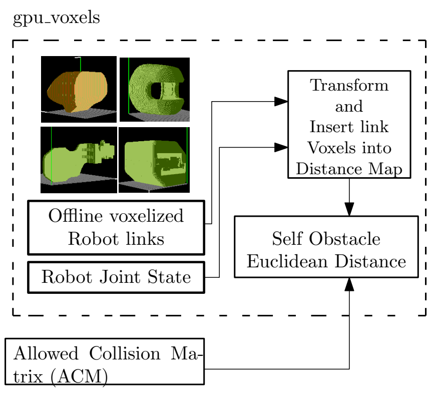

gpu_voxels
Transform and Insert link Voxels into Distance Map
Offline voxelized Robot links
Robot Joint State
Self Obstacle Euclidean Distance
Allowed Collision Matrix (ACM)

FIGURE 3.5: Pipeline for Self collision minimum distance domputation: The voxel model of Links that are not in the ACM are transformed and Inserted inside a GPU_voxels map followed by EDT.

Algorithm 3: Update_Col_Jac

```
Input: $\{\mathbf{C_1}\ldots\mathbf{C_m};\mathbf{O_{s_i}}\ldots\mathbf{O_{s_m}};\mathbf{O_i}\ldots\mathbf{O_m}\}_{\text{global\_frame}}$
1:  for $i = 1$ to $m$ do
2:      $J_{c_i}(\mathbf{q}) \leftarrow$ Compute Position Jacobian at $\mathbf{C_i}$
3:      $x_{s_i} \leftarrow \|\mathbf{O_{s_i}} - \mathbf{C_i}\|_2$
4:      $x_{c_i} \leftarrow \|\mathbf{O_i} - \mathbf{C_i}\|_2$
        Compute Self Collision Avoidance Jacobian at $\mathbf{C_i}$
5:          $J_{s_i}(\mathbf{q}) \leftarrow (-\frac{\mathbf{O_{s_i}}-\mathbf{C_i}}{x_{s_i}})^T J_{c_i}(\hat{q})$
        Compute Obstacle Avoidance Jacobian at $\mathbf{C_i}$
6:          $J_i(\mathbf{q}) \leftarrow (-\frac{\mathbf{O_i}-\mathbf{C_i}}{x_{ci}})^T J_{c_i}(\mathbf{q})$
        Compute Activation Value $A_{s_{(i,i)}}$, $A_{o_{(i,i)}}$
7:          $A_{s_{(i,i)}} \leftarrow \text{Sigmoid}(x_{s_i}, x_{M_i}, b_i)$
            $A_{o_{(i,i)}} \leftarrow \text{Sigmoid}(x_{c_i}, x_{M_i}, b_i)$
        Compute Control $\dot{\tilde{x}}_{s_i}$, $\dot{\tilde{x}}_{c_i}$
8:          $\dot{\tilde{x}}_{s_i} \leftarrow \kappa(x^* - x_{s_i})$
            $\dot{\tilde{x}}_{c_i} \leftarrow \kappa(x^* - x_{c_i})$
9:  end for
10: Return   $X_c, \dot{\tilde{X}}_c, J_{CA}, J_{SA}, A$
```

The manifold of all solutions can be derived by introducing null space projector that ensures task velocity remains unchanged:

$$\dot{\mathbf{q}} = (\mathbf{J}^T\mathbf{J})^{\#}\mathbf{J}^T\dot{\tilde{\mathbf{x}}} + (\mathbf{I} - (\mathbf{J}^T\mathbf{J})^{\#}\mathbf{J}^T\mathbf{J})\dot{\mathbf{q}}_0 \tag{3.3.6.4}$$

where $(.)^{\#}$ represents generalized pseudo-inverse and $\dot{q}_0$ denote a joint velocity that produce an orthogonal component of $\dot{\tilde{x}}$ i.e. not affecting the desired task.

Task insertion and removal is handled using a diagonal task activation matrix $A \in \mathbb{R}^{m\times m}$ associated with each task $x \in \mathbb{R}^m$ whose value is given by

$$A_{(i,i)} = \begin{cases} 1 & x_i \geq x_{M_i} \text{ (Activated)} \\ s_i(x_i) & x_{Mi} - b_i \leq x_i \leq x_{Mi} \text{ (transition)} \\ 0 & x_i \leq x_{Mi} - b_i \text{ (Deactivated)} \end{cases} \tag{3.3.6.5}$$

where $s_i(x_i)$ is the sigmoid function given by

$$s_i(x_i) = \frac{1}{2}\left(\cos\frac{(x_i - x_{Mi})\pi}{b_i} + 1\right) \tag{3.3.6.6}$$

Where b_i, x_i and x_{M_i} are the i-th task transition buffer, i-th elements of the task x and the task activation threshold of the i-th task respectively. with the activation matrix A, the minimization problem in eq. (3.3.6.3) takes the form $\min_{\dot{q}} \left\|\mathbf{A}(\dot{\tilde{\mathbf{x}}} - \mathbf{J}\dot{\mathbf{q}})\right\|^2$ showing that only tasks that are active will be considered in the minimization problem.

To avoid large joint velocities due to task singularity, singular value oriented regularization term is required to penalize joint velocities in the task singularity direction. Even more, to tackle practical discontinuity due to task activation and deactivation, a task oriented regularization term is added to selectively penalize scalar tasks that are in transition phase. Thus, introducing both singular value oriented and task-oriented regularization in to the minimization problem in (3.3.6.3) gives

$$\min_{\dot{q}} \left\|\mathbf{A}(\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}})\right\|^2 + \|\mathbf{J}\dot{\mathbf{q}}\|_{\mathbf{A}(\mathbf{I}-\mathbf{A})}{}^2 + \left\|\mathbf{V^T}\dot{\mathbf{q}}\right\|_{\mathbf{P}}{}^2 \tag{3.3.6.7}$$

where V^T is the right orthonormal matrix of the SVD decomposition of $J^T AJ = U\Sigma V^T$ and P is a diagonal regularization matrix where each diagonal element $p_{(i,i)}$ is a bell-shaped function of the corresponding singular value of J, or zero if the corresponding singular value do not exist and the notation $\|\cdot\|_P$ indicates the weighted norm, i.e. $\|\dot{q}\|_P^2 = \dot{q}^T P\dot{q}$. The generalized solution of (3.3.6.7) is then given by

$$\begin{aligned}\dot{\mathbf{q}} &= (\mathbf{J}^T\mathbf{AJ} + \mathbf{V}^T\mathbf{PV})^{\#}\mathbf{J}^T\mathbf{A}\ \mathbf{A}\dot{\mathbf{x}} \\ &\quad + (\mathbf{I} - (\mathbf{J}^T\mathbf{AJ} + \mathbf{V}^T\mathbf{PV})^{\#}\mathbf{J}^T\mathbf{AAJ})\dot{\mathbf{q}}_0\end{aligned} \tag{3.3.6.8}$$

where $\dot{q}_0$ can be used to perform lower priority tasks in hierarchy.

3.4 Experiments

3.4.1 Experimental Setup

The 7-DoF Tiago robot arm is used in the experiments to interact with a dynamic obstacle. The scene is captured at 30 Hz by an ASUS Xtion 3D camera available on the Tiago robot head and eventually transformed to the base frame of the robot using the approach described in section 3.3.1. The obstacle and self-collision avoidance Alg. 2 calls the GPU kernel from GPU_voxels library in lines 1-8. An external computer hosting an NVIDIA GeForce GPU (GeForce GTX 1080 Ti) with 3584 CUDA cores for parallel distance computation is used for this purpose. Even though the robotic arm can only span a volume of less than 1.2 m x 1.2 m x 1.8 m, the experiments are performed on a 1.92 m x 1.92 m x 1.92 m physical volume and evaluated the real time property of maps as large as 20.48 m x 20.48 m x 5.12 m, i.e (512x512x128 with 4cm voxel size). During all experiment scenarios, the following task hierarchy is considered: (1) Joint limiter (2) Collision Avoidance and (3) End-effector Pose control, where the priority goes from the highest to the lowest with the joint limiter and collision avoidance tasks are activated and deactivated according to an activation matrix.

3.4.2 Dynamic Obstacle Avoidance Results

The real time obstacle avoidance is demonstrated by selecting two test scenarios: the first test case is to interact with a dynamic scene formed by an approaching person while the robot is tracking a set point and the second test scenario is to follow an end-effector trajectory while also avoiding dynamic obstacle. During our experiment, the PBA updates distance map at about 350 Hz which is much higher than other robot obstacles distance evaluation methods. Even though scene point cloud is updated at 30 Hz, the control points continue to move along with the robot link, therefore their corresponding minimum distance evaluation should be performed at higher rate. In the first scenario, the Tiago arm avoids collisions in real time from a dynamic obstacle, as shown in the sequence of Figs. 3.6 and accompanying video[1]. A person walking faster than an average human walk speed (approximately 1.4 m/s)

[1]https://drive.google.com/file/d/1uu7fmqTX9dWkxG90lXoo85LdTbw-HKkt/view?usp=sharing

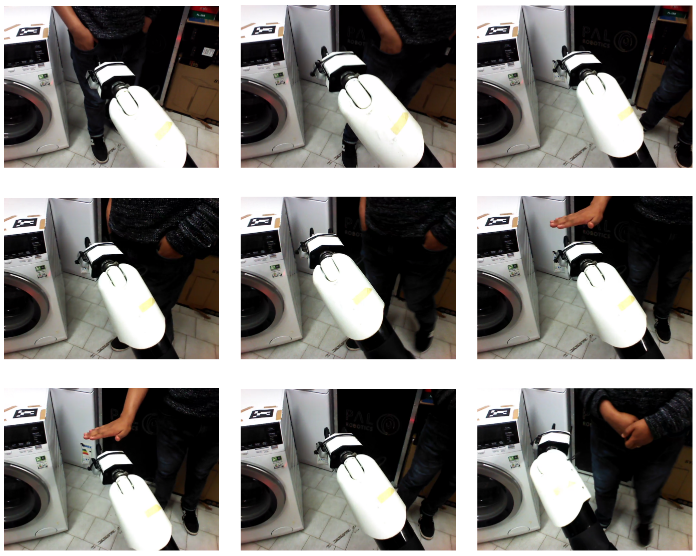


FIGURE 3.6: Experiment 1: The sequence of images top-down demonstrate real time collision avoidance from a dynamic obstacle while tracking end-effector setpoint.

approaches the robot from different directions. With reference to Fig. 3.6, the first two images in the sequence show a person approaching the robot from the front, in the next three the person approaches the robot from the side, in the next two the person moves his hand close to the robot end effector from the top and finally full body approach is shown in the last two images. In all these cases, the robot successfully avoids collision. As the person retreats from the scene, the end-effector control task pushes the end-effector to the desired pose. The real time velocity command, obstacle avoidance task activation value and minimum distance associated to this maneuver is given in Fig. 3.7. Note that the activation value and minimum obstacle distance for the 12th, 11th, 10th and 9th Bounding Spheres indicated in the plot of Fig 3.7 are associated to the last four spheres of the robot model and the remaining spheres are omitted because they remain deactivated throughout the experiment. The collision avoidance activation matrix A is also seen to get activated only when the person enters the scene and deactivated when the person exits, see Fig 3.7.

In our second experiment, the robot end-effector follows a trajectory composed of two way-points, moving back and forth between them. A fast dynamic obstacle is formed by a box hanging through a thread, oscillating and spinning toward the robot, therefore requiring a fast reaction. As shown in Fig. 3.8 and the accompanying video[2], the robot avoids collision while also following end-effector trajectory as a lower priority task. A small part of the 3d Cartesian waypoints(red) and end-effector trajectory (blue) is shown in Fig 3.9.

3.4.3 Task Oriented Regularization: Evaluation

To specifically evaluate concern with the practical discontinuity, an experiment (experiment 3) is performed in which the robot arm followed an identical trajectory with and without task oriented regularization. As shown in Fig. 3.10, the joint velocity command at the top contain regularization term which suppresses sudden velocity jumps during collision avoidance task activation.

[2]https://drive.google.com/file/d/1Gqkal-3iWbv-Q39KaJLa1sPInsDdwUBF/view?usp=sharing

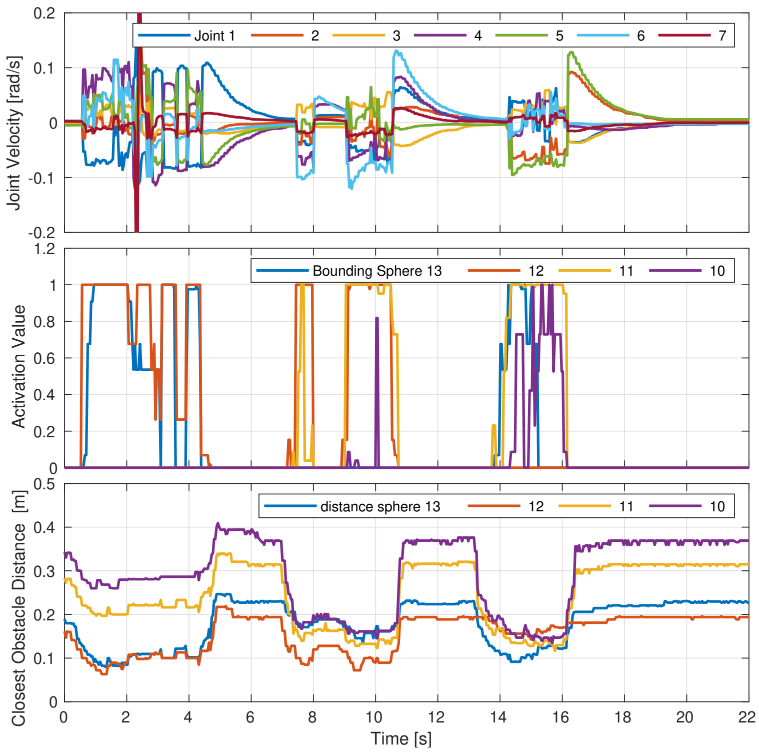


FIGURE 3.7: Experiment 1: Commanded joint velocities, activation value and minimum obstacle distance during the collision avoidance from a moving person.

3.4.4 Comparison to related works

The real time property is evaluated by examining the time elapsed to insert the scene point cloud to a GPU and compute distance map with various the map size. As it can be seen from Table 3.2, the rate of GPU based EDT depends linearly on the map size. But for a map sizes of practical interest (i.e. within the limit of the robot workspace) the distance map updates in the range of 200-500 Hz, see Table 3.2. A GPU based implementation comparable to our work is given in Kaldestad et al., 2014 which utilizes large set of vertices sampled from the mesh model of the robot as opposed to bounding spheres in our work to represent the robot. The reported time for distance field computation in Kaldestad et al., 2014 is at least 10 ms (100 Hz)

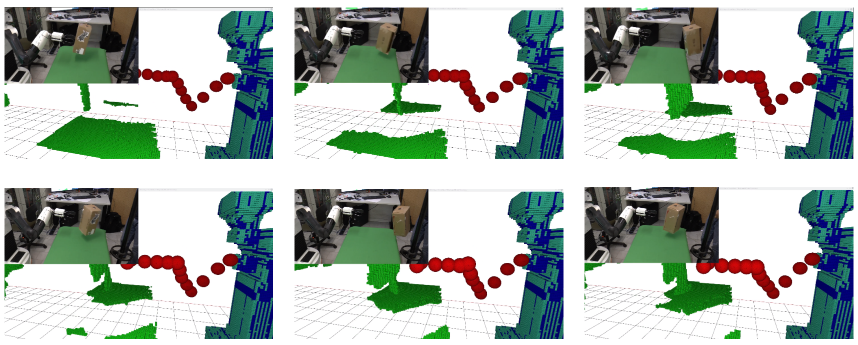


FIGURE 3.8: Experiment 2: Image sequence for a motion through two way points with collision avoidance. An oscillating and spinning hanging box approaching the robot arm in the workspace

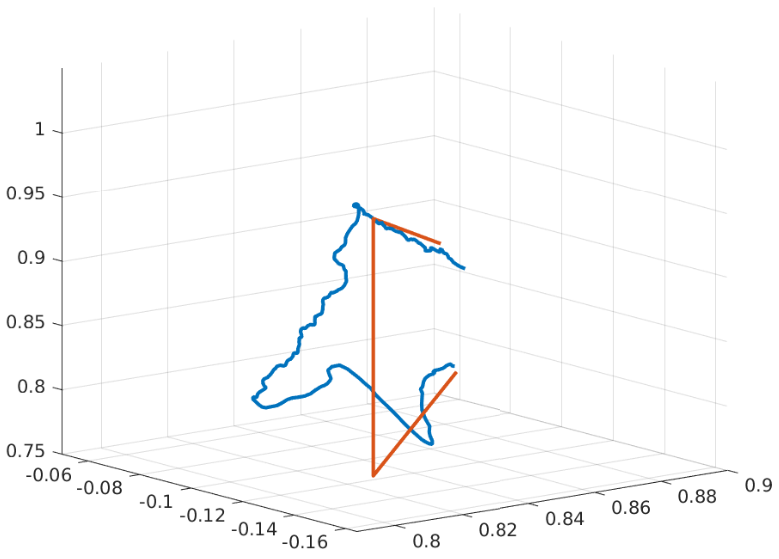


FIGURE 3.9: Experiment 2: Tiago End-effector trajectory. The red line shows the desired trajectory points and The blue line shows the modified end-effector trajectory by the fast oscillating obstacle described in Fig 3.8.

