

# Optimization of recyclable materials collection on conveyor belts

*Aikaterini Artemis Agiomavriti*

Thesis submitted in partial fulfillment of the requirements for the  
*Masters' of Science degree in Computer Science and Engineering*

University of Crete  
School of Sciences and Engineering  
Computer Science Department  
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisor: Prof. *Panos Trahanias*

---

This work has been performed at the Computational Vision and Robotics Laboratory (CVRL) of the Institute of Computer Science (ICS) of the Foundation for Research and Technology–Hellas (FORTH) .

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS).





UNIVERSITY OF CRETE  
COMPUTER SCIENCE DEPARTMENT

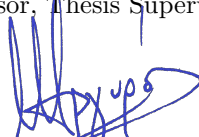
**Optimization of recyclable materials collection on conveyor belts**


Thesis submitted by  
**Aikaterini Artemis Agiomavriti**  
in partial fulfillment of the requirements for the  
Masters' of Science degree in Computer Science

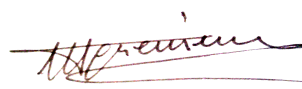
THESIS APPROVAL

Author:   
Aikaterini Artemis Agiomavriti

Committee approvals:   
Panos Trahanias  
Professor, Thesis Supervisor

  
Antonis Argyros  
Professor, Committee Member

  
Dimitris Tsakiris  
Research Director, Committee Member

Departmental approval:   
Polybios Pratikakis  
Assistant Professor, Director of Graduate Studies

Heraklion, March 2021



# Optimization of recyclable materials collection on conveyor belts

## Abstract

With the need for recycling of used materials growing steadily in order to save valuable resources of our planet, and given also that the rate at which recyclable materials reach the recycling factories and consequently the flow at which they fall on the conveyor belts in order to separate them is particularly large, the need for more efficient ways of separating and collecting these materials becomes apparent. In real life industrial set-ups, a large percentage of objects pass through the conveyor belts without being collected, at least not immediately. In addition, the main concern of a recycling factory is profit. Based on the above, the main focus of the present work is the optimization of the collection of the materials via the use of a robotic arm. The named optimization is based on the capabilities of the employed robotic arm and also on the market value of recyclable materials.

Our approach is separated into two interrelated parts, the prediction of the material of the objects we expect to pass through the belt and their collection. In the first part, the materials are classified into three classes (paper, plastic and aluminum) using only information from previous throws on the belt and the characteristics of the materials (color and size). For this part, we employ Hidden Markov Models that are capable of accomplishing the required prediction. In the second part, a Path Planner is implemented targeting the optimization of the materials' collection in terms of their cost. This is implemented via a Reinforcement Learning algorithm, specifically a Q-learning algorithm. Using a reward function the algorithm decides which is the next material to be collected. Finally, our approach is evaluated via simulated and real results, and its performance is also compared with that of a Proximity (Random) picker.



# Βελτιστοποίηση της συλλογής ανακυκλώσιμων υλικών σε ιμάντες

## Περίληψη

Με την ανάγκη για ανακύκλωση των υλικών που χρησιμοποιούμε να μεγαλώνει μέρα με τη μέρα με σκοπό να αποφορτιστεί ο πλανήτης και να σωθούν πολύτιμοι πόροι και έχοντας σαν δεδομένο πως ο ρυθμός με τον οποίο τα ανακυκλώσιμα υλικά φτάνουν στα εργοστάσια ανακύκλωσης και κατέπλεταση η ροή με την οποία πέφτουν στους ιμάντες με σκοπό το διαχωρισμό τους είναι πολύ μεγάλη, γίνεται φανερή η ανάγκη για πιο αποτελεσματικούς τρόπους διαχωρισμού και συλλογής των υλικών. Σε πραγματικές συνθήκες μεγάλο ποσοστό των αντικειμένων περνά από τον ιμάντα χωρίς να συλλεχθεί, όχι άμεσα τουλάχιστον. Επιπλέον, κύριο μέλημα ενός εργοστασίου ανακύκλωσης είναι το κέρδος. Με κύριο γνώμονα τα παραπάνω, ο βασικός άξονας της παρούσας εργασίας είναι η βελτιστοποίηση της συλλογής των αντικειμένων μέσω ενός ρομποτικού βραχίονα. Η παραπάνω βελτιστοποίηση πραγματοποιείται με βάση τις δυνατότητες του ρομποτικού βραχίονα που κάνει το διαχωρισμό, αλλά και την αξία των ανακυκλώσιμων υλικών.

Η προσέγγιση μας χωρίζεται σε δύο επιμέρους αλληλένδετα κομμάτια, την πρόβλεψη του είδους των αντικειμένων που αναμένουμε να περάσουν από τον ιμάντα και την αυτή καθαυτή συλλογή τους. Στο πρώτο κομμάτι γίνεται ουσιαστικά μια προβλεπτική ταξινόμηση των υλικών σε τρεις κλάσεις (χαρτόνι, πλαστικό και αλουμίνιο) μόνο με τη χρήση πληροφοριών από τις προηγούμενες ρίψεις στον ιμάντα και κάποιων χαρακτηριστικών των υλικών (χρώμα και μέγεθος). Αυτό το κομμάτι αναπτύχθηκε με την υλοποίηση Hidden Markov Models για τη πρόβλεψη. Στη συνέχεια υλοποιήθηκε ένας Path Planner με στόχο τη βελτιστοποίηση της συλλογής ως προς την αξία των υλικών που συλλέγονται. Σε αυτό το κομμάτι υλοποιήσαμε έναν Reinforcement Learning αλγόριθμο και πιο συγκεκριμένα Q-learning. Μέσω μιας reward function ο αλγόριθμος παίρνει την απόφαση για το ποιο θα είναι το επόμενο υλικό που θα συλλεχθεί. Τέλος, για να γίνει αποτίμηση των αποτελεσμάτων μας, έγινε μια σύγκριση της υλοποίησής μας με έναν Proximity (Random) picker.





## Acknowledgments

I would like to thank all those who contributed to the completion of this work. First of all, I would like to thank my thesis supervisor, Professor Panos Trahanias, for his support and guidance throughout the preparation of my master's thesis. In addition, I'm also greatly indebted to the members of my thesis committee, Prof. Antonis Argyros and Dr. Dimitris Tsakiris. Their views on my work and thesis document contributed to the improvement of the final result.

I would also like to thank the postdoc researcher at CVRL, Dr. Markos Sigalas, who supervised the day-to-day progress of this work, for his help and continuous support. Also, the graduate student at CVRL, Freddy Raftopoulos, who assisted me with the conduction of many of the real experiments of my thesis. Moreover, I would like to thank many of the current or former members of CVRL (specifically, Prof. Maria Pateraki, Dr. Michalis Maniadakis, Manos Pappadakis, Dr. Marianna Koskinopoulou, Dr. Stelios Piperakis and Nikos Tavoulari) for the excellent cooperation and friendly atmosphere we had during my tenure at CVRL. Of course I owe a special thanks to the Institute of Computer Science, Foundation for Research and Technology - Hellas (ICS-FORTH) for providing the resources and financial support to pursue my graduate thesis.

Special thanks to my parents Telis and Sofia, without their support nothing of what I have achieved would be possible; also to my brother Fotis who always helped me with a scientific discussion. Closing, I would like to thank my friends (Ageliki, Nelly, Taso, Angelo, Dimitra T., Nestora, Costa S., Giannis A., Antoni, Nikona, Stella, Sofia, Dimitra L.) who helped me, supported me and gave me some wonderful years in Heraklion, as well as all my friends from Athens and Patra who were with me all these years and made me the person I am today.



# Contents

<b>Table of Contents</b>	<b>i</b>
<b>List of Tables</b>	<b>iii</b>
<b>List of Figures</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.1.1 Recycling . . . . .	1
1.2 Problem Formulation . . . . .	4
1.3 Proposed Approach . . . . .	4
1.4 Related Work . . . . .	6
1.5 Thesis contribution . . . . .	8
1.6 Thesis structure . . . . .	9
<b>2 Employed tools and Theory</b>	<b>11</b>
2.1 Background Infrastructure . . . . .	11
2.1.1 The ANASA project . . . . .	11
2.1.2 Delta Robots . . . . .	13
2.1.3 Conveyor Belts . . . . .	14
2.1.4 Vacuum Gripper . . . . .	15
2.2 Mathematical tools and methods . . . . .	15
2.2.1 Hidden Markov Models . . . . .	15
2.2.1.1 Terminology and Equations . . . . .	16
2.2.2 K-Means Clustering . . . . .	21
2.2.3 Reinforcement Learning and Q-Learning . . . . .	22
2.2.4 Global Path Planner . . . . .	25
2.2.4.1 Path Planning . . . . .	26
2.2.5 Image Preprocessing . . . . .	27
2.2.6 Simulation and Dataset Generation . . . . .	28
<b>3 Methodology</b>	<b>31</b>
3.1 Framework . . . . .	31
3.2 HMM Implementation . . . . .	32

3.2.1	HMM Design . . . . .	32
3.2.2	HMM Initialization . . . . .	34
3.2.3	HMM Training . . . . .	35
3.3	Processing of HMMs Predictions . . . . .	36
3.4	Decision Manager . . . . .	37
3.4.1	Q Calculation . . . . .	37
3.4.2	Final Path . . . . .	40
<b>4</b>	<b>Results</b>	<b>43</b>
4.1	Qualitative Evaluation . . . . .	43
4.1.1	HMM Assessment . . . . .	43
4.1.2	Predictions Processing . . . . .	44
4.1.3	Objective Function Evaluation . . . . .	46
4.1.4	Decision Making Evaluation . . . . .	47
4.2	Comparative Evaluation . . . . .	49
<b>5</b>	<b>Discussion</b>	<b>55</b>
5.1	Our Contribution . . . . .	55
5.2	Future Work . . . . .	56
5.3	Other Applications of the Framework . . . . .	56
5.4	Epilogue . . . . .	56
	<b>Bibliography</b>	<b>57</b>
	<b>Annex: Code</b>	<b>59</b>

# List of Tables

2.1	HMM Components . . . . .	17
3.1	Classes . . . . .	33
3.2	Observation Coding . . . . .	33
3.3	Value of each class in euros . . . . .	38
4.1	Analysis of HMM's performance. . . . .	44
4.2	Pick and place accuracy. . . . .	48
4.3	Extreme Value cases . . . . .	48
4.4	Long-term accuracy. . . . .	49
4.5	Performance analysis at conveyor belt speed x1. . . . .	50
4.6	Performance analysis at conveyor belt speed x1.2. . . . .	50



# List of Figures

1.1	Total projected waste generation [1]. . . . .	2
1.2	Global waste treatment and disposal percentage [1]. . . . .	3
1.3	Conveyor Belt's separation into stripes. . . . .	5
1.4	Flow Diagram of the proposed approach. . . . .	5
2.1	ANASA Robotic Sorter[2] . . . . .	12
2.2	ABB IRB 360 FlexPicker . . . . .	12
2.3	Parallel robot configuration . . . . .	13
2.4	Typical examples of conveyor belts . . . . .	14
2.5	Vacuum Gripper . . . . .	15
2.6	Hidden Markov Model . . . . .	17
2.7	Typical examples of conveyor belts . . . . .	22
2.8	Reinforcement Learning . . . . .	23
2.9	Subdivision in RL . . . . .	23
2.10	Markov Process . . . . .	24
2.11	Block diagram: Q-learning . . . . .	25
2.12	Path Planning [3] . . . . .	26
2.13	Image pre-processing steps. (a) Original, distorted, image. (b) Undis- torted image. (c) Overlaid grid on undistorted image. . . . .	27
2.14	Conveyor Visualization . . . . .	28
3.1	Flowchart . . . . .	32
3.2	HMMs Framework . . . . .	35
3.3	Conveyor Representation for our implementation . . . . .	36
3.4	Q-learning: An off-policy TD control algorithm [4] . . . . .	38
3.5	Basic steps for $q$ calculation [5] . . . . .	39
3.6	Flowchart for $q$ calculation . . . . .	40
4.1	Clusters of each class for a part of the conveyor and the centroids cor- responding to an item of each class . . . . .	45
4.2	Accuracy and Profit comparison for the three objective functions . . . . .	46
4.3	Different path plans depending on the reward function. . . . .	47
4.4	Accuracy comparison of the two methods at speeds x1 (a) and x1.2 (b) . . . . .	51
4.5	Time comparison of the two methods at speeds x1 (a) and x1.2 (b) . . . . .	52

4.6	Time comparison of the two methods at speeds $x1$ (a) and $x1.2$ (b) . . .	52
4.7	Average accuracy of the two methods for class 2 in different dataset sizes	53
4.8	Average accuracy of the two methods for class 3 in different dataset sizes	53
4.9	Average accuracy of the two methods for class 4 in different dataset sizes	54



# Chapter 1

## Introduction

Arm conveyors and Automated Storage and Retrieval Systems (AS/RS), in the form of an endless belt or chain to which are attached projecting arms or shelves which carry the materials [6], are used for a variety of applications, ranging from chemical and food processing to mining or medical applications to state a few. One area, where arm conveyor systems are critical to the success of the industry, is that of material recycling, referring either to end-of-life products, post-consumer waste and industrial excess, or otherwise collected materials for reuse. This thesis investigates the improvement of the performance of arm conveyor recycling systems, through modeling of the materials' flow and optimization of the material collection procedure.

### 1.1 Motivation

In many industrialized nations, material consumption takes place at an unsustainable rate with multiple negative effects for both the environment -e.g. pollution or resources depletion- and the economy -e.g. waste management or material and product price fluctuations. Recycling is an effective means of tackling with the negative impacts of this over-consumption, by increasing the product's and material's re-usability and, thus, reducing the excessive needs (and cost) of manufacturing systems. However, and despite the constantly growing societal interest and demand, material recycling is far from being considered as "common practice", both in the EU and worldwide, requiring costly and, at times, inefficient equipment and facilities. This creates the necessity for more efficient, optimized, recycling technologies, that will enhance the performance of recycling installations, reduce recycling costs and, thus, make recycling a competitive alternative against the usage of raw materials, bringing several societal, financial and environmental benefits.

