Semi-Autonomous Behaviour Tree-Based Framework for Sorting Electric Vehicle Batteries Components

Rastegarpanah, Alireza; Gonzalez, Hector Cruz; Stolkin, Rustam

DOI:
10.3390/robotics10020082

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (Harvard):
Rastegarpanah, A, Gonzalez, HC & Stolkin, R 2021, 'Semi-Autonomous Behaviour Tree-Based Framework for Sorting Electric Vehicle Batteries Components', *Robotics*, vol. 10, no. 2, 82.
https://doi.org/10.3390/robotics10020082

robotics

MDPI

Article

Semi-Autonomous Behaviour Tree-Based Framework for Sorting Electric Vehicle Batteries Components

Alireza Rastegarpanah [1,2,*], Hector Cruz Gonzalez [2] and Rustam Stolkin [1,2]

[1] Department of Metallurgy & Materials Science, University of Birmingham, Birmingham B15 2TT, UK; r.stolkin@bham.ac.uk
[2] The Faraday Institution, Quad One, Harwell Science and Innovation Campus, Didcot OX11 0DG, UK; HLC945@student.bham.ac.uk
* Correspondence: a.rastegarpanah@bham.ac.uk; Tel.: +44-742-922-4991

Abstract: The process of recycling electric vehicle (EV) batteries currently represents a significant challenge to the waste management automation industry. One example of it is the necessity of removing and sorting dismantled components from EV battery pack. This paper proposes a novel framework to semi-automate the process of removing and sorting different objects from an EV battery pack using a mobile manipulator. The work exploits the Behaviour Trees model for cognitive task execution and monitoring, which links different robot capabilities such as navigation, object tracking and motion planning in a modular fashion. The framework was tested in simulation, in both static and dynamic environments, and it was evaluated based on task time and the number of objects that the robot successfully placed in the respective containers. Results suggested that the robot's success rate in accomplishing the task of sorting the battery components was 95% and 82% in static and dynamic environments, respectively.

Keywords: Behaviour Trees; electric vehicle batteries; recycling; mobile manipulator; semi-autonomous robot

Citation: Rastegarpanah, A.; Gonzalez, H.C.; Stolkin, R. Semi-Autonomous Behaviour Tree-Based Framework For Sorting Electric Vehicle Batteries Components. *Robotics* **2021**, *10*, 82. https://doi.org/10.3390/robotics10020082

Academic Editors: Dan Zhang and Juan M. Corchado

Received: 24 March 2021
Accepted: 12 June 2021
Published: 17 June 2021

1. Introduction

Efficient management of Electrical Vehicle Battery (EVB) packs is essential given the increase in the number of Electrical Vehicles (EVs) around the world. According to the work in [1], in just 2017 the world sales of EVs exceeded one million of units, which in turn, will generate around 250,000 tonnes of obsolete lithium-ion battery packs after completing their service life. Handling and recycling this type of material represents a high risk to the human health if not properly managed. This risk can be minimised if robots takeover the task of testing [2] or dismantling of EV batteries (EVB) in high demand situations [1,3]. However, a significant amount of research has to be carried out in order to ensure high reliability.

The process of recycling an EVB involves many stages, from the logistics process to removing raw material from the battery cells. Nonetheless, the literature has been mainly focused on the recycling process of lithium ion batteries and forsaken the other stages within the recycle process [4]. This is why there is an opportunity to cover the logistics problems such as sorting EVB packs and lithium ion battery modules using intelligent automation [5].

EVB designs have been developed to suit the manufacturer and the model of the car, therefore, as yet, no common standard has been developed. However, according to the work in [6] there are some fundamental steps for dismantling any EVB starting with opening the battery system until dismantling the battery modules and cells. Furthermore, most of the automation literature in EVB is oriented and tailored either towards the automation of detecting and unscrewing bolts [3] or the disassembly of specific parts such as battery modules [7]. As proposed in [3], fixed robotic arms have the capability to open

an EVB case and unscrew its components. However, due to their workspace limitation, assisted technology is required to remove, and to sort, the dismantled components from the workspace. Mobile manipulation addresses this problem as it offers more flexibility of the robot's workspace to achieve a desired task [8]. In general, effective industrial recycling highly depends on the quality of sorting. In addition, automating the process of sorting the battery components could considerably increase the safety of workers while reducing the labour costs and making this process economically viable.

In this paper, we propose a proof of concept framework that contributes towards automating the process of sorting the EVB components using a mobile robot. The proposed method is developed by a Behaviour Tree-based framework which semi-automates the process of removing EVB components from the workspace in an industrial environment. Behaviour Trees (BT) have been widely used in the game industry and in the recent years have been gaining traction in robotics for task execution monitoring as they offer better modularity than earlier control architectures such as Finite State Machines (FSM). FSM and its variant, Hierarchical-Finite State Machines, have been the standard choice of control architecture in the game and robotics industries. They are mostly used to programme autonomous task-level processes. They are composed of states, transitions and events. The transitions between states are defined by a response from events or conditions. They are considered one-way control transfers, where the process jumps to another state and continues executing from there. This creates a significant disadvantage over BTs when designing a complex task as stated in [9], as a significant amount of states and transitions has to be implemented to define a full task. Therefore, adding or removing states forces the developer to revisit the entire design to ensure logical consistency. This is not practical when handling large processes. Instead, BTs are composed of sufficient expressive and independent sub-behaviours which make the design and development of a BT highly independent and scalable. Thus, our proposed system opens the possibility to separate the development of complex robot skills from the architecture design allowing non-robot expert users to create new sub-behaviours.

Outline of the Proposed Method

The proposed framework is composed of three main modules: navigation, object pose tracker and grasping. As its name implies, the navigation module allows the robot to navigate within its workspace by generating collision-free paths as well as localising itself inside the environment. The object pose tracker finds a pose relative to the end-effector frame by using a model-based visual method and initialisation points introduced by a user. The grasping module receives the pose of an object and ensures collision-free grasping by using Rapidly-Exploring Random Trees (RRT) motion planner [10], inverse kinematics solver and vacuum gripper. The main contribution in this work is a novel Behaviour Tree architecture that manages and monitors the aforementioned modules and their sub-modules to collect and sort EVB components. The schematic diagram that outlines the proposed framework is illustrated in Figure 1.

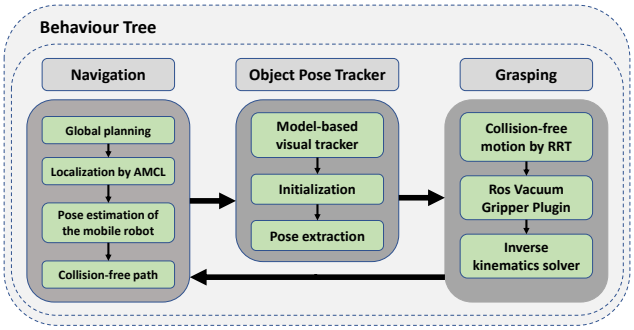


Figure 1. Schematic diagram of the proposed framework. It includes three main of Navigation, Object Pose Tracker and Grasping which are linked together using Behaviour Tree structure.

2. Related Work

Over the past years, different approaches for high-level dismantling EV batteries packs have been proposed. A methodology to develop a disassembly plan based on the trade-off between a partial and complete dismantling of an EVB has been proposed in [11]; it offers an optimised plan that ensures the best profitability with the least environmental impact. In another study, a disassembly plan was developed for an Audi Q5 battery by comparing one-to-one parts and defining a priority matrix based on their precedents [6]. Furthermore, a high-level disassembly process of the lithium-ion battery module has also been proposed. For instance, a method to characterise the battery model and recover material while providing safety guidelines on each disassembly stage has been developed and detailed in [12]. Note that our work centres its attention on sorting EVB packs components rather than the management of lithium-ion battery modules. Moreover, the aforementioned works have been developed to be handled manually by human, therefore automating this process offers new challenges for the automation industry.