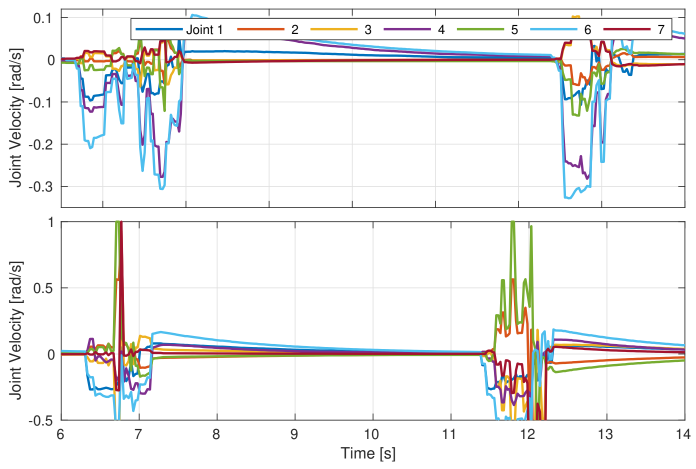


FIGURE 3.10: Experiment 3: Joint command: with Task oriented regularization (top) and without (bottom). A sharp velocity change is observed at time 5.2, 8.6, 10.5 Sec for the same obstacle scene without regularization.

for a map dimension of 2 m × 2 m × 1.8 m, in addition to the 4 ms for potential field computation. Note that the approach given in Kaldestad et al., 2014 represents the environment in 2.5D and still our approach delivers higher performance.

To evaluate the efficiency of the PBA compared to other GPU-compatible EDT algorithms, we considered two approaches from literature that could exploit modern GPU's. In the first case, we evaluate a naive approach in which the parallelism is limited to one thread per row. This is the baseline scenario achieved by setting the banding parameters to $m_1 = 1, m_2 = 1$ and $m_3 = 1$. Secondly, a Jumping Flood algorithm(JFA) in Rong and Tan, 2007 that works by creating outward ripple effect starting from the occupied voxels so that each voxel in the grid can decide which occupied voxel is it's closest one is also considered. Another algorithm comparable to JFA is also presented in Schneider, Kraus, and Westermann, 2009 with time-work complexity of $\mathcal{O}(3^d N)$, where d is the size of each dimension in the grid. However, It is reported in Cao et al., 2010 that this algorithm performs slower in all scenarios considered compared to the PBA, therefore omitted from the comparison in Table 3.1. In our comparison we considered three important factors: the level of parallelism (GPU utilization), algorithm run-time and work complexity and computation time for a practical map size. Although both PBA and JFA exploit GPU resource quite well, JFA has higher work-complexity that grows very fast with the map dimension as shown by the significant jump in time from $256^2 \times 128$ map size to $512^2 \times 128$.

The comparison to algorithms whose distance computation relies on CPU based 3D representations such as Octomap and other methods utilizing approximations are given in Tab. 3.3. Due to significant amount of latency in capturing and updating dynamic environment using a CPU based hierarchical data structures such as Octree (the underlying data structure of of Octomap), which is usually less than 15 Hz, comparison to our work is limited to static or very slowly varying scenes. The work in Pan et al., 2013 proposed two approaches to represent obstacles: the first utilizes Octomap directly while in the second approach they generate box collision objects from Octomap as an approximation. The results in Tab. 3.3 shows that, the approach in Pan et al., 2013 is notably affected by the resolution of the Octomap as well as the robot model on which distance query is performed. Usage of high resolution Octomap in combination with mesh model for distance query leads to prohibitive computational time. On the other hand, for 11 bounding spheres distributed along the arm of Tiago robot, it takes about 20 ms to complete distance computation from an Octomap of 2 cm resolution. Although this time is acceptable in terms of real time execution, the performance deteriorates as the number of occupied nodes in the map rises, see Fig. 3.12. Another approach proposed in Simoni et al., 2018 approximates the Octomap at different depth with sphere by replacing each occupied node with sphere. Here, it is important to note that the depth of query on the Octomap affects the computation time as well as the specific collision avoidance application. To elaborate this we performed a separate experiment (experiment 4) in which the robotic arm reaches inside a washing machine to perform inspection and grasping in the drum. To achieve this, An offline generated point cloud model of a complete washing machine is registered in to the scene using fiducial markers. For obstacle avoidance task during entering and existing the interior of an appliance, depth of 14 leafs on Octomap wasn't sufficient and going higher would incur larger computation time. However, with the proposed method in this work, a one time EDT for this static scene was sufficient. As the robot arm moves minimum distance to the control points is extracted for collision avoidance.

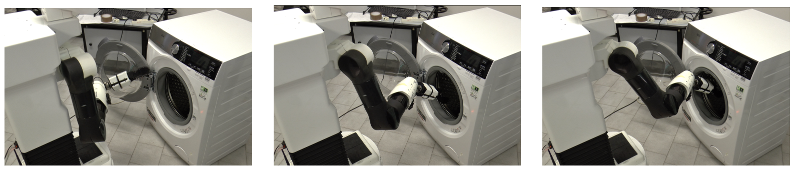


FIGURE 3.11: Experiment 4: Tiago robot arm entering the washing machine drum.

3.4.5 Self-Collision Avoidance

This experiment demonstrates Tiago arm self collision avoidance with the entire body in real time. The distance between the bounding spheres of the arm and the rest of the robot part is computed and a self-collision avoidance velocity is generated according to Alg. 2 and eqs. (3.3.5.2)-(3.3.5.3) as Tiago arm navigate from initial pose in Fig. 3.13 to a final pose in Fig. 3.13 along the rim of the circular base.

The minimum distance of the bounding spheres on the gripper and the corresponding activation matrix and joint velocities shown in Fig. 3.14 demonstrate that the robot arm maintained threshold distance during the movement toward the set-point.

3.5 Conclusions

This chapter presented a unified real time self and obstacle collision avoidance method that is based on kinematic task priority control. The main idea in the proposed algorithm is to perform minimum distance computation for both external obstacle and self collision on a GPU at a very high rate so that the robot is able to react instantaneously to avoid collision from dynamic obstacles as well as it's own link. Multiple experiments on our mobile robot platform Tiago demonstrated the real time effectiveness of the method. A comparison to related works on reactive collision avoidance has also been given showing that the level of occupancy, resolution and depth of query of the environment representation significantly affect their performance.

Future work will involve estimation of the velocity of the dynamic obstacle which will improve the obstacle avoidance task. In addition we also intend to introduce safety constraints for human interacting with robot.

TABLE 3.1: work and time complexity comparison of GPU-based EDT computation algorithms for 3d voxel grid $N = n^3$

Factors	PBACao et al., 2010 m_1=m_2=1 m_3=8	JFARong and Tan, 2007	PBA-naive m_1=m_2=1 m_3=1
GPU Utilization (Threads and Bandwidth)	Very high	Very high	under-utilized
time-work complexity	$\mathcal{O}(N)$	$\mathcal{O}(N \log n)$	$\mathcal{O}(N)$
Run-time in (ms): 256×256×128	3.18	26.67	3.52
Run-time in (ms): 512×512×128	7.782	114.47	11.271

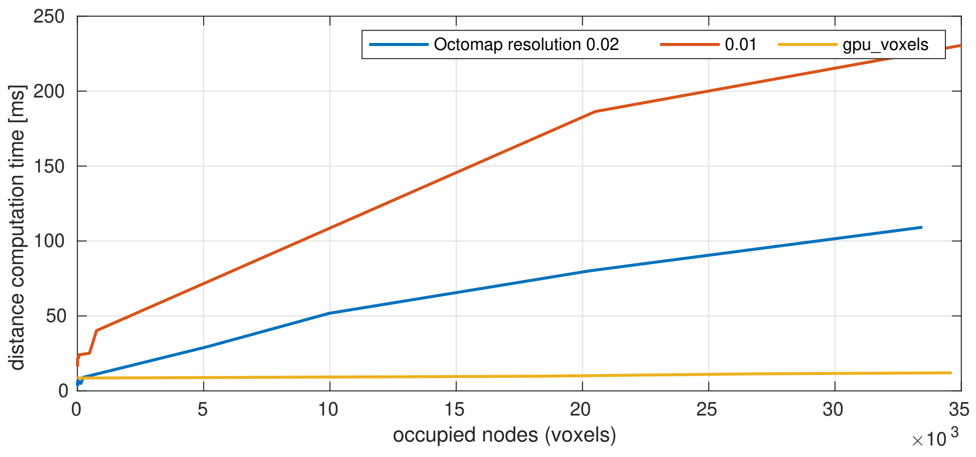


FIGURE 3.12: 11 distance queries on Octomap as the number of occupied nodes increase shows significant change in the computation time (Red and Blue). On the other hand, time for distance queries are relatively unaffected as number of voxels occupied increases GPU voxels (Yellow).

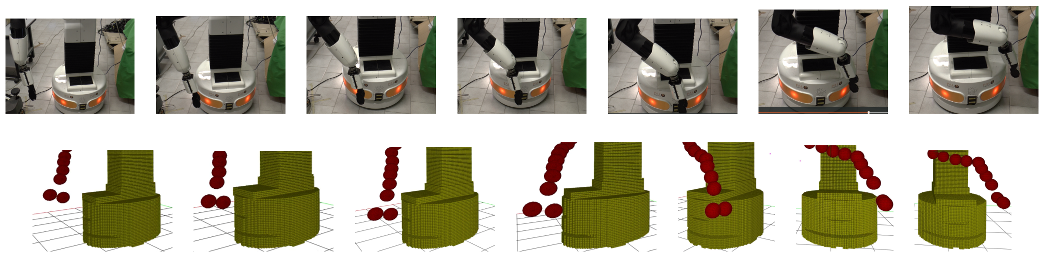


FIGURE 3.13: Experiment 5: Sequence of images (top) showing The arm navigating around the rim of the circular base platform to Avoid self collision while also moving towards a set point. Visualization of robot body as a collision object (bottom sequence).

TABLE 3.2: GPU Voxels Exact Euclidean Distance Computation Time (in ms) with Voxel size of 0.5cm, 1cm and 2cm.

Map Dimension[Vox]	No. of Occupied Voxels	Clear Map	Insert PointCloud	Compute Distance PBA
Voxel size 2cm				
192x192x128	6214	0.2284	0.3281	1.7601
256x256x128	9080	0.4565	0.3179	3.2475
512x512x128	12274	0.8268	0.2580	8.1972
Voxel size 1cm				
192x192x128	6863	0.9978	1.0639	2.4013
256x256x128	10014	1.2085	0.9460	3.9767
512x512x128	18274	1.3446	0.9426	11.5601
Voxel size 0.5cm				
192x192x128	7263	1.0478	1.2639	2.8013
256x256x128	12014	1.2985	0.9962	4.2767
512x512x128	19304	1.5446	1.1426	12.0601

TABLE 3.3: Average minimum distance computation time (in ms) between robot and obstacle for CPU based models.

	Full Octomap obstacle representation (Pan et al., 2013)		
Robot model	1cm res	2cm res	4cm res
mesh Model	269.842	145.571	107.289
bounding spheres	77.8657	19.7948	6.7741
	Occupied Box approximation for obstacles (Pan et al., 2013)		
	1cm res	2cm res	4cm res
mesh Model	142.446	48.1075	17.459
bounding spheres	131.719	36.56	8.3969
	Occupied Spheres approximation for obstacles (Simoni et al., 2018)		
	14 depth	11 depth	9 depth
mesh Model	142.446	48.1075	17.459
bounding spheres	17.724	11.661	9.351

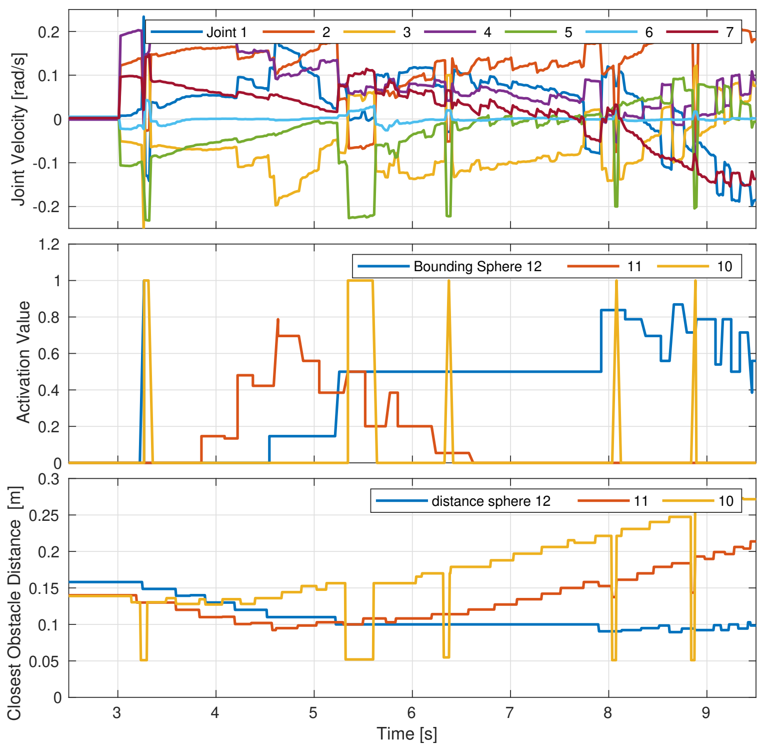


FIGURE 3.14: Self collision avoidance: joint velocity, activation and minimum distance during Experiment 5 shown in Fig 3.13 .

Chapter 4

Efficient 6D Model Registration of Large Objects

4.1 Introduction

The application of shape registration nowadays spans multiple areas of robotic assistant tasks that require a complete understanding of the 3d scene. In this regard, the main practical problem usually emerges due to partial information acquired by range finders or 3D vision sensors because of occlusions. This problem becomes particularly relevant when the robot objective is to interact in a specific way with certain objects in the environment, having a previous knowledge about its functionality but only limited information about its location and that can be also considered as obstacles form the point of view of the robot motion.

The scenario here considered is related to a mobile manipulator interacting with a washing machine in household environments. Because of its size, the washing machine can be seen as an obstacle, but precise knowledge about its location is needed to interact with it and perform manipulation tasks. Due to the complex geometry of the drum's interior, the robotic arm requires up to 1 cm accuracy in the nearest obstacle measurement when performing inspection and grasp inside the appliance drum which can only be achieved through full model registration. Besides, to perform grasping of pieces of laundry inside or on the exterior of the appliance, laundry region segmentation is important. Our registration algorithm can contribute to the correct removal of the point clouds related to the appliance out of the scene. The same principle can be applied to similar scenarios in which robots need to interact with other appliances or machines in general. To solve this issue, shape registration algorithms provide information about the location of a known objects to the robot manipulator, assuming that the robot has some representation of the object embodiment, such as a 3D CAD model.

The main issue related to the case of relatively big objects is that only limited information can be obtained by 3D vision sensors. This is due to the limited field of view of devices such as 3D scanners, RGB-D cameras and LiDAR, and due to the shape and size and occlusions of target object. This issue is also made harder by the fact that very likely those objects have very limited features that can be exploited by the vision system. In facts, many registration algorithms highly depend on the accurate initial transformation provided by feature-based methods that have been widely used over the years as a solution. However, feature-based methods require sufficient shape textures on the given objects in order to calculate the local features.

In this chapter, we provide an efficient algorithm to align the 3D model of a domestic appliance with point cloud provided by a live RGB-D camera in a robust way by combining object detection based on deep learning, GPU-based implementation of the Iterative Closest Point (ICP) algorithm and exploiting OcTree structure. The

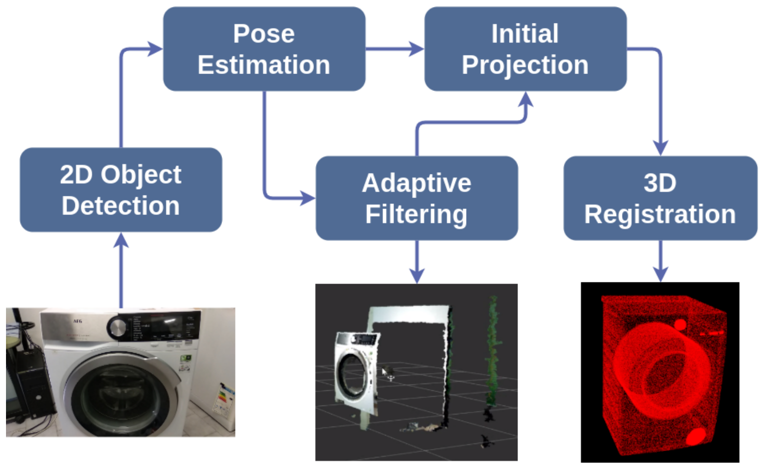


FIGURE 4.1: The workflow of shape registration.

proposed scenario is selected because, in common household environments, usually only the front of the object is visible, but the robot needs to have a complete knowledge about its spatial location and encumbrance during the interaction with the internal parts of the appliance.

A deep learning-based approach is implemented in order to have better initialization and avoid computational complexity of feature-based methods. The first step in our approach is to determine relevant points on the image plane using deep learning based object detection techniques. This result is exploited to obtain a preliminary information about the appliance location in the camera frame. The initial pose guess allows us to apply adaptive filtering on the scene data as well, while localizing the robot with respect to the appliance. The initial pose estimate integrate the rigid transformation obtained from pose estimation module to the solution of initial estimation problem. The proposed approach has the following important contributions:

- Improved shape registration in terms of accuracy and speed by using featureless technique in combination with usage of OcTree structure in the ICP computation on GPU.
- Comparisons to the different registration approaches that involves Feature-based technique, involving various data structure and computing hardware (CPU and GPU).

4.2 Related Works

Object Detection: Nowadays, deep learning based object detection is one of the hot topics in computer vision. In recent years, one-stage models (Liu et al., 2015), where localization and classification processes are conducted in a single step by eliminating initial region proposals provide usually faster and more efficient results than two-stage approaches (Girshick et al., 2013; Ren et al., 2015). The evaluation metrics related to the different neural networks are indicated in Table 4.1. It is however, visible from table 4.1 that, one-stage models sacrifice performance for detection of small objects to improve on speed.

