Bilkent University
Department of Computer Engineering

Senior Design Project

T2323

GroceryBee

Final Report

21901548, Efe Beydoğan, efe.beydogan@ug.bilkent.edu.tr
21903350, Arda Önal, arda.onal@ug.bilkent.edu.tr
21901645, Mert Barkın Er, barkin.er@ug.bilkent.edu.tr
21903100, E. Melih Erdem, melih.erdem@ug.bilkent.edu.tr
21902615, Eren Polat, eren.polat@ug.bilkent.edu.tr
Supervisor: Özgür S. Öğüz
Course Instructors:
Selim Aksoy
Tağmaç Topal
Erhan Dolak

<19.05.2023>

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS491/2.

Contents

Final Report

Project Short-Name: GroceryBee

1 Introduction

Online grocery shopping has been offered by numerous markets for years now, however, especially with the advent of the COVID virus and the following quarantine period, there has been a considerable surge in the number of online grocery orders. Specifically, Turkey's online market grew by 434% and reached $244.9 million in the first half of 2020 [1], during the time when COVID first became a part of our lives. The supermarket chain "Migros" was already offering an online shopping option, and with the popularity of online grocery shopping booming, new companies have also started investing in online grocery shopping options for their platforms, such as Morhipo and Trendyol. It was reported that the growth in fast delivery startups, such as Getir and Banabi, exceeded 50% and the provinces Migros delivers to have gone from 58 to practically all provinces in Turkey. During the pandemic, the demand for virtual markets almost quadrupled [1]. In the US, 70% of shoppers stated they would be buying groceries online by 2022, and Walmart's US e-commerce sales grew by 79% in the third quarter of 2020. It is estimated that e-commerce warehouses will be fully automated by the year 2029 [2].

It is apparent from the aforementioned information that demand for online grocery shopping has skyrocketed in the past couple of years, and investing in this field is surely a worthwhile engagement. To this end, we will be developing a robot, named "GroceryBee", to automate and expedite the task of gathering grocery items from the shelves of supermarkets for online orders. GroceryBee is a robot designed to assist grocery stores in keeping up with their online orders. GroceryBee will help grocery store employees by collecting the items which are required for an online order from the store shelves, thus speeding up the delivery process. Additionally, GroceryBee will restock shelves that are empty and keep updated stock information in order to inform customers on whether a product is available in the store or not. Consequently, supermarket employees will have received an assist in their jobs, and they will be able to work simultaneously with GroceryBee to serve customers. There are similar projects for accomplishing tasks akin to what we are trying to achieve. However, GroceryBee will not only safely and accurately gather grocery items, but it will at the same time help grocery store employees with keeping the markets in order. Moreover, although there are robots that are engineered to accomplish tasks such as managing inventory or notifying employees about shelves that need to be restocked [3], none of these robots are designed to do all of these, and even more, altogether, unlike GroceryBee.

1.1 Purpose of the System

We are developing "GroceryBee" for the automation and acceleration of gathering grocery items from the shelves of supermarkets for online orders. For this, we aim to leverage theories from different fields in science and engineering. GroceryBee will have to make decisions in an uncertain environment, and in this uncertainty, it is not possible to plan everything precisely beforehand, therefore our robot will exploit the real-time information from its camera (both RGB and depth) to make real-time decisions and changes in planning (a robot might see a person blocking its path, and change its path to the destination, etc.). For perception, connectivist

approaches which have achieved state-of-the-art results will be employed. Therefore, neural models will be used and perhaps trained according to our purposes; both for the detection of the product to be handled and for sequential decision-making. For example, the robot might have to analyze the situation of the item and check if it is stuck, or has other items around it. Deep learning approaches will pave the way for the robot to assess the situation and segment the item of interest from a window of frames. Different architectures have been shown to achieve SOTA results (such as transformers) in generative and understanding tasks regarding computer vision. Robot actuators and grippers apply control theory in motion/trajectory planning and as mentioned above to leverage the sequential information gathered from the feedback system. This is important for the robot to execute sequential decision-making and be responsive to its environment. We are using the Drake framework, which uses control theoretic design and analysis to enhance the robustness of the physics engine. Open problems in motion planning, perception, and decision-making are being worked on using these approaches from various disciplines.

GroceryBee will be able to collect required items in response to the orders while navigating between the supermarket aisles without crashing. The robot will have access to the locations of products within aisles and navigate to the location when an order is requested. It will detect the requested object on the shelf from the camera frames, identify relative 3D coordinates, use the gripper to pick up the object safely, and bring the object to the delivery point without damaging the item. GroceryBee will also calculate the most optimal path to collect the orders, ensuring the fastest delivery possible for customers.

1.2 Design Goals

Usability

The application should require only the information of the items such as the images, names and locations. GroceryBee should be able to collect the requested items and bring it to the delivery point within the supermarket.

Efficiency

When multiple items are requested in the same order, GroceryBee should calculate the shortest path and collect the orders to save time, energy and money.
When GroceryBee arrives at the aisle to collect the item, it should calculate the most optimum way for its arm to move to grab the item.

Extensibility

GroceryBee should be designed so it could be operated in environments with multiple robots which could communicate with each other. In the future if a system with multiple robots is introduced, the GroceryBee should easily be converted to a multi-robot application.
In the future, it could work in environments that have humans.

Reliability

GroceryBee should collect, carry and deliver items without damaging them. When an order is requested, GroceryBee should ensure successful delivery.

Performance

GroceryBee should calculate the most optimum path with a low computation time. When an order is made, the robot should move fast and try to collect items as fast as possible to make sure that the order is completed in a short time span.

Marketability

GroceryBee as a software is designed with a real physics engine and uses models of robots of "KUKA" which is a robotics company. The final product can be advertised with its ease to use in real life scenarios by stating the fact that it can be used in real life. There are millions of supermarkets across the globe and having a robot which can collect items from shelves would certainly attract these supermarket companies. GroceryBee should be designed so it is cheap, robust and accurate so that it could be used in real supermarkets.

Scalability

GroceryBee should be built so that it could be scaled to systems with multiple robots. These robots can be operated in an efficient manner to complete an order in the shortest time possible. Furthermore, it should be adaptable to environments with human beings for its incorporation to daily supermarkets.

Maintainability

GroceryBee should be implemented following object-oriented programming and other programming principles to ensure its ease to maintain.

1.3 Definitions, acronyms, and abbreviations

SOTA: State-Of-The-Art. It refers to the current best level of performance achieved in a particular field or task and is commonly used in the context of technology and research.