The variability in the configuration of designs of EVBs is high as it depends on the manufacturer and on the car model. Therefore, tailored automation processes have to be carried out to disassemble specific EVB packs. For instance, a fixed robot-assisted battery disassembly workstation has been proposed in [3]. This allows both the human and robot to have access to the EVB simultaneously. The robot autonomously detects the locations of the fastener by either using visual recognition or human instruction. Then, it changes the tool bit to match the head of the bolt and plans to approach it and removes it. The detection was performed using a training set of M5 bolts along with the Haar Cascade classifier and 90% of true positives were obtained.

The industry has benefited from the use of mobile manipulators as they offer a better solution when a robot has to operate in a larger workspace than fixed manipulators. In addition to that, the use of redundant manipulators increases manipulability as well as defines secondary task such as avoiding the joint limits or singularities as stated in [13].

Behaviour Trees (BT) have been recently applied in industrial projects such as SCARA [14] which aims to develop a programming framework that allows non-expert users to plan complex robotics tasks with an ABB Yumi robot. Due to the expressiveness nature of BTs, plans can be developed and set up in a robot in a matter of days. The Collaborative System for Task Automation and Recognition (CoSTAR) [15] is another software framework that provides a graphic–user interface (GUI) that allows the training of robots to perform numerous tasks involving human cooperation. Furthermore, the Intera project from Rethink Robotics aims to develop an affordable "world-fastest" deployable robot using its well-known Sawyer manipulator [16]. Besides, BTs have also been used in areas of autonomous vehicles and international competitions such as The Amazon Pick challenge [9].

Different industrial applications had been developed using BTs. For instance, the software components required to perform autonomous robotic grasping and dexterous manipulation with high-level supervision using the Behaviour Architecture for Robotic Task (BART) have been detailed in [17]. They demonstrated that their system is able to grasp successfully nine different types of objects as well as to perform six different dexterous manipulation tasks such as stapling, flashlight on, drill hole, hang up a phone, open and locking a door with success rates of 92% and 91%, respectively. Nevertheless, the BART architecture defines a Behaviour Tree as a binary tree rather than a directed rooted tree as in our approach. This limits control flow nodes to only execute at maximum 2 action nodes, which significantly impacts the design complexity. Moreover, our work follows a Behaviour Tree model stated in [18], which is an unified definition in robotics.

A Behaviour Tree-based end user framework has been proposed in [19]; it integrates the CoSTAR framework [15] and different robot capabilities to set up a fixed UR5 manipulator to execute tasks such as unloading and kitting. Their framework extends the Behaviour Tree model to a task specification level such that the robot capabilities and the task plan are separated. This enables an end user to easily understand the robot capabilities and

therefore design a task plan without involving any code programming. They demonstrated that their kitting plan is robust to pose changes of objects achieving 82 successful unloading operations from a blender machine. In our framework, we leverage the capabilities of an holonomic mobile manipulator in order to add workspace flexibility to pre-designed tasks.

Bin-picking is a common application problem in many industry areas and therefore has been highly automated. This task is usually limited by the workspace of fixed robot arms. On the other hand, mobile bin-picking exploits the flexibility of a mobile base to increase the robot's workspace. For instance, a mobile bin-picking application was achieved using an Anthropomorphic Service robot [20]. The robot was programmed to collect unordered pipe connectors from a transportation box and place them in a different container. Active recognition was utilised in order to capture different points of view of the target scene thus, detecting objects and dealing with occlusion on the scene. After selecting the best object to be grasped, the robot navigates to a desired placement location. In our approach, we deal with larger objects and we leverage the use of the mobile base to reach any object on the scene and place them into containers, rather than just transport them. The focus of this work is not centralised in object pose estimation, but, a model-based object tracker is used. Despite the different industrial applications mentioned above, to the best of our knowledge, no work has been done to tackle the sorting of electric vehicle batteries components using a mobile manipulator and Behaviour Trees.

Assembly and disassembly tasks can be highly beneficial if different candidate grasps and re-grasp poses are considered. For instance, in [21] the authors developed a grasping RRT-based planner to simultaneously find the best grasping and place pose of an assemble part as well as the collision-free path between the initial and final pose. Furthermore, they introduced an orientation graph search-based method to find intermediate poses if the initial pose of a part needs to be reoriented and re-grasped to be assembled. Given that the grasping point is not limited by the initial pose of a part, reorientation of parts is an object of study [22]. Consequently, reorienting a target object considerably increases the chances of obtaining a successful disassembly or assembly. Furthermore, the authors of [23] propose a high-speed assembly planner which considers the environment and the current part poses in order to create a high and low-level assembly motion sequence. The aforementioned processes could be also applied to EVB disassembly tasks. As the main focus of this work is the combination of different robot capabilities using Behaviour Trees, a general motion planner approach is used.

The main contribution of this work is the development of a Behaviour Tree-based framework that enables the modular combination and monitoring of different robot capabilities to semi-automate the process of sorting the components of an EVB. The Behaviour Tree depicted in Figure 2 represents the architecture proposed in this work. This architecture executes and monitors a sequence to sort different objects from an EVB using a mobile manipulator. The flowchart diagram describing this sequence is shown in Figure 3. Furthermore, this diagram shows that an edge-based tracker is exploited to estimate the pose of a texture less object and the arm motion planner is used to plan and execute the suction grasping. Moreover, navigation capabilities are implemented such as A* for collision-free global path planning and Eband planner for path tracking and collision avoidance in a 2D map.

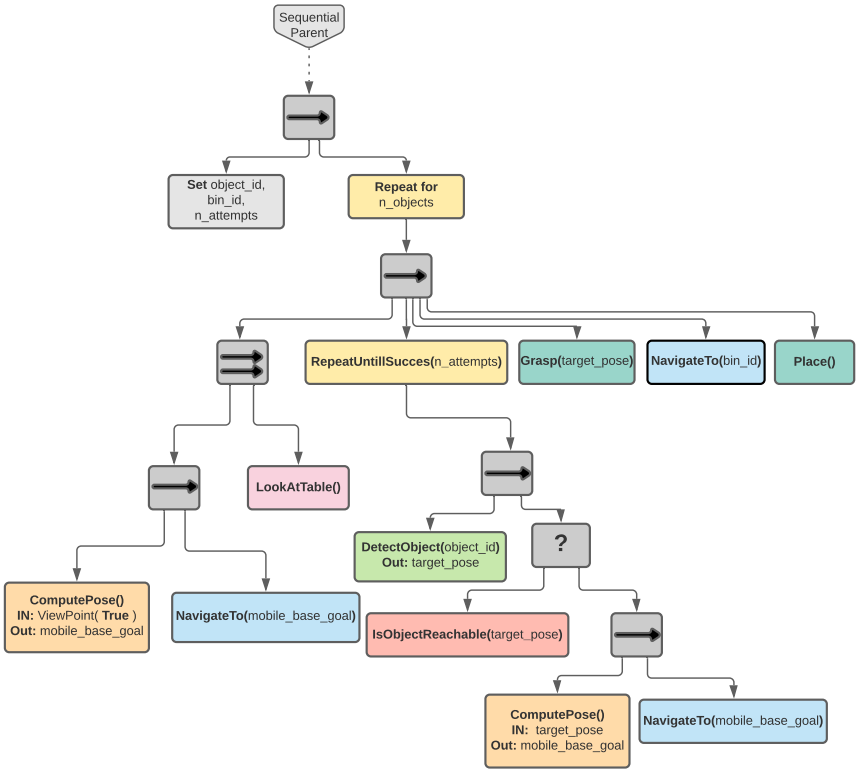


Figure 2. Behaviour Tree of the system to collect N number of $object_{id}$ class objects from a workstation and place them inside a container using a mobile manipulator.