Pose Estimation: Gauss-Newton (Lowe, 1987; Araújo, Carceroni, and Brown, 1996), Levenberg-Marquardt (Lowe, 1991; Weng, Ahuja, and Huang, 1989) and the

TABLE 4.1: Detection results on the MS COCO dataset (Lin et al., 2014b) regarding different values of IoU thresholds and variety of sizes of targeted objects. **mAP$_L$** - Mean Average Precision (mAP) with large objects, **mAP$_M$** - mAP with medium objects, **mAP$_S$** - mAP with small objects, **mAP@.50IoU** - mAP with 50% of IoU threshold, **mAP@.75IoU** - mAP with 75% of IoU threshold.

Method	**Data**	**Backbone**	**AP**	**AP$_{50}$**	**AP$_{75}$**	**AP$_S$**	**AP$_M$**	**AP$_L$**
YOLOv3	trainval35k	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD300	trainval35k	VGG-16	25.1	43.1	25.8	6.6	22.4	35.5
SSD513	trainval35k	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513	traincal35K	ResNet-101	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet800	trainval35k	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
ssd	**trainval4200**	**ResNet-50**	**0.35**	**0.7562**	**0.2472**	**0.4891**	**0.3604**	**0.2343**

orthogonal iteration (Lu, Hager, and Mjolsness, 2000) methods are three widely utilized non-linear optimization techniques to solve Perspective-n-Point (PnP) (Xiao-Shan Gao et al., 2003; Lepetit, Moreno-Noguer, and Fua, 2009) problems. However, the first algorithm is highly dependent on the initial projection and prone to failures if poorly initialized. To avoid incorrect convergence and to obtain maximal precision, non-iterative EPnP (Lepetit, Moreno-Noguer, and Fua, 2009) algorithm has been proposed to initialize non-linear approach which produces both higher stability and faster convergence. On the other hand, the Levenberg-Marquardt method can be considered as an interpolation of steepest descent and the Gauss-Newton algorithm. In case of incorrect solution, it provides slow convergence to the desired solution, behaving like a steepest descent method. When the solution is close to a correct convergence, it behaves like Gauss-Newton method. This method always guarantees convergence of the solution.

Point Cloud Registration: Several approaches are reported in literature that can be comparable in terms of accuracy and convergence time. The ICP algorithm (Besl and McKay, 1992; Chen and Medioni, 1991) is a dominant registration method for geometric alignment of three-dimensional models when an initial estimate of the relative pose is known. The ICP iteratively refines a closed form solution, minimizing an error metric by targeting maximal convergence between two data. The original approach of the ICP uses the closest point matching which results in an exhaustive search. Over the years, several improvements have been proposed to decrease time needed for convergence and improve the accuracy. These variants can be classified according to their data sampling approaches, matching and pairs weighting strategy, threshold value for match rejection and their error metrics which give various results depending on level of noise, shape features in three-dimensional data (Rusinkiewicz and Levoy, 2001). It is reported in literature that the ICP can likely diverge to a local solution depending on initial pose (Yang et al., 2016).

A probabilistic solution for both rigid and non-rigid registration problems, called Coherent Point Drift (CPD) algorithm, is provided in (Myronenko and Song, 2010). The ICP has been reinterpreted in (Jian and Vemuri, 2011) as an alignment problem of Gaussian mixtures in a robust way that statistical discrepancy measure between the two corresponding mixtures is minimized. Initial correspondence matching is necessary step for most registration algorithms including the ICP, whereas the ICP is highly dependent on a correct correspondence matching. In (Le et al., 2019) semi-definite relaxation based randomized approach is proposed as a correspondence-absent registration. To speed up the ICP algorithm, several search methods (Elseberg et al., 2012; Samet, 1989; Wehr and Radkowski, 2018) in finding the correspondences have been proposed.

4.3 Initial Pose Estimation

There are four different region of interests on a washing machine which can be targeted in the detection step, namely the Knob, the Detergent Box, the User Interface and the Glass Door. The points corresponding to the region of interest on the washing machine is obtained in two-dimensional image plane using deep learning based object detection. The accuracy of the detection is a crucial to improve the performance of the point cloud registration process as explained in the previous section. Table 4.1 shows the performance of different neural architectures trained on MS COCO (Lin et al., 2014b) dataset with respect to different values of Intersection over Union (IoU) threshold and the special average precision for the detection of small, medium and large objects. The choice of the network is motivated by the need of accuracy and tolerable detection speed of the selected architecture, whereas SSD architecture with ResNet-50 backbone (He et al., 2015) has been considered as a suitable solution.

The robot should localize the domestic appliance in order to interact with its pieces and to implement numerous tasks. Besides, our shape registration algorithm initially requires the estimated pose of the appliance with respect to the robot as already shown in Figure 4.1. For that aim, an iterative approach for the PnP problem is implemented in order to determine the position and orientation of a calibrated camera with respect to the world frame attached on the knob of the domestic appliance as shown in Figure 4.2, given four 3D-2D point correspondences and the intrinsic parameters. The main reason behind choosing the iterative approach is to have robust precision in spite of being slower solution compared to non-iterative ones. The accuracy of the initial projection can save computation time needed for overall convergence. In addition, it has huge impact on the overall accuracy of the ICP algorithm (Rusinkiewicz and Levoy, 2001). Therefore, we are feeding the estimated pose of the appliance into registration algorithm, whereas the general procedure is explained in detail in Section 4.4.

To start estimating the pose, we measure the object points of interests on the appliance with respect to the world frame. Three out of four points are the center of the bounding boxes of Knob, User Interface and Glass Door respectively which are shown with red colour in Figure 4.2. The fourth point alters depending on the robot pose. Variability of the fourth point adds stability to pose estimation task, preventing incorrect projection caused by detection so that detection on the image plane may be mismatched with the corresponding points in the world frame depending on the variety of camera poses. It causes significant pose estimation error and instability. Two possible options for the last point are indicated with blue colour in Figure 4.2.

Given the homogeneous representation of 4 known object points $p_k^w = [X_k\ Y_k\ Z_k\ 1]^T$, $k \in \{1, 2, 3, 4\}$ expressed with a reference frame attached to the observed object, the homogeneous image points $p'_k = [u_k\ v_k\ 1]^T$ with respect to the camera frame are obtained by projecting the points p_k^w into the image plane by means of a generalized

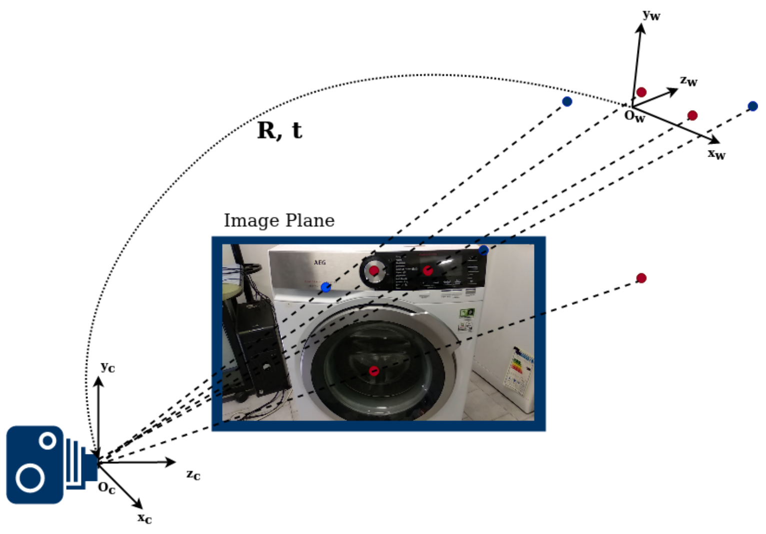


FIGURE 4.2: PnP estimation.

camera model which can be expressed as

$$p'_k = \frac{1}{s} A \, [R_{\text{init}} | t_{\text{init}}] \, p^w_k \tag{4.3.0.1}$$

where

$$A = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3.0.2}$$

is the camera matrix containing the camera intrinsic parameters, such as the focal length coefficients f_u, f_v and the principal points c_u and c_v, s is a scalar parameter and R_{init} and t_{init} are respectively the rotation matrix and the translation vector representing the relative position of the object frame with respect to the camera. The rotation matrix $R_{\text{init}} \in \text{SO}(3)$ is orthogonal, which satisfies the constraints $R^T_{\text{init}} R_{\text{init}} = I_3$ and $\det(R_{\text{init}}) = 1$. The matrix R_{init} can be specified with three consecutive rotations around the frame axis using Euler angles $\{\theta_x, \theta_y, \theta_z\}$

$$R_{\text{init}} = R_{\text{init}}(\theta_x) R_{\text{init}}(\theta_y) R_{\text{init}}(\theta_z) \tag{4.3.0.3}$$

From eq. (4.3.0.1) we define a homography matrix H as

$$H = \frac{1}{s} A \, [R_{\text{init}} | t_{\text{init}}] = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 \\ h_5 & h_6 & h_7 & h_8 \\ h_9 & h_{10} & h_{11} & h_{12} \end{bmatrix} \tag{4.3.0.4}$$

By defining the vector $\bar{h} = [h_1, \cdots, h_{12}]$ containing the elements of H, the projection of the 4 considered object points into the image plane can be represented by the function $f(\bar{h})$

$$f(\bar{h}) = \text{Blockdiag}([H, H, H, H])[p^w_1 \; p^w_2 \; p^w_3 \; p^w_4]^T \tag{4.3.0.5}$$

where $\text{Blockdiag}([H, H, H, H])$ is the block diagonal matrix having four H matrices along the diagonal. Considering that the intrinsic parameters in A are known, the elements of $\bar{h}$ can be defined as a function of the pose vector $\bar{\theta} = [\theta_x \; \theta_y \; \theta_z \; t_x \; t_y \; t_z]^T$ containing the 3 Euler angles $\{\theta_x, \theta_y, \theta_z\}$ and the three components $\{t_x, t_y, t_z\}$ of the translation vector t, i.e. $\bar{h} = g(\bar{\theta})$,

Assuming the vector b' contains 4 detected image points, the re-projection error can be computed as

$$E_{\text{init}}(\bar{\theta}) = \left\| b' - f(g(\bar{\theta})) \right\|^2 \tag{4.3.0.6}$$

An iterative solution based on the non-linear Levenberg-Marquardt optimization allows us to compute the rotation matrix R_{init} and the translation vector t_{init} in order to minimize the re-projection error, which is the sum of squared distances between the actual image points and the projected object points.

Then, we compute Jacobian matrix J of the re-projection error $E(\bar{\theta})$ by combining two Jacobian matrices J_f and J_g

$$J = \frac{\partial f(g(\bar{\theta}))}{\partial \bar{\theta}} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial \bar{\theta}} = J_f J_g \tag{4.3.0.7}$$

To minimize re-projection error, the pose vector $\bar{\theta}$ is updated recursively at each step k as

$$\bar{\theta}_{k+1} = \bar{\theta}_k + (J_k^T J_k + \lambda \, \text{diag}(J_k^T J_k))^{-1} J_k^T (\bar{b}' - f(g(\bar{\theta}_k))) \tag{4.3.0.8}$$

where $\bar{\theta}_k$ and J_k are the estimated parameter vector $\bar{\theta}$ and the related Jacobian J at the generic step k respectively, $\lambda > 0$ is a damping factor and $\text{diag}(J_k^T J_k)$ means a diagonalized matrix of $(J_k^T J_k)$. We iterate the computation until the re-projection error becomes smaller than the certain threshold, i.e. $E_{\text{init}}(\bar{\theta}_k) < \alpha$. Once the position and the orientation of the washing machine with respect to the robot is estimated, the robot can exploit this information to feed into the point cloud registration algorithm.

4.4 Point Cloud Registration

Reliability of initial transformation has a significant influence on computation time and accuracy of the ICP algorithm. To this end, feature extraction algorithms are largely used in the initial guess, but they tends to be error-prone in case of large occlusions and in the presence of significant noise in the data. In the considered scenario, removing non-appliance points from the scene by filtering has huge impact on accuracy of the result and computational complexity such that running search algorithms through each point in model data in the correspondence matching between the scene and the model points is computationally expensive, in particular in the presence of noisy data. Therefore, the scene point cloud captured by a RGB-D camera of the robot is filtered using Passthrough filter in order to decrease computation time and to increase accuracy of the registration algorithm.

We estimate the initial projection as described in the previous section (section 4.3) as opposed to the traditional approaches that require feature extraction and correspondence computation, thereby speeding up the point cloud registration process and make it more reliable. Assuming $\mathcal{B}$ is a finite set of points obtained by uniformly sampling the appliance CAD model, which points in the object frame are named $b_j \in \mathcal{B}$, $j = \{1 \ldots, N_m\}$, the initial transformation in the camera frame can be described as:

$$o_j = R_{\text{init}} \, b_j + t_{\text{init}}, \quad j = \{1 \ldots, N_m\} \tag{4.4.0.1}$$

where o_j, $j \in \{1, \ldots, N_m\}$ represents the model point cloud in the camera frame after initial transformation, we will refer to this point cloud as $\mathcal{O}_0$ in the following

After the initial transformation provided by eq. (4.4.0.1), we recursively refine the registration using the ICP algorithm. Let $\mathcal{S}$ be the scene point cloud obtained by filtering the camera data, which points are referred by s_i, $i \in \{1, \ldots, N_s\}$. Since the scene point is fixed, the center of mass of the scene cloud is computed and a new point could with origin on the center of mass is created:

$$C_{\mathcal{S}} = \frac{1}{N_s} \sum_{i=1}^{N_s} s_i, \quad \forall s_i \in \mathcal{S} \tag{4.4.0.2}$$

$$\mathcal{S}' := \{s_i{}' : s_i{}' = s_i - C_{\mathcal{S}}, \quad \forall s_i \in \mathcal{S}\}, \tag{4.4.0.3}$$

Thereafter, the algorithm is initialised by the initial projection eq. (4.4.0.1), i.e. $\mathcal{O}_k = \mathcal{O}_0$ at the initial step. Hereafter, the model point cloud is updated at each iteration k in order to achieve a suitable alignment of the model with the point cloud provided by the 3D camera, that is considered fixed. In order to find the correspondences between closest points between model and scene point clouds, the conventional ICP algorithm computes the Euclidean distance between each point in model set $o_j \in \mathcal{O}_k$ and each scene point $s_i \in \mathcal{S}$. This generates a new point cloud

$$\mathcal{M}_k := \left\{ m_i : \min_{o_j} \left\{ \sqrt{o_j^2 - s_i^2} \right\}, \; \forall s_i \in \mathcal{S}, \; \forall o_j \in \mathcal{O}_k \right\} \tag{4.4.0.4}$$

composed by N_s points o_j from the model point cloud $\mathcal{O}_k$ representing the point having the smaller distance to their counterpart in the filtered scene point cloud $\mathcal{S}$. It results that the conventional ICP algorithm (Besl and McKay, 1992) looks through every model point in order to compare distances and to find nearest neighbor related to given source point. This approach causes great computational complexity in the order of $O(N_S N_M)$, i.e. the computation time increases proportionally with respect to the product between the number of points in each set. To improve the effeciency of the algorithm, the model point cloud is partitioned into an optimized data structure by exploiting OcTree, a special type of space partitioning which provides faster solution in nearest neighbor searches in many applications. OcTree speeds up this process by subdividing recursively each node in a tree into eight children, as it implements searching only through points inside its octant. The average computation time for the nearest neighbor search using OcTree structure emerges to be in the order of $O(N_S \log N_M)$.

To converge to the global minimum in the registration, one useful approach is filtering correspondences (Holz et al., 2015) to reduce the number of the false matches, once corresponding pairs of the points are determined. There are several filtering policies of correspondence rejection based on distance, duplicity, surface properties and statistics. In this work, the outliers are eliminated according to their distance in GPU-based algorithms. It filters out the matches with a distance larger than given threshold where it is also formulated in the original ICP algorithm, see Figure 4.3.

The center of mass of the model cloud is found and subtracted from each model point m_i at each iteration:

$$C_{\mathcal{M}} = \frac{1}{N_s} \sum_{i=1}^{N_s} m_i, \quad \forall m_i \in \mathcal{M}_k \tag{4.4.0.5}$$

$$\mathcal{M}'_k := \{m_i{}' : m_i{}' = m_i - C_{\mathcal{M}}, \quad \forall m_i \in \mathcal{M}_k\}, \tag{4.4.0.6}$$

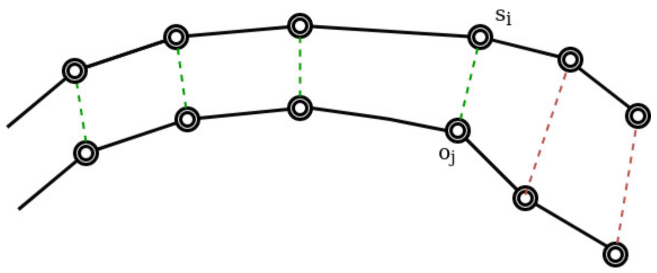


FIGURE 4.3: Outliers filtering: Rejection based on the distance between the points

The goal in the registration process at each k-th iteration is to determine best fit transformation T_k between point clouds defined as

$$T_k = \begin{bmatrix} R_k & t_k \\ 0\,0\,0 & 1 \end{bmatrix} \tag{4.4.0.7}$$

that allows to optimally match the model point cloud with the one provided by the 3D camera. To achieve this goal, the convergence to zero of the following point-to-point error metric must be ensured

$$E_{\text{icp}}(R_k, t_k) = \frac{1}{N_s} \sum_{i=1}^{N_s} w_i \, \| m_i - R_k \, s_i - t_k \|^2 \tag{4.4.0.8}$$

where the weighting factor w_i can be used to normalize the matches in the least squares formulation. The proposed algorithm then updates the rotation matrix R_k and the translation vector t_k in the transformation matrix T_k by means of Singular Value Decomposition (SVD). To do so, the point clouds are organized in the column vectors M_k, M'_k and S' containing the points of $\mathcal{M}_k$, $\mathcal{M}'_k$ and $\mathcal{S}'$ respectively. Then, to define the optimal transformation, the cross-covariance matrix is computed in order to apply the decomposition:

$$P = M'_k S'^T \tag{4.4.0.9}$$

By applying SVD to P

$$P = U \Lambda V^T \tag{4.4.0.10}$$

the rotation matrix R_k and the translation vector t_k can be computed as

$$R_k = UV^T \tag{4.4.0.11}$$

$$t_k = C_{\mathcal{M}} - R_k C_{\mathcal{S}} \tag{4.4.0.12}$$

Assuming M_k^h is homogeneous representation of the vector M_k, the model point could can be aligned with the filtered scene point cloud using the inverse of T_k

$$M_{k+1}^h = T_k^{-1} M_k^h \tag{4.4.0.13}$$

Considering the orthogonality condition of the rotation matrix $R_k^{-1} = R_k^T$:

$$M_{k+1}^h = \begin{bmatrix} R_k^T & -R_k^T t_k \\ 0\,0\,0 & 1 \end{bmatrix} M_k^h \tag{4.4.0.14}$$

This procedure is iterated until the termination criteria is satisfied. Termination criteria (Holz et al., 2015) can be determined according to maximum number of iterations, relative minimum transformation threshold between iterations, maximum number of similar iterations and the absolute or relative value of error metric, i.e. $E_{\text{icp}}(R_k, t_k) < \alpha$, where α is a positive threshold. Since this work aims to compare different variants of the ICP algorithms, the refinement of the scene data is processed recursively until the given iteration count is obtained.