Heuristic: A heuristic involves the approach to solving a problem leveraging a calculated guess from previous experiences. In order to find a (near) optimal solution, instead of trying/calculating all possible solutions to a problem, say a traversal problem, heuristic approaches enable agents to make a decision without having to consider all paths, eliminating unnecessary amounts of work.

Sequential decision-making: Sequential decision-making is the process of making a series of decisions over time, where the outcome of each decision affects the available options for subsequent decisions. In the context of robotics, it is a key aspect as robots often need to make decisions based on sensory input and previous actions. This process involves gathering information about the environment, determining the current state of the robot, selecting an action based on this information and state, and then updating the state based on the outcome of the action.

End-effector: In robotics, the end-effector is the tool that is attached to the end of a robotic arm or manipulator, which is designed to interact with the environment. In our context, the end-effector is the gripper.

Motion/path planning: Motion planning, also known as path planning, is the process of determining a sequence of valid motions that meet certain movement constraints to achieve a desired goal, and perhaps, optimize the movement in certain ways. The goal can be a desired end-effector position or a more complex task, such as navigating through an environment or manipulating an object.

Kinematics: Kinematics in robotics is a branch of study that deals with the motion of robots without taking into account the forces or torques that cause that motion. It focuses on analyzing the position, velocity, and acceleration of different parts of a robot, as well as how they change over time. By understanding the kinematics of a robot, engineers can optimize its movement, ensure that it moves efficiently, and make it perform specific tasks accurately. There are two main types of kinematics in robotics: forward kinematics and inverse kinematics. **Forward kinematics** is concerned with determining the position and orientation of a robot's end-effector (gripper in our case) based on the joint angles of the robot. In contrast, **inverse kinematics** is concerned with finding the joint angles of a robot that will result in a desired end-effector position and orientation.

Differential inverse kinematics (DiffIK): Differential inverse kinematics is a technique used in robotics to control the motion of a robot's end-effector based on changes in its position or orientation. While inverse kinematics involves computing the joint angles required to achieve a desired end-effector position, differential inverse kinematics involves computing the change in joint angles required to move the end-effector in a desired direction. This results in a smooth motion of the end-effector since DiffIK takes into account the current state of the robot, including the velocity and acceleration of the end-effector.

Kinematic trajectory optimization: Kinematic trajectory optimization is a technique used in robotics for computing the optimal trajectory of a robot's end-effector over time, subject to constraints on the robot's motion. The goal is to find a motion that minimizes some cost function while satisfying constraints on the robot's movement, such as joint limits, joint velocities, or avoiding collisions with obstacles.

Sampling-based methods: Sampling-based methods are a set of techniques for motion planning which involve randomly sampling the robot's state space to explore potential configurations, and finding a collision-free path from the robot's current position to the desired goal by connecting those. They offer an efficient way to plan the robot's motion.

Perception: Perception in the robotic context involves the algorithms and approaches to enable the robots to understand data received from sensors and cameras. These algorithms may involve traditional Vision algorithms or ML/DL approaches that have been used in the current.

Point cloud: A set of points in 3D space that represent a 3D shape, this point cloud can be generated by the camera on the robot.

RGBD: Red Green Blue Depth. An RGBD camera is a type of camera that captures both color and depth information.

Normal vector: The normal vector, often simply called the "normal," to a surface is a vector that is perpendicular to the surface at a given point. When normals are considered on closed surfaces, the inward-pointing normal (pointing towards the interior of the surface) and outward-pointing normal are usually distinguished [4].

Segmentation: Segmentation is the process of dividing an image into different regions, based on some criteria such as color, texture, or shape. In our context, it is necessary to identify and separate different objects within a scene and determine how to grasp them.

2 Requirements Details

2.1 Functional Requirements

- GroceryBee should be able to collect required items in response to the orders.
- GroceryBee should be able to navigate in the supermarket aisles without crashing.
- GroceryBee should have access to locations of products within aisles and navigate to the location when an order is requested.
- GroceryBee should be able to detect the requested object on the shelf from the camera frames, identify relative 3D coordinates, use the gripper to pick up the object safely, and bring the object to the delivery point without damaging the item.
- GroceryBee should be able to calculate the most optimum path for requests with multiple items and collect them accordingly.
- GroceryBee should inform the employees about stock information and alert them if stock is low when it arrives on the shelf.
- GroceryBee should be able to restock the shelves in the supermarket.
- GroceryBee should be simulated in a simulation rather than a physical robot.

2.2 Non-functional Requirements

Usability

The application should require only the information of the items such as the images, names, and locations. GroceryBee should be able to collect the requested items and bring them to the delivery point within the supermarket.

Efficiency

When multiple items are requested in the same order, GroceryBee should calculate the shortest path and collect the orders to save time, energy, and money.

When GroceryBee arrives at the aisle to collect the item, it should calculate the most optimum way for its arm to move to grab the item.

Extensibility

GroceryBee should be designed so it could be operated in environments with multiple robots which could communicate with each other. In the future, if a system with multiple robots is introduced, GroceryBee should easily be converted to a multi-robot application.

In the future, it could also work in environments that have humans.

Reliability

GroceryBee should collect, carry and deliver items without damaging them. When an order is requested, GroceryBee should ensure successful delivery.

Data should be processed securely to ensure that it is protected and not leaked to third parties.

2.3 Pseudo Requirements

1. Drake is a module that is available in C++ and Python, but Python will be used for implementation.
2. Github will be used for version control.
3. scikit-learn, PyTorch, pandas, NumPy, and matplotlib modules will be used for perception purposes.
4. The robot will be free for the users, but will be paid for customers:
5. The website of GroceryBee will be hosted on GitHub.
6. Visual Studio Code will be our IDE.

3 Final Architecture and Design Details

3.1 Overview

In order to make the development of GroceryBee easier, we have divided the system into subsystems, all of which are critical to the project's success, and also complex enough to be separated from each other. The "Subsystem Decomposition" diagram shows the subsystems we came up with for our project, as well as the relationships between them, as in which subsystem communicates with which other ones to obtain the necessary data it needs. With this design, we are aiming to simplify the maintenance of our system, as well as ensure its performance.