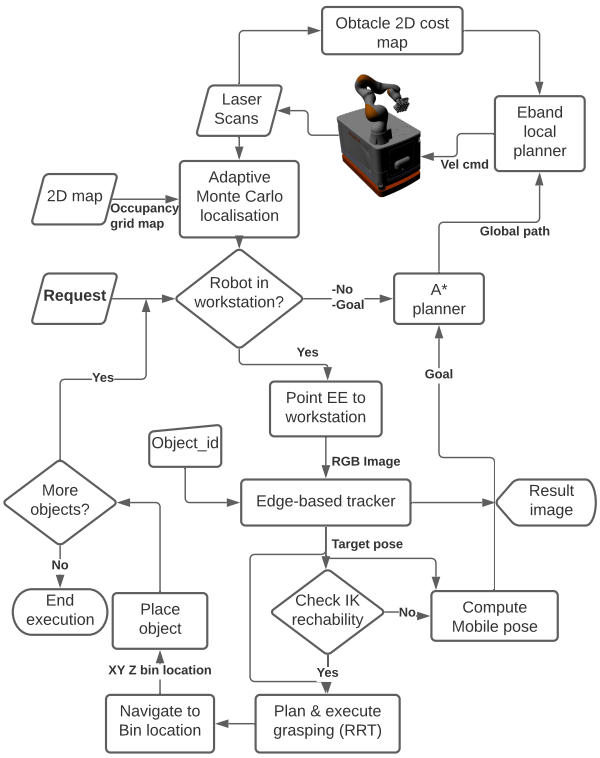


Figure 3. Flowchart diagram of the proposed framework showing the sequence of actions to retrieve and sort objects from a Electrical Vehicle Battery (EVB).

The remainder of this paper is organised as follows. Section 3 describes the industrial scenario utilised in this work as well as the system overview of the framework. Section 4

explains the methodology to detect and extract the pose of textureless objects from an RGB-D camera. Section 5 explains the autonomous navigation approach utilised in this work to move the robot inside its task-space. Section 6 explains the grasping and placing techniques of objects using a vacuum gripper. Section 7 outlines the characteristics of a Behaviour Tree as well as describes our behaviour architecture along with its action nodes. Sections 8 and 9 outline the experimental setup and the results of the performance of the robot in accomplishing the tasks in two different case studies respectively. Finally, a conclusion is presented and our future work is stated in Section 10.

3. System Overview

For the purpose of this work, we consider a scenario where an EVB pack is already dismantled using two KUKA KR500 robots, and the components are spread out as in the workstation depicted in Figure 4. For simulation purposes, we differentiate four different EVB-type objects such as plates (two types), battery module and L-shape brackets.

Objects are to be collected from the EVB pack and placed in four different bins located in the four corners of the workstation. Locations of the bins are recorded offline respective to the map coordinate frame. For testing and experiments, we use a KUKA KMR iiwa robot, which is a mobile manipulator with a 4-wheeled omnidirectional base and 7-DoF LBR arm mounted on the top of the base with a VG10 suction gripper [24] attached on the end-effector.

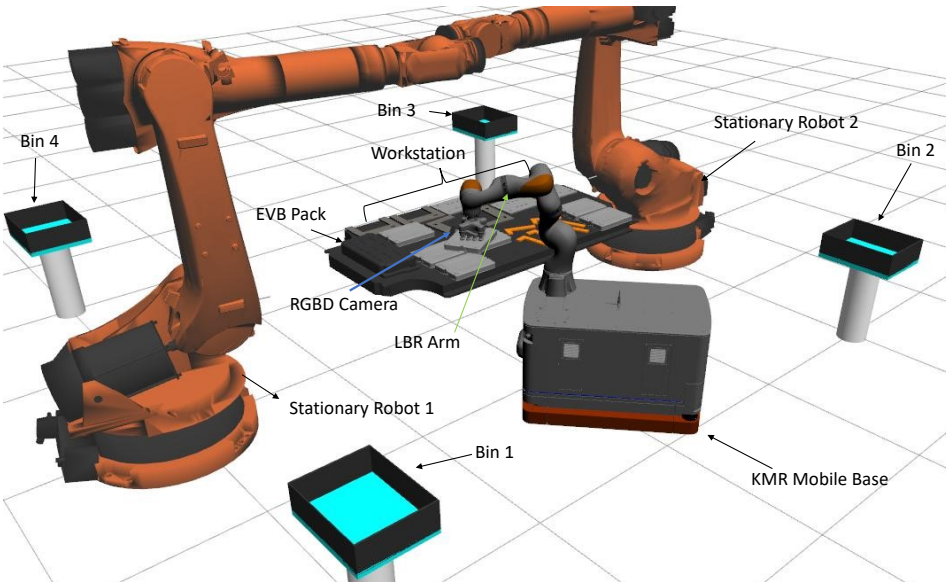


Figure 4. Bin-picking scenario. Mobile platform grasp objects from the simulated open electric vehicle battery and place them into boxes around the workstation.

The robot has two lasers scanners: one on the front-right and one on the rear-left side of the mobile base. These sensors allow the robot to detect static and dynamic obstacles in a 2D map. For object detection, an RGB-D camera is configured in an eye-in-hand fashion, which, in turn, adds flexibility to the detection point of view.

The target object and the bin number ID are specified at the beginning of the tree to ensure that every node on the tree reads the same variables at running time. The system is modelled based on pre-recorded poses around the workstation referred in this work as candidate poses, where the locations of each bin and the available poses to perform grasping are included. This helps to simplify the problem when an object is detected but is not reachable. In that case, the system will choose the best mobile pose candidate to perform the grasping of the target object. Moreover, the following poses of the mobile manipulator were recorded to ensure that the eye-in-hand camera has a panorama view of the entire workstation: one configuration of the LBR arm and two 2D poses for the mobile

base, and one for each side of the workstation. In this work, these mobile base poses are referred as view points.

The robot starts in an arbitrary position outside the workstation and it will navigate to the closest view point in order to execute the first detection. As the layout of the content of an EVB is known, the number of instances of the same class and its priority of collection (i.e., which object class has to be collected first and which one afterwards) can be set. Afterwards, a window showing the current camera view will be prompted to the user where they could manually select and initialise the object's pose, and thus initialise the tracking system. Therefore, the robot will attempt to approach the object and plan for grasping. After grasping the object, the robot will navigate to the respective bin where the arm is required to place it. The block diagram describing the aforementioned execution sequence is depicted in Figure 3.

4. Object Pose Tracker

Object pose detection becomes important when a robot is planning to grab specific objects from a workstation, but it becomes crucial when the objects represent a high risk for human safety (e.g., manipulating lithium-ion batteries). For this work, we set up a model-based visual tracker [25] that allows a user to initialise the tracking system in order to secure proper handling of the objects. Thus, it minimises the risk of a wrong detection and therefore avoids an undesired interaction between the suction gripper and the target object.

According to the work in [25], the pose estimation of an object could be considered as an optimisation problem. Define $\mathbf{q} = (\mathbf{t}^c, \mathbf{R}^c)$, where $\mathbf{t}^c$ and $\mathbf{R}^c$ are the translation and rotation matrix of the object respective to the camera frame. The goal is to approximate the independent parameters in $\mathbf{q}$ by minimising the error between the features x^* of the model M expressed in the camera domain and the features x of the projected model. The optimisation problem is solved using Levenberg–Marquard algorithm, which is an iterative non linear optimisation technique and the projection method used in this approach is stated in [26]. The objective function is defined as

$$\mathbf{q}^* = \underset{q}{\operatorname{argmin}} \sum_i \epsilon_i^2(x_i^*, x_i(M_i, \mathbf{q}, \gamma)) \tag{1}$$

where $x_i(M_i, \mathbf{q}, \gamma)$ is the function that represents the i-th feature of the model M_i using camera intrinsic parameters γ. Moreover, $\mathbf{q}^*$ is the optimal pose that minimises the objective function such that $\epsilon_i^2 = 0, \forall i$. It is assumed that the intrinsic parameters of the camera are known. Please note that this tracker is tested in a simulated environment, therefore, the virtual camera sensor has no distortion. As the objects to be grasped are textureless, the features x^* are extracted by using the moving edges algorithm [27]. In this work, the visual servoing platform C++ library (VISP) is used to estimate the pose of the object on an image, which integrates the aforementioned optimisation technique, the edge-based features extractor and the projection algorithm [28].