4.5 Experimental Results

Object Detection: Our data set consists of 3700 training and 500 validation images. The data samples are collected using a camera mounted on a mobile robot. Training images have been labelled in both automatic and manual ways. The automatic labelling is based on the fiducial markers (Garrido-Jurado et al., 2014; Romero-Ramirez, Muñoz-Salinas, and Medina-Carnicer, 2018) and annotation type with 2500 samples of training data in order to increase the number of samples and speedup labelling process. The rest of training images have been labelled manually on the marker-free scenes to increase accuracy of detection and to avoid the network learning the marker itself.

SSD architecture with ResNet-50 backbone is retrained on NVIDIA GeForce RTX 2080Ti graphics adapter over 65K steps using TensorFlow library (al., 2015). The learning rate is initially assigned to 0.04. However, it adaptively decreases over training process relative to the number of an iteration in order to provide a smooth convergence of the cost function to zero value. We set the momentum optimizer coefficient γ to 0.9. All the input images are normalized and augmented by rotation, flipping and contrast variance. Table 4.2 indicates that training with our dataset shows satisfying results for large objects.

Pose estimation: Once the localization of the corner points of the bounding boxes surrounding targeted appliance parts on the image plane through the detection module is achieved, four points on the image plane which are not preserving collinearity condition are selected. Those points of interest are directly fed into pose estimation algorithm for each frame of the scene. This implementation avoids additional computation for feature extraction and correspondence matching between scene and model images. Given three-dimensional points in the world frame and the selected points on the image plane, SSD integrated algorithm provides only four correspondences which results in a reliable and stable pose estimation.

The results of the pose estimation algorithm can be seen in Figure 4.5 and in Figure 4.7 respectively. The pose accuracy becomes satisfying due to iterative non-linear optimization. The stability of pose allows us to implement the initial projection of point cloud which highly influences the accuracy of the ICP registration algorithm.

TABLE 4.2: mAP values after 65K steps of training process. $\mathbf{mAP_L}$ - mAP with large objects, $\mathbf{mAP_M}$ - mAP with medium objects, $\mathbf{mAP_S}$ - mAP with small objects, **mAP@.50IoU** - mAP with 50% of IoU threshold, **mAP@.75IoU** - mAP with 75% of IoU threshold.

mAP	**@.50IoU**	**@.75IoU**	$\mathbf{mAP_L}$	$\mathbf{mAP_M}$	$\mathbf{mAP_S}$
0.3537	0.7562	0.2472	0.4891	0.3604	0.2343

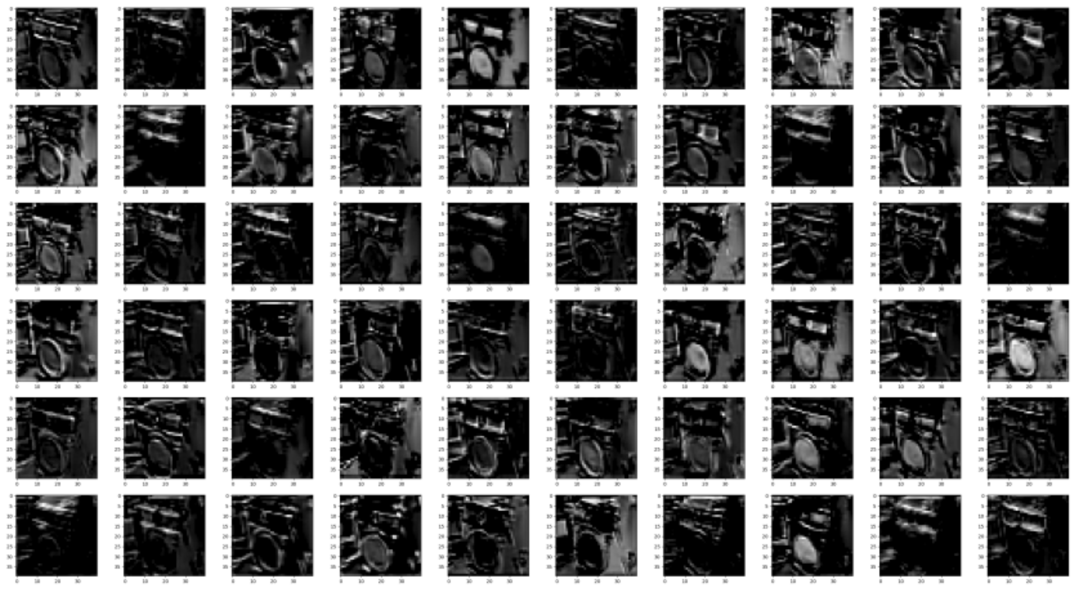


FIGURE 4.4: Washing machine: Feature maps fetched from the certain hidden layer of the neural network.

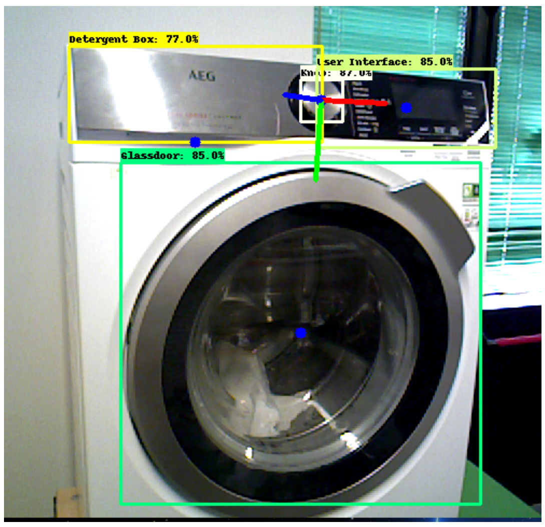


FIGURE 4.5: Washing machine pose.

Besides, localizing a domestic appliance in a three-dimensional space with respect to the robot allows us to implement manipulation tasks on the appliance.

Point cloud registration: Table 4.3 reports the comparison of the proposed method to various of registration approaches to highlight their differences in computation time of the convergence with respect to the device on which they are implemented, i.e. CPU or GPU. Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz has been deployed for the CPU-based operations. GPU-based parallel programming is implemented in order to create a registration structure handling the ICP algorithms using conventional and OcTree search methods. The GPU algorithms are conducted using NVIDIA GeForce RTX 2080Ti graphical adapter. The 3D model and the scene point set are fetched using PCL library and stored initially in memory for the CPU operations. However, they are transferred in the graphical adapter's memory addresses to implement CUDA-based algorithms. The threshold distance for correspondence rejection has been set to 0.6 in GPU-based implementations. Multiple scenarios are

TABLE 4.3: Comparison between the algorithms. Scene point cloud: 12599 points, Model point cloud: 117542 points.

Algorithm	**Initial Projection [s]**	**ICP [s]**
ICP CPU Feature-based	10.857	5.522
ICP CPU PnP	0.268	4.106
ICP GPU conventional	0.266	1.597
ICP GPU OcTree	**0.258**	**0.184**

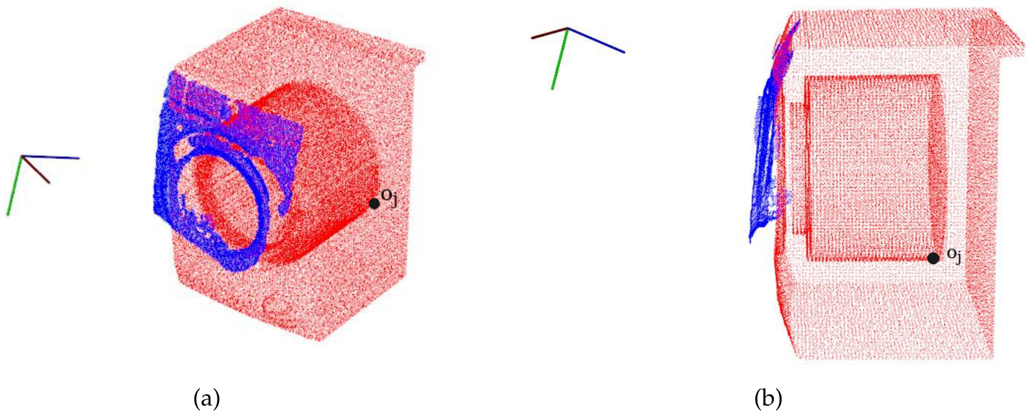


FIGURE 4.6: Projection of the model cloud after initial estimation. **blue**: filtered scene point cloud, **red**: the model point cloud after DL-based initial transformation, $\mathbf{o_j}$: the model point in the camera frame.

considered to validate the efficiency of our algorithm, i.e with open and closed glass door cases and with the presence of occlusions in the scene. The ICP computation performance in terms of time with GPU implementations is about 30 times faster than their CPU counterparts. If the initial pose estimation is taken into account, feature-based estimation takes up to 42 times more compared to a deep neural network employed in our work.

The model point cloud contains three-dimensional points of 117.542 after uniform sampling of the mesh data, which is shown in red colour in Figure 4.6. Considering the original representation of the scene point cloud contains the high number of non-appliance points fetched directly by the robot camera, the effect of filtering can apparently be seen in Figure 4.6 such that it completely eliminates background points from the scene data. One of the main impact of filtering in our experiment is a glass door-open configuration of the appliance. The registration algorithms tend to diverge to a local minimum in the presence of a glass door of the washing machine without priori filtering. The reason behind is absence of the glass door in our model, causing incorrect mismatches between point clouds. However, filtering provides the algorithm to converge to desired solution both in opened and closed configurations of the glass door.

In our experiments, a vague estimation of transformation between the point clouds for the initial projection is conducted in the absence of feature extraction. The estimation of pose is conducted with four 3D-2D point correspondences, whereas image points obtained by deep learning based object detection method. The model

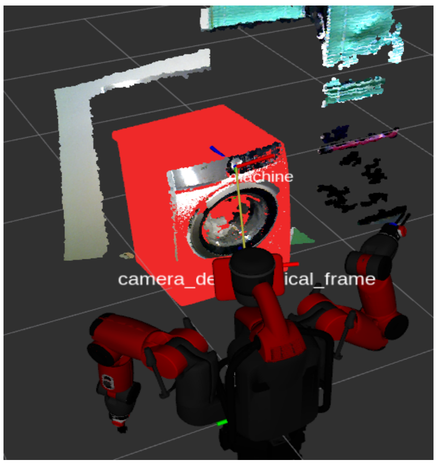
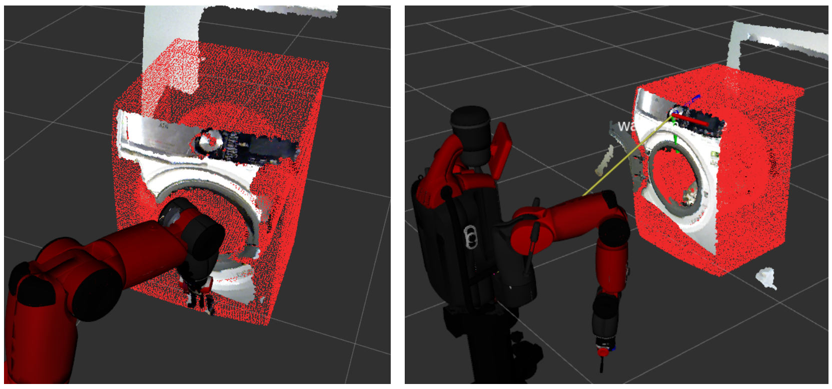


FIGURE 4.7: The results of the ICP algorithm. The closed glass door configuration.

point cloud after initial transformation is represented with green points in Figure 4.6. The final alignment of the model onto the scene point set, in case of the closed glass door configuration of the appliance, is represented in Figure 4.7.

In order to see the effect of our solution on the accuracy of the ICP algorithms, we also implemented feature-based calculation of the initial projection based on FPFH (Rusu, Blodow, and Beetz, 2009) descriptors. The results in multiple trials show a superiority of deep learning based approach over the classical one. It allows us to skip computational complexity for feature matching in the correspondence estimation step and gives higher accuracy with respect to former method. Table 4.3 indicates a computational difference in terms of time complexity between feature-based approaches and our algorithm. While determining the correspondences and estimating initial transformation took 10.857 seconds on average after 6 trials, it lasted averagely only 0.258 and 0.266 seconds at the same number of trials in GPU-based implementations. In addition, feature-based approach is highly sensitive to camera noise and partial occlusions. In such cases, feature-based algorithms produce less success rate in multiple trials due to the lack of information. On the other hand, the 3D model could align to the scene point set in every trial using our approach despite of the presence of occlusions and the state of open glass door as shown in Figure 4.8.



(a) (b)

FIGURE 4.8: Different states of the washing machine: occluded (a), open glass door configuration (b).

4.6 Conclusions

In this work a two step 6d shape registration problem of large objects is addressed. We exploited both the speed of OcTree search method in CUDA environment and featureless characteristic of initial estimation of the object pose. Integrating the solution of PnP problem into the registration allows us to eliminate computational burden for the initial transformation. In addition, the approach gives accurate registration in the presence of noisy point clouds. While the presence of occlusion and noise can have a significant influence on the results of the registration in the classical methods, our proposed method demonstrated robustness in these scenarios. Using the OcTree structure in the GPU-based implementation decreased the duration of the registration process almost 30 times than time needed for the CPU-based implementation. Future activities will be devoted on the generalization of the process to different scenarios using multiple large objects and on the automatic definition of features for the initial alignment.

Chapter 5

Robotized Laundry Manipulation and Appliance User Interface Interpretation

5.1 Introduction

The need for automating various stages laundry washing and related operations has many useful application in domestic as well as industrial scenarios. In particular, appliance manufacturing industries conduct accelarated life test (ALT) on the small fraction of appliances statistically sampled from the production line and are supervised by a human operators. The test cycle usually involves loading and unloading of clothes and adjusting the washing cycle through the display interface. In this scenario, the deployment of robotic system requires autonomous capability to detect the graspable regions of the cloth and target appliance and robust user-interface interpretation.

The detection and manipulation of deformable objects (DOs) like clothes has a wide range of applications such as cloth washing, ironing and folding tasks (Jiménez and Torras, 2020). Clothes are DOs characterized by having one dimension considerably smaller than the other two (i.e. the thickness of the fabric) (Sanchez et al., 2018). The challenge of manipulating and sensing DOs is massive due to their intrinsic property of being deformable. In fact, their shape and appearance change spatially and over time. This implies that the vast majority of the approaches and algorithm developed for rigid objects need to be modified or are not applicable at all to DOs. A comprehensive review of the literature on sensing and manipulation of DOs is provided in a recent survey (Sanchez et al., 2018). Extensive work has also emerged in the literature specifically related to cloth(es): state estimation of clothes (Kita et al., 2011); grasp point detection (Ramisa et al., 2012) and (Yamazaki, 2014); manipulation for garment picking-up (Shibata, Ota, and Hirai, 2009) and (Monsó, Alenyà, and Torras, 2012), manipulation for garment reconfiguration (Cusumano-Towner et al., 2011) and (Doumanoglou et al., 2014). These previous works either targets a particular region of the cloth (colar, sleeve) or are trying to extract particular features such as color for identifying graspable areas. While, In this work we directly operate on a live pointcloud thereby doesn't rely on the these specific features for grasp pose detection. By developing algorithm to detect important 3d shapes in the cloth (wrinkles and blobs) from 3d pointcloud, the proposed approach can generate stable grasp from a large of clothes.

In this work, there are two core algorithms developed: The first one is a pointcloud-based algorithm for the identification of optimal grasping poses based on wrinkles and blobs in a set clothes inside a bin or during recovery. The second algorithms developed in this chapter deals with the user interface interpretation for washing cycle setting using deep neural networks.

5.2 Laundry Manipulation Strategy

The laundry operations can be generalized as picking laundry from a source container, e.g. a bin or the washing machine drum, and placing laundry inside a destination container, i.e. the washing machine drum or a bin respectively. Moreover, two intermediate undesired conditions are considered: 1) a portion of cloth lays outside the washing machine door; 2) a cloth drops to the floor during manipulation between the source and the destination container. A generalization of the proposed laundry manipulation strategy is provided in Alg. 4. We assume in this work that the approximate location of the source and destination container is known.