#### 1.1.1 Recycling

The management of commercial and household waste is a highly challenging issue of critical importance for modern societies. Poorly managed waste contaminates the

world's oceans, clogs drains and causes flooding, facilitates disease transmission, increases respiratory problems through airborne particles from burning of waste and harms animals that consume it unknowingly. Humans have created a harmful environment, both for ourselves and for all the animals that inhabit our planet, which affects various economic activities -such as tourism or agriculture- and suppresses economic growth worldwide [1]. To better conceive the size of the problem and the negative impact it has on the natural resources and the environment, some of the most indicative facts and statistics are stated below:

- Plastic production between 1950 and 2015 was 7,8 billion tones.
- In 2015 only 20% of the plastic wastes was recycled.
- Americans use over 2 and a half million plastic bottles every thirty minutes, and most of them are simply thrown away rather than recycled.
- Over 60% of the trash that ends in dustbin could be recycled.
- Every week half a million trees must be cut down for Sunday newspapers only.
- Due to the fact that people aren't recycling as much as they should, the rainforests are actually be cut down by about 100 acres a minute [7].
- If we recycled all our paper 25.000.000 trees could be saved each year.
- For every tone of recycled glass 1,2 tones of raw material and 180-200 litres of petroleum are saved.
- Recycling an aluminium can help to save a great deal of energy, in fact, enough to run your home television for about three hours.

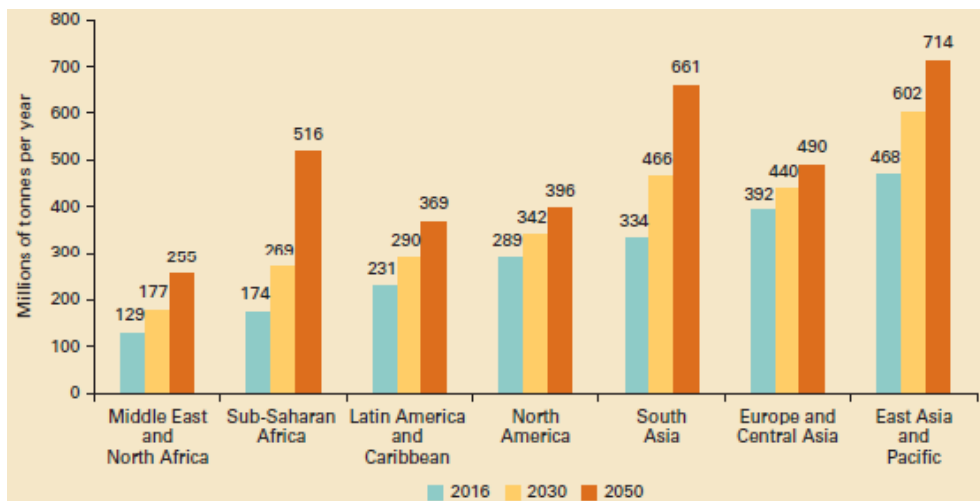


Figure 1.1: Total projected waste generation [1].

Considering the constantly increasing rate of waste generation, as illustrated in Figure 1.1, it becomes obvious that the situation is only going to be aggravated with unpredictable consequences for the environment. A possible way to reverse, or at least decelerate, this trend is by recycling old and used material to produce new items and equipment, and thus not only reducing the generated waste but also reducing the exploited natural resources.

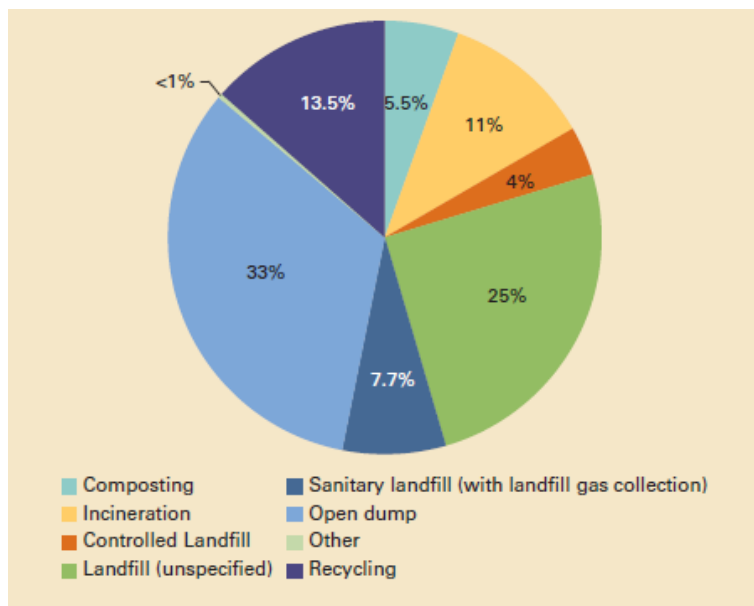


Figure 1.2: Global waste treatment and disposal percentage [1].

The benefits from adopting recycling are numerous with respect to both social and financial aspects as well for the environmental preservation, since it reduces the usage of natural resources, such as water, while also decreasing the necessity for mining and processing of raw materials. Moreover, the production based on recyclable materials saves energy, causes less air pollution and less greenhouse gases. Nevertheless, and despite its undisputed advantages, recycling represents only a small portion of worldwide waste treatment, i.e. a 13.5% as illustrated in Figure 1.2. This is mostly due to the cost of material recycling, which, at times, is higher to that of extracting new ones [8]. Thus, developing efficient and effective methods and technologies, will make recycling more approachable and, consequently, will reinforce our environmental education and conscience and will improve our living.

## 1.2 Problem Formulation

The most common procedure for recycling worldwide is firstly to collect all the recycling materials piled and mixed altogether somewhere. Subsequently, the materials are thrown as they are –mixed– on a conveyor belt where their separation takes place. The separation process is performed either manually by humans or is automated via robotic separators. It is obvious that the robotic sorting is much more effective, especially when we have to deal with huge masses of recyclable materials. At this point is where the use of an efficient separation system becomes vital. Separation systems perform a core function in material recycling. They facilitate the fast and precise sorting of the materials in their corresponding categories via fully automated processes.

The Computational Vision and Robotics Laboratory (CVRL) at FORTH-ICS participates in the ANASA project, which aims at developing, integrating and commercializing an autonomous robotic system for categorizing and separating recyclable materials. It comprises by a conveyor belt, where materials are unloaded in bulks, a visual system which identifies them and a Delta Robot-based Waste Separator, which collects and distributes them accordingly. Although existing work focuses on how to pick and place materials after identifying, on the spot, the type of the material with recognition and classification algorithms, we observed that predicting the materials' flow/sequence on the conveyor belt could possibly facilitate and enhance the robot's sorting process. The concept behind this is that past waste sequences could be used to predict the type of the succeeding materials that will fall on the conveyor belt, and thus provide for the optimum picking path for the Delta robot. This could enhance the performance of the robotic sorter by means of (i) volume of processed materials, (ii) value of collected materials and (iii) time of the sorting process.

In this thesis we aim at filling this necessity that emerges from the above. Using Hidden Markov Models (HMM), as a first step we predict the next sequence of materials on the conveyor belt based on the previous sequences and the materials' probabilities of appearance. Chapter 2 analyzes in detail the way these probabilities are formulated and the employed HMM's structure. Following that, our ultimate goal is to generate a sequence of steps (path) for the Delta Robot to follow, in order to collect the most valuable materials on the conveyor belt. This goal is achieved using a Reinforcement Learning technique, more specifically Q-Learning, by applying the results of the HMM to the Q-Learning Learning process. Overall, our framework can be considered as a Decision Manager that regards the robot's path.

## 1.3 Proposed Approach

Our approach consists of two parts. The main idea for the first part, is to use Hidden Markov Models in order to successfully predict the materials that will next pass through the conveyor belt. In practise, to accurately predict the materials via the HMMs use, we want to infer the spatial dependency of the materials' flow on the conveyor belt. To facilitate the preprocessing of the data and the implementation of the HMMs, the belt is

effectively divided into 5 virtual stripes (Figure 1.3). For each of the stripes, a separate HMM is employed in order to predict the materials that are expected (on the said stripe) given the material sequence of the previous throws. Each of the separate HMMs operates with its own priors, transition matrix and observation matrix ( $\pi$ ,  $A$  and  $B$ ), respectively. Accordingly, re-estimation of these matrices is performed separately. For the latter we also use information about the materials that appeared on the neighboring HMMs (practically the neighboring stripes) at  $t - 1$ . The input information for the HMMs (observed features of the materials and re-estimated matrices) is different at every moment  $t$ . The observed features are given to the HMMs as new input every second and the matrices, as mentioned above, are re-estimated every second based on the previous throws, which are now known. In this way time dependency is also part of the system.

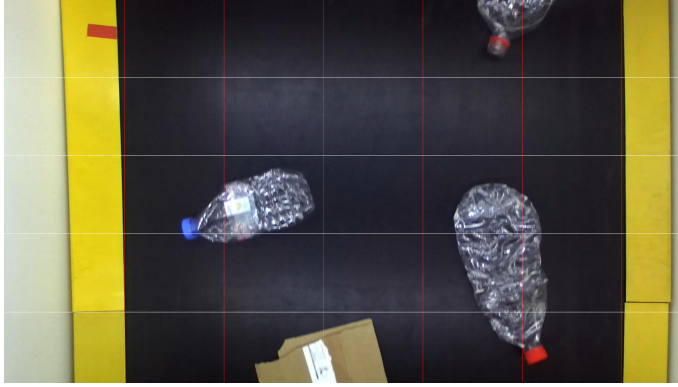


Figure 1.3: Conveyor Belt's separation into stripes.



Figure 1.4: Flow Diagram of the proposed approach.

Given the extracted material predictions, in the second part of our work we process the output matrix to acquire the position of each of the predicted materials on the conveyor belt. In order to do so, we use k-means clustering to acquire the individual items from each class and their centroids. Then, by taking into account the position of each material on the belt, as well as the cost of the materials and the cost of the Delta robot's movement, our system will make a decision about the most lucrative path for

the robot to follow.

Concluding in the final part of our framework, we import the labeled materials and their positions on the conveyor belt into a decision manager. More specifically, the employed method is a Reinforcement learning algorithm (Q-Learning). Considering the parts of the above described workflow, it is evident that a Global Path Planner for the Arm - Conveyor system is effectively formulated. The flow diagram of the entire process is illustrated in Figure 1.4.

## 1.4 Related Work

The literature review is separated in three main sections. The first section includes the literature regarding robotic sorters. In the second section we present works that focus on prediction techniques. Finally, in the third section we review Reinforcement Learning applications.

In the framework of the ANASA project extensive research has been made with respect to the pick and place process. In [9] the vacuum griper's grasp is optimized by adding a custom suction. Application of a new transfer paradigm for recyclable sorting is proposed in [10], by replacing the usual Pick-and-Place process with the much faster Pick-and-Toss process.

There are numerous works in the literature focusing in waste separation, investigating the problem from a variety of aspects. A review about the existing works in the field is made in [11], covering matters as the collection and logistics in recycling, machines and waste treatment plants, business models and data tools. They assess tools and methods already applied in waste management, as well as methods that could be applied successfully. Designing an algorithm that minimizes cycle duration with focus on the kinematics as well as optimizing the picking process strategy and the order of the picked object is the main goal in [12]. Another approach that uses deep learning technology to detect and locate solid waste is proposed in [13]. Also in this work the visual area on the conveyor belt is captured by the depth camera and the information of the geometric center coordinates and the angle of the long side of the target object are sent to the robot to complete the classification and grabbing of the solid waste. Another system for automatically sorting garbage based on machine vision is presented in [14], where a deep neural network model is used. Specifically, Region Proposal Network (RPN) and the VGG-16 [15] model for object recognition and pose estimation are optimized to the object detection.

In [16] the authors formulate an application in which objects of different types arriving on a moving conveyor belt are first automatically identified, then grasped and picked up by an industrial UR10 robot arm, and finally sorted into predisposed bins based on the object type. For this task they use Gilbreth software architecture which consists of multiple ROS (Robot Operating System) [17] nodes. As reported in [18], an efficient algorithm that finds collision-free paths for a manipulator, which solves the problem for four-degree-of-freedom pick-and-place operations by describing free space in two ways: as freeways for the hand and payload ensemble and as freeways for the

upper arm.

In the work reported in [19], an approach for a real time inspection and selection of objects in continuous flow is proposed. They use color sorting system solution with the application of image processing which senses the objects in an image captured in real-time by a webcam and then identifies color and information out of it and process them for pick-and-place mechanism. There are works in the literature that focus on image processing to facilitate the pick and place process. For example, in [20] the objects are detected with a feature extraction algorithm, then the extracted image (parameters in compliance with the classifier) is sent to the classifier to recognize what object it is and once this is finalized, the output would be the type of the object along with it's coordinates to be ready for the Robotic Arm to execute the pick and place task. Another approach is analyzed in [21], where first the polygon models of both the object and the environment are clustered, with each cluster being approximated by a planar region, so position and orientation of an object can be determined by selecting a pair of clusters: one from the object and the other from the environment.

Many different methods are used for prediction and classification of objects, materials or images. A common technique for such tasks regards Neural Networks. Convolutional neural network (CNN) trained with deep-learning algorithms have been widely applied in demanding computer vision applications [22]. In [23] the authors propose a recurrent Convolutional Neural Network (RCNN) for object recognition by incorporating recurrent connections into each convolutional layer, with static input, the activities of RCNN units evolve over time so that the activity of each unit is modulated by the activities of its neighboring units. Interestingly, CNNs have been also used to perform waste classification [24], being however applied in a relative small dataset with single-item images and without any type of occlusion. In [25], Support Vector Machines are employed. They propose a Gabor wavelets and support vector machine (SVM)-based framework for object recognition. In [26], the authors propose Hidden Markov Models for learning and classification of two-dimensional objects based on segmented grey-level images.

Some works in contemporary literature focus exclusively on waste classification. In [27] the goal is to develop a deep learning application which detects types of garbage into trash in order to provide recyclability with vision system. They use two different classifiers, specifically Softmax and Support Vector Machines in order to test performance of fine-tuned models, Alexnet, VGG16, Googlenet and Resnet. Using Support Vector Machines (SVM) with scale-invariant feature transform (SIFT) features and a convolutional neural network (CNN) they attempt to classify between six classes a recycling material in [28], with their experiments showing that the SVM performed better than the CNN.