Four different CAD models were created in order to represent different components of EVB depicted in Figure 5. The tracking system is triggered once the robot is strategically positioned, the user is asked to define n initialisation points which partially denotes the shape of the model inside the image. The initialisation and the pose estimation results of two different objects are depicted in Figure 6.

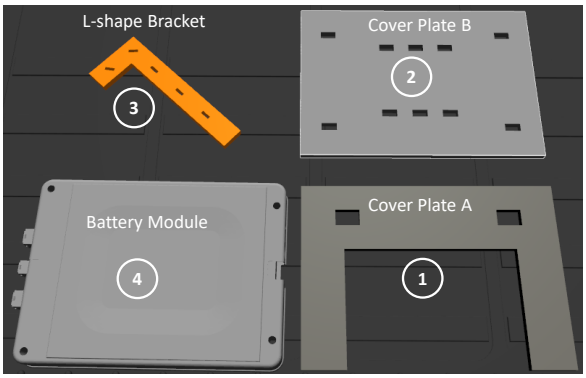


Figure 5. Different EVB-class objects used in the proposed work; (**1**) Cover Plate A, (**2**) Cover plate, (**3**) L-shape bracket, (**4**) Battery module.

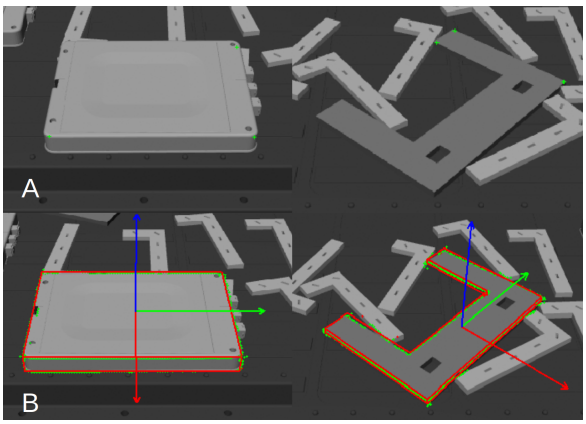


Figure 6. Initialisation of visual features of the objects and estimating the objects' poses using model-based visual tracker. (**A**) Initializing the visual features, (**B**) Extracting the pose of the object

5. Navigation

In order to safely move the robot around the workstation and to approach the bin's location, we used a global and local planner approach. Global planning is achieved by representing the environment in a 2D occupancy grid map generated from 2 laser scan sensors and to solve path planning problems we employed A* to find an optimised obstacle-free path to the goal [29].

For localisation, we implemented Adaptive Monte Carlo Localisation (AMCL), a common method and robust particle filter to estimate the pose of a mobile robot which compensates the error generated by the odometry motion model. It is based on the posterior probability that the robot is located in a pose **q** given a map and sensor data [30]. Local planning uses a local online 2D occupancy grid map (also know as 2D Cost Map) created from the current lasers scans. This leverages the navigation to detect dynamic objects (e.g., people or other dynamic objects) and replan if necessary. In order to follow the path generated by the global planning, we employ the elastic band approach [31], a reactive planner that uses sensor-based data to continuously deform the global path generated by the planner. Thus, it creates a new local collision-free path based on local changes around the robot. Furthermore, it implements a subset of the free-collision configuration space, which the authors refer as *bubbles*, as contemplating the entire free space of a three-dimensional configuration space is expensive. A subset of the free space around the configuration b is given by (2)

$$B(b) = \{q :\| b - q \| < p(b)\} \tag{2}$$

where the function $p(b)$ represents the minimum distance between the robot at configuration b and the obstacles (q) in the environment. As the robot is required to reach specific points in the space, poses for the mobile base were recorded (Figure 7) and labelled offline.

In this way, the navigation action is initialised by feeding a *goal_id* to it as described in Algorithm 1.

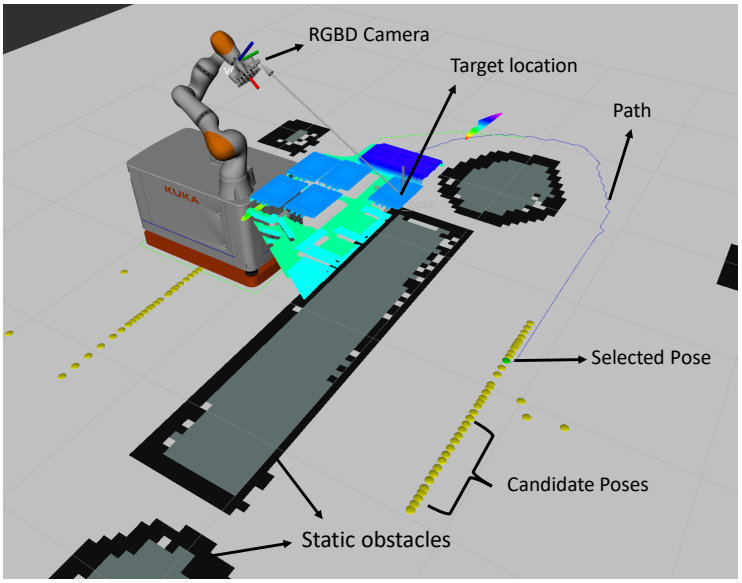


Figure 7. Robot detecting a new target location and planning for a better pose for grasping.

Algorithm 1: NavigateTo

Input: $goal\_id \in \mathbb{R} \geq 0$
$\mathbf{q}_{goal} \in \mathbb{R}^3 \leftarrow$ candiatePosesHashTable($goal\_id$);
$\mathbf{q}_{init} \in \mathbb{R}^3 \leftarrow$ AMCL();
$gloabalPath \leftarrow$ AStarPlanner($\mathbf{q}_{init}$, $\mathbf{q}_{goal}$);
initEbandPlanner($gloabalPath$) ;
while *goalIsNotReached()* **do**
  $\mathbf{q}_{current} \leftarrow$ AMCL();
  $obstacles \leftarrow$ 2DCostMap();
  $elastic\_path \leftarrow$ EbandPlanner($obstacles$, $\mathbf{q}_{current}$);
  pathTracker.Track($elastic\_path$, $\mathbf{q}_{current}$)
end
if *isRobotInXYTolerance()* **then**
  return *SUCCESS* ;
else
  return *FAILURE* ;
end

6. Object Grasping and Placing

Grasping objects with irregular shapes and specifications (e.g., metallic cases, looms and cables) would be less feasible using conventional 2–3 finger grippers. Therefore, in this study we employed suction grasping using a VG10 vacuum gripper [24]. In addition, suction grasping increases the chance of a grasp if the estimated pose was not very accurate. Moreover, suctions cups are more suitable to interact with deformable or sensitive objects (e.g., lithium-ion batteries) as they minimise the damage risk. In this study, we used Ros Vacuum Gripper Plugin to simulate the action of grasping [32].

The Rapidly-Exploring Random Trees (RRT) method [10] was used to generate a collision-free motion to grasp the objects. The sequence of grasp is depicted in Figure 8. Following grasp, a new motion plan is generated to move the arm from its current configuration to a safe pose. This pose was introduced to the system as it ensures that the grasped object has an adequate altitude respective from the battery pack such that it does not collide with other objects on the EVB while the mobile robot base is moving. Furthermore, the safe

pose places the end-effector above any container once the mobile base approaches it. In order to abolish the possibility of dropping a hazardous object while grasping, moving the arm to the safe pose or when placing the object into the container, an orientation constrain was applied to the motion planner. This ensures that the motion planner only takes into account movements where the arm always maintains the same post-grasp orientation while moving to the safe pose as illustrated in Figure 8. The algorithm describing the grasping pipeline is depicted in Algorithm 2.