Algorithm 4: Laundry Manipulation

```
Result: Empty Source Container
Input: Source Container Type $C$
SourceNotEmpty = True
do
    for $C_{tmp} \in \{C, Door, Floor\}$ do
        MoveToSourcePOVLocation($C_{\text{tmp}}$)
        $P_I$ = GetPointCloud()
        $\mathcal{Z}$ = GraspPoseDetection($p_{\text{ref}}(C_{\text{tmp}}), P_I$)
        if $\mathcal{Z} \neq \emptyset$ then
            LaundryPickAndPlace($C_{\text{tmp}}, \mathcal{Z}$)
        else
            if $C_{tmp} = C$ then
                SourceNotEmpty = False
while SourceNotEmpty
```

The laundry task is basically the repetition of the same steps specialized each time for the specific scene, i.e. the bin, the drum, the door and the floor. Note that the bin and the drum are mutually exclusive and passed as input (line 1) depending on the source container, i.e. if a loading or a unloading phase is considered. Then the task starts by considering the source container: the robot moves to the desired scene point of view (line 5) and the pointcloud P_I is acquired (line 6); the grasp pose detection is then executed on P_I taking into account the specific requirements of the current scene C_{tmp} (line 7); if some laundry is detected in the scene, the grasps in $\mathcal{Z}$ are ranked based on specific properties depending on the current scene C_{tmp} and the best one is selected to execute the laundry pick and place (line 9). In case the input container C is empty, i.e. no grasp is found, the task terminates.

5.3 Grasping Pose Detection

Two complementary approaches for grasp poses identification for cloth-like objects are discussed in this section. The first method was a Wrinkle detection in 3D scene. The second approach is based on identifying graspable region: based on blob of graspable points in the pointcloud. To improve grasp success, both wrinkle and

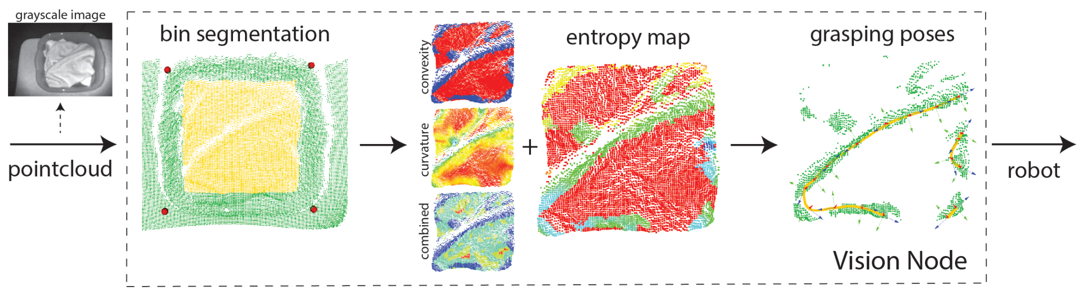


FIGURE 5.1: Wrinkle detection algorithm. The input pointcloud is segmented retrieving only the interior point of the bin (yellow). The entropy map is build utilizing the knowledge embedded into the convexity, curvature and combined (depth and edges) maps. The grayscale image is shown for visualization purpose only.

blob based approaches are combined to generate a set of graspable poses $\mathcal{Z}$. A cost function based pose scoring is finally utilized to rank all the poses in $\mathcal{Z}$.

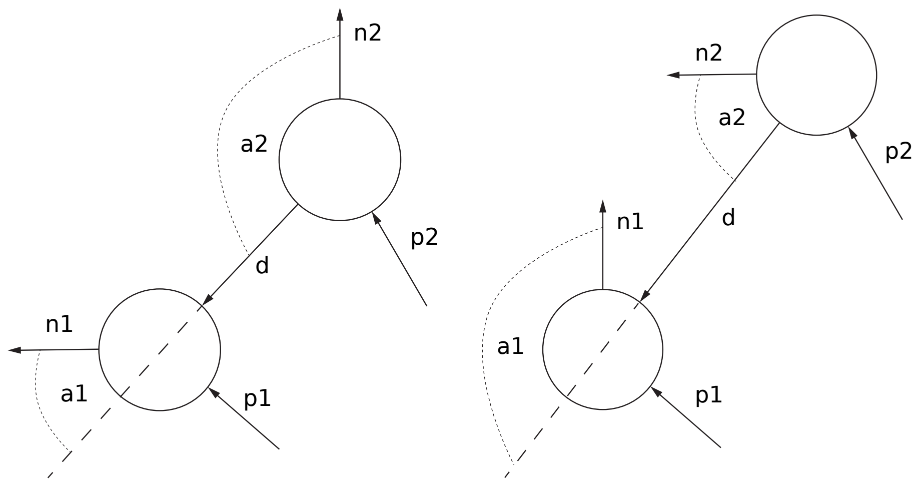
5.3.1 Wrinkle Detection

When dealing with the problem of grasping cloth-like objects, wrinkles are distinctive features for grasp where the information about the wrinkledness is embedded in the 3d surface topology and therefore requires a 3d sensor to capture it. The wrinkle detection algorithm presented here targets the task of bin picking and recovery grasp which consists of four distinct steps. In the first step, the segmentation of the source pointcloud is performed by identifying the bin in the scene and segmenting only the region in the pointcloud that are associated to the clothes only. The second step applies a wrinkledness measure combined with additional cues (convexity and depth) in order to find the areas of the segmented pointcloud with wrinkle-like structures. The third step is responsible for the creation of graph structure and path building in each detected wrinkle area. The last step, estimates a grasping pose for each piecewise wrinkle-path. The starting with live scene pointcloud, the algorithm produces poses along wrinkles described by piecewise curves (see Fig 5.1). In the following the main components of the algorithm are discussed. The segmented pointcloud is processed in the second step in order to detect graspable regions. A graspability measure is employed to gain an understanding of the location of highly wrinkled areas in the cloth. This information is encoded into an *entropy map*. A *depth map* and *convexity map* are used as auxiliary cues to robustify the detection of the wrinkled areas.

Normals Estimation

The first step in the detection of the regions is based on low-level features as the surface normals of the pointcloud. They are computed relying on the Moving Least Squares algorithm (Alexa et al., 2003) which smooth out the pointcloud surface by fitting a polynomial curve before estimating the surface normals. By using this approach, a reduction in the noise in the estimation process is obtained.

The normal vectors obtained are expressed in Cartesian coordinates as (n_x, n_y, n_z). They are transformed in spherical coordinates since only two components are relevant. The transformation involved is documented in (Rusu, 2010). The pair of angles



(a) Convex (b) Concave

FIGURE 5.2: Drawing showing the convex and concave conditions.

(ϕ, θ) are calculated as: $\phi = \text{atan}\left(\frac{n_z}{n_y}\right), \quad \theta = \text{atan}\left(\frac{\sqrt{n_z^2+n_y^2}}{n_x}\right)$ The angle ϕ is denoted as azimuth angle while θ as inclination angle.

Convexity Map

Detecting the concavity or convexity of a local area is important to remove from the highly wrinkled areas the regions that are not easily graspable. The method presented here is capable of providing such understanding with a very small computation footprint. This procedure can determine if a point's neighborhood is convex or concave (Christoph Stein et al., 2014). Given the input and normal vectors pointclouds in Cartesian coordinates, for each point a local neighborhood is found by choosing local patches composed by 9 points. Focusing on a given patch, lets denote by $\vec{p}_1$ the (x, y, z) coordinates of the considered point and by $\vec{n}_1$ its normal. Lets $\vec{p}_2$ and $\vec{n}_2$ be in turn one of its neighborhood points. The distance vector between $\vec{p}_1$ and $\vec{p}_2$ is computed as $\vec{d} = \vec{p}_1 - \vec{p}_2$. Then, the angle α_1 between $\vec{n}_1$ and $\vec{d}$ is compared with the one α_2 between $\vec{n}_2$ and $\vec{d}$. A convex connection between $\vec{p}_1$ and $\vec{p}_2$ is defined if α_1 is smaller than α_2. The condition can be expressed as:

$$\alpha_1 < \alpha_2 \Rightarrow \cos(\alpha_1) - \cos(\alpha_2) > 0 \iff \vec{n}_1 \cdot \hat{d} - \vec{n}_2 \cdot \hat{d} > 0 \tag{5.3.1.1}$$

$$\hat{d} = \frac{\vec{p}_1 - \vec{p}_2}{||\vec{p}_1 - \vec{p}_2||}$$

If the condition (5.3.1.1) is not satisfied, the two points will exhibit a concave connection between them. The computation is performed for all the neighborhood points that satisfy a check based on the normal vectors difference angle: the convex connectivity is calculated only if the two normal vectors n_1 and n_2 have a significant angle difference between them. The original point is set to be convex if all of its neighborhood exhibit a convex connection with him. Figure 5.2 displays the convex and concave conditions. The results show that this simple approach is able to detect convex regions as wrinkles and edges in the clothes, at least in an approximated way. The combination of the entropy filter with the convexity check allows the robust detection of convex wrinkles only.

Depth Map

The depth map is built by using a reference plane and by evaluating the distance between each point in the input pointcloud and this plane. The choice of the reference plane depends on the scenario. In the bin picking, the knowledge of the four top vertices location is used for the computation of the bin-top plane. The point to plane distance is computed for all the points of the segmented pointcloud. At the height of the point corresponding to the largest distance is fit a plane parallel to the bin-top one using RANSAC. This plane is used as reference level for the calculation of the map.

Entropy Filter

The entropy filter is employed in order to quantify how much information exists in a given local region. In particular, the goal is to discover regions of the clothes with a sparse distribution of normals. They will result in a high value in the entropy measure. Instead, regions with normals mostly aligned with each other will be characterized by a low value in the entropy measure. For each point in the input pointcloud a local region is considered and a two-dimensional histogram is constructed. The histogram is built with the two spherical components of the surrounding normal vectors. Hence, It is used to model the spherical coordinate angles distributions. The entropy measure is defined as:

$$H(x) = -w_x \sum_{i=1}^{n} p(x_i) \log p(x_i) \tag{5.3.1.2}$$

where x is the point considered to which a two-dimensional histogram of orientation angles in spherical coordinates (azimuth and inclination) is associated. The histogram is made of n bins for each dimension. The parameter w_x is the weight related to point x. With x_i we are denoting the i-th bin of the histogram and with $p(x_i)$ its associated value. The weight factor comes from the depth map (Sec. 5.3.1). In particular, w_x represent the intensity value in the depth map of the point considered. As the point is far away from the reference plane, its associated intensity value is larger and the weight factor increases. As result, the points that are more distant from the plane are preferred. If the not weighted version of the formula is needed, w_x can be set equal to one.

Wrinkles as Interpolated Splines

To improve the original piece-wise curve fitting reported in (Caporali and Palli, 2020) on the nodes, the new idea exploited here is to build an undirected graph $G = (V, E)$ where the nodes (V) are interconnected by the edges (E) based on some properties. These nodes are ordered according to a predefined sequence and a spline curve is interpolated to them.

The supervoxel clustering algorithm (Papon et al., 2013) is utilized to create the graph structure, by applying it to the projected segmented entropy map (i.e. the output of Entropy Filtering step described in Section 5.3.1.

The algorithm provides, for each supervoxel, its centroid points and the set of adjacent neighbours. The supervoxels centroids will be the nodes of the graph, the adjacency information is used for the introduction of edges. Each node is also augmented with an *intensity* attribute related to the wrinkledness score of the related

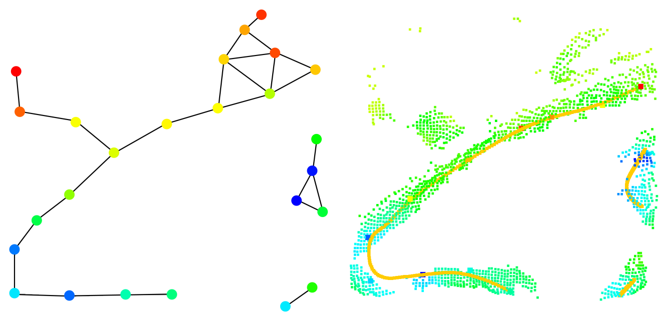

(a) (b)

FIGURE 5.3: Graph structure (a) and interpolated spline (b). In (a), the points in the graph denote the supervoxels centroids and the segments connecting them resemble the graph edges. The different colors in the nodes highlight their different intensities values. In (b), the segmented entropy map is shown in the background with the spline in yellow.

area in the entropy map. Figure 5.3(a) displays the result of the clustering with the generated graph.

The obtained graph is clustered based on connected components resulting into a set of clusters (subsets) $\mathcal{C}_i$, $i = 1, \ldots, c$ where c is the number of clusters, such that $G = \cup \mathcal{C}_i$.

The path building process reported in Alg. 5 build a path $\mathcal{P}_i = \{v_{i,1} \ldots v_{i,l}\}$ over the generic i-th cluster $\mathcal{C}_i$ as an ordered set of distinct alternating nodes and edges, where l is the total number of nodes in $\mathcal{P}_i$. First, the node of $\mathcal{C}_i$ exhibiting the maximum intensity value is selected as v^* (line 1). The adjacent nodes of v^* are stored in $\mathcal{N}$ via the function adj(v^*) (line 2) and the two nodes with the larger intensity value are selected from $\mathcal{N}$ (lines 3 and 5). Thus, two partial paths are built (lines 8 to 20) starting from v^* and its adjacent nodes with the higher intensity (line 8). Then, an iterative procedure adds nodes in $\mathcal{P}_t$ based on a score s_j computed as the product between the intensity and curvature scores (line 14) considering the last path node and each of its adjacent nodes. In fact, given a sequence of node positions, i.e. $\{v_{i-1}, v_i, v_{i+1}\}$, the angle between the two 3D vectors $\vec{d_i} = v_{i-1} - v_i$ and $\vec{d}_{i+1} = v_{i_i} - v_i$ is $\gamma_i = \Gamma(\{v_{i-1}, v_i, v_{i+1}\}) = \vec{d_i} \cdot \vec{d}_{i+1}$. Given a Von Mises distribution $M(\cdot)$ centered on π and with variance 1, the curvature score is defined as $s_j^c = M(\gamma_i)$. In The sequence $\{v_{l-1}, v_l, v\}$, denoting the last two elements of the path and the candidate node under test, is considered in the computation of the curvature score (line 13). Since $M(\cdot)$ is concentrated around the value of π, it is more likely to select a straight line path. The node with the greatest product score v_{j^*} is chosen as next node and added to the path (line 16). If all the candidate nodes $v \in \mathcal{N}$ have a product score of zero, the path is closed and the loop stops. If only one neighbor is present in $\mathcal{N}$, then the method described consists simply in checking its curvature and intensity, similarly to a flood-fill algorithm with threshold. Finally, the path $\mathcal{P}_i$ is obtained just by merging rev$(\mathcal{P}_1)$, i.e. the reversed set obtained form $\mathcal{P}_1$, and $\mathcal{P}_2$, using v^* as junction point (line 20).

Spline Interpolation

The nodes centroids are translated from the projected space back to the original 3D space. Then, the nodes of each $\mathcal{P}_i$ are fed as set of control points to be interpolated

Algorithm 5: Wrinkle Path Building

```
   Input: $\mathcal{C}_i$
   Output: $\mathcal{P}_i$
1  $v^* \leftarrow \arg\max_v(\{I(v), \forall v \in \mathcal{C}_i\})$
2  $\mathcal{N} \leftarrow \mathrm{adj}(v^*)$
3  $v_1^n \leftarrow \arg\max_v(\{I(v), \forall v \in \mathcal{N}\})$
4  $\mathcal{N} \leftarrow \mathcal{N} \setminus v_1^n$
5  $v_2^n \leftarrow \arg\max_v(\{I(v), \forall v \in \mathcal{N}\})$
6  $\mathcal{P}_\mathrm{t} \leftarrow \emptyset \ \forall t \in \{1, 2\}$
7  for $\mathcal{P}_t$ do
8      $\mathcal{P}_\mathrm{t} \leftarrow \{v^*, v_t^n\}$
9      $\mathcal{N} \leftarrow \{\mathrm{adj}(v_t^n) \setminus \mathcal{P}_\mathrm{t}\}$
10     do
11         $\mathcal{S} \leftarrow \emptyset$
12         for $v \in \mathcal{N}$ do
13             $s^c = M(\Gamma(\{v_{l-1}, v_l, v\}))$
14             $s = I(v)s^c$
15             $\mathcal{S} \leftarrow \mathcal{S} \cup s$
16         $j^* = \arg\max_j(\mathcal{S})$
17         $\mathcal{P}_\mathrm{t} \leftarrow \mathcal{P}_\mathrm{t} \cup v_{j^*}$
18         $\mathcal{N} \leftarrow \mathrm{adj}(v_{j^*}) \setminus \mathcal{P}_\mathrm{t}$
19     while $\mathcal{N} \neq \emptyset \cup s_{j^*} \neq 0$
20 $\mathcal{P}_i \leftarrow \mathrm{rev}(\mathcal{P}_1) \cup \mathcal{P}_2$
21 return $\mathcal{P}_i$
```

by a spline in the 3D space with a degree of 2. Notice that this is an approximated solution and different interpolation strategies can be implemented to refine it. The result of the interpolation for a sample pointcloud is denoted in Fig. 5.3(b) as a yellow curve.

5.3.2 Extension to Washing Machines

Robotized insertion of clothes into the washing machine drum is a complex task. Due to many factors related to the initial point of grasp and topology/dimension of the cloth considered, it may happen that, after the insertion, a portion of the object lays outside the opening door of the washing machine drum. A possible strategy for the identification and removal of clothes laying outside the drum door is here discussed by extending the approach presented in Sec. 5.3 about the bin segmentation.

The idea consists of using a pre-computed pointcloud model P_O of the washing machine for calculating a new pointcloud, named *difference map* P_D, as the difference between P_O and the current scene P_S pointclouds. The obtained P_D can be used for understanding if a misplaced cloth is present. Then, the grasping pose identification algorithm presented in Sec. 5.3 can be employed for computing the recovery poses for inserting it. Fig. 5.4 provides a summary view of the approach described in this section.