Hidden Markov Models are also widely used to determine a robot's trajectory. For instance in [29], the parameters of an HMM are determined by historical data. Subsequently, using a Viterbi algorithm to find the double layers hidden states sequences corresponding to the current trajectory and finally propose a new algorithm for vehicle trajectory prediction based on the hidden Markov model of double layers hidden states, and predict the nearest neighbor unit of location information of the next  $k$

stages. Markov models are used also for a variety of prediction tasks. In [30] they use Markov Models in order to acquire a topological model of indoors environment by means of visual sensing and subsequent localization given the model. A new learning method based on HMM techniques estimations, to built a model for classification is presented in [31]. Their approach consists of evaluation of the probability to belonging in one group, given the observations by a linear classifier. The developed algorithm in this work is based on discrete states and discrete observations cases of HMM. The experimental results shown that the method has strong performance not only in supervised, but also in unsupervised problems. As hidden Markov models are graphical interpretable models they can be used as a first step towards 3D structure prediction is stated in [32]. It was proven that HMM achieves valuable prediction results using only a limited number of parameters. Which leads to an interpretable framework for protein secondary structure architecture. Furthermore, it can be used as a tool for generating protein sequences with a given secondary structure content.

Reinforcement learning use in robotics field is growing. To start with, the reward shaping problem (rewards that guide the learning system quickly to success) was investigated by [33]. In [34], they combine motor primitives with the theory of stochastic policy gradient learning, as a framework for reinforcement learning for humanoids. They state that in real-world domains, the shortcomings of the discounted formulation are often more critical than those of the average reward setting as stable behavior is often more important than a good transient. As was investigated by [35] there are numerous reinforcement learning algorithms for Markov Decision Processes with performance guarantees are known [36] [37], [38].

Q-Learning techniques specifically have also gained increasing popularity in path planning. As stated in [39] Q-Learning has the potential to reduce robot programming effort and increase the range of robot abilities. They introduce an algorithm that deals with continuous state and action variables without discretising. In [40] the concept of partially guided Q-learning is introduced and the flower pollination algorithm (FPA) is used to improve the initialization of Q-learning, so that the convergence of Q-learning is accelerated when Q-values are initialized appropriately using the FPA to optimize the robot's path selection. In [41] they introduce a reinforcement learning method for exploring a corridor environment with the depth information from an RGB-D sensor. Using this method the robot controller achieves obstacle avoidance ability by pre-training of feature maps using the depth information. Also in [42] they present a continuous-action Q-learning algorithm over the standard discrete-action version in terms of both asymptotic performance and speed of learning.

## 1.5 Thesis contribution

This work aims at optimizing the recycling process in numerous aspects. The main goal is to develop a fully automated recycling process using a robotic sorter. Such a procedure allows for precise and fast sorting of the materials, eliminates human errors, and gives rise to economic benefits due to the market value of the recycled materials.



To achieve these goals, we initially formulate Hidden Markov Models (HMMs) to facilitate the classification process. The employment of HMMs for unsupervised classification provides robust predictions regarding the materials on the conveyor belt. Accordingly, we end up with a prediction about the sequence of materials which will appear next on the belt, which provides enhanced information to the main classification method and helps avoiding wrong classifications. Moreover, we specify each material's centroid using k-means clustering, which is a simple and low complexity method. By means of this step we provide the sorter with the optimum spot for the gripper to pick an object.

After having established the above, our final step to complete the optimization is to create a decision maker which will at any moment determine the most profitable material to be picked. We achieve that by implementing a Reinforcement Learning algorithm, more precisely a Q-Learning algorithm. Such learning methods are widely used in the Robotics field for path planning with great success, which is also proved in our work.

Experimental results obtained with our implementation demonstrate that our method significantly outperforms existing approaches. Our method increased the picking accuracy of the robotic sorter by better planning the picking of objects, achieving to collect more items than previous methods, and finally selecting items of higher value. Experimentally, we also showed that our method scales smoothly to larger number of items of the conveyor belt. The latter constitutes a very desirable characteristic, especially in real world applications, where the management of huge amounts of items is required.

## 1.6 Thesis structure

In the next Chapters of this thesis we present in detail the tools and the developed methodological framework. We explain the formulation of the proposed framework and finally the experimental results are presented and discussed. More precisely, in Chapter 2 all the methods that were implemented in this work are thoroughly analyzed and the theoretical background is given. In Chapter 3 we explain the steps of the formulated framework, with details regarding our implementation. In Chapter 4 we illustrate the results of a qualitative analysis of our method and offer comparative results with an existing method. Finally, we reach our conclusions in Chapter 5 and propose further enhancements for the framework and other applications.



## Chapter 2

# Employed tools and Theory

In this chapter we describe the background knowledge and infrastructure that is used for the purposes of this thesis. As mentioned in the previous chapter, we initially employ a set of HMMs to infer the state of the system and predict the future material flow on the conveyor belt. Given the initial predictions a Q-Learning reinforcement learning procedure is then employed, formulating a Decision manager, in order to come up with a sequence of actions, which define the optimum path for the robotic separator. Employed equipment, techniques and methodologies, along with the data processing and generation procedures, are explained in detail below.

### 2.1 Background Infrastructure

This thesis is inspired by and capitalizes on the ongoing project ANASA[2], which takes place in the Computational Vision and Robotics Laboratory (CVRL) at FORTH-ICS and aims at optimizing the performance of conveyor belt-equipped robotic waste separator systems. For the project's purposes, CVRL is equipped with an arm conveyor robotic system, consisting mainly of a conveyor belt, a visual detection system and a robotic gripper, that is used for recyclable waste separation.

#### 2.1.1 The ANASA project

Main objective of the project is the development and optimization of an automated procedure for recyclable waste separation. The ANASA Robotic Waste Separator (RWS) has significant advantages over the existing ordinary recycling systems, i.e. high reliability in object recognition (material detection), short separation cycle (high speed), significantly low installation volume, low cost and ease of application to both old and new recycling industries. All the above aim to evolve and facilitate the recycling process.



Figure 2.1: ANASA Robotic Sorter[2]



Figure 2.2: ABB IRB 360 FlexPicker

Figure 2.1 shows the integrated mechanical system, installed in the robotics laboratory of CVRL, which is composed by three main parts: (i) the ABB IRB360 DELTA robot figure 2.2, (ii) an external vision camera system for object detection, (iii) a conveyor belt of 4.5x0.8m length and 135-277 mm/sec speed range, all jointly secured by a steel cage, allowing for high speed movements with low vibrations. To facilitate gripping, the DELTA robot is complemented by a vacuum gripper suction cup, designed and made in-house [9] for the needs of the project's application and a vacuum generator with pressurized air supply of 10 bars which provides a low pressure to the

suction cup.

As far as it concerns the visual system of the project a standard stereo full HD ZED camera is used. The camera is placed 148cm before the robot (in the direction recyclables are transferred towards the robot), at a height of 75cm above the conveyor belt. To ensure constant light conditions during data acquisition and system operation, the camera was placed inside a box with LED-lighting equipment installed. The camera field of view covers the entire width of the conveyor belt providing information about the shape and color of the transported waste. For the categorization of the recyclable materials the well-known open source network Mask R-CNN [43] is employed.

### 2.1.2 Delta Robots

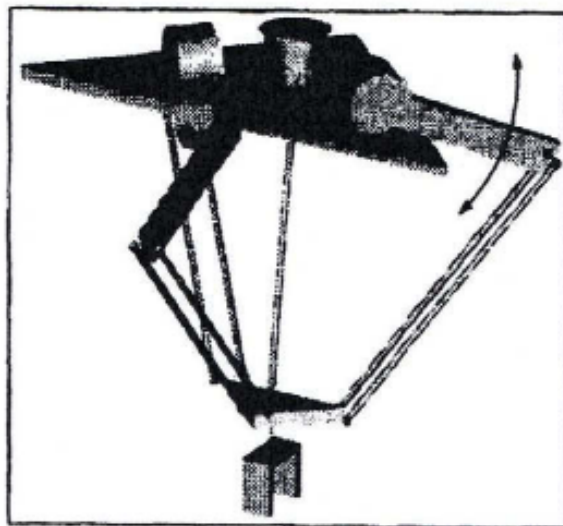


Figure 2.3: Parallel robot configuration

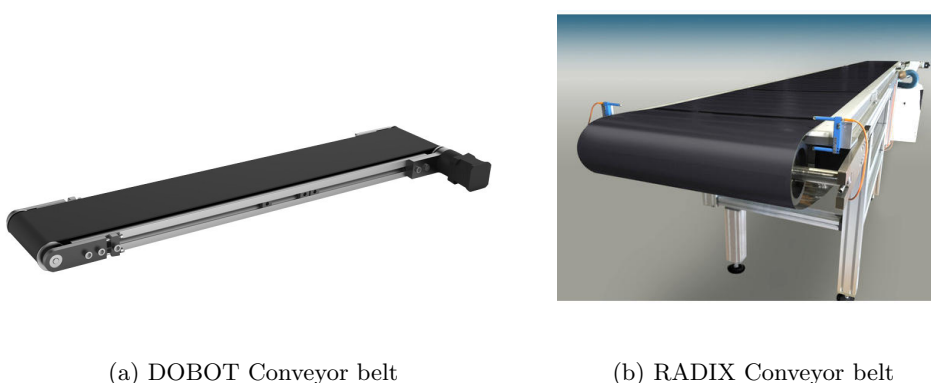
The parallel or Delta robot configuration, shown in figure 2.3, designed by Clavel[44], is where all the state of the art parallel robot configurations were based their development. It includes overhead mounted robotic arms with concurrent prismatic or rotary joints, controlled by the motors within the base.

Typically, these types of robots have four degrees of freedom, the fourth being fixed on the mobile platform and allowing the end-effector to rotate around the vertical axis. The moving platform always remains parallel to the base and is connected to it by three identical kinematic chains having a R-(RR)-(RR) architecture (where each RR denotes two parallel axis, the parenthesis indicates that the axes meet at one point) [45].

The parallel chains are actuated by the revolute joints, which are close to the base, using DC motors fixed to the base. The benefit of this approach is that it reduces the weight within the arms and therefore provides very high acceleration and speed capabilities. However they do have a low payload capacity. Typically they can

manipulate two pieces of 10g per second. These robots also have the advantage of having a relatively large workspace. Therefore, they are mainly used in pick-and-place applications, particularly on packing lines for the food industry, as well as assembly applications[46][47].

### 2.1.3 Conveyor Belts



(a) DOBOT Conveyor belt

(b) RADIX Conveyor belt

Figure 2.4: Typical examples of conveyor belts

Conveyor belt systems consist of two or more pulleys (a.k.a. drums) and an endless loop of carrying medium—the conveyor belt—rotates about them. To move the belt and the material that it carries, one or both pulleys are powered. The powered pulley is called “drive pulley”, while the unpowered one is known as “idler pulley”. Conveyor belts in general are handling materials. Depending on the materials they handle they are separated into respective categories, for example a conveyor belt used to move boxes inside a facility can not be used to transport large volumes of resources and agricultural materials. Conveyors are used for packing, transporting, sorting, industrial, manufacturing and agricultural processes, all of which are leading to diversities in shape and material of the belts.

Based on the proposed use, conveyor belts are manufactured using either PVC or rubber. The belt consists of one or more layers of material. Most belts in general material handling consist of two layers: the bottom layer is called carcass and provides linear strength and shape, while the top layer is called cover. Carcass is usually made of polyester, nylon and cotton while the cover is made of a variety of rubber or plastic compounds specified by the specific tasks where the belt will be used. Belts with regularly spaced partitions are known as elevator belts and typically are used to transport loose materials up steep inclines. Belt conveyors are also used in self-unloading bulk freighters and in live-bottom trucks. In figure 2.4 conveyor belts from two different manufactures are illustrated.

### 2.1.4 Vacuum Gripper

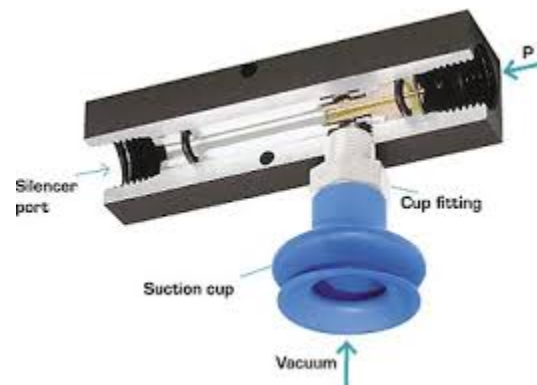


Figure 2.5: Vacuum Gripper

A vacuum gripper is a robot component that uses a suction cup connected to a vacuum source to lift and handle objects[6]. This type of grippers are most effective and will provide good handling if the objects are smooth, flat, and clean. Vacuum grippers work when the difference between atmospheric pressure and the vacuum, or negative pressure, is enough to provide the ability to lift, hold or move items and more. This is ideally achieved when one side of an item is large and flat enough for a vacuum gripper to create enough difference in pressure and achieve a firm grip. In non-ideal cases, which are the most common in real world (permeable, rough or uneven materials) high-flow vacuum generators are used. Generally, the vacuum cups (suction cups) have round shape and are manufactured by rubber or other elastic material in order to provide powerful and safe grips for collaborative robot applications with light, flat parts[48]. A characteristic example of a vacuum gripper, similar to the one employed on the ANASA waste sorter, is shown in figure 2.5.

## 2.2 Mathematical tools and methods

In this section we analyze all the tools employed in this work. We explain each of their definitions and state their mathematical background.

### 2.2.1 Hidden Markov Models

A Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov chain with unobservable ("hidden") states, i.e. a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous one. An HMM assumes that there is another, observable, process  $Y$  which can be used to infer the model's hidden states. A tangible example that can help us understand how HMMs function

is weather prediction. Imagine we want to predict tomorrow's weather, we are able to examine today's weather but we are not allowed to look at yesterday's weather [49]. In our implementation today's weather corresponds to the materials on the conveyor belt at the moment  $t - 1$ , yesterday's weather the materials already passed through the conveyor and we have indirect information about them through the probabilities they formulated. Finally tomorrow's weather corresponds to the materials we want to predict at moment  $t$ .