Algorithm 2: Grasp

Input: $target\_pose \in \mathrm{SE}(3)$, $safe\_pose \in \mathrm{SE}(3)$, $preGrasp\_offset$, $postGrasp\_offset$

$eeGrasp\_pose \leftarrow$ trasnformToEE($target\_pose$)
$preGrasp\_pose \leftarrow eeGrasp\_pose + preGrasp\_offset$
$postGrasp\_pose \leftarrow eeGrasp\_pose + postGrasp\_offset$
$currentEE\_pose \leftarrow$ Robot.GetCurrentEEPose()
RRTPlanner.init()
$preGrasp\_plan \leftarrow$ RRTPlanner.Plan($currentEE\_pose$, $preGrasp\_pose$)
$approach\_plan \leftarrow$ RRTPlanner.Plan($currentGrasp\_pose$, $eeGrasp\_pose$)
$retreat\_plan \leftarrow$ RRTPlanner.Plan($eeGrasp\_pose$, $postGrasp\_pose$)
Robot.Execute($preGrasp\_plan$, $approach\_plan$, $retreat\_plan$)
`/* After executing the above plan, EE will be at post-grasp pose, therefore the same orientation is used as constraint */`
$P \leftarrow$ getCurrentEEOrientation()
RRTPlanner.SetOrientationConstraint(P)
$safe\_plan$ = RRTPlanner.plan($postGrasp\_pose$, $safe\_pose$)
if *Robot.Execute(safe_plan)* **then**
  return *SUCCESS*
else
  return *FAILURE*
end

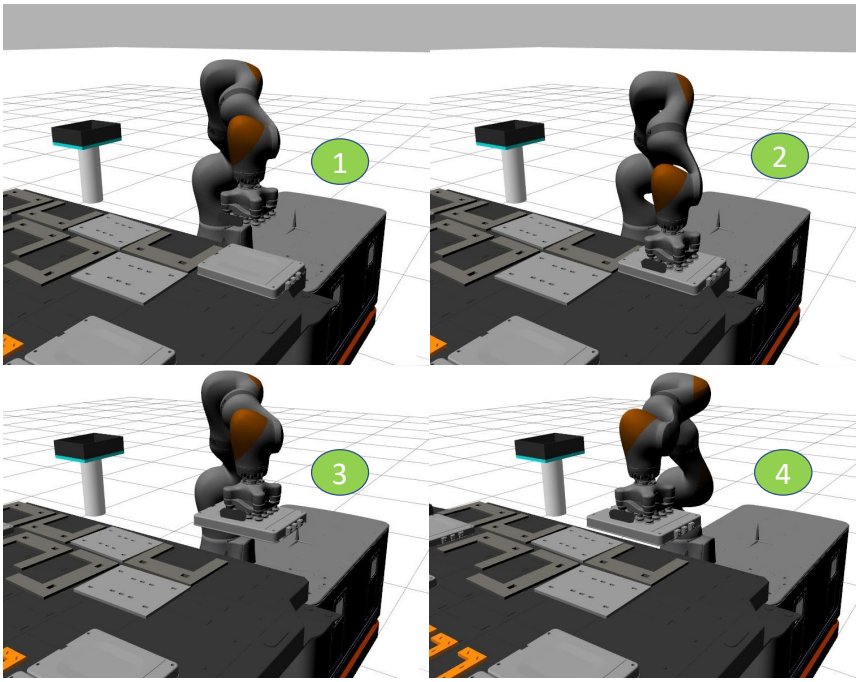


Figure 8. Grasping pipeline: (**1**) pre-grasp, (**2**) grasp, (**3**) post-grasp and (**4**) safe poses.

Once the robot approaches the container, a new motion plan is generated to place the object and to retreat the end-effector. These plans also take into account the orientation

constraint described above. For this end, inverse kinematics solver library from [33] was used as well as Moveit! C++ API for high-level interface with the sampling-based motion planner (RRT).

7. Control Architecture

In essence, the proposed framework, depicted in Figure 3, consists of a Behaviour Tree architecture. This tree is in charge of switching and monitoring the task execution, in other words, it executes the robot skills needed to perform the task given the state of the robot. BTs have been proved to offer better modularity and reactivity to an autonomous agents than conventional control architectures such as Hierarchical Finite State Machines (HFSM) and Decision Trees (FT) [34] which is the motivation behind this work.

As such a detailed explanation of a BT is out of the scope of this work only the characteristics are outlined. According to [9], a BT is composed of internal nodes known as *control flow* nodes and leaf nodes known as *execution* nodes. The execution of the tree starts from its root by sending *ticks* to its children. An internal node will run if and only if it constantly receives ticks from its parent (analogically, this can be seen as a clock signal that a CPU receives to execute tasks). The execution order of the tree is defined by its configuration; therefore, it can be either defined from the top to the bottom or from left to the right of the tree. A child node can either return *RUNNING*, *SUCCESS* or *FAILURE* to its parent. Internal nodes propagate their ticks to their children as long as they continuously return *RUNNING*.

Furthermore, *control flow* nodes are mainly divided in *sequential* (marked with an arrow $\rightarrow$), *fallback* (denoted with question mark?) and *parallel* (denoted with a double arrow $\rightrightarrows$). The first one will return *SUCCESS* if and only if all their children return *SUCCESS* or it will return *FAILURE* otherwise. The second one will success if and only if at least one of its children return *SUCCESS* and it will fail if all of its children fail. The final node can send *ticks* simultaneously to all of its children; thus, *parallel* executions can be achieved. This node will only return *SUCCESS* if N number of its children, where N can be lower than the total number of children, return *SUCCESS FAILURE* otherwise.

Each control node has one *parent* and at least one *child*. They will return *RUNNING* if both its success and fail conditions have not been met. Execution nodes represent the leaves of the tree and correspond to the actual primitives that a robot can execute which are divided in *condition* and *action* nodes. Usually, conditions serve as a guard of action nodes, thereby, ensure that the system meets the right requirements in order to execute a primitive or even another sub-tree.

For removing parts from the EVB we implemented six action nodes $A = \{Grasp, Place, NavigateTo, LookAtTable, DetectObject, ComputePose\}$ and one condition node $C = \{IsObjectReachable\}$. The BT architecture used in this work to collect N number of objects of the same type is depicted in Figure 2. The execution order is defined from the left to the right side of the tree. The *object_id* is set at the start of the tree to denote the name of the object to be collected as well as its respective bin index (*bin_id*), the number of attempts (*n_attempts*) and the number (*n_objects*) of same-class objects that are present in the workstation. This tree can be seen as a parametrised sub-behaviour tree and to fully programme the robot to collect the entire set of objects a *sequential* control flow node is applied on the top of the sub-trees. Consequently, the priority of collection can be set by using the execution hierarchies of the sequential node.

The sub-tree will be executed for a n number of objects with the same ID; this will be repeated for the four different object types. Therefore, the total number of executions will be $N = \sum_{i=1}^{4} \sum_{j=1}^{n} obj_{i,j}$ where $obj_{i,j}$ corresponds to the object of type i and j represents the instance number inside the scene. Note that every action node was interfaced in ROS and the tree was built using `BehaviourTree.CPP` C++ framework [9,35]. The description of each action node is explained in the following sub sections.

7.1. NavigateTo

As the locations of the bins and the locations where the robot should be to have a full panorama of the workstation are pre-recorded, this action receives only the index of the target pose. Furthermore, it takes into account the current pose of the robot respective to the map frame and evaluates if the desired goal is already achieved.

7.2. ComputePose

One of the main characteristic of the system is to make the most of the omnidirectional mobile base without requiring complex computations. If the robot is not able to reach an object, the system should find a new mobile pose that makes the detected object reachable. Additionally, this node will return the closest *view point* if required. This action receives as input the current target's 3D pose and returns a pose that the mobile base should be in to make the object reachable by the arm. In Algorithm 3, the logic behind this node is described. The *candidate poses* are recorded off-line and stored in a hash table. In addition, this node can return the closest *view point* pose if required.