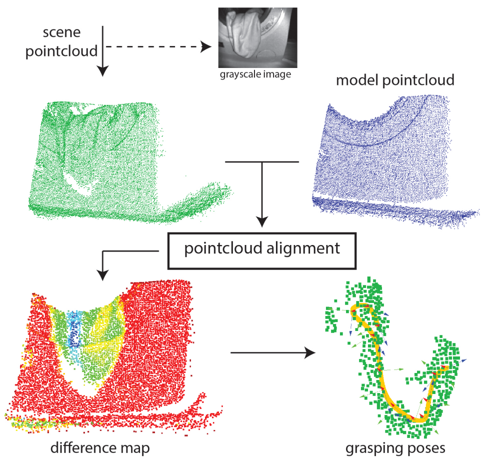


FIGURE 5.4: Schema of the vision approach for the washing machine recovery picking. Grayscale image shown for clarity.

Pointclouds Registration

In order to calculate P_D, P_S and P_O should be aligned (i.e. registered). In particular, we need to find the transformation that would bring P_O to overlap P_S, obtaining the aligned model pointcloud $P_{\hat{O}}$. Notice that P_O is computed offline and represents a portion of a washing machine with the door opened. The alignment operation can be split into 1) the problem of determining the initial (rough) transformation between P_0 and P_S, and 2) the optimization of the alignment.

The initial alignment is obtained by exploiting the Samples Consensus Prerejective (SCP) (Buch et al., 2013) method. It evaluates the correspondences between feature points, as features we select the Fast Point Features Histograms (FPFH) (Rusu, Blodow, and Beetz, 2009), and provides the initial guess for the transformation. This is refined by the Iterative Closest Point (ICP) algorithm (Rusu, 2010) which minimize the Euclidean distance error metric between the overlapping areas of the pointclouds.

Difference Map

A reference plane is estimated from $P_{\hat{O}}$ and both pointclouds are projected on this plane. Each point in P_S is matched with a point in $P_{\hat{O}}$ and the two point-to-plane distances between each of them and p_{ref} are computed. Then the distances are evaluated and their difference stored inside P_D, Fig. 5.4 provides an example of *difference map* where the point intensity values are encoded in the color-map (reddish means close to zero). Thus, P_D is used to display the regions of the washing machine where $P_{\hat{O}}$ and P_S differ the most. P_D is segmented by discarding all the points with a negligible difference (distance) based on an user-defined threshold. Notice that if all the points in the difference map do not satisfy the threshold, then this can be interpreted as a signal of not presence of misplaced cloth.

5.3.3 Blob Detection

Blobs are another important features that could easily be computed to augment scenarios where the wrinkledness criterion produces very small number of graspable poses. This happens in situations where clothes form a flat top whith other laundry pilled up underneath. Therefore, By combining wrinkledness measure and blob

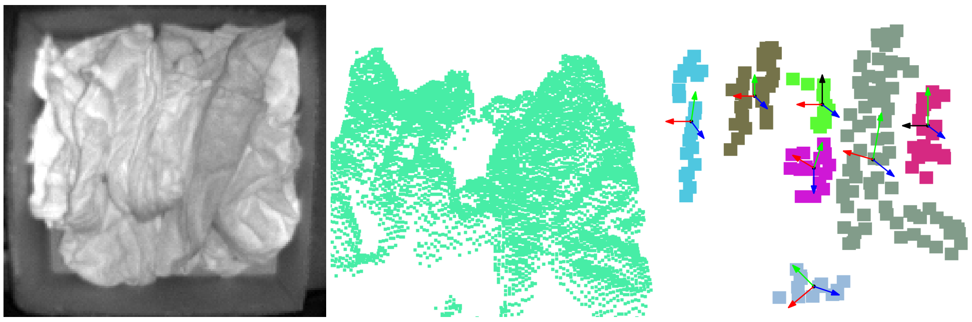


FIGURE 5.5: Grasp pose detection using blob regions. The colored set of points represent the cluster of blobs with their principal and normal direction computed for grasp pose estimation.

detection, the laundry grasping algorithm gives higher rate of success. Due to deformable nature of clothes, a robot can form the grasp shape by pushing against the laundry where there are enough surface to make contact, particularly when parallel grippers are utilized. In this regard, a set of densely populated clusters of points i.e. blobs can offer another alternative.

To extract the blobs from the segmented pointcloud, we first search for the local maxima points with respect to other points within predefined radius along the normal direction from the surface of the container. To create graspable regions, we perform clustering to the locally maximum points, effectively culling regions that are not suitable for the grasp. For each cluster, a centroid and Principal component directions are computed to determine the grasp point and it's orientation (see Fig. 5.5). By integrating the wrinkle and blob based techniques, there are a number of graspable points of which the optimal one is selected for grasp execution using the robot. To identify optimal grasp pose, a score is computed for each of grasp points based on their location in the container, the direction of the wrinkle or blob w.r.t the container. In the case of recovery grasp, the poses ranked by considering the point along the outlaying part of the laundry.

5.4 User Interface Interpretation

Another important problem to autonomous robotic laundry operation is the robust interpretation of the digital display and setup of the washing cycle. Performing user interface interpretation from a mobile robotic platform using conventional 2d cameras has its own challenges. The angle view, lighting condition and reflection from the glass screen are all varying during the operation. These factors has made the traditional computer vision approach (feature based) difficult to utilize in our scenario and a deep convolutional neural network based approach is proposed.

The First step in this regard is to Identify important coordinate points on the display. This requires matching points between the scene image and model image of the display by computing the homography matrix $\mathcal{H}$. The output of this step will be coordinates of digits, symbols and leds which indicate preferred washing program and options of the washing machine. The second task will be recognizing the value of digits and identify programs, functions and option at which the washing machine is working. For this, a MobilenetV2 convolutional neural network architecture has been used to recognize 10 number of digits and additional two symbols ('-','h') that possibly appear on the appliances digital display.

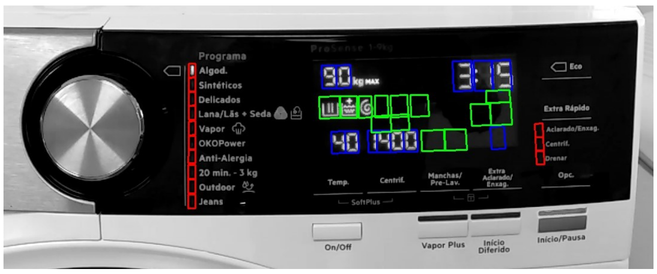


FIGURE 5.6: The points of interest on the reference image (red, green and blue dots marking the coordinates of programs, functions and digits respectively).

Point of Interest Coordinate Estimation

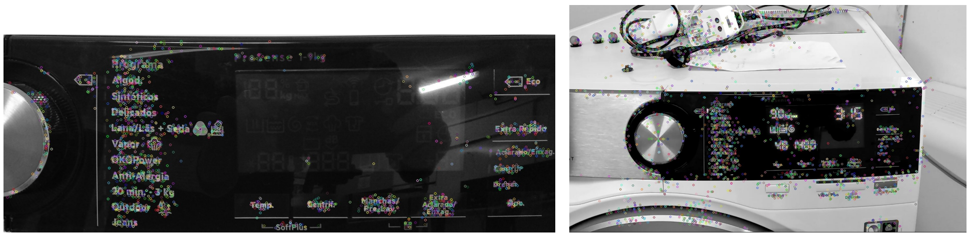
These are Points that should robustly be identified on the scene image i.e (image during the actual robotic execution) to perform user interface interpretation by matching them with the model image. The model image of the user interface and each desired point to be projected on the scene are indicated as shown in Fig 5.6. By solving the homography problem between the scene and model images, we will get the homography matrix $\mathcal{H}$. The first step to do so is by utilizing efficient feature descriptors (in our case SURF (Bay, Tuytelaars, and Van Gool, 2006)) to determine the key-points in the reference and scene images (see Fig. 5.7(a) and Fig. 5.7(b)). The second step is to Determine the correspondence between the keypoints in scene and model image by performing a two step procedure: first an approximate matching is computed by performing nearest neighbour search (Muja and Lowe, 2009) to rapidly generate pairs of matching keypoints which might include false matches. Then, a homography matrix is computed by applying RANSAC (Fischler and Bolles, 1981) to the matching points from the previous step, thereby eliminating false positives as shown in Fig. 5.7(c)

The idea of using this two step procedure helps to eliminate outliers (false-matches) while also being consistent with the inliers. It also improves finding desired points of the user interface on the scene image with high accuracy from various positions and orientations of a camera without a need for a prior camera calibration.

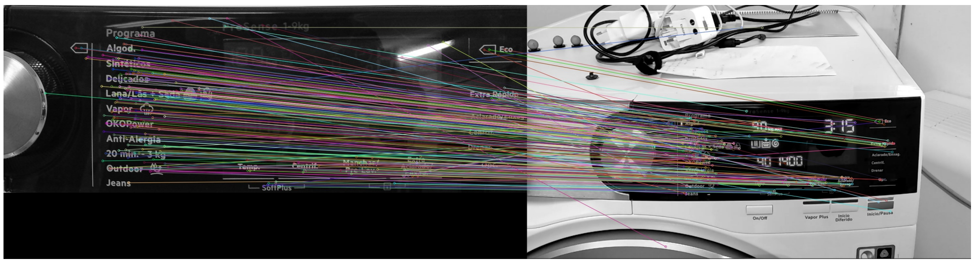
Display Recognition

A MobilenetV2 convolutional neural network which takes an input tensors as the size of 96×96 in three dimension and gives output of (32,12) where the number 12 indicates the number of classes in Logits layer and the 32 indicates the batch size which has been set during training step. Considering the small number of classes to be classified (12), there was no need for training the neural network from the scratch. Therefore, two dense layers and ten convolutional layers of MobilenetV2 network are fine tuned in training process, giving a final total loss value converging to 0.5462 starting from nearly 3.1.

Once Homography estimation has been implemented, the scene frame captured by the camera and the coordinates of the regions of interests are sent to the recognition steps. The regions of interests on the user interface are divided into three parts according to their functionality (see Figure 5.10(a)). The different classes(digits, functions, programs) of the target areas are enclosed with rectangles in corresponding colours(blue, green, red).



(a) Key points on the reference image. (b) Key points on the scene image.



(c) Final matches after eliminating the incorrect matches.

FIGURE 5.7: Keypoints from SURF descriptor. In (a) keypoints are shown for the reference image (user interface) (b) shows descriptors from the robot scene view. (c)Final keypoint matches after eliminating the incorrect matches.

5.5 Experimental Results

Collision Avoidance

A relevant problem during the execution of the loading and unloading tasks is related to avoiding collisions. Apart from kinematic constraint, the movement of the arm inside the drum is also constrained by the appliance itself. Thus, in the experiments, the robotic arm is velocity controlled and, for grasp execution, the redundancy of the arm is exploited in a task-priority framework such that low priority tasks are fulfilled in the null space of higher priority tasks (Bedada et al., 2020). The task-priority framework allows us to smoothly insert a collision avoidance task in the priority hierarchy, task that is required for the safety of the appliance as well as the robotic arm. The full appliance model registration is performed following the approach described in Chapter. 4. To this end, the registered model shown in Fig. 5.8 is used as collision object for the washing machine. A spherical approximation of the robotic arm and an Octomap representation from full model pointcloud regisration of the washing machine are utilized for reactive collision avoidance, see Fig. 5.8(b). The minimum distance and the corresponding nearest point between the bounding spherical volumes and the Octomap is computed by applying distance queries from Flexible Collision Library (FCL) (Pan, Chitta, and Manocha, 2012b). The collision avoidance task generates a repulsive velocity in the direction of the vector connecting the center of the sphere bounding part of a link and the closest obstacle in case the distance is lower than a certain safety threshold. As the robot arm moves minimum distance to the control points is extracted for collision avoidance. Figure 5.9 shows the commanded joint velocity, activation value and minimum obstacle distance of the baxter arm during entering and exit while also successfully avoiding collision.

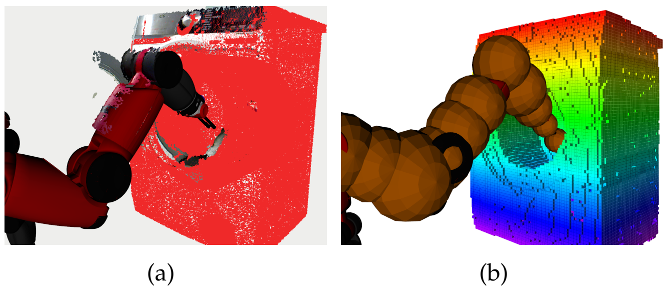


FIGURE 5.8: (a) Scene registration with the appliance model in red, (b) robot and appliance collision model for safe manipulation inside the drum.

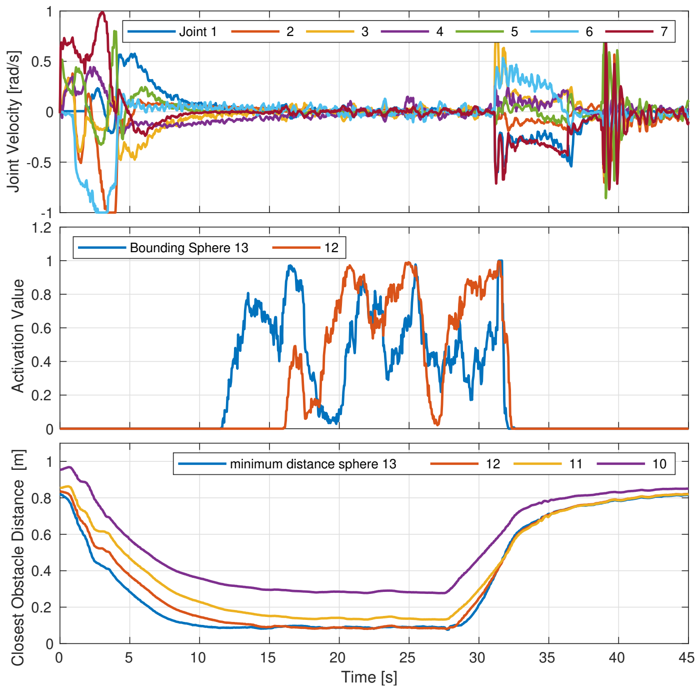


FIGURE 5.9: Entering and exiting the washing machine drum: joint velocity and collision avoidance activation.

5.5.1 Laundry Grasping Task

To validate the proposed approach in Sec. 5.3 and Sec 5.4 the algorithms are implemented in the ROS environment with the 7-DoF Baxter robot arm, while a 3D PicoFlexx camera in an eye-in-hand configuration is employed for the vision system. The schematic representation of the proposed cyber-physical system is reported in Fig. 5.10(a). An external PC is dedicated to run the cloth and UI vision algorithm which provide poses as an output. A task-priority based control algorithm is then utilized to perform grasp execution. In the experiments, the performance of the robot in the grasps with the poses provided by the proposed method is tested.

Five clothes shown in Fig. 5.10(b) are used as test set. These clothes are selected in order to increase the variance for what concerns the dimensions of the item but also the type of fabric (e.g. "harder" or "softer") aiming at providing a comprehensive analysis. In this regard, the shirt has the softest fabric in the set, whereas the jeans have the hardest.

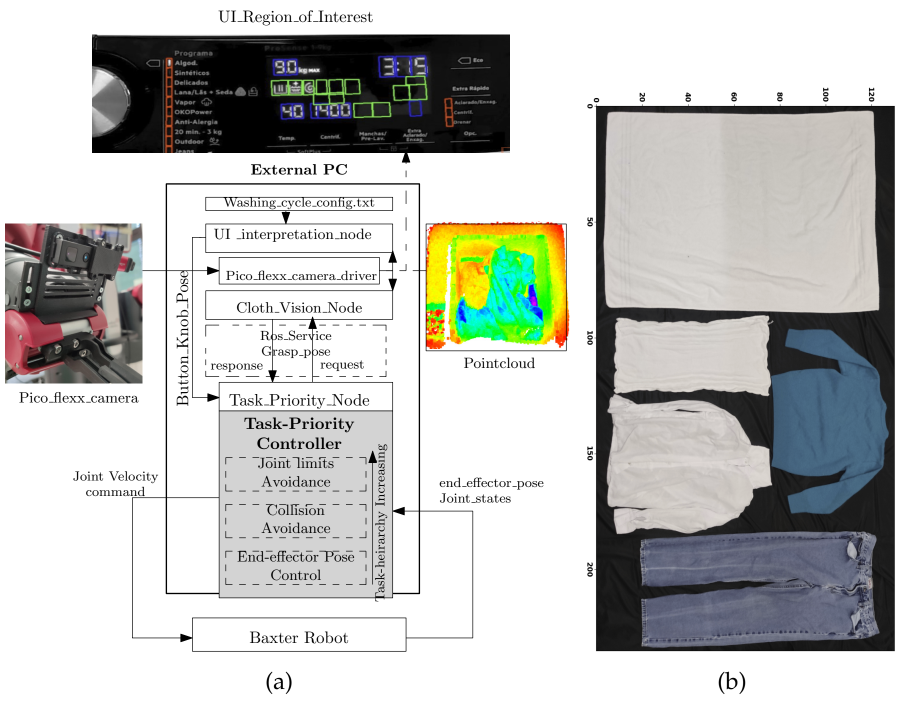


FIGURE 5.10: Experimental Layout. In (a) The overall system layout: detection, control and UI. (b) Clothes test set with rulers in centimeters for scale.

5.5.2 Graspability Tests

The experiments are carried out for the two tasks already introduced throughout this chapter: the bin picking and washing machine recovery picking.

Bin Picking Task

This test is experimented having a single cloth of Fig. 5.10(b) in a bin. At the beginning of the experiment, the cloth is arranged into a random configuration. Then, the execution of the picking task is performed. In sequence, the vision system provides a new target pose (as explained in Sec. 5.3), the robot attempt the grasp, lift the item 1 meter and release it. Hence, the cloth falls back down in the bin assuming a new random configuration. The picking task sequence is repeated 10 times for each cloth.