More formally, consider a sequence of state variables  $q_1, q_2, \dots, q_i$ . A Markov model embodies the Markov assumption on the probabilities of this sequence that states that when predicting the future, the past doesn't matter, only the present:

$$\text{MarkovAssumption} : P(q_i = a | q_1 \dots q_{i-1}) = P(q_i = a | q_{i-1}) \quad (2.1)$$

Additionally, the probability of an output observation  $o_i$  depends only on the state that produced the observation  $q_i$  and not on any other states or any other observations:

$$\text{OutputIndependence} : P(o_i | q_1 \dots q_i, \dots, q_T, o_1, \dots, o_i, \dots, o_T) = P(o_i | q_i) \quad (2.2)$$

To put it with respect to the task at hand, that of recyclable waste flow prediction, the problem formulates as attempting to infer/predict the type  $q_t$  (i.e. hidden state) of the material at time  $t$ , based on the current observations (i.e. features of the material such as color, size or shape) and the previous material type, i.e. state  $q_{t-1}$  at time  $t - 1$ .

### 2.2.1.1 Terminology and Equations

Typically, an HMM is specified by the following components:



$N$ :	number of hidden states, numbered $\{1, \dots, N\}$
$M$ :	number of output symbols, numbered $\{1, \dots, M\}$
$T$ :	length of the observation sequence
$Q$ :	distinct states of the Markov process, $q = (q_1, \dots, q_t, \dots, q_T)$ where $q_t \in \{1, \dots, N\}$
$V$ :	set of possible observations $\{0, 1, \dots, M\}$
$O$ :	observation sequence $o = (o_1, \dots, o_t, \dots, o_T)$ where $o_t \in \{1, \dots, M\}$
$A$ :	state transition matrix, $a_{ij} = P(q_{t+1} = j   q_t = i)$
$B$ :	per-state observation distributions, $b_i(k) = P(o_t = k   q_t = i)$
$\pi$ :	initial state distribution, $\pi_i = P(q_1 = i)$
$\lambda$ :	all numeric parameters defining the HMM considered together, $\lambda = (A, B, \pi)$
<i>indices</i> :	$i, j$ index states, $k$ indexes output symbols $t$ indexes time

Table 2.1: HMM Components

A generic hidden Markov model is illustrated in figure 2.6 , where  $X_i$  represents the hidden state sequence and all other notations are as given above. The Markov process, which is hidden behind the dashed line, is determined by the current state and the  $A$  matrix. We are only able to observe the  $O_i$ , which are related to the (hidden) states of the Markov process by the matrix  $B$ .

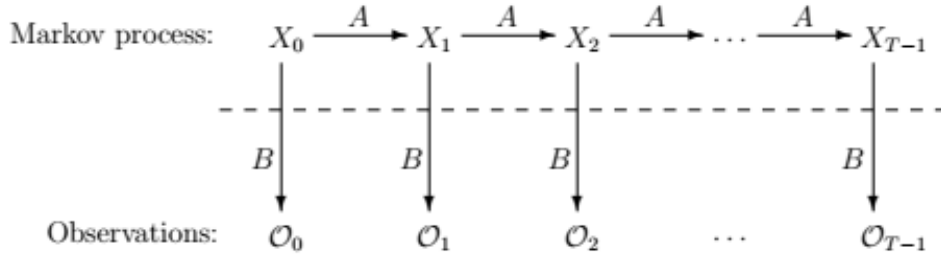


Figure 2.6: Hidden Markov Model

Note that matrix  $A = a_{ij}$  is a  $N \times N$  matrix, denoting the probability of the transition from one state to another (transition matrix), with

$$a_{ij} = P(Q(t+1) = j | Q(t) = i) \quad (2.3)$$

This is calculated for all the possible combinations of states.

Matrix  $B = b_j(k)$  is a  $N \times M$  matrix which correlates observations and states (emission matrix), with

$$b_j(k) = P(O(t) = k | Q(t) = q_j) \quad (2.4)$$

Matrices  $\pi$ ,  $A$  and  $B$  are row stochastic, meaning that each element is a probability and each row is a probability distribution (row elements sum to 1).

In order to have a better understanding of what  $a_{ij}$  and  $b_j$  represent, we can formulate the definitions as follows:

$$a_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected transitions from state } i} \quad (2.5)$$

$$b_j(O) = \frac{\text{expected counts of observing } O \text{ while at } j}{\text{expected counts of going through state } j} \quad (2.6)$$

It is easier now to explain what each of the notations represents in our work and how the Dataset is used to create matrices  $\pi$ ,  $A$  and  $B$  in Chapters 3 and 4. Beginning with matrices  $\pi$ ,  $A$  and  $B$  as initial information and an observation sequence  $O$  the HMM computes  $a_t$ ,  $b_t$  and re-estimates the priors (matrix  $\pi$ ) using  $O_t$  where  $t$  denotes the time. In this context, the initial computations needed are listed below.

Our goal is to find an optimal state sequence for the given observations  $O$  with respect to the given model  $\lambda$ . For that we need to determine a score for the sequence given the model, this score is denoted  $P(O|\lambda)$ .

To find  $P(O|\lambda)$ , the so-called forward algorithm, or  $\alpha$ -pass, is used. For  $t = 2, \dots, T$  and  $i = 1, \dots, N$ , we define:

$$\alpha_t(i) = P(O_1, O_2, \dots, O_t, x_t = q_i | \lambda) \quad (2.7)$$

Then  $\alpha_t(i)$  is the probability of the partial observation sequence up to time  $t$ , where the underlying Markov process is in state  $q_i$  at time  $t$ . The crucial insight here is that the  $\alpha_t(i)$  can be computed recursively as follows.

1. For  $i = 0, 1, \dots, N$  let

$$\alpha_1(i) = \pi_j b_j(O_1) \quad (2.8)$$

2. For  $t = 2, \dots, T$  and  $i = 1, \dots, N$  compute

$$\alpha_t(i) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(O_t) \quad (2.9)$$

3. From equation 2.7 it is clear that

$$P(O|\lambda) = \sum_{i=1}^N \alpha_{T-1}(i) \quad (2.10)$$

We define the backward algorithm, or  $\beta$  – *pass*. This is analogous to the  $\alpha$  – *pass* discussed above, except that it starts at the end and works back toward the beginning.

For  $t = 1, \dots, T - 1$  and  $i = 1, \dots, N$ , we define:

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \dots, O_{T-1}, x_t = q_i | \lambda) \quad (2.11)$$

Then the  $\beta_t(i)$  can be computed recursively (and efficiently) as follows.

1. For  $i = 0, 1, \dots, N$  let

$$\beta_T(i) = 1, \quad (2.12)$$

2. For  $t = 2, \dots, T$  and  $i = 1, \dots, N$  compute

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \quad (2.13)$$

Based on the above, the problem of waste type prediction can be formulated as following: Given an observation sequence  $O$  and the dimensions  $N$  and  $M$ , find the model  $\lambda = (A, B, \pi)$  that maximizes the probability of  $O$ . This can be viewed as training a model to best fit the observed data. Alternatively, we can view this as a (discrete) hill climb on the parameter space represented by  $A$ ,  $B$  and  $\pi$ .

The sizes of the matrices ( $N$  and  $M$ ) are fixed but the elements of  $A$ ,  $B$  and  $\pi$  are to be determined, subject to the row stochastic condition. The fact that we can efficiently re-estimate the model itself is one of the more amazing aspects of HMMs.

For  $t = 1, \dots, t - 1$  and  $i, j \in 1, \dots, N$ , we define "di-gammas" as:

$$\gamma_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \quad (2.14)$$

For  $t = 1, \dots, T - 1$  the  $\gamma_t(i)$  and  $\gamma_t(i, j)$  are related by:

$$\gamma_t(i) = \sum_{j=0}^{N-1} \gamma_t(i, j) \quad (2.15)$$

Given  $\gamma$  and di-gamma, the model  $\lambda = (A, B, \pi)$  can be re-estimated as follows:

1. For  $i = 1, \dots, N$  let

2. For  $i = 0, 1, \dots, N$  let

$$\pi_i = \gamma_0(i) \quad (2.16)$$

3. For  $i = 1, \dots, N$  and  $j = 1, \dots, N$  compute

$$a_{ij}(i) = \frac{\sum_{t=0}^{T-2} \gamma_t(i, j)}{\sum_{t=0}^{T-2} \gamma_t(i)} \quad (2.17)$$

4. For  $j = 1, \dots, N$  and  $k = 1, \dots, M$  compute

$$b_j(k) = \frac{\sum_{O_t=k, t \in \{1, \dots, T\}} \gamma_t(j)}{\sum_{t=0}^T \gamma_t(j)} \quad (2.18)$$

The numerator of the re-estimated  $a_{ij}$  can be seen to give the expected number of transitions from state  $q-i$  to state  $q-j$ , while the denominator is the expected number of transitions from  $q_i$  to any state. Then the ratio is the probability of transiting from state  $q_i$  to state  $q_j$ , which is the desired value of  $a_{ij}$ . The numerator of the re-estimated  $b_j(k)$  is the expected number of times the model is in state  $q_j$  with observation  $k$ , while the denominator is the expected number of times the model is in state  $q_j$ . The ratio is the probability of observing symbol  $k$ , given that the model is in state  $q_j$ , which is the desired value of  $b_j(k)$ . Re-estimation is an iterative process. First, we initialize  $\lambda = (A, B, \pi)$  with a best guess or, if no reasonable guess is available, we choose random values such that  $\pi_i \approx 1/N$  and  $a_{ij} \approx 1/N$  and  $b_j(k) \approx 1/M$ . It's critical that  $A, B$  and  $\pi$  be randomized, since exactly uniform values will result in a local maximum from which the model cannot climb. As always,  $\pi, A$  and  $B$  must be row stochastic.

The solution to our problem can be summarized as follows.

1. Initialize  $\lambda = (A, B, \pi)$ .
2. Compute  $\alpha_t(i), \beta_t(i), \gamma_t(i, j)$  and  $\gamma_t(i)$ .
3. Re-estimate the model  $\lambda = (A, B, \pi)$ .
4. If  $P(O|\lambda)$  increases, go to 2.

It might be desirable to stop if  $P(O|\lambda)$  does not increase by at least some predetermined threshold and/or to set a maximum number of iterations. In our work we set a maximum number of iterations.

This method thought, require computations involving products of probabilities. It is easy to see, for example, that  $\alpha_t(i)$  tends to 0 exponentially as  $T$  increases. Therefore, any attempt to implement the formulae as given above will inevitably result in underflow. The solution to this underflow problem is to scale the numbers. However, care must be taken to insure that, for example, the re-estimation formulae remains valid. First, consider the computation of  $\alpha_t(i)$ . The basic recurrence is:

$$\alpha_t(i) = \sum_{j=1}^N \alpha_{t-1}(j) a_{i_j} b_i(O_t) \quad (2.19)$$

It seems sensible to normalize each  $\alpha_t(i)$  by dividing by the sum (over  $j$ ) of  $\alpha_t(j)$ . However, we must verify that the re-estimation formulae hold.

For  $t = 0$ , let  $\tilde{\alpha}_0(i) = \alpha_0(i)$  for  $i = 1, \dots, N$ . Then let

$$c_0 = 1 / \sum_{j=1}^N \tilde{\alpha}_0(j) \quad (2.20)$$

and finally,  $\hat{\alpha}_0(i) = c_0 \tilde{\alpha}_0(i)$  for  $i = 1, \dots, N$ . Then for each  $t = 1, 2, \dots, T - 1$  do the following.

1. For  $i = 1, \dots, N$  compute

$$\tilde{\alpha}_t(i) = \sum_{j=1}^N \alpha_{t-1}(j) a_{i_j} b_i(O_t) \quad (2.21)$$

2. Let

$$c_t(i) = \frac{1}{\sum_{j=1}^N \tilde{\alpha}_t(j)} \quad (2.22)$$

3. For  $i = 1, \dots, N$  compute

$$\hat{\alpha}_t(i) = c_t \tilde{\alpha}_t(i) \quad (2.23)$$

The same scale factor is used for  $\beta_t(i)$  as was used for  $\alpha_t(i)$ , namely  $c_t$ , so that we have  $\hat{\beta}_t(i) = c_t \beta_t(i)$ . We then compute  $\gamma_t(i, j)$  and  $\gamma_t(i)$  using the formulae show in equations (1.14) and (1.15) with  $\hat{\alpha}_t(i)$  and  $\hat{\beta}_t(i)$  in place of  $\alpha_t(i)$  and  $\beta_t(i)$ , respectively. The resulting gammas and "di-gammas" are then used to re-estimate  $\pi, A$  and  $B$ .

### 2.2.2 K-Means Clustering

K-means clustering is a popular unsupervised machine learning algorithm. A cluster refers to a collection of data points aggregated together because of certain similarities. So, what we give as input in such models is a set of observations or points (representing positions in our case) on a 2D space and we want to unravel the relation between them in teams. The clustering methods create these teams for us. The number of those teams (clusters) is denoted by the  $K$ . We define that target number  $k$ , which also coincides with the number of centroids we need in the dataset. In figure 2.7(a),(b) we can see what is explained above and how clustering works by an indicative example.