Algorithm 3: ComputePose

Input: Candidate poses list $Q = q_1, q_2 \ldots q_n$, Target pose T and ViewPoint condition U
Output: Mobile base pose $q \in \mathbb{R}^3$

```
if U then
    RobotPose_current ← AMCL();
    return ClosestViewPoint (RobotPose_current)
else
    scores ← 0;
    targetPose ← trasformToMapFrame(T);
    foreach c in C do
        scores.append(ComputeEuclideanDistance(c, targetPose);
    end
    return scores.Min()
end
```

7.3. LookAtTable

This node creates a motion planning request to set the arm in a configuration such that the camera mounted at the end-effector has a full view-panorama of the workstation. This configuration was defined after an arduous experimentation to evaluate the field of view of the camera. Nevertheless, due to the modularity of the system, the arm configuration can be replaced at any time.

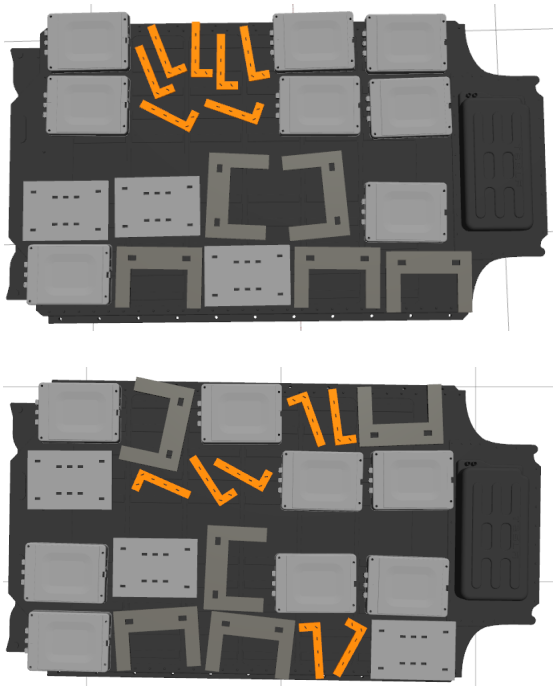
7.4. DetectObject

This node encapsulates what has been described in Section 4. It receives the *object_id* of the target, and it will prompt a window where the user is required to introduce the initialisation points of the given object. In addition, the user has this option to visualise the resulted pose (Figure 6) and to evaluate its reliability. Afterwards, the node sets the target pose inside the knowledge blackboard of the tree as other actions will read from it.

8. Experimental Setup

Two different workstation setups (which will be referred in this paper as case studies A and B) were used to test our framework. The first one, depicted in Figure 9a, is characterised as containing the objects clustered (except the battery modules) either to the bottom or the top side of the workstation as well as it respective containers. In other words, once the robot is on one side of the EVB, it does not need to navigate to the other side to grasp an object of the same class. However, the robot is still required to navigate to the other side once all the same-class objects have been collected. Furthermore, it is assumed that the

EVB pack is mounted and fixed on an industrial table, such that this one does not move during the robot operation.



(**a**)

(**b**)

Figure 9. Two different configuration of objects on the EVB, the robot is more often required to traverse around the workstation in scenario B rather than in A since, the objects are located in both sides (top and bottom) of the battery. (**a**) Workstation for the first scenario experiments. (**b**) Workstation for the second scenario experiments.

From the second workstation scenario (Figure 9b), it can be seen that at least two objects of the same class are located in one of the side of the EVB and the remainder are located at the opposite side. The left and right hand sides of the EVB were not considered as they are obstructed by the two fixed robots (KUKA KR500) as illustrated in Figure 4.

Furthermore, in case study A, only static objects were considered inside the scenario. In this work, a static object is referred as all the collision objects that were present when the 2D mapping of the environment was performed. In this scenario, the two side manipulators, the space underneath of the EVB and the containers were considered as static objects. On the other a hand, in case study B primitive-shape dynamic objects were added and placed into the scene such that they interfere with the planned path of the robot every time it needs to navigate to the other side either to place or to approach an object. This was performed to simulate any new obstacles such as humans or even other platforms that the robot may encounter while performing the task. The addition and removing of primitive objects was achieved by exploiting the spawning plugging of the simulator engine. In both case studies, before starting the task, the robot is initialised outside the workstation area, thus the first thing that the robot has to do is to navigate to the closest *viewpoint*.

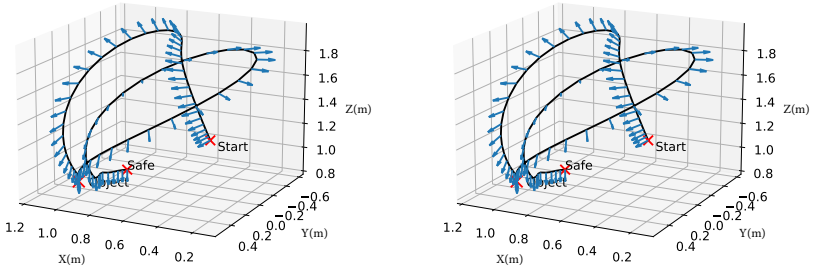
Given that the component's layout of a particular EVB can be known off-line. The order of collection or priority can be established by swapping the order of the different sub-behaviour trees (Figure 2), corresponding to the pick and place routine to sort all the instances of the same object class, inside a *sequential control flow* node. In both case studies, the order of collection task was set in the following order: *cover plate A*, *cover plate B*, *Battery Module* and *L-shape bracket* (Figure 5) where the number of instances for each class were 5, 3, 7 and 8, respectively. Both case studies where designed and simulated using Gazebo 9.0 with a 2.4 Ghz 4-core processor and a GeForce GT 750 M graphics card.

9. Results

As there are no similar approaches that are highly related to EVB dismantling process using task-control architectures such as FSM or Behaviour Trees in the literature, the proposed framework was evaluated in terms of success rates and execution times. We would like to highlight that these results are in terms that prove the concept and are based on simulation results which may vary if other computer capabilities are used.

After running the framework several times, the best performance was obtained by setting a maximum linear velocity v_{max} of 9.0 m/s, maximum absolute angular velocity θ_{max} of 0.4 rad/s as well as maximum translational a_{max} and rotational accelerations α_{max} of 2.0 m/s^2 and 1.5 rad/s^2, respectively.

Furthermore, as explained in Section 6, the end-effector of the robot should go to a safe pose after grasping an object while maintaining the current end-effector orientation and thus, the object orientation. However, it was observed that the motion planner was occasionally unable to find a solution that satisfies this constraint within the time limit. Therefore, the planner could execute undesirable trajectories such the one depicted in Figure 10a; it can be observed that the orientation of the end-effector is drastically flipped which caused the dropping of the object from the end-effector. Therefore, it was decided to constraint joint 5 from the arm with a certain tolerance T. This, in turn, reduces the amount of trajectories that the motion planner takes into account where the orientation constraint is violated. Consequently, preferred grasping trajectories were obtained, one example of them is depicted in Figure 10b where it can be observed that the end-effector successfully maintained its orientation while moving to *safe* pose. Moreover, this constrain was applied only to the motion plan to go from the post-grasp pose to the safe pose and for both case studies.



(**a**) (**b**)

Figure 10. Examples of end-effector trajectories. (**a**) End-effector trajectory of a unsuccessful grasping. (**b**) End-effector trajectory of a successful grasping.

9.1. Case Study A

For this experiment, only static objects were considered as part of the sorting task (Figure 11). Table 1 summarises the results of the proposed framework after running it three times. An overall success rate of 95.6% was achieved in an average time of 51.63 min. As described in the Table 1, 100% success rate was achieved for the majority of objects. For the *Bracket* object, the robot was unable to grasp the object during the experiments due to limitations of the simulation engine to successfully simulate the suction grasping for small objects. However, as the evaluation is based on simulation we considered it as a failure. Note that the time of accomplishing the task will also depend on the expertise of the user to initialise the pose of the object given the current view of the camera as explained in Section 4.

Table 1. Results of case study A.