Washing Machine Recovery Picking Task

This test is carried out similarly to the bin picking one. The clothes of Fig. 5.10(b) are tested one at the time. For each cloth tested, 10 grasp trials are performed. The cloth is placed along the drum door partially hanging out from it to simulate an incorrect insertion. The vision system computes a target pose as described in Sec. 5.3.2. The robot attempts the grasp, moves the grasped cloth vertically toward the drum center, then it releases the grasp. In this case, after each grasp, the cloth is rearranged manually in a new configuration along the drum door.

In Fig. 5.11, the sequence of actions performed for each task is depicted for clarity. Instead, Fig. 5.12 provides a chart showing the success rates of the wrinkle-based grasps for the two tasks from the point of view of each type of cloth. The average success rate in the bin picking task is 0.70, while it is 0.82 in the washing machine one. The success rates of the grasps for the two tasks is also shown to improve by

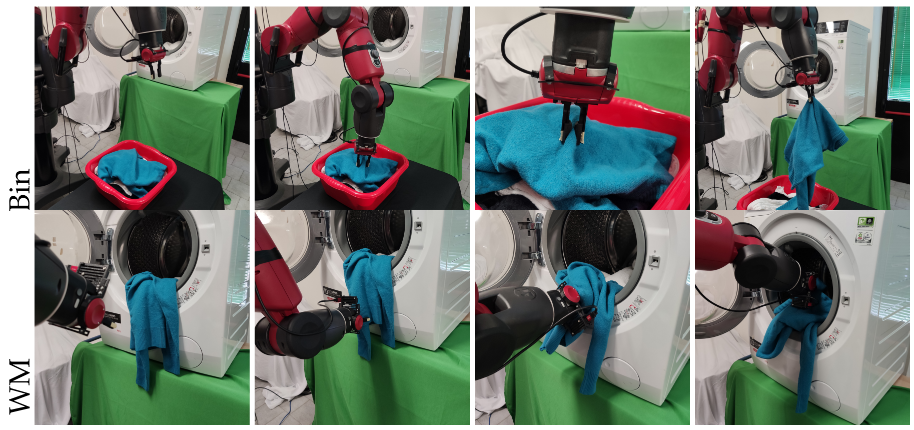


FIGURE 5.11: Sequence of motion during laundry manipulation from Bin and Washing Machine (WM).

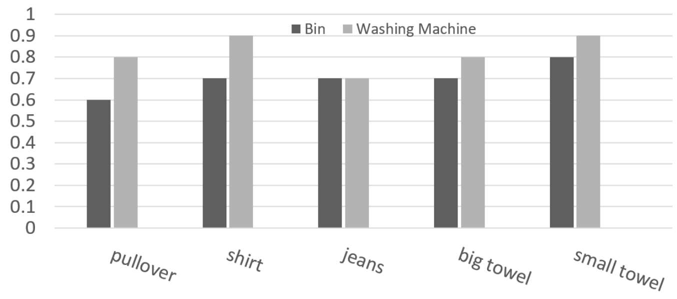


FIGURE 5.12: Success rates for the bin picking and washing machine recovery picking actions.

combining wrinkle and blob based approaches. The average success rate in the bin picking task is 0.850 with the proposed approach and 0.7 with only wrinkle based approach. Similarly, a success rate of 0.87 has been achieved during recovery grasp from the wachine machine in contrast to the 0.8 success rate with the wrinkle only approach. Considering the total set of 158 grasps attempted, our algorithm provides a target pose that results on a successful grasp in 84% of the cases. The full cycle execution of the laundry loading and unloading task is shown in the accompanying video[1]. The failure in the grasps are mostly related to wrong orientations of the target frame resulting from an erroneous identification of the wrinkle path. In a smaller extent, failures can be also associated to measurement errors of the camera, calibration errors for the eye-in-hand and accuracy of the robot arm itself.

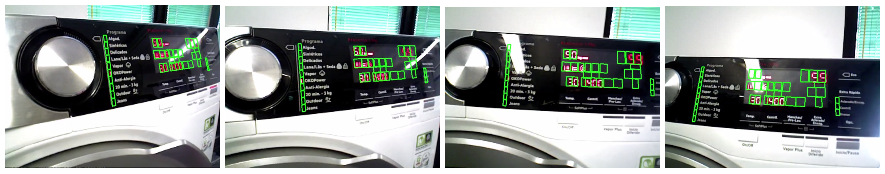
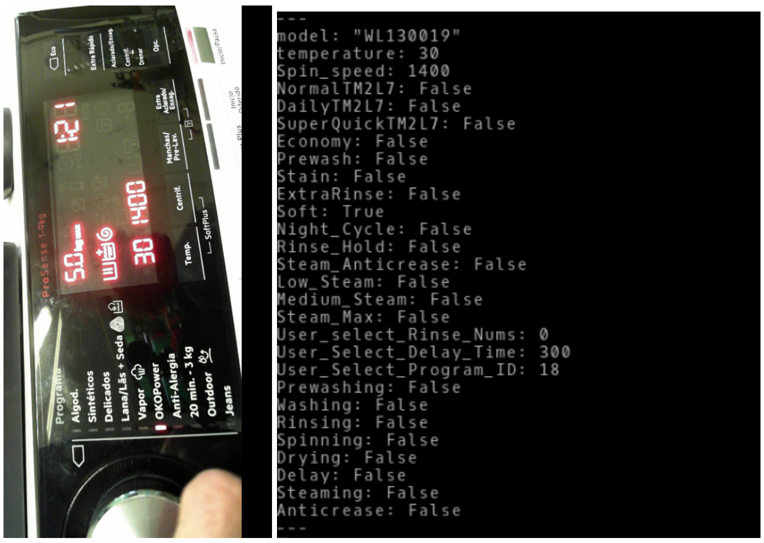


FIGURE 5.13: UI homography estimation for recognition from sharp angles and varying glass reflection.

[1]https://drive.google.com/file/d/1YwSVt8GzLccz_W9lTMr4pb5_r1pcGYdl/view?usp=sharing



```
---
model: "WL130019"
temperature: 30
Spin_speed: 1400
NormalTM2L7: False
DailyTM2L7: False
SuperQuickTM2L7: False
Economy: False
Prewash: False
Stain: False
ExtraRinse: False
Soft: True
Night_Cycle: False
Rinse_Hold: False
Steam_Anticrease: False
Low_Steam: False
Medium_Steam: False
Steam_Max: False
User_select_Rinse_Nums: 0
User_Select_Delay_Time: 300
User_Select_Program_ID: 18
Prewashing: False
Washing: False
Rinsing: False
Spinning: False
Drying: False
Delay: False
Steaming: False
Anticrease: False
---
```

FIGURE 5.14: UI interpretation for robotic vision: For instance Spin speed and washing temprature are set to 1400 and 30 respectively as indicated in the interpretation output.

Additional Experiment

A second type of experiments are performed related not only to the assertion of the correctness of the target poses provided by the algorithm, but also to the capability of those poses to allow the robot to reach a goal state with a reasonable amount of grasps. In particular, the capability of emptying a bin full of 4 test clothes is tested. Performing 5 trials, an average number of 5.2 grasps are required for accomplishing this tasks.

The capability of inserting completely a cloth hanging down from the drum door is also evaluated by performing 5 trials for each cloth tested. In this case, we adopt the small and big towels from the test set: for the first, an average number of 1.8 grasps are necessary; for the second, 3.2 grasps are needed;

5.5.3 Appliance User interface Interpretation

In Fig 5.13 a UI homography estimation of sequence of images from a camera mounted on a robotic arm from various angles is shown. The robotic arm moves to various poses, the alignment was successfully computed under varying glass reflectance and lighting conditions. Two dense layers and ten convolutional layers of mobileNet neural network is fine tuned (instead of training from the scratch) for display recognition on 1280 training data samples. The resulting neural network was able to correctly classify the 12 classes of the display (sample output is shown in Fig 5.14). The three categories of outputs shown on the display, i.e. digits, function and programs are all recognized accurately as shown in this sample video [2].

[2]https://drive.google.com/file/d/1PSZtzydWnqhokPWT6oKgcYd01sRJ_9T3/view?usp=sharing

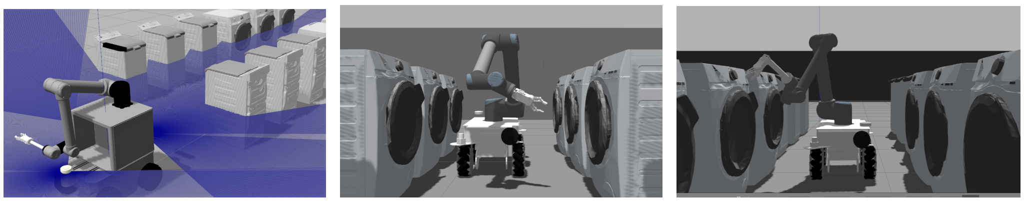

(a) (b) (c)

FIGURE 5.15: Three candidate for final deployment has been evaluated: (a) MMO-500 mobile robot with a ur-10 from Universal robot and an extender. The size of this combination takes considerable space which limits reachable space of robot. (b) rbkairos mobile base from Robotnik and ur10 robotic arm. This option offers acceptable size thanks to the relatively small size of the mobile base apart from the large wrist of the ur10 arm. (c) The final choice for deployment was rbkairos with ur10 arm and an extender with a gripper mounted at angle at the tip of the extender.

5.6 Deployment and Testing

The final evaluation in the lab floor using industrial robot for laundry manipulation and appliance operation has been performed to verify deployability. For this, various industry grade robotic platforms have been evaluated for visibility. A combination different mobile base and manipulator arms are considered: rbkairos mobile base from Robotnik fig 5.15(b), mmo-500 mobile base Neobotix fig 5.15(a) together with ur5 and ur10 collaborative arms and wrist extenders. The comparison is based on the capability to reach safely the interior of the drum, navigate safely in a corridor with narrow passage and ability to perform grasp inside the drum of an appliance and it is performed in simulation environment. The simulation result has shown that utilizing ur10 arm and gripper directly in the appliance drum leads to collision due to small distance margin. Therefore, a mechanical extender shown in figs 5.16(a) and 5.16(b) mounts the gripper at the end is developed and deployed. The study on the dimension of the extender and angle of mounting of the gripper produced a configuration with a small extension and an inclination of 15 °, could make all points accessible for the most common types of washing machines, with wrist staying outside the drum (figs 5.16(c) and 5.16(d)).

5.6.1 Final Results

The capability of laundry loading and unloading by utilizing the approach proposed in this chapter is tested on the final robotic platform. First, camera and grippers are calibrated using standard technique to improve accuracy. With the improved robotic hardware and high quality gripper, the success rate of the grasp from the bin has been slightly higher to 86 % while the recovery laundry picking task has always resulted in success in all attempts.

5.7 Conclusions and Future Work

In this chapter, a pointcloud-based approach for the perception of clothes aiming at the robotized insertion of clothes inside a washing machine is proposed. A perception pipeline for washing machine display interpretation is also developed to perform washing cycle setting autonomously. The approach is validated extensively

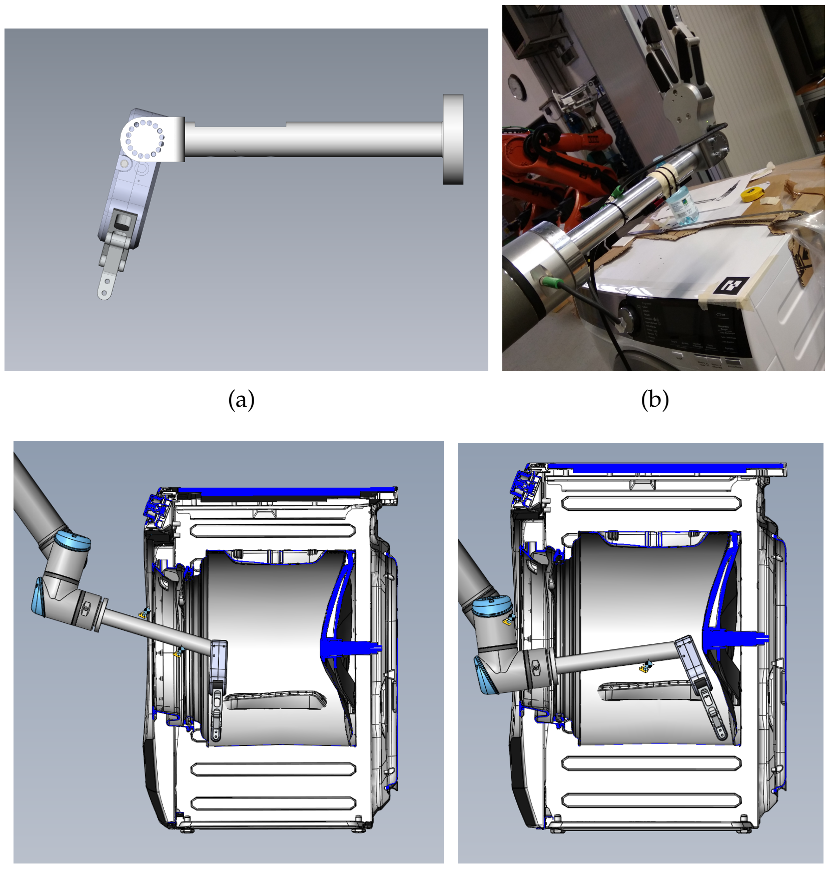


FIGURE 5.16: (a) The mechanical model of the extender with rg6 gripper fron onRobot. (b) Extender and gripper mounted on ur10. (c) Reachability testing by overlying the extender and appliance CAD model at the near end of the drum and (d) far end. It can be seen that the wrist of the ur10 arm remains outside in both cases.

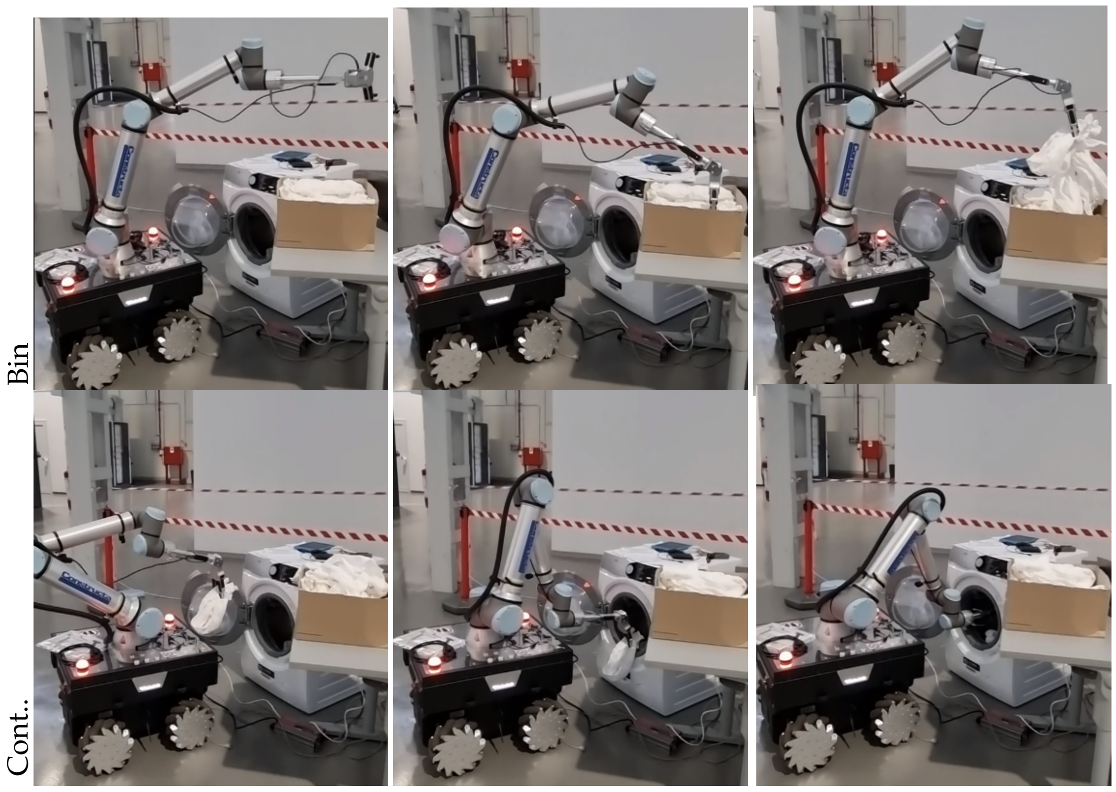


FIGURE 5.17: Sequence of motion during laundry insertion from Bin and Washing Machine (WM).

for both the bin picking and washing machine recovery picking tasks. The grasping performances are satisfactory allowing a success in 84% of the attempts while the user interface interpretation works correctly under varying environmental conditions and sharp angle of view. The work to integrate the approach presented into a complex robotic behavior to achieve full autonomy is underway.

Chapter 6

Conclusions

In this dissertation, various aspects of autonomous robotic system for domestic appliance test automation is evaluated and implemented. A complete pipeline for safe navigation and collision avoidance and a perception system for appliance and laundry detection has been developed.

In chapter 2 , a navigation system that takes into account the movement of human operators using a high resolution predictive full body collision checker is detailed. A safety constraint is also implemented on the local planner level of the navigation system. The presence of human operator is taken into consideration by rendering virtual cylindrical volume around the detected person and depending on the distance from the human, various modes of operation are employed. This behaviour is implemented to comply with the human-robot collaboration standard defined for industrial setup. A task priority based versatile and efficient robotic arm controller; **MOVE-RT** is designed and implemented in C++. This controller is able to incorporate various kinematic constraints dynamically. Considering a battery powered mobile manipulator, energy efficiency is improved by penalizing joints associated to large inertia within this controller. Tests that involve inspection of the interior an appliance drum using a simulation model of the final robotic hardware and the task priority priority controller developed in this chapter is also discussed.

The control of the robotic arm from chapter 2, is further extended to encompasses real-time collision avoidance within the task priority control framework in chapter 3. A GPU based distance field computation from live 3d sensor data is integrated with the task priority controller of the arm to achieve successful collision avoidance from a fast moving obstacles. This approach is also extended to self-collision avoidance by offline voxelization of the robot model, real-time update of the link voxels and euclidean distance computation. This approach helps to unify obstacle and self-collision avoidance within one framework.