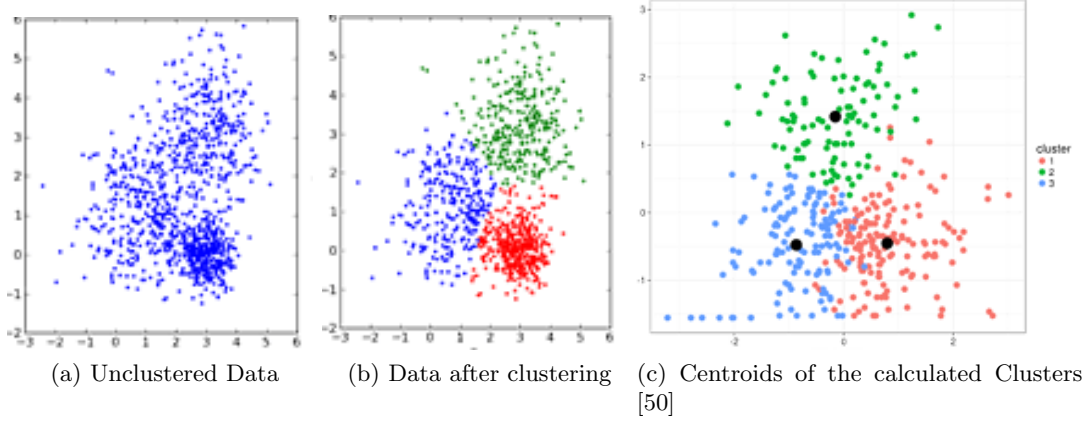


Figure 2.7: Typical examples of conveyor belts

A centroid is the imaginary or real location representing the center of each cluster. Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. In other words, the  $K$ -means algorithm identifies  $k$  number of clusters, and then allocates every data point to the nearest cluster, while keeping the clusters as small as possible. The ‘means’ in the  $K$ -means refers to averaging of the data, which in this case is finding the centroid. In our work, each cluster refers to an item and each centroid to the corresponding item’s center. In figure 2.7(c) we see the calculated centroids for each of the clusters found from the algorithm on the indicative data we show above in figure 2.7(b).

### 2.2.3 Reinforcement Learning and Q-Learning

Reinforcement learning (RL) [4] is a principled mathematical framework for experience-driven, goal-directed learning and decision making. RL starts with interaction between agent and environment, agent taking actions in the environment, which drives it to next state and receives a rewards based on the goodness of that action. As shown in figure 2.8 , at any time-step  $t$ , agent being in state  $s_t$  takes action  $a_t$  in its environment, gets reward  $r_t$  and observes next state  $s_{t+1}$ . This notion is suitable for our problem and how we have structured it so far. The time, states and actions are coherent with our notions. Reinforcement learning algorithms, push up the probabilities of taking good actions to achieve desired goals. In this work we solve a path problem using Q-learning technique.

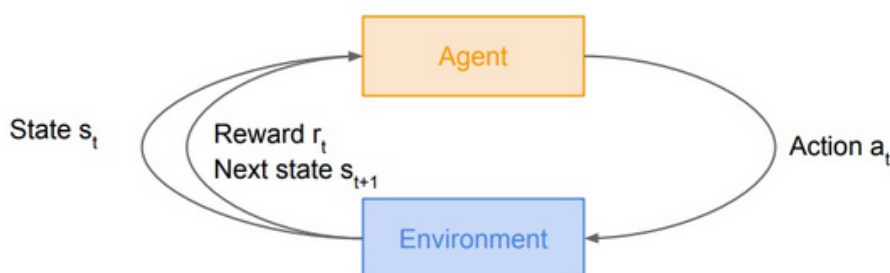


Figure 2.8: Reinforcement Learning

Reinforcement learning can generally be sub-divided into model-free & model-based as shown in figure 2.9. In model-based RL dynamical model of the environment is used and in model-free RL, a policy or value function is learnt.

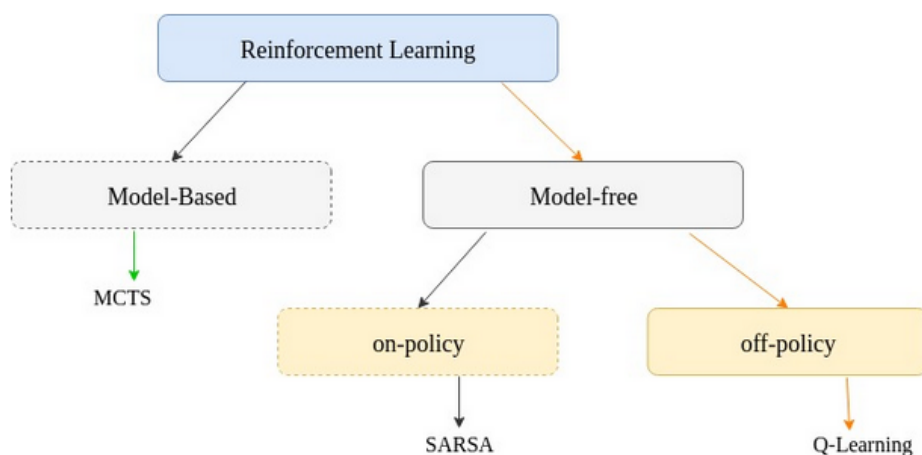


Figure 2.9: Subdivision in RL

Model-free RL is divided into two broad categories, off-policy and on-policy learning. In off-policy methods, which we use in this work, the policy used to generate behaviour, called the behaviour policy, may be unrelated to the policy that is evaluated and improved, called the estimation policy. An advantage of this separation is that the estimation policy may be deterministic (e.g. greedy), while the behaviour policy can continue to sample all possible actions. Our agent could even behave randomly and despite this, off-policy methods can still find the optimal policy. Mainly,

off-policy methods gather information from (partially) random moves, evaluate states as if a greedy policy was used and finally, slowly reduce randomness.

Q-learning as shown in figure 2.9, a model-free, off-policy learning proposed by [51]. Specifically, Q-learning can be used to find an optimal action-selection policy for any given (finite) Markov decision process (MDP), finds an optimal policy in the sense of maximizing the expected value of the total reward over any and all successive steps, starting from the current state. It works by identifying and learning an action-selection policy for any MPD given infinite exploration time and a partly-random policy.

At this point to note that, by the term Markov decision process we refer to a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker. More precisely, a Markov Decision Process is a discrete time stochastic control process. At each time step, the process is in some state  $s$ , and the decision maker may choose any action  $a$  that is available in state  $s$ . The process responds at the next time step by randomly moving into a new state  $s'$ , and giving the decision maker a corresponding reward  $R_a(s, s')$ . Markov decision processes are an extension of Markov chains, the difference is the addition of actions (allowing choice) and rewards (giving motivation). A Markov process diagram example is show in figure 2.10. Markov processes are explained in detail above in this Chapter without the reward part which is now added in our work.

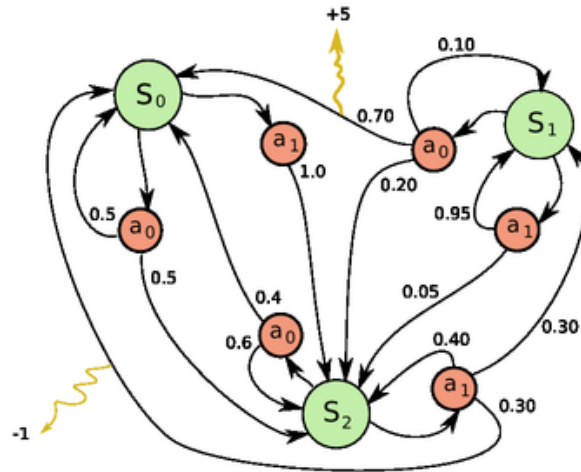


Figure 2.10: Markov Process

To continue with the Q-Learning analysis, "Q" names the function that the algorithm computes with the maximum expected rewards for an action taken in a given state figure 2.11. Ultimately this process gives the expected utility of taking a given action in a given state and following the optimal policy thereafter.



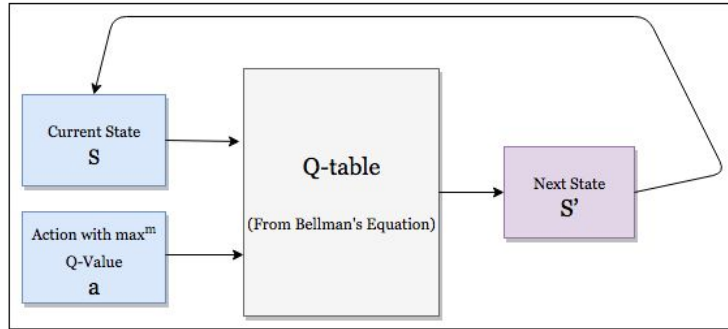


Figure 2.11: Block diagram: Q-learning

Q-value at state  $s$  and action  $a$ , defines the expected cumulative reward from taking action  $a$  in state  $s$  and then following the policy. In other words, Q-Learning is off-policy learning as one can choose the best action just by looking at  $Q(s', a')$ , without worrying about what happens next. In each iteration, Q-values of current state  $s$  are updated using the Q-value of next state  $s'$  and the greedy action  $a'$  using Bellman Optimality Equation which is given below.

$$Q^*(s, a) = [r + \gamma \max_{a'} Q^*(s', a') | s, a] \quad (2.24)$$

Bellman Equation for Q-value Function serves as a target (greedy) policy.

#### 2.2.4 Global Path Planner

By the term Global Path Planner we mean a group of navigation algorithms for planning an optimal path that connects a point of origin to a given goal in a known environment. Considering everything explained in this Chapter, we know can better understand why the group of algorithms we use in this work consists a Global Path Planner.

Global Path Planners are broadly used in Robotics field in many cases, from Robotic Arms to Humanoid Robots to acquire the robot's trajectory for a specific task. In our work the environment is unknown at the beginning of our process. We only have knowledge of our workspace dimensions (conveyor belt) and some features. Using the HMMs, as described in detail, we acquire complete knowledge for our environment and based on that we plan, via a Q-Learning algorithm, the Delta robot's trajectory (our optimal path). Details about this planning are explained in detail in Chapter 3.

### 2.2.4.1 Path Planning

Path planning is a computational problem to find a sequence of valid configurations that moves the object from the source to destination. It can only be applied when a map of the environment is known. So, path-planning requires a map of the environment and the robot to be aware of its location with respect to the map.

In figure 2.12 we see a characteristic example of path planning. An agent in a well defined environment (map) and the goal position denoted by  $x$  on the left of each image. In the case we have in figure 2.12 an optimum path planning. In such cases the goal is to find the optimum path between the initial and goal point. If the space is divided into a grid of cells (cell size depends on the robot dimensions), the goal of this planning is to visit only the cells needed to reach the goal position once. This problem is also known as the traveling salesman problem. Once the optimum path is found the robot can systematically traverse the space and therefore be more time and energy efficient (Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?).

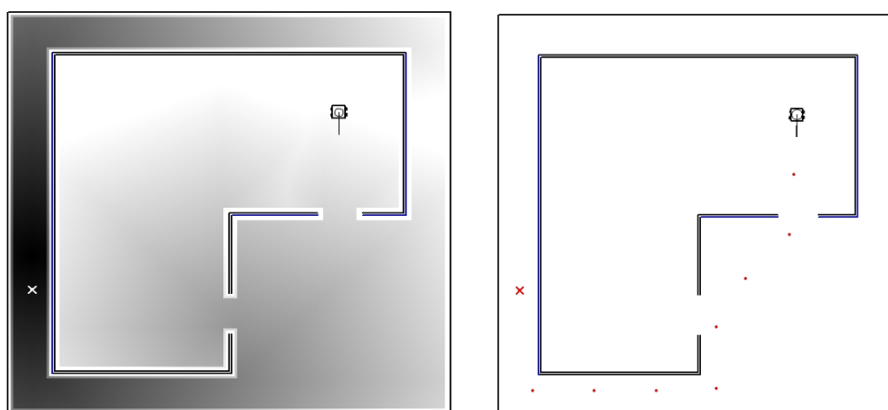


Figure 2.12: Path Planning [3]

In path planning problems we have the problem of complete coverage. In such cases the goal is to find the optimum path so the robot covers the entire space. In other words, the goal of optimum coverage is to visit every cell at least once [52].

Real-Time Path Planning is a term used in robotics that consists of motion planning methods that can adapt to real time changes in the environment. This includes everything from primitive algorithms that stop a robot when it approaches an obstacle to more complex algorithms that continuously take in information from the surroundings and create a plan to avoid obstacles.

Our path planning approaches an optimum and real-time path planning. Optimum

by the means of the reward in our case. So, the robot's goal is not to pass through the entire grid or only one cell, but from the cells that contain the most expensive materials and are arranged conveniently for the agent. Another difference is that there is not a specific point as a goal location in our notions. We want the robot to move based on materials on the belt. Also we study a static robot, which is not actually moving, but its end effector (the last link (or end) of the robot) does.

### 2.2.5 Image Preprocessing

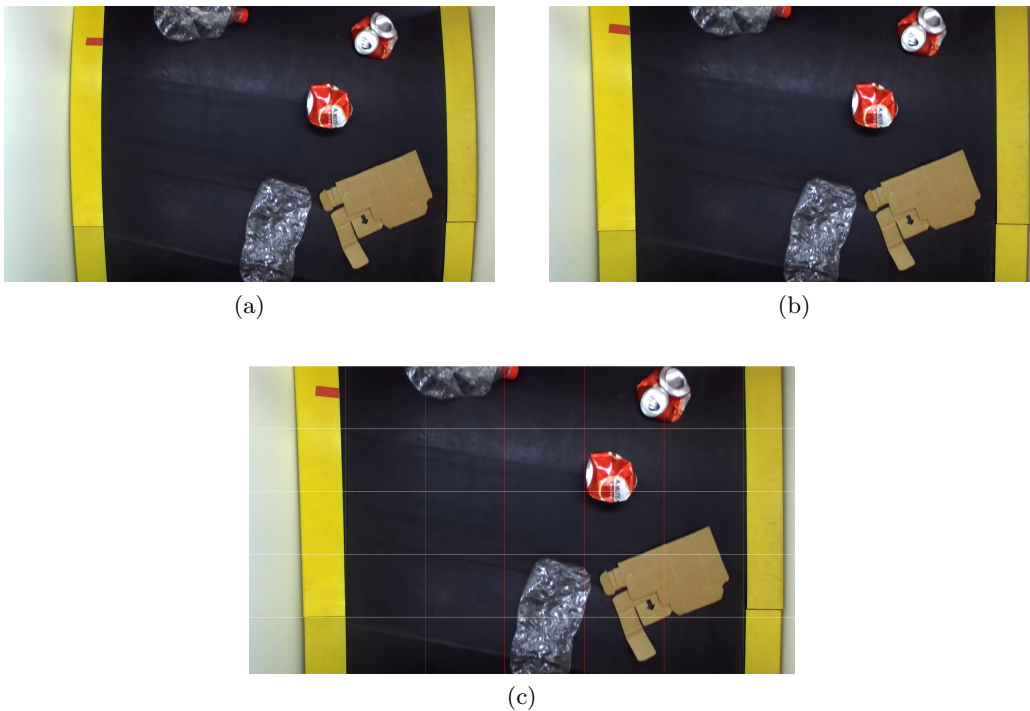


Figure 2.13: Image pre-processing steps. (a) Original, distorted, image. (b) Undistorted image. (c) Overlaid grid on undistorted image.