3.2 Subsystem Decomposition

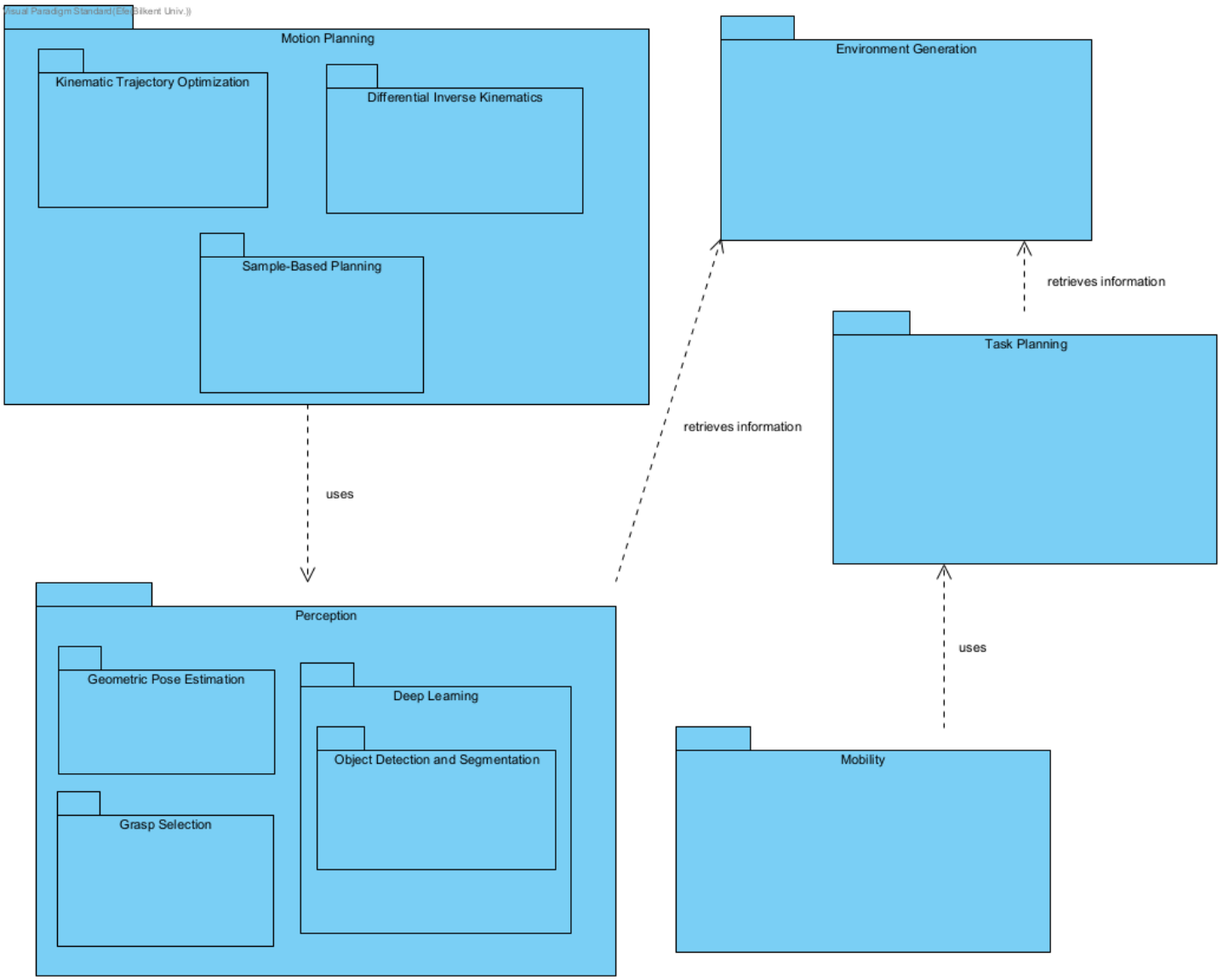

Figure 1. Subsystem Decomposition Diagram

Subsystem Explanations

Environment Generation:
The “Environment Generation” subsystem consists of functionalities responsible for generating a supermarket environment, which GroceryBee will traverse. The environments will be generated dynamically depending on different supermarket interiors. GroceryBee will adapt to the given environment and plan its paths accordingly to ensure a fast and efficient delivery.

Task Planning:
The “Task Planning” subsystem includes the functionalities for the robot to find an optimum order for tasks in any generated supermarket environment. To do so, this subsystem retrieves information from the “Environment Generation” subsystem. The information regarding the

placement of shelves, items, etc. is received, and the “Task Planning” subsystem utilizes this information to come up with an optimum path to gather items as quickly as possible.

Mobility:
The “Mobility” subsystem is responsible for integrating the two dimensional planar movement of the robot to move along the aisles and navigate within the supermarket. This subsystem will be built separately and then integrated to the robot. It will use the “Task Planning” subsystem to plan the order of items that will be picked up to navigate with the optimum path.

Perception:
The “Perception” subsystem will be solely responsible for the understanding of the robots environment. In the real world, the robot will not know the locations of the items therefore, camera systems are required for the robot to perceive its surroundings. Cameras and depth sensors will be added to the system. The images and the information retrieved from them will be used with image processing and deep learning algorithms to detect and pick up the objects. It will consist of three subsystems. The “Geometric Pose Estimation” subsystem is the system responsible for understanding the orientation of the object with regards to the world. This information is crucial to determine how to pick the object up. The “Grasp Selection” subsystem is the system for deciding the pinching points of the gripper for picking up the item. As the shapes and the sizes of the items vary, sophisticated algorithms are necessary for finding valid grasp candidates. Lastly, the “Deep Learning” subsystem is the implementation of neural networks and deep learning algorithms to detect and segment the objects with the images and the depth information retrieved from the cameras.

Motion Planning:
The “Motion Planning” subsystem includes the functionalities that will let GroceryBee calculate the best ways to pick an item from the shelves and put them in the bin for online delivery orders. This subsystem consists of 3 further subsystems. The first one, named “Kinematic Trajectory Optimization”, encompasses the functions for optimizing the movement of the robot’s arm when reaching objects on the shelves. By optimizing this process, we ensure that the robot doesn’t touch the shelves by mistake when picking up an item, or it doesn’t knock over anything while reaching or retracting to put an item in the bin. The second subsystem is “Differential Inverse Kinematics”. Differential IK is necessary for GroceryBee to calculate where to place its gripper when picking up an item, so the robot will know exactly which movements to make prior to reaching the shelves, and thus the possibility of mistakes will have been minimized. The final subsystem is called “Sample-Based Planning”. This subsystem consists of functionalities that are both fast and efficient as they don’t require fully exploring the configuration space. Rather, random configurations are sampled and checked for plausibility, finally helping the robot determine a solution. The “Motion Planning” subsystem uses information from the “Perception” subsystem to make informed decisions when optimizing the picking up of items.