The actual perception and manipulation challenges related to washing machine and clothe like deformable objects are discussed chapter 4 and chapter 5. To insure safe interaction between the robot and washing machine, 6d model registration of the appliance is computed by combining a deep learning based initial pose estimate and a gpu-implemented octree data structure. This achieved from only a partial view and registration includes all the internal components of the appliance. Finally, Chapter 5 discusses efficient technique acquire stable grasp of clothes in various scenarios i.e from the bin, door and washing machine. Wrinkled like features combined with blobs on the surface of laundry are detected by segmenting regions of the clothes with the highest entropy and convexity measure. By integrating it with appliance user interface algorithm, full robotic laundry loading and unloading experiment has been conducted.

Bibliography

Aalerud, Atle, Joacim Dybedal, and Geir Hovland (2018). "Scalability of GPU-Processed 3D Distance Maps for Industrial Environments". In: *Proc. IEEE/ASME Int. Conf. on Mechatronic and Embedded Systems and Applications*, pp. 1–5.

al., Martín Abadi et (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org.

Alexa, Marc et al. (2003). "Computing and rendering point set surfaces". In: *IEEE Transactions on visualization and computer graphics* 9.1, pp. 3–15.

Araújo, Helder, R. Carceroni, and C. M. Brown (1996). "A Fully Projective Formulation for Lowe's Tracking Algorithm". In:

Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool (2006). "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer, pp. 404–417.

Bedada, Wendwosen Bellete et al. (2020). "A Safe and Energy Efficient Robotic System for Industrial Automatic Tests on Domestic Appliances: Problem Statement and Proof of Concept". In: *Procedia Manufacturing* 51, pp. 454–461.

Besl, P. J. and N. D. McKay (1992). "A method for registration of 3-D shapes". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 14.2, pp. 239–256.

Bohlin, Robert and Lydia E Kavraki (2000). "Path planning using lazy PRM". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE Int. Conf. on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 1, pp. 521–528.

Bosscher, Paul and Daniel Hedman (2011). "Real-time collision avoidance algorithm for robotic manipulators". In: *Industrial Robot: An International Journal*.

Brock, Oliver and Oussama Khatib (1999). "High-speed navigation using the global dynamic window approach". In: *Proc. IEEE Int. Conf. on Robotics and Automation*. Vol. 1, pp. 341–346.

Buch, Anders Glent et al. (2013). "Pose estimation using local structure-specific shape and appearance context". In: *Proc. IEEE ICRA*.

Burgard, Wolfram et al. (1998). "The interactive museum tour-guide robot". In: *Aaai/iaai*, pp. 11–18.

Cao, Thanh-Tung et al. (2010). "Parallel banding algorithm to compute exact distance transform with the GPU". In: *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pp. 83–90.

Caporali, Alessio and Gianluca Palli (2020). "Pointcloud-based Identification of Optimal Grasping Poses for Cloth-like Deformable Objects". In: *Proc. IEEE Int. Conf. on Factory Automation and Emerging Technologies*.

Cesetti, Andrea et al. (2010). "Development of a flexible test platform for household appliance testing based on mobile agents". In: *2010 IEEE International Conference on Automation Science and Engineering*. IEEE, pp. 855–860.

Chen, Y. and G. Medioni (1991). "Object modeling by registration of multiple range images". In: *Proc.. 1991 IEEE Int. Conference on Robotics and Automation*, 2724–2729 vol.3.

Choi, Byung June, Sung Moon Jin, Shin, et al. (2008). "Development of flexible laboratory automation platform using mobile agents in the clinical laboratory". In: *Proc. IEEE Int. Conf. on Automation Science and Engineering*, pp. 918–923.

Christoph Stein, Simon et al. (2014). “Object partitioning using local convexity”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 304–311.

Cusumano-Towner, Marco et al. (2011). “Bringing clothing into desired configurations with limited perception”. In: *Proc. IEEE ICRA*.

Di Lillo, Paolo et al. (2018). “Safety-related tasks within the set-based task-priority inverse kinematics framework”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 6130–6135.

Doumanoglou, Andreas et al. (2014). “Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning”. In: *Proc. IEEE ICRA*.

Dozier, Gerry et al. (1998). “Artificial potential field based robot navigation, dynamic constrained optimization and simple genetic hill-climbing”. In: *Proc. IEEE Int. Conf. on Evolutionary Computation Proceedings*, pp. 189–194.

Elseberg, Jan et al. (Jan. 2012). “Comparison on nearest-neigbour-search strategies and implementations for efficient shape registration”. In: *Journal of Software Engineering for Robotics (JOSER)* 3, pp. 2–12.

Escobar, Luis A and William Q Meeker (2006). “A review of accelerated test models”. In: *Statistical science*, pp. 552–577.

Fischler, Martin A and Robert C Bolles (1981). “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6, pp. 381–395.

Flacco, Fabrizio et al. (2012). “A depth space approach to human-robot collision avoidance”. In: *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 338–345.

Fox, Dieter, Wolfram Burgard, and Sebastian Thrun (1997). “The dynamic window approach to collision avoidance”. In: *IEEE Robotics & Automation Magazine* 4.1, pp. 23–33.

Garrido-Jurado, Sergio et al. (June 2014). “Automatic generation and detection of highly reliable fiducial markers under occlusion”. In: *Pattern Recognition* 47, 2280–2292.

Girshick, Ross B. et al. (2013). “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *CoRR* abs/1311.2524.

Greenspan, Michael and Nestor Burtnyk (1996). “Obstacle count independent real-time collision avoidance”. In: *Proc. of IEEE Int. Conf. on Robotics and Automation*. Vol. 2, pp. 1073–1080.

Hamner, Brad et al. (2010). “An autonomous mobile manipulator for assembly tasks”. In: *Autonomous Robots* 28.1, p. 131.

He, Kaiming et al. (2015). “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385.

Hermann, Andreas et al. (2013). “GPU-based real-time collision detection for motion execution in mobile manipulation planning”. In: *Proc. IEEE Int. Conf. on Advanced Robotics*, pp. 1–7.

Hermann, Andreas et al. (2014a). “Mobile manipulation planning optimized for GPGPU Voxel-Collision detection in high resolution live 3d-maps”. In: *Int. Symp. on Robotics*, pp. 1–8.

Hermann, Andreas et al. (2014b). “Unified GPU voxel collision detection for mobile manipulation planning”. In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 4154–4160.

Holz, D. et al. (2015). “Registration with the Point Cloud Library: A Modular Framework for Aligning in 3-D”. In: *IEEE Robotics & Automation Magazine* 22, pp. 110–124.

Hornung, Armin et al. (2013). "OctoMap: An efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous robots* 34.3, pp. 189–206.

Jian, B. and B. C. Vemuri (2011). "Robust Point Set Registration Using Gaussian Mixture Models". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 33.8, pp. 1633–1645.

Jiménez, Pablo and Carme Torras (2020). "Perception of cloth in assistive robotic manipulation tasks". In: *Natural Computing*, pp. 1–23.

Juelg, Christian et al. (2017). "Fast online collision avoidance for mobile service robots through potential fields on 3D environment data processed on GPUs". In: *Proc. IEEE Int. Conf. on Robotics and Biomimetics*, pp. 921–928.

Kaldestad, Knut B et al. (2014). "Collision avoidance with potential fields based on parallel processing of 3D-point cloud data on the GPU". In: *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3250–3257.

Karaman, Sertac and Emilio Frazzoli (2011). "Sampling-based algorithms for optimal motion planning". In: *The Int. Journal of Robotics Research* 30.7, pp. 846–894.

Karaman, Sertac et al. (2011). "Anytime motion planning using the RRT". In: *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1478–1483.

Khatib, Oussama (1986). "Real-time obstacle avoidance for manipulators and mobile robots". In: *Autonomous robot vehicles*. Springer, pp. 396–404.

Kita, Yasuyo et al. (2011). "Clothes handling based on recognition by strategic observation". In: *Proc. IEEE ICRA*.

Koenig, Sven and Maxim Likhachev (2002). "D^* lite". In: *Aaai/iaai* 15.

Koren, Yoram and Johann Borenstein (1991). "Potential field methods and their inherent limitations for mobile robot navigation". In: *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1398–1404.

Le, Huu et al. (2019). "SDRSAC: Semidefinite-Based Randomized Approach for Robust Point Cloud Registration without Correspondences". In: *CoRR* abs/1904.03483.

Lei, Maolin et al. (2020). "Real-Time Kinematics-Based Self-Collision Avoidance Algorithm for Dual-Arm Robots". In: *Applied Sciences* 10.17, p. 5893.

Lepetit, Vincent, Francesc Moreno-Noguer, and Pascal Fua (Feb. 2009). "EPnP: An accurate O(n) solution to the PnP problem". In: *Int. Journal of Computer Vision* 81.

Lin, Tsung-Yi et al. (2014a). "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer, pp. 740–755.

Lin, Tsung-Yi et al. (2014b). "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312.

Liu, Wei et al. (2015). "SSD: Single Shot MultiBox Detector". In: *CoRR* abs/1512.02325.

Liu, Wei et al. (2016). "SSD: Single Shot MultiBox Detector". In: *ECCV*.

Lowe, D. G. (1991). "Fitting parameterized three-dimensional models to images". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 13.5, pp. 441–450.

Lowe, David G. (1987). "Three-dimensional object recognition from single two-dimensional images". In: *Artificial Intelligence* 31.3, pp. 355–395. ISSN: 0004-3702.

Lu, C.P., Gregory Hager, and Eric Mjolsness (July 2000). "Fast and globally convergent pose estimation from video images. Patt Anal Mach Intell, IEEE Trans". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 22, pp. 610 –622.

Mansard, Nicolas, Oussama Khatib, and Abderrahmane Kheddar (2009). "A unified approach to integrate unilateral constraints in the stack of tasks". In: *IEEE Tran. on Robotics* 25.3, pp. 670–685.

Marder-Eppstein, Eitan et al. (2010). "The office marathon: Robust navigation in an indoor office environment". In: *Proc. IEEE Int. Conf. on robotics and automation*, pp. 300–307.

Marvel, Jeremy A (2013). "Performance metrics of speed and separation monitoring in shared workspaces". In: *IEEE Tran. on Automation Science and Engineering* 10.2, pp. 405–414.

Moll, Mark, Ioan A Sucan, and Lydia E Kavraki (2015). "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization". In: *IEEE Robotics & Automation Magazine* 22.3, pp. 96–102.

Monsó, Pol, Guillem Alenyà, and Carme Torras (2012). "Pomdp approach to robotized clothes separation". In: *Proc. IEEE/RSJ IROS*.

Muja, Marius and David G Lowe (2009). "Fast approximate nearest neighbors with automatic algorithm configuration." In: *VISAPP (1)* 2.331-340, p. 2.

Myronenko, A. and X. Song (2010). "Point Set Registration: Coherent Point Drift". In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 32.12, pp. 2262–2275.

Nooruddin, Fakir S. and Greg Turk (2003a). "Simplification and repair of polygonal models using volumetric techniques". In: *IEEE Tran. on Visualization and Computer Graphics* 9.2, pp. 191–205.

— (2003b). "Simplification and repair of polygonal models using volumetric techniques". In: *IEEE Tran. on Visualization and Computer Graphics* 9.2, pp. 191–205.

O'Mahony, Niall et al. (2019). "Deep learning vs. traditional computer vision". In: *Science and Information Conference*. Springer, pp. 128–144.

Pages, Jordi, Luca Marchionni, and Francesco Ferro (2016). "Tiago: the modular robot that adapts to different research needs". In: *Int. workshop on robot modularity, IROS*.

Pan, Jia, Sachin Chitta, and Dinesh Manocha (2012a). "FCL: A general purpose library for collision and proximity queries". In: *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3859–3866.

— (2012b). "FCL: A general purpose library for collision and proximity queries". In: *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3859–3866.

Pan, Jia, Christian Lauterbach, and Dinesh Manocha (2010a). "g-Planner: Real-time motion planning and global navigation using GPUs". In: *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

— (2010b). "g-Planner: Real-time motion planning and global navigation using GPUs". In: *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Pan, Jia et al. (2013). "Real-time collision detection and distance computation on point cloud sensor data". In: *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3593–3599.

Papon, Jeremie et al. (2013). "Voxel cloud connectivity segmentation-supervoxels for point clouds". In: *Proc. IEEE Conf. on computer vision and pattern recognition*.

Quigley, Morgan et al. (2009). "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan, p. 5.

Ramisa, Arnau et al. (2012). "Using depth and appearance features for informed robot grasping of highly wrinkled clothes". In: *Proc. IEEE ICRA*.

Ren, Shaoqing et al. (2015). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497.

Romero-Ramirez, Francisco, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer (June 2018). "Speeded Up Detection of Squared Fiducial Markers". In: *Image and Vision Computing* 76.

Rong, Guodong and Tiow-Seng Tan (2007). "Variants of jump flooding algorithm for computing discrete Voronoi diagrams". In: *4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)*. IEEE, pp. 176–181.

Rusinkiewicz, S. and M. Levoy (2001). "Efficient variants of the ICP algorithm". In: *Proc. Third Int. Conference on 3-D Digital Imaging and Modeling*, pp. 145–152.

Rusu, R. B., N. Blodow, and M. Beetz (2009). "Fast Point Feature Histograms (FPFH) for 3D registration". In: *2009 IEEE International Conference on Robotics and Automation*, pp. 3212–3217. DOI: 10.1109/ROBOT.2009.5152473.

Rusu, Radu Bogdan (2010). "Semantic 3d object maps for everyday manipulation in human living environments". In: *KI-Künstliche Intelligenz*.

Rusu, Radu Bogdan, Nico Blodow, and Michael Beetz (2009). "Fast point feature histograms (FPFH) for 3D registration". In: *Proc. IEEE ICRA*.

Safeea, Mohammad and Pedro Neto (2019). "Minimum distance calculation using laser scanner and IMUs for safe human-robot interaction". In: *Robotics and Computer-Integrated Manufacturing* 58, pp. 33–42.

Safeea, Mohammad, Pedro Neto, and Richard Bearee (2019). "On-line collision avoidance for collaborative robot manipulators by adjusting off-line generated paths: An industrial use case". In: *Robotics and Autonomous Systems* 119, pp. 278–288.

Samet, Hanan (1989). "Neighbor finding in images represented by octrees". In: *Computer Vision, Graphics, and Image Processing* 46.3, pp. 367 –386. ISSN: 0734-189X.

Sanchez, Jose et al. (2018). "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey". In: *The Int. Journal of Robotics Research* 37.7, pp. 688–716.

Sandler, Mark et al. (2018). "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520.

Schneider, Frank E and Dennis Wildermuth (2003). "A potential field based approach to multi robot formation navigation". In: *Proc. IEEE Int. Conf. on Robotics, Intelligent Systems and Signal*. Vol. 1, pp. 680–685.

Schneider, Jens, Martin Kraus, and Rüdiger Westermann (2009). "GPU-based realtime discrete Euclidean distance transforms with precise error bounds." In: *VISAPP (1)*, pp. 435–442.

Shea, Roberta Nelson (2013). "Robot Safety standard Update". In: *RIA meeting*, pp. 20–23.

Shibata, Mizuho, Tsuyoshi Ota, and Shinichi Hirai (2009). "Wiping motion for deformable object handling". In: *Proc. IEEE ICRA*.

Simetti, Enrico and Giuseppe Casalino (2016a). "A novel practical technique to integrate inequality control objectives and task transitions in priority based control". In: *Journal of Intelligent & Robotic Systems* 84.1-4, pp. 877–902.

— (2016b). "A novel practical technique to integrate inequality control objectives and task transitions in priority based control". In: *Journal of Intelligent & Robotic Systems* 84.1, pp. 877–902.

Simoni, Roberto et al. (2018). "A novel approach to obstacle avoidance for an I-AUV". In: *Proc. IEEE/OES Autonomous Underwater Vehicle Workshop*, pp. 1–6.

Slotine, Siciliano B and B Siciliano (1991). "A general framework for managing multiple tasks in highly redundant robotic systems". In: *Proc. of 5th Int. Conf. on Advanced Robotics*. Vol. 2, pp. 1211–1216.

Steinbach, Klaus et al. (2006). "Efficient collision and self-collision detection for humanoids based on sphere trees hierarchies". In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*, pp. 560–566.

Sucan, Ioan A, Mark Moll, and Lydia E Kavraki (2012). "The open motion planning library". In: *IEEE Robotics & Automation Magazine* 19.4, pp. 72–82.

Sverdrup-Thygeson, J. et al. (2017). "Kinematic singularity avoidance for robot manipulators using set-based manipulability tasks". In: *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 142–149.

Thrun, Sebastian et al. (1999). “MINERVA: A second-generation museum tour-guide robot”. In: *Proc. IEEE Int. Conf. on Robotics and Automation*. Vol. 3.

Wehr, David and Rafael Radkowski (Dec. 2018). “Parallel kd-Tree Construction on the GPU with an Adaptive Split and Sort Strategy”. In: *Int. Journal of Parallel Programming* 46.

Weng, J., N. Ahuja, and T. S. Huang (1989). “Optimal motion and structure estimation”. In: *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 144–152.

Xiao-Shan Gao et al. (2003). “Complete solution classification for the perspective-three-point problem”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 25.8, pp. 930–943.

Yamazaki, Kimitoshi (2014). “Grasping point selection on an item of crumpled clothing based on relational shape description”. In: *Proc. IEEE/RSJ IROS*.

Yang, J. et al. (2016). “Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 38.11, pp. 2241–2254.

Yoshikawa, T. (1985). “Manipulability and redundancy control of robotic mechanisms”. In: *Proceedings. 1985 IEEE Int. Conf. on Robotics and Automation*. Vol. 2, pp. 1004–1009.