To acquire the datasets needed for this work, we initially recorded videos of recyclable materials passing through the conveyor belt in real time speed. Specifically, we recorded eight videos from which we created a training set and a test set. The training set consists of the first six videos and the test set from the remaining two. Both training and testing videos were separated in frames. Each of the videos lasted approximately one minute with frame rate 30 frames/second. Consequently, the videos were divided in frames, specifically we stored an image every 15 frames of the video.

Our first step for the preprocessing was to eliminate the camera distortion, as it appears in figure 2.13(a). We used the camera specifications to accurately eliminate the distortion from all the frames. The resulted image with no distortion is shown in figure 2.13(b). It becomes obvious by comparing the two images, that the elimination

of the camera distortion is necessary, as it could easily affect our analysis.

Last, we added a grid on our images as seen in figure 2.13(c). The grid assists us to visualize and understand the time and spatial aspect of our analysis. Namely, the vertical stripes divide the conveyor in five smaller stripes which helps the spatial analysis by importing the spatial dependency in our work (positioning of the materials on the belt). The horizontal lines, depict the time factor, meaning that each horizontal stripe represents a time snap of our materials' sequence, so time dependency is imported as well.

## 2.2.6 Simulation and Dataset Generation

In order to facilitate the performance analysis and to enhance the robustness of the model, we developed a simulator which produces simple representations of the materials' flow on a conveyor belt. Figure 2.14 illustrates a snapshot of the simulated conveyor belt, where the colored shapes represent the 3 material classes (AL: aluminium, PL: plastic, PAP: paper) and the  $x$  mark represents the vacant space. As will be discussed later, there is also a provision for the side bins, where each material should be placed after picking.

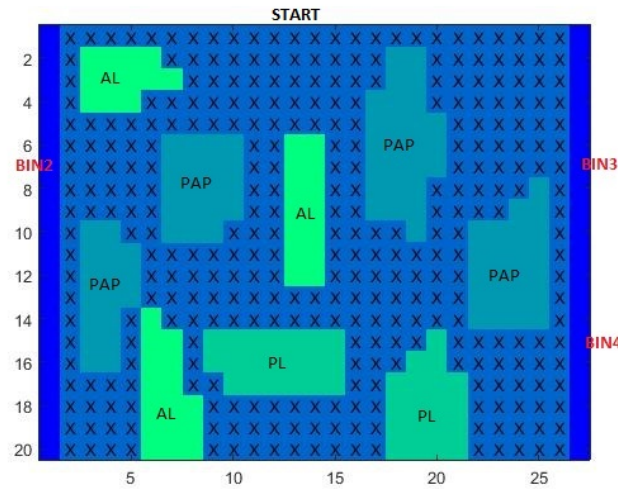


Figure 2.14: Conveyor Visualization

Capitalizing on the findings of the ANASA project, we trained a Neural Network to estimate the prior probabilities about the location -i.e. where is more likely a certain material to fall on the belt- and the features -i.e. type, size or shape of each of the materials. A simple random number generator, biased by the extracted probabilities, was then employed in order to simulate the flow of the materials on the conveyor belt.

Based on the developed simulator, we extracted a series of sequences of various complexities which were used for training and evaluation purposes. Specifically, we

created in total 100 waste flow simulations of varying conveyor belt speed, sequence duration, number of material classes and total number of materials.

The position of each material, in relation with the horizontal as well as the vertical lines of the grid, denotes in which of the five HMMs this material (more correctly its features) will be given as input.

A very important detail, at this point is that the observations i.e. size and possible colors of each class- used in the generated dataset, differ from the ones on the real dataset. Instead of having, for example only transparent plastic bottles, we inserted in the dataset via the observations, blue or green plastic items. In this way we assure our framework has efficient and unbiased performance under any circumstances.



## Chapter 3

# Methodology

As discussed in the previous chapters, our framework formulates a Decision Manager which consists of two main parts: prediction and selection (path planning). The first part (prediction) consists of five independent HMMs and the second part (selection) is a reinforcement learning algorithm. The HMMs predict the class of each material on the conveyor belt and the reinforcement learning algorithm (Q-Learning) selects which items are the most lucrative for the Delta robot to pick. In this chapter we go through the steps of our methodology, analyzing and explaining each step.

### 3.1 Framework

Firstly our data were exported from videos of the recyclable materials passing through the conveyor belt in a random way, which is different for each of the videos. The videos were acquired from real time experiments conducted in Computational Vision and Robotics Laboratory in ICS-FORTH using the conveyor belt and Delta Robot from ANASA project [2].

We used the data (features of the materials) exported from the videos as input to the HMMs in order to predict the class of each material. More specifically, the information acquired from the real dataset processing were used for the HMM's initialization and consequently, information from both real and generated dataset were used for the HMM's training. Afterwards, by processing these outputs we acquired information about the materials' positions on the conveyor belt, specifically their centroids. Finally, we gave the classes and positions of the materials as an input to the Q-Learning algorithm in order to ultimately get the optimal picking path for the Delta Robot, based on the values of the  $Q$  table. In figure 3.1 the framework is visualized for better understanding of connection of the processes mentioned above.

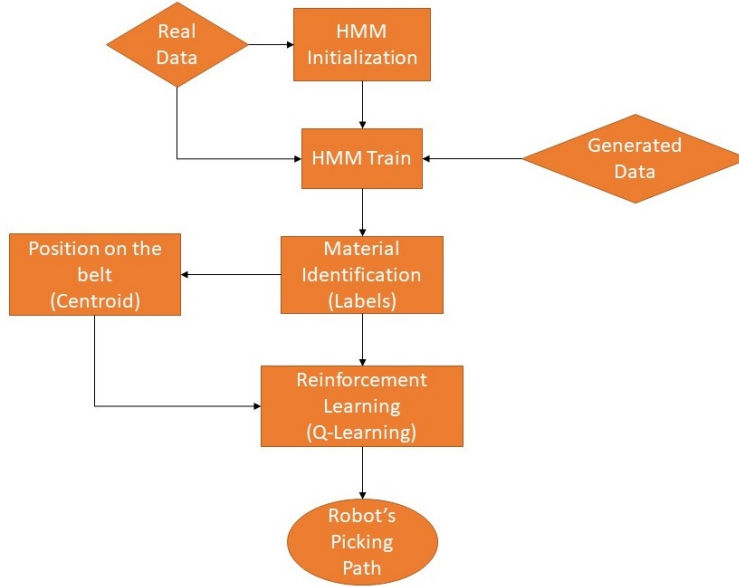


Figure 3.1: Flowchart

## 3.2 HMM Implementation

The choice of implementing Hidden Markov Models in our work is justified by the need of a model that can accurately predict the upcoming state of the conveyor belt, only by using knowledge of the exactly previous one. Applying HMMs allows us exactly that, adding the use of some basic information (transitions and observations) about our kind of data. Furthermore, it is an efficient, low cost and complexity model which enables real time calculations.

### 3.2.1 HMM Design

In all datasets we use three types (classes) of recyclable materials: paper, plastic and aluminum. We also added an extra "void" class which indicates that in this position of the belt we have no material present. The resulting four classes define the number of model states, i.e.  $N = 4$  as described in section 2.2.1.1. In the first column we have the material that appears on the conveyor belt and in the second column the number that each class is annotated with.

As explained in Chapter 1 Hidden Markov Models need as an input an observation sequence which is necessary for the predictions. Each observation indicates one or more of the classes we aim to be predicted by the HMM, meaning a material might share a feature with another material. In our work, these observations refer to the color of the materials (red, blue etc) and the space they occupy on the grid (Small,



Material	Class
Void	1
Paper	2
Plastic	3
Aluminum	4

Table 3.1: Classes

Observations (Features)	Corresponding Numbers
Small and Transparent	1
Small and Brown	2
Small and White	3
Small and Blue	4
Small and Red	5
Small and Green	6
Medium and Transparent	7
Medium and Brown	8
Medium and White	9
Medium and Blue	10
Medium and Red	11
Medium and Green	12
Large and Transparent	13
Large and Brown	14
Large and White	15
Large and Blue	16
Large and Red	17
Large and Green	18
None and Black	19

Table 3.2: Observation Coding

Medium, Large), meaning the occupied space in each of the cells that are created by the horizontal and vertical grid lines in figure 2.13(c), and are merged into a unified matrix which will be used as input to the HMMs. In that basis, an object is characterized small when it occupies a small part of the cell ( 1% to 25%), medium when it occupies over the half of the cell (up to 65%) and large when it almost occupies the whole cell. Each element of the resulting matrix contains a number, which corresponds to one of the observations table 3.2. Note that the conveyor itself is characterized by a specific unique observation. The horizontal position of each material denotes in which of the five HMMs this material belong.

In table 3.2 we have listed the observation as used in our work. On the first column we have all the possible combinations of the two separate observations (color and size). On the second column we have the numbers with which we have noted each of these observations throughout our work, i.e.  $M = 19$  as described in 2.2.1.1.

Normally we would also have each color and each size alone as observations, meaning having for example only medium or only green. But in our case, practically we will never acquire such information separately. So we excluded these observations as they will always have zero occurrence.

Regarding the output of the HMMs', it is in the form of five 1-D matrices of dimension  $X$ , with being the number of predicted classes of the material on the corresponding conveyor stripe. These matrices are afterwards put together and give us the full 2-D predictions matrix of dimension  $X \times 5$ , representing the class predictions for the whole conveyor belt.

### 3.2.2 HMM Initialization

As discussed, we used five independent HMMs, one for each of the stripes we split the conveyor belt into. As a first step, prior to that of the HMM training, we exploit the recorded video sequences to initialize the parameters of each HMM, namely the initial state distribution  $\pi$ , the transition matrix  $A$  and the observation matrix  $B$ . Starting with almost (with added noise) uniform distribution across the states to form the initial  $\pi$ , we "train" each HMM using the training set from the real laboratory recordings, by unifying the observations into a train-set matrix consisting of five columns, each column corresponding to a vertical stripe of the conveyor as shown in figure 2.13(c).

The priors calculation is an easy process, we count the appearances of each class in each column and we divide with the number of the total rows.

The transition matrix is, in our case a 4x4 matrix (because of the four classes we have). It gives as the probability of transition from the present state (class) to each of the other ones, and is formulated as:

$$a_{ij} = P(q_{t+1} = j | q_t = i) \quad (3.1)$$

To acquire the transition matrix  $A$  (as referred in section 2.2.1.1) we counted, for each column separately, the transitions from one state (class) to each of the remaining ones and we divided by total number of appearances of the corresponding material each time. The transition matrix is a row stochastic matrix (the elements of each row are summing to 1)

Finally, we calculated the observation matrix  $B$  (as referred in section 2.2.1.1). This is a  $N \times M$ , with  $N = 4$  representing the four states (classes) of the model and  $M = 19$  representing the observations (features of the materials) as stated in table 3.2. To export this matrix from our data we needed to match each of the materials appearing in our matrix with the observation that corresponds to them. To acquire this matrix we counted the appearances, in our initial matrix, of a material from a specific class with a specific feature (this was done for all the classes and all the observations) and

we divided these appearances by the total number of appearances of the corresponding material. The observation matrix is also a row stochastic matrix.

It is important to be said, that in case we include the "single" observation in our analysis (only small or only blue) the observation matrix would not be a row stochastic matrix. As items counted for one of our present observations would be counted a second time for their size alone and a third time for their color alone. In that case the row summing could not possibly be one as the overall number of materials would still be the same. It is not though mandatory from the HMM's statement for the observation matrix to be row stochastic. The observation matrix is an  $N \times M$  matrix, with  $N=4$ ,  $M=19$  and is formulated as:

$$b_j(k) = P(O(t) = k | q(t) = j) \quad (3.2)$$

### 3.2.3 HMM Training

Having an initial estimation about the HMMs' parameters, we proceed with the models' training using a mixture of both real and generated data. Datasets are shuffled and split into two sets, the training set which will be used to update the models and the test set that will be used to evaluate it. Similarly to the above, the input is given to each HMM in the form of 1-D matrix representing the sequence of material classes on the conveyor belt. To maintain the time dependency of material flow, each dataset is processed one row at a time.

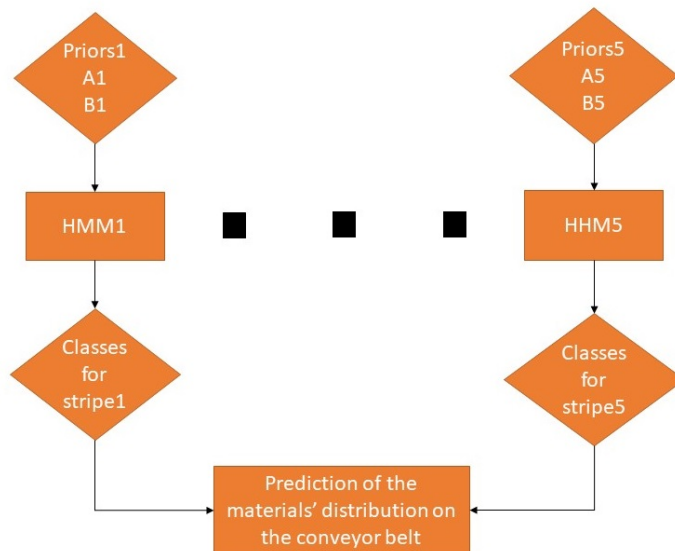


Figure 3.2: HMMs Framework

After having all the states predicted we unified the outputs in a matrix in order to extract relevant statistics and assess the models' performance.

For the implementation of the HMMs, a MATLAB version, similar to the pseudocode provided in [53], has been developed.