4 Development/Implementation Details

To complete the motion planning aspect—the movement of the robotic arm and its mobile base—we used Drake’s differential inverse kinematics and kinematic trajectory optimization modules. In addition to the task/motion planning problems we have approached with Drake, we made use of the traditional Computer Vision algorithms and combined them with state-of-the-art deep learning models for semantic segmentation. We used semantic masks created by the LangSAM model which involves a BERT-like model for providing text

embeddings for text prompting, and SAM model by Meta for semantic segmentation. We used Python>= 3.10, and several libraries including but not limited to pydrake, torch, numpy etc. We have used the Hugging Face API to load the deep learning models at runtime. The 3D relative coordinates of the segmented objects in the RGB image were calculated using the pinhole camera model and camera intrinsics. After the 3D coordinate space belonging to the object is calculated, the object is picked up with grasp calculation using the iterative closest point algorithm and differential inverse kinematics calculations for setting the joint positions and velocities of the robot. Meshcat was used to remotely run and view 3D simulations on the web.

5 Test Cases and Results

Test ID	01	Category	Functional	Severity	Critical
Objective	This test case is to verify that the robot should fit in the aisle.				
Steps	1. Check if the robot is smaller than the aisle.				
Expected	The robot fits in the aisle.				
Date-Result	10/05/2023 - Pass				

Test ID	02	Category	Functional	Severity	Critical
Objective	This test case is to verify that the items should fit in the shelves.				
Steps	1. Check if the items safely fit in the shelves.				
Expected	The items fit in the shelves without problems.				
Date-Result	10/05/2023 - Pass				

Test ID	03	Category	Functional	Severity	Major
Objective	This test case is to verify that the robot can deal with an item not being on a shelf.				
Steps	1. The robot is tasked with picking up an item which does not exist on a shelf.				
Expected	The robot skips picking up that item and moves on with the rest of the item list.				
Date-Result	11/05/2023 - Pass				

Test ID	04	Category	Functional	Severity	Major
Objective	This test case is to verify that the robot can still move if an obstacle blocks its path.				
Steps	1. The robot is tasked with picking up an item.				

	2. An obstacle is added to the robot’s path.
Expected	The robot chooses another path for itself.
Date-Result	11/05/2023 - Pass

Test ID	05	Category	Functional	Severity	Major
Objective	This test case is to verify that the robot can deal with an item that is higher than the robot can reach.				
Steps	1. The robot is tasked with picking up an item that is higher than the robot can reach.				
Expected	The robot skips picking up that item and moves on with the rest of the item list.				
Date-Result	N/A				

Test ID	06	Category	Functional	Severity	Major
Objective	This test case is to verify that the robot can skip over an item if that item is too deep.				
Steps	1. The robot is tasked with picking up an item that is too deep in the shelf to pick up.				
Expected	The robot skips picking up that object.				
Date-Result	12/05/2023 - Pass				

Test ID	07	Category	Functional	Severity	Major
Objective	This test case is to verify that the robot can function even if the item the robot is tasked with picking up is blocked by some obstacle(s).				
Steps	1. The robot is tasked with picking up an item. 2. Some obstacle(s) is added between the robot and the item.				
Expected	The robot skips picking up that object.				
Date-Result	13/05/2023 - Pass				

Test ID	08	Category	Functional	Severity	Minor
Objective	This test case is to verify that the robot can create a path between itself and the shelf it’s going to pick the item from.				
Steps	1. The robot is tasked with picking up an object.				
Expected	The robot creates a movement path between itself and the item it is tasked with picking up.				

Date-Result	10/05/2023 - Pass

Test ID	09	Category	Functional	Severity	Minor
Objective	This test case is to verify that the robot can display the expected motion time after it creates a motion plan.				
Steps	1. The robot is tasked with picking up an item.				
Expected	The robot displays the expected motion time.				
Date-Result	10/05/2023 - Pass				

Test ID	10	Category	Functional	Severity	Major
Objective	This test case is to verify that the robot drops the items in the bin correctly.				
Steps	1. The robot is tasked with picking up an item. 2. The robot drops the item while attempting to put the item in the bin.				
Expected	The robot picks the item again and puts it back in the bin.				
Date-Result	13/05/2023 - Fail				

Test ID	11	Category	Functional	Severity	Major
Objective	This test case is to verify that the robot fills the bin with the items it is tasked with.				
Steps	1. The robot is tasked with picking up a list of items.				
Expected	The robot fills the bin with the tasked items.				
Date-Result	15/05/2023 - Pass				

Test ID	12	Category	Functional	Severity	Major
Objective	This test case is to verify that the robot only cares about the item it is picking up from the shelf.				
Steps	1. The robot is tasked with picking up an item. 2. The items are set up in a way that when the robot picks up the item, it drops another item in the same shelf.				
Expected	The robot does not fix the dropped item, and it drops the picked up item in the bin.				
Date-Result	12/05/2023- Pass				

Test ID	13	Category	Functional	Severity	Critical
Objective	This test case is to verify that the environment information is transferred correctly to the robot.				
Steps	1. An environment is created. 2. The robot is started. 3. Robot's information about the environment is checked if it's correct.				
Expected	The environment information is correctly transferred to the robot, and the robot is able to process the information.				
Date-Result	12/05/2023 - Pass				

Test ID	14	Category	Functional	Severity	Critical
Objective	This test case is to verify that the item grasp trajectory is correctly calculated.				
Steps	1. The robot is tasked with picking up an item. 2. The trajectory is checked after the robot calculates it.				
Expected	The trajectory calculated by the robot is correct and the item is retrieved successfully.				
Date-Result	10/05/2023 - Pass				

Test ID	15	Category	Functional	Severity	Major
Objective	This test case is to verify that the robot does not hit the shelves.				
Steps	1. The robot is tasked with picking up a list of items.				
Expected	The robot does not hit the shelves when retrieving the items.				
Date-Result	12/05/2023 - Pass				

Test ID	16	Category	Functional	Severity	Critical
Objective	This test case is to verify that the robot's camera functions correctly.				
Steps	1. The robot is tasked with picking up an item. 2. Some RGB-D cameras are connected to the robot.				
Expected	The cameras work correctly and the robot retrieves the item by using the image retrieved from the cameras.				
Date-Result	10/05/2023 - Pass				

Test ID	17	Category	Functional	Severity	Critical
Objective	This test case is to verify that the robot's joints work correctly.				