Object Type	Avg Time (min)	Std (min)	Objects Placed	Task Achieved
Battery Module	25.54	1.31	8/8	100%
Plate A	10.96	1.47	5/5	100.0%
Plate B	5.68	0.67	3/3	100%
Bracket	9.45	1.08	6/7	85.71%
Total	51.63		22/23	95.6%

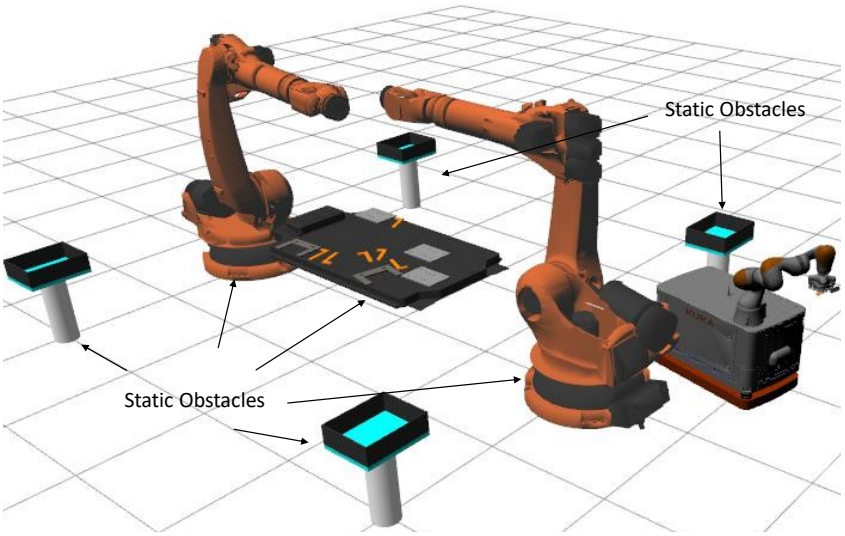


Figure 11. Case study A: Robot navigating while holding an object (no dynamic obstacles).

9.2. Case Study B

For the second experiment, the same task was executed. As depicted in Figure 12, in this case study different dynamic obstacles in different primitive shapes were instantiated in front of the robot every single time it needed to either navigate to a container or to the opposite side of the workstation. Table 2 summarises the results obtained after running the framework. It can be observed that the average time to collect the four objects increases approximately by 60% and the success rate decreases to 82.6% in order to sort all the objects. As expected, it takes more time for the robot to sort all the objects given that it needs to navigate more often between both sides of the workstation. Furthermore, whenever a dynamic obstacle appears in front of it, it is necessary to re-plan and generate a new path to avoid it. The performance of the reactivity of the navigation approach used in this work is highly affected by the CPU resources to obtain a better solution. Thus, the average time to complete the task is also affected by it.

Table 2. Results of case study B.

Object Type	Avg Time (min)	Std(min)	Objects Placed	Task Achieved
Battery Module	40.34	1.49	7/8	87.5%
Plate A	15.86	1.54	4/5	80%
Plate B	8.72	1.26	3/3	100%
Bracket	17.43	1.13	5/7	71.4%
Total	83.09		19/23	82.6%

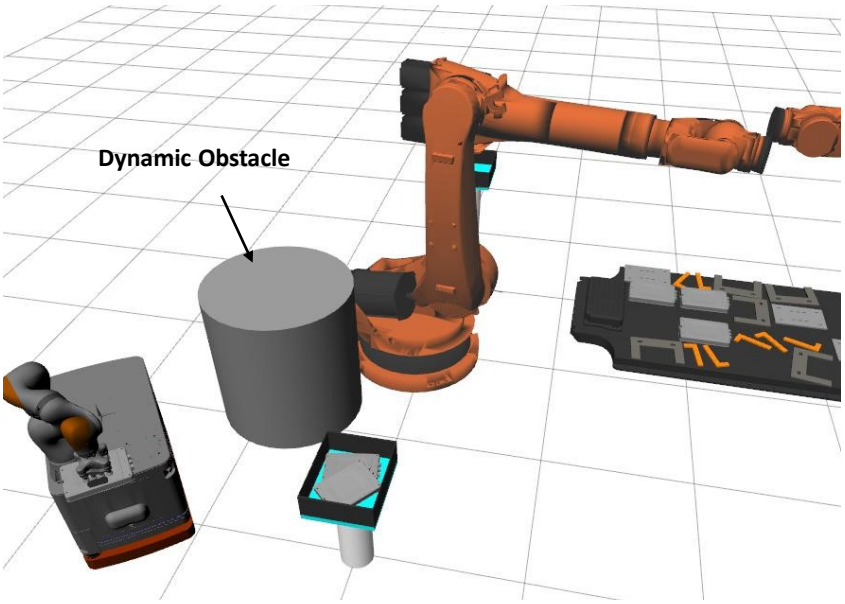


Figure 12. Case study B: Robot encountering a dynamic obstacle and re-planning its original path to avoid it.

In both Tables 1 and 2, the standard values (Std) are relatively small and they indicates that the required time for sorting a specific object is almost the same over all the trials. However, this value could be reduced significantly by replacing the self-initialisation tracker with a more advanced learning-based tracker without requiring a user to select the desired object.

10. Conclusions

This work, for the first time, proposes a generic/modular framework to automate the process of sorting the EVB components in an unstructured environment using Behaviour Tree architecture. This strategy enables the user to execute and monitor the process of extracting and sorting the components from an EVB pack. The proposed framework was tested in simulation in two different environments: the first environment was modelled with static objects in which the mobile robot navigated towards the workspace and accomplished the task of sorting in 51 min with success rate of 95.6%, and in the second environment, dynamic objects were modelled to randomly make obstacle for the mobile robot and in this case the robot could successfully complete the task of sorting in 83 min with the success rate of 82%. The proposed framework benefits from its deployed Behaviour Tree which has this potential to be generalised with appropriate actions for various EVB packs.

The implementation of learning-based methods to autonomously detect the objects and classify them along with a pose estimation using point cloud data could add benefit to the current work. Furthermore, the full coordination between the manipulator and the mobile base to create more accurate motion plans and sampling the 3D free space will add robustness to the system to approach and grasp objects. This in turn can increase the level of task success.

Transferring this work to a real-world scenario should not represent difficulties in terms of the task-control architecture (BT). However, more action or condition nodes may be required to handle future case failures. Examples of them: handling properly when the object could not be grasped and the object has fallen from the end-effector, verifying that the grasped object is the correct one by adding force-torque sensors or planning when the robot should go back to home station due to low battery. Furthermore, as mentioned before, Behaviour Trees offer a great modular architecture, thus adding more sub-behaviours or new robot capabilities can be efficiently scalable.

Author Contributions: Conceptualization, A.R. and H.C.G.; methodology, A.R. and H.C.G.; software, H.C.G.; validation, A.R. and H.C.G.; formal analysis, A.R. and H.C.G.; investigation, A.R. and H.C.G.; resources, A.R. and H.C.G.; data curation, H.C.G.; writing—original draft preparation, A.R., H.C.G. and R.S.; writing—review and editing, A.R., H.C.G. and R.S; visualization, H.C.G.; supervision, A.R. and R.S; project administration, A.R., and R.S.; funding acquisition, A.R. and R.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was conducted as part of a project called Reuse and Recycling of Lithium-Ion Batteries (RELIB). This work was supported by the Faraday Institution [grant number FIRG005].