### 3.3 Processing of HMMs Predictions

Our next step is to process the matrix we acquired from the HMMs. For visualization purposes and calculation of the materials' centroids, which is explained in detail below, we turned the five columns of the prediction matrix into twenty five. Basically we "expanded" the size of the conveyor and at the same time we expanded the size of the recyclable materials, with respect to their volume (the volume of a can is different than the one of a plastic bottle) and their position on the conveyor belt. Namely, each of the five columns is now expanded in five more columns.

In this step we must note that we added two additional columns, one at the beginning of the matrix (column 1) and one at the end of the matrix (column 27), which indicate the positions of the bins the recyclable materials must be placed after they are picked by the Delta Robot.

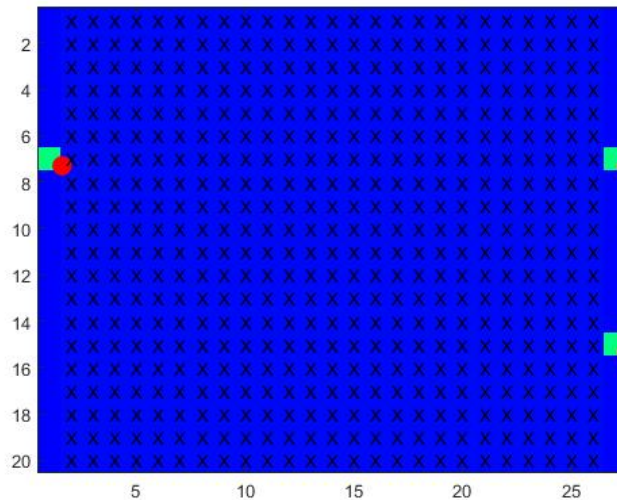


Figure 3.3: Conveyor Representation for our implementation

Being placed in one of the three bins, whose position is outside but next to the conveyor, is the target for each and every item and closes the cycle of the sorting process for each material. These two bin columns, have three bins virtually placed on them, one on the first column and two on the last. We have chosen their position based on the workspace of the Delta robot and they are indicated by a single cell, which represents the centroid of the corresponding bin. The bins' centroids are hardcoded

in our algorithm and they remain always the same. The elements of the matrix on these two columns are noted by the number -1 so that we avoid confusion with our classes. The conveyor's representation with the bins illustrated as a red dot is shown in figure 3.3. The "x" on the figure 3.3 denote the conveyor itself (class 1), meaning that each cell with an "x" is an empty position on the conveyor.

### 3.4 Decision Manager

In this step of our framework, our aim is to acquire the best picking path for our robot to follow. As an input to the Q-Learning algorithm we have the matrix we acquired from the HMMs and was processed to have 27 columns in total. To maintain the time dependency of our method, we split the matrix in batches of  $20 - by - 27$  and give them separately as input; i.e. we split it to smaller matrices of twenty rows each.

Following, we needed to acquire the exact position of every item. More specifically, we need to know each item's centroid, in order to be precise about the picking position and maximize the force of the vacuum's grip.

For this purpose we isolated in different matrices the members of each class. Namely we created three matrices (there was no need for a matrix for class 1 as it corresponds to the conveyor itself) with zeros in every element of the matrix except the ones that the class corresponding to each of the matrices appears. In this way we have a list of pointers for every item. Using these pointers as an input to a k-means clustering algorithm with the appropriate k (where k is the number of clusters expected and is different for every class) we have a list of centroids. Each centroid indicates the center of an item on the conveyor belt along with its class. The calculation of these centroids is done before the Q-Learning process itself begins, separately for each splitted matrix of twenty rows. The matrix with the centroids and the class corresponding to each centroid is used in the Q-Learning algorithm for two tasks. Firstly, to help in calculating the reward of each action of the Delta robot and secondly to specify the available picking points of the recyclable materials and their class.

#### 3.4.1 Q Calculation

In order to proceed to the calculation of  $q$  we have one final step, the reward calculation via an objective function. The reward function is an integral part of the Q-Learning formulation, as it is necessary for the calculation of the Q matrix which is the main criterion for the algorithm's selection of the final path. The first step is to assign a weight to each of the materials on the belt by using a combination of their value and the distance that the robot needs to cover for picking and placing. Specifically, the weight assign to each item is formulated as:

$$w_x = \frac{vx_i}{d_x} \quad (3.3)$$

where  $x$  denotes the specific identified item,  $i = 2, 3, 4$  is the identified material class,  $vx_i$  is the value of that item belonging to class  $i$  and  $d_x$  is the total distance required

for the robot to pick item  $x$  and place it in the respective bin. For simplicity purposes, we assigned indicative values to each class, as shown in table 3.3.

Class	2 (Paper)	3 (Plastic)	4 (Aluminum)
Value (euros)	0.22	0.25	0.27

Table 3.3: Value of each class in euros

Given that processing of the dataset in batches of size 20, a cumulative weight is calculated for each batch, taking into account both the selected and non-selected items. The sum of all batch weights provides with the value of our reward function, which is formulated as:

$$O_{total} = \sum_{b=1}^B \left( \sum_{x_b=1}^{X_b} \frac{vx_i}{d_x} - \sum_{y_b=1}^{Y_b} \frac{vy_j}{d_y} \right) \quad (3.4)$$

with  $b = 1, \dots, B$  denoting each of the  $B$  batches,  $x_b = 1, \dots, X_b$  denoting the selected items in batch  $b$  and  $y_b = 1, \dots, Y_b$  denoting the rest of the items in batch  $b$ .

During experimentation, i.e. real-time decision making, the reward is iteratively calculated and updated in each step of the algorithm, due to the continuous change of the robot's position which directly affects it. Robot's possible movements are selected based on their reward. The  $q$  is calculated every time for the selected action (each action corresponds to an item). The  $q$  matrix is randomly initialized and is iteratively updated. The logic of  $q$  calculation is shown in figure 3.4.

```

Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$ 
Repeat (for each episode):
  Initialize  $S$ 
  Repeat (for each step of episode):
    Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
    Take action  $A$ , observe  $R, S'$ 
     $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$ 
     $S \leftarrow S'$ ;
  until  $S$  is terminal

```

Figure 3.4: Q-learning: An off-policy TD control algorithm [4]

The size of the matrix is the same as the reward matrix, which is 20x27. Equation 3.5 is used for the calculation of  $q$ .

$$q(x) = w_x + \gamma * q_{max} \quad (3.5)$$

where  $\gamma$  is the discount factor and determines the importance of future rewards. A factor of 0 will make the robot "myopic" (or short-sighted) by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward.

The main steps for calculating  $q$  are shown in figure 3.5 below:

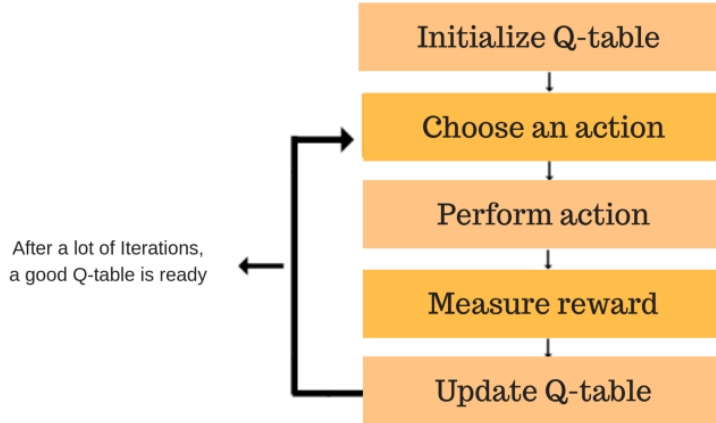
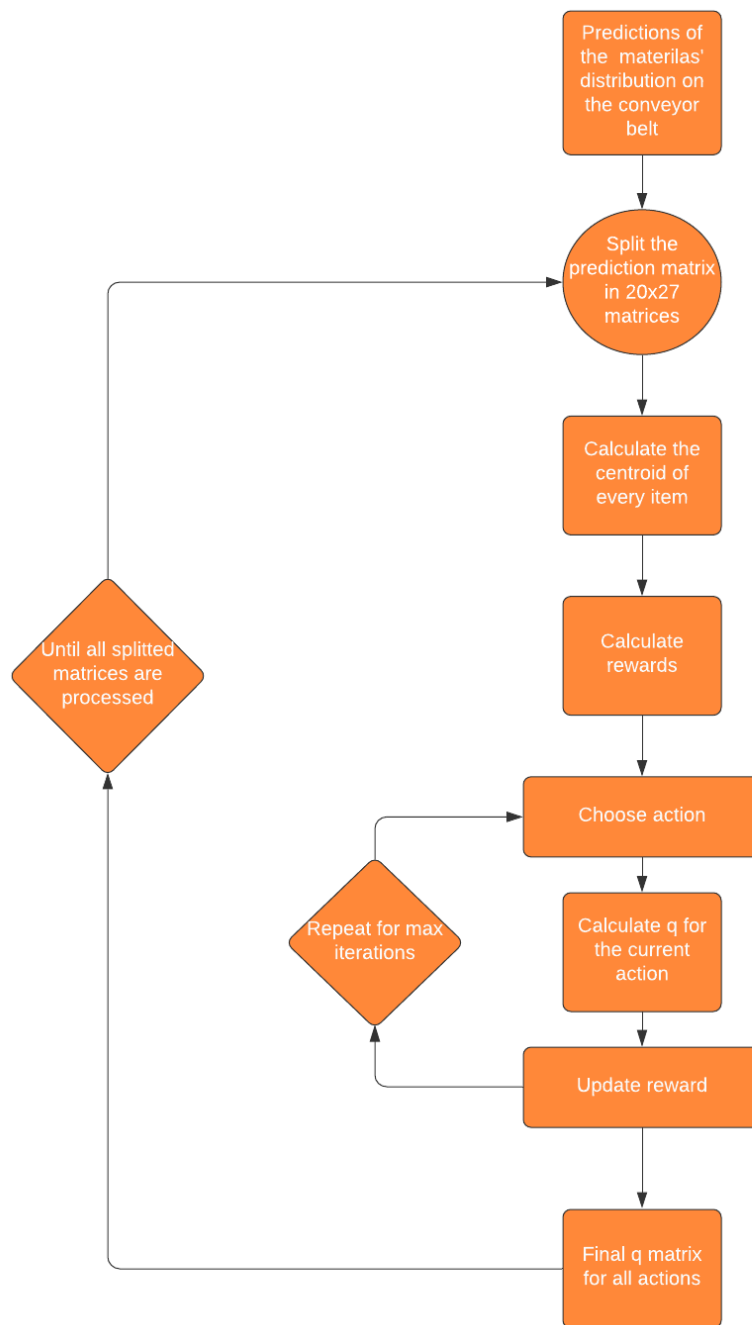


Figure 3.5: Basic steps for  $q$  calculation [5]

After  $q$  is estimated according to equation 3.5, the reward corresponding to this  $q$  becomes zero. This is needed because in our problem it is not accepted for the robot to pass through the same position twice, as it happens in regular path planning problems. In this problem once the material is picked it must disappear from our list of materials, as it is no longer on the conveyor but inside one of the bins. To continue with, after the picking action we automatically send the robot to one of the bins (to the one corresponding to the class of the selected material). So, the current position of the robot becomes the bin position.

This learning procedure stops when the sum of rewards of the remaining items is smaller than the reward of the most "precious" item, meaning that the remaining items on the belt are no longer worthy enough or not many enough. The procedure is repeated several times until it acquires the best  $q$  matrix possible or reaches the maximum iterations. In our case the maximum iterations are experimentally set to 50 while the gamma ( $\gamma$ ) used for the  $q$  calculation in equation 3.5 is 0.9. To note that the main idea of the implementation for the main framework of the Q-Learning code is from [54].

The complete flowchart for the Q-Learning part of our work is shown in figure 3.6. The final  $q$  matrix is a 20x27 matrix referring to one of the splitted matrices each time. For each of these matrices given the calculated  $q$  we now proceed in the last step of our work, which is to estimate the optimum pick and place path of the robot.

Figure 3.6: Flowchart for  $q$  calculation

### 3.4.2 Final Path

In the final part of our method the Decision manager selects which items the robotic arm will pick and in which order. Thus, we achieve the main goal of this work, to



acquire a final picking path for our robot which is the most efficient on both time and cost aspects. For this reason we integrated the time counting. Meaning that from the beginning of our calculation process we started counting the time till the moment we got the final  $q$  matrix, for each of the splitted matrices. Given that time, we restrain our simulation to be kept in real time bases and make the picking selection based on how much time the robot would have in the real world. If this time was not part of our simulation the algorithm would continue picking items, always based on their value, starting from the most lucrative one, until there were no items left on the conveyor. This is not a realistic scenario in real world applications, as the robot will eventually lose items due to the movement of the conveyor belt. The response time of the Delta robot was integrated in this part between the picking and placing process.

Based on the final  $q$  matrix the item that will be picked from the robot is selected. More specifically, we find the maximum  $q$  that appears in the matrix and then we find to which material it corresponds. This is the most "precious" material on the conveyor at the current moment, so it is the robot's choice. Once the material is picked,, it is deleted from the list of the potential picks, as it would have been in real world tasks. After the selection the material is placed inside the bin corresponding to its class. So in the robot's path the bin's position must be present. After each material on the list of the robot's movements we see the bin we collect this type of materials.

The path is calculated individually for each of the splitted matrices. The starting point of the robot is selected to be at the beginning of the conveyor's sequence only for the first of the matrices. Afterwards it is the last position of the robot from the exact previous sequence, as it is logical. Finally, the path sequence consists of the positions the robot picked a recyclable material each time and consequently the position of the corresponding bin.



# Chapter 4

## Results

This chapter is divided in two main parts, the qualitative evaluations and the comparative evaluation of the proposed method. In the first part, we begin with the HMM's performance assessment. Next we proceed with an evaluation of the Q-Learning method, giving examples of the centroids calculation. We also explain in detail, via experimental results, how we chose our objective function. Finally, we analyze the experimental process for the evaluation of the Decision maker. In the second part, we compare our method with an existing one, developed in our laboratory, in numerous datasets for different system configurations.