Steps	1. The robot is tasked with picking up an item.
Expected	The robot's joints function correctly and the robot successfully picks the item up.
Date-Result	10/05/2023 - Pass

Test ID	18	Category	Functional	Severity	Critical
Objective	This test case is to verify that the robot collects items from a list.				
Steps	1. The robot is tasked with picking up a list of items.				
Expected	The robot collects all of the items given in the list.				
Date-Result	11/05/2023 - Pass				

Test ID	19	Category	Functional	Severity	Critical
Objective	This test case is to verify that the robot can calculate an optimal path for collecting items.				
Steps	1. The robot is tasked with picking up a list of items.				
Expected	The robot calculates a path and starts picking the items up.				
Date-Result	11/05/2023 - Pass				

Test ID	20	Category	Functional	Severity	Critical
Objective	This test case is to verify that the robot can deal with an item not being on a shelf.				
Steps	1. The robot is tasked with picking up an item which does not exist on a shelf.				
Expected	The robot skips picking up that item and moves on with the rest of the item list.				
Date-Result	11/05/2023 - Pass				

Test ID	21	Category	Functional	Severity	Major
Objective	This test case is to verify that the robot can try more than one try to pick an item.				
Steps	1. The robot is tasked with picking up an item. 2. That item is placed such that the robot is unable to pick the item on the first try.				
Expected	The robot tries to pick the item again.				
Date-Result	10/05/2023 - Pass				

Test ID	22	Category	Functional	Severity	Minor
Objective	This test case is to verify that the robot can pick up an item when the texture of the item is incorrect.				
Steps	1. The robot is tasked with picking up an item. 2. The texture of the item is changed.				
Expected	The robot picks up another item of the same kind as the item given to the robot.				
Date-Result	N/A				

Test ID	23	Category	Functional	Severity	Major
Objective	This test case is to verify that the robot can pick an item when the item geometry is not perceived correctly.				
Steps	1. The robot is tasked with picking up an item. 2. The item is placed such that the robot’s cameras can not perceive the item correctly.				
Expected	The robot picks up another item of the same kind that can be perceived.				
Date-Result	15/05/2023 - Pass				

Test ID	24	Category	Functional	Severity	Major
Objective	This test case is to verify that the robot does not pick up a damaged item.				
Steps	1. The robot is tasked with picking up an item. 2. The item is placed such that the cameras can perceive that the item is damaged.				
Expected	The robot picks up another item of the same kind that is not damaged.				
Date-Result	N/A				

Test ID	25	Category	Functional	Severity	Critical
Objective	This test case is to verify that the robot can wait for the delivery point to be available..				
Steps	1. The delivery point is set up such that the robot is unable to find a path there. 2. The robot is given a list of items to retrieve.				
Expected	The robot waits for the delivery point to be available.				
Date-Result	N/A				

Test ID	26	Category	Functional	Severity	Major
Objective	This test case is to verify that the previous delivered basket not picked up and new delivery is attempted				
Steps	1. Delivery system is set up such that orders are made before previous basket is picked up				
Expected	Hang on until the basket is picked up				
Date-Result	11/05/2023 - Pass				

Test ID	27	Category	Functional	Severity	Critical
Objective	This test case is to verify what happens when the position in front of shelf is too far				
Steps	1. Place the robot to a far position from the shelf				
Expected	Get closer to shelf				
Date-Result	11/05/2023 - Pass				

Test ID	28	Category	Functional	Severity	Major
Objective	This test case is to verify what happens if an item type is not suitable for being picked up by two grippers				
Steps	1. Recognize certain items that are not suitable to be picked up by two grippers				
Expected	Use a different kind of gripper (such as vacuum)				
Date-Result	11/05/2023 - Pass				

Test ID	29	Category	Functional	Severity	Critical
Objective	Differential inverse kinematics not working properly				
Steps	1. Give a task to the robot 2. See if pick objective is reached when the task is given				
Expected	Try to pick up using trajectory optimization				
Date-Result	11/05/2023 - Pass				

Test ID	30	Category	Functional	Severity	Major
Objective	Segmentation not working as expected				
Steps	1. Start grasp selection process				

	2. Detect if segmentation fails with the constructed point cloud
Expected	Use alternative approaches such as deep learning
Date-Result	11/05/2023 - Pass

Test ID	31	Category	Functional	Severity	Critical
Objective	Environment generation shouldn’t lead to unreachable areas				
Steps	1. Generate areas that are unreachable by the robot 2. Give tasks to the robot that involves unreachable areas				
Expected	Calculate the reachability and not take the assigned task of pickup				
Date-Result	11/05/2023 - Pass				

Test ID	32	Category	Functional	Severity	Critical
Objective	This test case is to verify that robot shuts when it falls over				
Steps	1. Place the robot lying down				
Expected	Shuts itself down				
Date-Result	N/A				

Test ID	33	Category	Functional	Severity	Major
Objective	One item unavailable but other items remain on the list				
Steps	1. Recognize that the item is unavailable and plan accordingly				
Expected	Proceeds to collect other items				
Date-Result	13/05/2023 - Pass				

Test ID	34	Category	Functional	Severity	Critical
Objective	This test case is to verify that the robot does not crash into shelf				
Steps	1. Place the item so that a pickup would be likely to cause a collision 2. Give the robot the task to pick it				
Expected	Optimize the trajectory to avoid collisions				
Date-Result	13/05/2023 - Pass				

Test ID	35	Category	Non-Functional	Severity	Minor

Objective	An item is attempted to be picked up at most 5 times
Steps	1. Attempt a pick up of an item 2. Count the times an item is attempted to be picked up
Expected	Try to pick up another object
Date-Result	13/05/2023 - Pass

Test ID	36	Category	Non-Functional	Severity	Minor
Objective	The process of picking up an item with all attempts should take at most 2 minutes				
Steps	1. Attempt picking up a certain item 2. Measure the pick up time for each item and print/check it				
Expected	Skip the item, try alternative perception methods such as deep learning				
Date-Result	13/05/2023 - Fail				

Test ID	37	Category	Functional	Severity	Major
Objective	This test case is to verify that a valid grasp candidate is found and optimum trajectory is calculated				
Steps	1. Run the robot on a certain task 2. Check the logs for the trajectory and grasps				
Expected	Robot moves its joints to pick up the object				
Date-Result	13/05/2023 - Pass				