Data Availability Statement: The data that support the findings of this study (Semi-autonomous Behaviour Tree-Based Framework For Sorting Electric Vehicle Batteries Components) are openly available in Figshare (https://figshare.com/s/ddb832077aff5ccda96e (accessed on 15 June 2021)) with doi (10.6084/m9.figshare.13360349)

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Harper, G.; Sommerville, R.; Kendrick, E.; Driscoll, L.; Slater, P.; Stolkin, R.; Walton, A.; Christensen, P.; Heidrich, O.; Lambert, S.; et al. Recycling lithium-ion batteries from electric vehicles. *Nature* **2019**, *575*, 75–86. [CrossRef] [PubMed]
2. Rastegarpanah, A.; Ahmeid, M.; Marturi, N.; Attidekou, P.S.; Musbahu, M.; Ner, R.; Lambert, S.; Stolkin, R. Towards Robotizing the Processes of Testing Lithium-ion Batteries. *Proc. Institut. Mech. Eng. Part I J. Syst. Control Eng.* **2020**, in press.
3. Wegener, K.; Chen, W.H.; Dietrich, F.; Dröder, K.; Kara, S. Robot assisted disassembly for the recycling of electric vehicle batteries. *Procedia Cirp* **2015**, *29*, 716–721. [CrossRef]
4. Melin, H.E. *State-of-the-Art in Reuse and Recycling of Lithium-ion Batteries—A Research Review*; The Swedish Energy Agency: London, UK, 2019.
5. Beaudet, A.; Larouche, F.; Amouzegar, K.; Bouchard, P.; Zaghib, K. Key Challenges and Opportunities for Recycling Electric Vehicle Battery Materials. *Sustainability* **2020**, *12*, 5837. [CrossRef]
6. Wegener, K.; Andrew, S.; Raatz, A.; Dröder, K.; Herrmann, C. Disassembly of Electric Vehicle Batteries Using the Example of the Audi Q5 Hybrid System. *Procedia CIRP* **2014**, *23*, 155–160. [CrossRef]
7. Schmitt, J.; Haupt, H.; Kurrat, M.; Raatz, A. Disassembly automation for lithium-ion battery systems using a flexible gripper. In Proceedings of the 2011 15th International Conference on Advanced Robotics (ICAR), Tallinn, Estonia, 20–23 June 2011; pp. 291–297. [CrossRef]
8. Dömel, A.; Kriegel, S.; Kaßecker, M.; Brucker, M.; Bodenmüller, T.; Suppa, M. Toward fully autonomous mobile manipulation for industrial environments. *Int. J. Adv. Robot. Syst.* **2017**, *14*. [CrossRef]
9. Colledanchise, M.; Ögren, P. *Behavior Trees in Robotics and AI: An Introduction*; Chapman & Hall/CRC Artificial Intelligence and Robotics Series; CRC Press: Boca Raton, FL, USA, 2018.
10. LaValle, S.M. Rapidly-Exploring Random Trees: A New Tool for Path Planning. 1998. Available online: https://www.cs.csustan.edu/~xliang/Courses/CS4710-21S/Papers/06%20RRT.pdf (accessed on 15 June 2021).
11. Alfaro-Algaba, M.; Ramirez, F.J. Techno-economic and environmental disassembly planning of lithium-ion electric vehicle battery packs for remanufacturing. *Resour. Conserv. Recycl.* **2020**, *154*, 104461. [CrossRef]
12. Marshall, J.; Gastol, D.; Sommerville, R.; Middleton, B.; Goodship, V.; Kendrick, E. Disassembly of Li Ion Cells—Characterization and Safety Considerations of a Recycling Scheme. *Metals* **2020**, *10*, 773. [CrossRef]
13. Prats, M.; Sanz, P.J.; del Pobil, A.P. The advantages of exploiting grasp redundancy in robotic manipulation. In Proceedings of the 5th International Conference on Automation, Robotics and Applications, Wellington, New Zealand, 6–8 December 2011; pp. 334–339. [CrossRef]
14. agreement ID: 644938, H.I...G. Smart Assembly Robot with Advanced FUNctionalities. 2015. Available online: https://cordis.europa.eu/project/id/644938 (accessed on 15 June 2021).
15. Paxton, C.; Hundt, A.; Jonathan, F.; Guerin, K.; Hager, G.D. CoSTAR: Instructing collaborative robots with behavior trees and vision. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 201; pp. 564–571. [CrossRef]
16. Robotics, R. Intera. 2021. Available online: https://www.rethinkrobotics.com/intera (accessed on 15 June 2021).
17. Bagnell, J.A.; Cavalcanti, F.; Cui, L.; Galluzzo, T.; Hebert, M.; Kazemi, M.; Klingensmith, M.; Libby, J.; Liu, T.Y.; Pollard, N.; et al. An integrated system for autonomous robotics manipulation. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 2955–2962.
18. Marzinotto, A.; Colledanchise, M.; Smith, C.; Ögren, P. Towards a unified behavior trees framework for robot control. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 5420–5427. [CrossRef]

19. Guerin, K.R.; Lea, C.; Paxton, C.; Hager, G.D. A framework for end-user instruction of a robot assistant for manufacturing. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 6167–6174.
20. Nieuwenhuisen, M.; Droeschel, D.; Holz, D.; Stückler, J.; Berner, A.; Li, J.; Klein, R.; Behnke, S. Mobile bin picking with an anthropomorphic service robot. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 2327–2334.
21. Ali, A.; Lee, J.Y. Integrated Motion Planning for Assembly Task with Part Manipulation Using Re-Grasping. *Appl. Sci.* **2020**, *10*, 749. [CrossRef]
22. Wong, C.C.; Yeh, L.Y.; Liu, C.C.; Tsai, C.Y.; Aoyama, H. Manipulation Planning for Object Re-Orientation Based on Semantic Segmentation Keypoint Detection. *Sensors* **2021**, *21*, 2280. [CrossRef] [PubMed]
23. Kang, T.; Yi, J.B.; Song, D.; Yi, S.J. High-Speed Autonomous Robotic Assembly Using In-Hand Manipulation and Re-Grasping. *Appl. Sci.* **2021**, *11*, 37. [CrossRef]
24. Onrobot. VG10 Vacuun Gripper. 2021. Available online: https://www.universal-robots.com/plus/urplus-components/handling-grippers/vg10-vacuum-gripper/ (accessed on 15 June 2021).
25. Trinh, S.; Spindler, F.; Marchand, E.; Chaumette, F. A modular framework for model-based visual tracking using edge, texture and depth features. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 89–96. [CrossRef]
26. Comport, A.I.; Marchand, E.; Pressigout, M.; Chaumette, F. Real-time markerless tracking for augmented reality: The virtual visual servoing framework. *IEEE Trans. Visual. Comput. Graph.* **2006**, *12*, 615–628. [CrossRef] [PubMed]
27. Bouthemy, P. A maximum likelihood framework for determining moving edges. *IEEE Trans. Pattern Anal. Mach. Intell.* **1989**, *11*, 499–511. [CrossRef]
28. Marchand, É.; Spindler, F.; Chaumette, F. ViSP for visual servoing: A generic software platform with a wide class of robot control skills. *IEEE Robot. Autom. Mag.* **2005**, *12*, 40–52. [CrossRef]
29. Marin-Plaza, P.; Hussein, A.; Martin, D.; Escalera, A.d.l. Global and local path planning study in a ROS-based research platform for autonomous vehicles. *J. Adv. Transp.* **2018**, *2018*. [CrossRef]
30. Dellaert, F.; Fox, D.; Burgard, W.; Thrun, S. Monte carlo localization for mobile robots. In Proceedings of the 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), Detroit, MI, USA, 10–15 May 1999; Volume 2, pp. 1322–1328.
31. Quinlan, S.; Khatib, O. Elastic bands: Connecting path planning and control. In Proceedings of the IEEE International Conference on Robotics and Automation, Atlanta, GA, USA, 2–6 May 1993; pp. 802–807.
32. Hsu, J. GazeboRosVacuumGripper. 2021. Available online: https://docs.ros.org/en/jade/api/gazebo_plugins/html/group__GazeboRosVacuumGripper.html#details (accessed on 15 June 2021).
33. Orocos Kinematics and Dynamics. The Kinematics and Dynamics Library. 2014. Available online: https://www.orocos.org/kdl_old.html (accessed on 15 June 2021).
34. Colledanchise, M.; Ögren, P. How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture, and Decision Trees. *IEEE Trans. Robot.* **2017**, *33*, 372–389. [CrossRef]
35. Faconti, D. BehaviorTree.CPP. 2017. Available online: https://www.behaviortree.dev/ (accessed on 15 June 2021).