Test ID	38	Category	Functional	Severity	Major
Objective	Object has been picked up				
Steps	1. Observe that the object is picked up				
Expected	Robot places object in the basket				
Date-Result	13/05/2023 - Pass				

Test ID	39	Category	Non-Functional	Severity	Minor
Objective	This test case is to verify that the completion of an order shouldn’t take more than 20 minutes				
Steps	1. Measure the time it takes for an order to arrive				
Expected	Cancel the order if more than 20 mins				

Date-Result	13/05/2023 - Pass

Test ID	40	Category	Functional	Severity	Major
Objective	Basket is too small for the whole order				
Steps	1. Fill the basket by manually placing items in it				
Expected	Robot should take several tours to complete the order				
Date-Result	13/05/2023 - Fail				

Test ID	41	Category	Functional	Severity	Minor
Objective	This case is to verify robot’s actions when the item is fragile				
Steps	1. Place a fragile item such as a glass container 2. Assign that item for the robot to pick it up				
Expected	Item should be placed in basket with care				
Date-Result	14/05/2023 - Fail				

Test ID	42	Category	Non-Functional	Severity	Minor
Objective	Robot mobility speed shouldn’t exceed 5 km/h				
Steps	1. Check if the robot speeds up more than 5 km/h				
Expected	Robot should slow down				
Date-Result	14/05/2023 - Pass				

Test ID	43	Category	Non-Functional	Severity	Minor
Objective	Joint speed shouldn’t exceed 0.2 m/s				
Steps	1. Measure the speed of the joint				
Expected	Report the time, make it faster				
Date-Result	14/05/2023 - Pass				

Test ID	44	Category	Functional	Severity	Minor
Objective	This case is to verify that top button is pressed				
Steps	1. Check if robot stops				

Expected	Robot should stop
Date-Result	14/05/2023 - Pass

Test ID	45	Category	Functional	Severity	Minor
Objective	Return robot button is pressed				
Steps	1. Robot stops whatever its doing safely and returns				
Expected	Robot should return to its dock and wait until it is given another instruction				
Date-Result	14/05/2023 - Pass				

Test ID	46	Category	Functiona	Severity	Minor
Objective	Robot out of power				
Steps	1. Cut the power to the robot				
Expected	Robot should shut down but should stay as it is to prevent the items from falling down				
Date-Result	N/A				

Test ID	47	Category	Functional	Severity	Minor
Objective	Items are on top of each other or are in contact, which impacts segmentation				
Steps	1. Place items on top of each other to test the robot				
Expected	Adjust an algorithm for the blocks world-like problem				
Date-Result	16/05/2023 - Fail				

Test ID	48	Category	Functional	Severity	Major
Objective	Item depth not perceived properly				
Steps	1. Check if the gripper reaches the correct depth				
Expected	Adjust the mobile base so that depth is perceived properly				
Date-Result	18/05/2023 - Pass				

Test ID	49	Category	Functional	Severity	Minor
Objective	Order cannot be received				

Steps	1. Check if the order is received from the outside stakeholders of the system
Expected	Report an error to outside users
Date-Result	16/05/2023 - Pass

Test ID	50	Category	Non-Functional	Severity	Minor
Objective	Environment generation shouldn't take longer than 5 minutes				
Steps	1. Measure the environment generation time				
Expected	Report the time				
Date-Result	13/05/2023 - Pass				

6 Maintenance Plan and Details

Whenever a problem pops up, such as an environment where the robot is unable to pick items from shelves, or a type of item that is particularly hard for the robot to grasp, the GroceryBee team will make the necessary adjustments to the code and the models to account for this problem. The changes could include the implementation of a different kind of gripper, such as a suction gripper, or a change in the infrastructure of the code, for example a more efficient way to pick grasp candidates to take objects from shelves. The possible extensions to the project in the future are discussed in the Conclusion and Future Work part of this report.

7 Other Project Elements

7.1 Consideration of Various Factors in Engineering Design

Here the constraints on the project (i.e implementation, design, time, language) and the needs of various factors like health, safety, environmental, and economic factors will be discussed. The section will mention how these factors have affected the analysis process and will affect the project in the future.

Implementation

The robot will be developed using Drake—a simulation toolbox developed at MIT Computer Science and Artificial Intelligence Lab (CSAIL) that enables model-based design and verification for robotics. The perception of the robot will be built in Drake.

Python will be the main language for developing the robotic skills, through Drake's Python integration.

OpenGL API will be used to render 2D and 3D vector graphics.

MeshCat will be used to remotely run and view 3D simulations on the web.

For computer vision, learning will be enabled by the PyTorch framework while the Open3D library will be used to facilitate geometric queries.

Libraries such as IPOPT, SNOPT, Gurobi, and MOSEK will be utilized as optimizers.

Design

The robot cannot be too wide in order to fit in between the shelves in the supermarkets.

The robot must be able to move between shelves without the need for external support.

Time

The reports and demos as part of the course CS-492 will be prepared and handed in before the deadlines stated in the course requirements.

The robot will be virtually implemented before the project delivery deadline and ready to be showcased at CSFair in Spring 2023.

Language

The robot will only recognize item names in English.

The shopping list will be provided only in English.

Economic Factors

Building a physical robot or buying a premade one would cost well over the budget of this project, and are out of the project's main focus. For that reason, the robot will only be built virtually through simulations since the gap between simulation and reality is sufficiently small in robotics nowadays.

The Drake simulator and other libraries and APIs are free to use, so will not affect the project's budget.

Health and Safety Factors

The robot should avoid any kind of collision, especially with humans, for the safety of the customers and employees around.

The robot cannot carry out sudden, unexpected maneuvers since that would threaten the individuals nearby.

The robot will not have pointy, sharp edges or parts made of harmful materials that may cause harm when in touch with human skin.

Global Factors

Since there are no global standards in supermarket plans, it is discussed that the project has to adapt to different supermarket environments and plans. To develop and demonstrate the project, an example virtual setting will be designed in MeshCat and used throughout the stages. However, the market plan will be changeable and the robot should be able to function in different environments.

Environmental Factors

Sustainability is a topic that is getting more and more prominent globally. With correct engineering practices that account for environmental factors, it is possible to sustain the engineering processes without harming the environment and wasting excessive amounts of energy. Sustainability in robotics is therefore gaining importance too. So, in the analysis, we had to consider how environmental factors may affect our design. Firstly, the energy consumption of the robots will be reduced by optimizing the paths the robots will take [5]. Next, the use of unsustainable materials in the production of the robot can be avoided, with the exception of sensors, computer chips, etc. [6], [7]. Lastly, to ensure sustainable development, no prototypes or physical components will be built. All testing and development will be carried out through virtual simulations.

7.2 Ethics and Professional Responsibilities

- Any and all libraries, frameworks, and tools used in GroceryBee should be licensed properly.
- The orders given to the robot should be securely stored to ensure the safety of all customers.
- Regardless of the security of the system, the users should be notified that their data is being processed in GroceryBee.
- The project might result in several lost jobs, but this might get fixed by the fact that GroceryBee might start new job areas of its own.
- The project is proprietary for now, however, it might be open source in the future.
- The data processing should be compliant with data protection laws such as GDPR.

7.3 Teamwork Details

7.3.1 Contributing and functioning effectively on the team

Efe: The responsibilities were divided among all of the team members in the implementation phase. Efe took part in creating the environment for the robot, pick and place without cameras, and placing the RGBD cameras to receive real time information so items could be picked up with information about the environment. He also contributed to implementing a mobile base for the robot. Moreover, Efe took the lead role in creating dynamic market environments, positioning shelves and cameras, as well as placing items on shelves. When writing the reports, he took active roles in all of them and helped with the parts that were assigned to him, like other members. All in all, as the rest of the group, he has fulfilled his responsibilities in every part of the project.

Arda: Arda contributed to creating the simulation environment, setting up the initial basic pick and place approach, configuring the robot's path for picking up objects from shelves, setting up the point clouds that are obtained from cameras to exclude parts that didn't belong to shelves,

converting the code from the deep note environment to local version and inclusion of git and version control and lastly, implementation of the mobile base of the robot.

Eren: Eren contributed to the initialization of the project, robot and the simulation environment. He has worked on basic pick and place, creating the shelves/placing objects of the simulation. He worked on using deep learning models for instance segmentation to replace classical methods used in perception for grasp selection. He worked on combining DL based semantic instance segmentation with ICP for robust grasp selection.

Mert: Mert contributed to initialization of the project, the first pick-and-place algorithms. He helped set up the simulation environments and created objects to place in the simulation environment. He also worked on making the robot mobile and trajectory optimization.

Emir: Emir contributed to building the simulation environment, initiating basic pick & place, planning the trajectory with DiffIK, and implementing the grasp selection system while ensuring that the robot functions reliably and collision-free in the first half of the project. Currently, he is working on motion planning—implementing kinematic trajectory optimization—and helping develop a mobile base.

7.3.2 Helping create a collaborative and inclusive environment

Efe: To create a collaborative environment, Efe helps his teammates whenever they need it and offers knowledge. Further, in any situation he makes sure to listen to his teammates' ideas on how to tackle an issue, and makes decisions accordingly. He is able to utilize his team members' strength and asks for help whenever he requires it, in order to save time and effort. All in all, he tries to maintain an inclusive environment for his teammates where everyone feels valued and listened.

Arda: Arda was always enthusiastic to involve others in trying to implement certain aspects of the project, even when they didn't believe in themselves that they could do it. He made sure that the workload was distributed equally amongst members and asked other members to join him in the implementation of crucial tasks to ensure teamwork. He listened to and considered everyone's suggestions while solving problems with regards to the project and debated ideas to make sure that everyone feels involved and their opinion is valued.

Mert: Mert helped his teammates whenever he could. He made sure that he was not doing less work than he was supposed to, and he helped other teammates if they had more work assigned to them than they were capable of doing.

Emir: Emir is always eager to help his friends in their tasks, provide his teammates with his understanding of Drake and other frameworks, and request support whenever he needs technical help or new ideas to progress. By working together with others in many aspects of the project and constantly exchanging ideas, he is helping create a collaborative and inclusive environment.

Eren: Eren is always helpful to his friends when they need it, and provides both emotional and technical support during the problems they faced throughout the project. He tried to work with others in an inclusive and accepting way to develop new ideas and solve problems.

7.3.3 Taking lead role and sharing leadership on the team

Efe: For the report aspect of the project, Efe usually shares leadership with Arda to assign responsibilities and make sure everything is done on time and in an acceptable way. For the implementation part, Efe, like other members of the team, leads a certain section, such as implementing various parts of the robot such as a joint for the mobile base or dynamic market environment generation. He usually shares the leadership of implementation with another person, for example with Mert in environment generation, to develop the project effectively and collaboratively.

Arda: Arda took responsibility in ensuring that the deliverables are implemented before the deadlines and distributed project's pieces to team members according to their interests. He also migrated the entire project from DeepNote (a collaborative web based notebook) which was inefficient in collaboration and version control to local development environments which reduced the loading time of the simulation significantly and improved version control. Lastly, he took shared leadership in perception and learning packages by implementing most of it.

Mert: Mert took a leadership role at the beginning of the project, helping set up the initial robot and the environment.

Emir: Emir shared leadership during most of the implementation process, helping navigate the progress on DeepNote and then the git repository. Currently, he is leading the "motion planning" aspect, which is a major subsystem of the project.

Eren: Eren took a leadership role at the beginning of the project, helping set up the initial robot and the environment. He worked on perception in the second semester.

7.3.4 Meeting Objectives

In our Analysis Requirements report, we had listed 4 work packages: Pick and Place, Perception, Navigation and Learning. Below, all of these work packages will be discussed separately, regarding which objectives were set for which work package and how much of them have been met.

Pick and Place

The main objective of this work package is to implement the main functionality of picking up the object with known coordinates and allowing the robot to pick and place objects. This has been done successfully, the robot could pick up items with known positions at the end of the Fall Semester, and this was shown in the demo for CS491. Moreover, in the Spring semester, RGB-D cameras have been added to the environment, and with the help of perception modules, objects whose coordinates are not previously known can also be picked up by the robot. GroceryBee exploits the real-time information it receives from the RGB-D cameras, evaluates grasp candidates and finally it can safely pick items up from the shelves and place them in the bin for delivery.

Perception

The first part of pick and place was done with known object coordinates. The purpose of this part was to find the object through frames obtained from the depth camera which is mounted on top of the robot. The object's 3D coordinates relative to the robot were envisioned to be detected, calculated, and fed into the pick and place functions so the robot would be able to detect and pick locations of unknown objects.

The first task of this work package was object detection, which has been successfully implemented and the robot can utilize information from the various RGB-D cameras in the environment to determine the locations of objects in the environment. For processing information, Lang-SAM and BERT models are used. SAM is the Segment Anything Model, developed by Meta, and Lang-SAM is a tool that combines text prompts with the SAM model to receive natural language input (the name of the object the robot is looking for) and segment the correct objects in the environment based on that input. With the use of cameras and the aforementioned models, the robot can determine a grasp candidate for any object and pick it up from a given shelf. However, we had initially aimed to mount the RGB-D cameras on the robot itself, so there would be no need for extra cameras in the environment. In the current state of the project, this isn't implemented and the cameras are positioned at every shelf to receive information about a particular shelf.

Navigation

The objective for this work package was: *When the robot is able to detect and pick up objects, it has to be able to move around the supermarket aisles without crashing to go to the desired location and come back to the delivery point.* This has been implemented fully, and at the moment, GroceryBee is able to navigate the supermarket aisles to locate objects and pick them up. This work package included "Environment Generation" as a task, which has also been implemented. In the project, there are currently two different market environment generations, one with several rows of shelves and one with a U-shaped pattern. The items are also created on the shelves dynamically, and the robot can move to one shelf, pick up an item and put the item inside the bin that is welded to its side. We were also planning shortest path calculation as part of this work package, however this is not currently implemented.

Learning

The objective for this work package was: *A machine learning model should be implemented to detect the desired objects with the camera frames. Therefore, the robot could be trained to detect and deliver local grocery items.* For this purpose, the models mentioned in the Perception part (Lang-SAM and BERT) were used to supply natural language input received from customers, and segment the required objects with the use of cameras so the robot can pick them up and return them for delivery. The tasks for this work package have been fulfilled and the models have been integrated into the project.

7.4 New Knowledge Acquired and Applied

First and foremost, Drake documentation was of paramount importance for making this project work. It involves the environment to make the simulations, and all the mathematical optimization modules and fundamentals of robotics that we may use with a black-box approach, as it would be impossible to mathematically deduce them throughout the project. The MIT course on robotics lectured by Russ Tedrake is a great source and as a group, we followed it quite closely. For basic tasks such as pick/place and movement in an ideal setting, the framework provides us with tools to work on. However, robotics is unavoidably related to many other disciplines. Different parts of this project required us to borrow knowledge from different disciplines, these included, but were not limited to, task/motion planning, perception (traditional CV algorithms, Deep Learning), and control. For perception, we reviewed state-of-the-art approaches using neural models.

These appeared in different architectures, and we read articles regarding Transformer architectures. We have used pre-trained models to fine-tune them on a specified dataset and customize the SOTA models giving top metric results for our work to achieve better perception. However, we have explored the SAM (Segment Anything) model developed by Meta, which is a powerful tool that needs no explicit fine-tuning and is very successful for semantic segmentation. Although we needed natural language prompts and Meta did not provide an API for text prompting, we explored other fine-tuned models for inference by following instructions given by researchers that trained the model. We found the Lang-SAM repository, which is a 3rd party modification for text prompting; combining both SAM and another segmentation model GroundingDINO. These third party models are often on open-source GitHub (and use models such as BERT, CodeT5, BART, etc.) for other researchers to use and exploit the optimized model. For task/motion planning, we worked on finding novel approaches to leverage the current algorithms and develop better ones. Russ Tedrake's Robotic Manipulation lecture notes were an excellent source for this purpose.

8 Conclusion and Future Work

As presented in the Meeting Objectives part, most functionalities of the project have been implemented with success, and GroceryBee is able to pick items from grocery shop shelves for speedy delivery to customers, with the help of RGB-D cameras. A few of the potential extensions for the project have been discussed below.

The maintenance and extensions of the project will be conducted by the GroceryBee team. The project can be extended by implementing the following features:

- Placing cameras on the robot to reduce total camera count
- Adding different grippers for better object coverage
- Extending/improving the supermarket environment
- Adding multiple robots to task planning
- Creating grocery stock management system for employees
- Collecting data about customer preferences and implementing business analysis

9 User Manual

The following installation guide can be followed to install GroceryBee and run it to view the simulation:

1. git clone https://github.com/ardaOnal/grocery-bee
2. Install python3, pip, venv:

   sudo apt update

   sudo apt install python3 python3-pip python3-virtualenv

3. Create a virtual environment in the project's root directory and install the requirements:

   python3 -m venv env

   source env/bin/activate

   pip install -r requirements.txt

4. Install the models for perception:

   pip install torch torchvision

   pip install -U git+https://github.com/luca-medeiros/lang-segment-anything.git

After having done the above steps, you can simply run the main.py file and see GroceryBee in action.

10 References

[1] “Turkey’s online supermarket grows 434% in H1, reaching TL 1.8B,” Daily Sabah, Sep. 01, 2020. [Online]. Available: https://www.dailysabah.com/business/economy/turkeys-online-supermarket-grows-434-in-h1-reaching-tl-18b. [Accessed: Mar. 5, 2023].

[2] M. Djordjevic, “20 Incredible Online Grocery Shopping Statistics for 2022,” SaveMyCent, May 19, 2022. [Online]. Available: https://savemycent.com/online-grocery-shopping-statistics/. [Accessed: Mar. 5, 2023].

[3] K. Matthews, “5 Robots Now in Grocery Stores Show the Future of Retail,” Robotics Business Review, Apr. 10, 2020. [Online]. Available: https://www.roboticsbusinessreview.com/retail-hospitality/5-robots-grocery-stores-now/. [Accessed: Mar. 5, 2023].

[4] “Normal Vector,” Wolfram MathWorld. [Online]. Available: https://mathworld.wolfram.com/NormalVector.html. [Accessed: Mar. 7, 2023].

[5] B. Lennartson and K. Bengtsson, “Reducing energy consumption through optimised robot systems,” *Open Access Government*, Sept. 22, 2017. [Online]. Available: https://www.openaccessgovernment.org/reducing-energy-consumption-robot-systems/28061/. [Accessed: Mar. 8, 2023].

[6] J. Snow, “How to Build a More Sustainable Robot,” *The Wall Street Journal*, Oct. 19, 2021. [Online]. Available: https://www.wsj.com/articles/how-to-build-sustainable-robot-11634585544. [Accessed: Mar. 8, 2023].

[7] F. Hartmann, M. Baumgartner, and M. Kaltenbrunner, “Becoming sustainable, the New Frontier in Soft Robotics,” *Advanced Materials*, vol. 33, no. 19, p. 2004413, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/adma.202004413. [Accessed: Mar. 8, 2023].