### 4.1 Qualitative Evaluation

#### 4.1.1 HMM Assessment

In order to assess the HMMs' performance, we first trained each HMM, as described in Chapter 3 and compared the predictions of each model to the ground truth in order to extract four, per class, performance statistics:

1. **Accuracy:**  $ACC = \frac{TP_i + TN_i}{P_i + N_i}$ , i.e. the proportion of correct predictions (both true positives and true negatives) among the total number of cases examined,
2. **Sensitivity:**  $TPR = \frac{TP_i}{P_i}$ , i.e. the proportion of positives that are correctly identified,
3. **Specificity:**  $TNR = \frac{TN_i}{N_i}$ , i.e. the proportion of negatives that are correctly identified, and
4. **Precision:**  $PPV = \frac{TP_i}{TP_i + FP_i}$ , i.e. the proportion of true positives w.r.t. the total number of positive predictions.

with  $i = \{1, 2, 3, 4\}$  representing the corresponding class as described in table 3.1,  $pr$  and  $gt$  denoting the predicted and ground truth class, respectively,  $P_i$  and  $N_i$  being the total number of positives (occurrence) and negatives (non-occurrence), respectively, for

class  $i$  and  $TP_i, FP_i, TN_i, FN_i$  representing the true positive, the false positive, the true negative and the false negative detections, respectively, for class  $i$ .

An overview of the HMMs' performance evaluation statistics is given in table 4.1, as the per class average of all employed HMMs.

$i$	Accuracy	Sensitivity	Specificity	Precision
1	0.8988	0.917	0.9233	0.878
2	0.8433	0.92	0.938	0.8457
3	0.9056	0.84	0.9307	0.9211
4	0.8788	0.918	0.8947	0.8996
<b>Average</b>	<b>0.8816</b>	<b>0.8987</b>	<b>0.9216</b>	<b>0.8861</b>

Table 4.1: Analysis of HMM's performance.

The above presented results are more than promising presenting high values for all extracted statistics, implying that the derived models are robust, not only regarding the successful class predictions ( $TP_i$ ) but also to the rejection of a class.

#### 4.1.2 Predictions Processing

As analyzed in detail in Chapter 3, in order to continue to the Q-Learning and the picking process we first needed to convert our prediction matrix into  $X-by-27$  matrix, with  $X$  being the size of corresponding dataset stripe and 27 being the number total columns counting also the two extreme columns indicating the bins where each material will be placed after picking, with respect to its class.

The first and last (27th) column of the said matrix, whose elements is -1, represents the bin. From the 2nd to 26th column we have the expansion of one of the original prediction columns in five columns. The materials occupy a cell number which respects the size of the class material (plastic bottles are bigger than aluminum cans). This matrix was given as input to the Q-Learning algorithm in bulks of  $20 \times 27$ , firstly to perform k-means clustering for acquiring the centroids of the materials and then to predict the final picking path.

In order to perform k-means clustering on the prediction data, we first need to determine the  $k$ , i.e. the number of clusters in our data. As already mentioned, class 1 refers to the conveyor, class 2 to paper, class 3 to plastic and finally class 4 refers to aluminum. A per-class analysis on the examined data is employed in order to estimate  $k$ , based on the expected size for each material (acquired after experimentation), assuming that paper items are larger than plastic bottles which, in turn, are bigger than the aluminium cans. As an example, consider the following: if in a 20 row part of the dataset, class 2 appears 127 times we expect to have 5.08 items of class 2. Obviously the results of these divisions were rounded every time, so we expect to have 5 papers on this part of the conveyor.

Figure 4.1 presents three different indicative examples of clustering and centroid

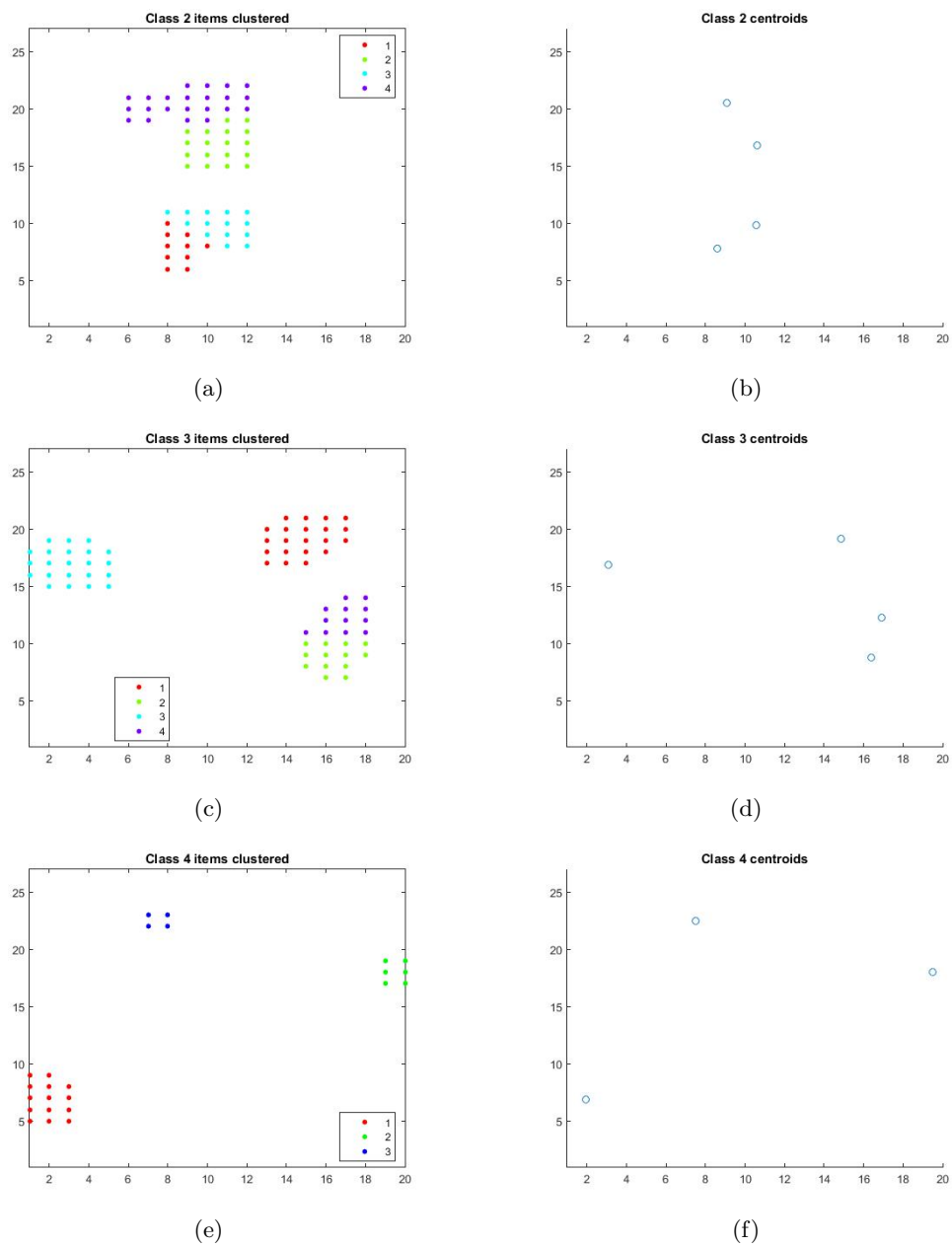


Figure 4.1: Clusters of each class for a part of the conveyor and the centroids corresponding to an item of each class

estimation. First we see the clusters for the class and then the corresponding centroids whose positions we use for the reward and  $q$  calculation.

### 4.1.3 Objective Function Evaluation

In order to assess the performance of the path planning module, we ran a series of experiments of different objective/reward function variations. Specifically, we assessed three variations of the objective function: (i) the first one takes into account exclusively the values of the materials to increase the final profit and overlooks the distance, i.e.  $d_x = 1$ , (ii) the second one has as a goal to pick the most items in the given time disregarding their value, i.e.  $v_i = 1$  and (iii) the third one is as described in section 3.4.1.

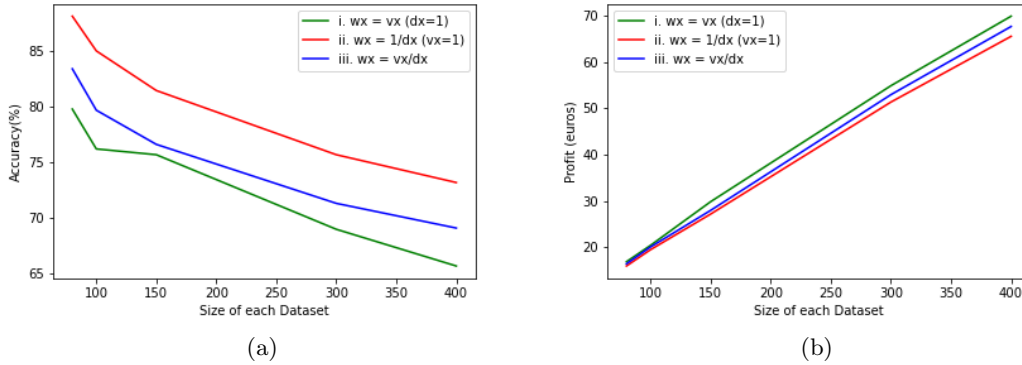


Figure 4.2: Accuracy and Profit comparison for the three objective functions

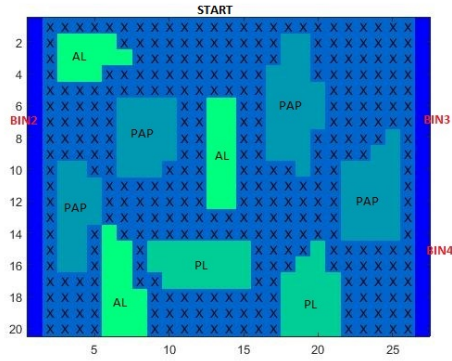
What is observed from the previous analysis is that in terms of accuracy, the second variation of the objective performs better, which is natural due to elimination of the value factor. On the other hand, in terms of profit, the second variation provides for the best results, since the distance modifier is excluded. As expected, the third variation (original reward formulation) lies somewhere in the middle of the other two, providing with balanced results, w.r.t. both accuracy and profit made.

In order to perceive the impact of each objective on the system's performance, we also assessed the path planning module under the same three reward variations. An illustrative example is shown in figure 4.3, with the three different paths for a specific dataset batch (figure 4.3(a)).

In figure 4.3(b) we observe that the robot picked 5 items. By the numbers on the figure we know in which order the pick and place actions were made. In this case all aluminum items were picked, which was expected as aluminum is the most valuable class and then all the plastic items were picked, which is the second most valuable class. In this case the path planner decided to pick the most valuable items leaving the less valuable class.

In figure 4.3(c) 6 items are picked, starting from the closer one and after placing in the corresponding bin continues picking trying to pick as many as it can without considering the value of the item.

Finally, in figure 4.3(d) we have again six picked items, in this case we observe again



(a) Conveyor visualization

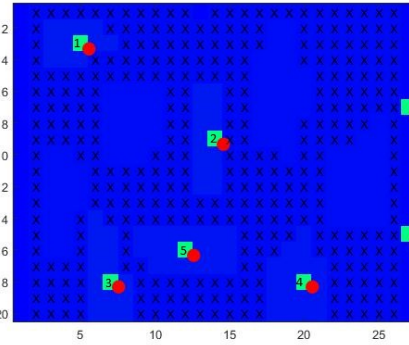
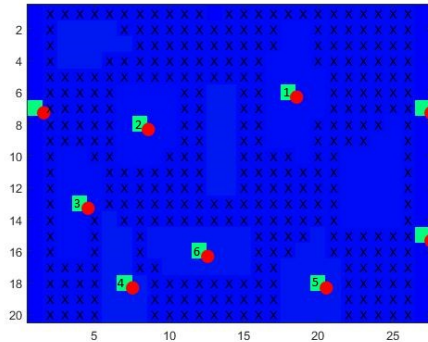
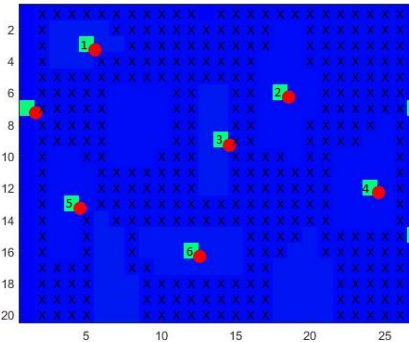
(b) Path planning using the 1<sup>st</sup> objective variation.(c) Path planning using the 2<sup>nd</sup> objective variation.(d) Path planning using the 3<sup>rd</sup> objective variation.

Figure 4.3: Different path plans depending on the reward function.

that the path is completely different than the previous two. It starts by picking the closest (first on the conveyor) and most valuable item. The path formulation continues in that way.

After thorough experimentation, we concluded that the third variation of the reward function is most appropriate to our task, combining both profit and accuracy optimization and providing for more balanced results.

#### 4.1.4 Decision Making Evaluation

The ultimate goal of our work is to acquire the best picking path the robot should follow. The final path is a matrix with 3 columns. The first two denote the position of the centroid of the picked material and the third one the class of the material.

The first row of the matrix represents a random starting point, where the robot is before the pick and place process begins, e.g. “homing” position. Every time we have a position for an item to pick, the next position is the corresponding bin in order to

place the material. We must note two things. It can be easily observed is that the positions -and paths- tend to "repeat" themselves, due the confined working space of the robot.

Number of items \ Speed	50	100	120	150
x1	0.8988	0.847	0.809	0.783
x1.2	0.8125	0.7659	0.726	0.637
x1.5	0.719	0.687	0.653	0.534

Table 4.2: Pick and place accuracy.

We evaluated the behaviour of our system under different experimental setups. Specifically, we tested the performance of our algorithm with varying conveyor belt speeds (x1, x1.2 and x1.5) and total number of items on the belt (50, 100, 120, 150), resulting into a total of 12 configurations. For each of the configurations, we generated 10 simulated datasets (as described in Chapter 2) of approximately 90 seconds of duration and fed it to the system. An overview of the performance, by means of average *pick and place accuracy* formulated as  $\mu_{\frac{SP}{SI}}$ , where *SP* denotes the "sum of items picked" and *SI* the "sum of items on the belt", is shown in table 4.2. However, note that, no matter if it was correctly predicted, an item that passes outside of the robot's working space (i.e. the robot didn't have enough time to pick) is discarded and does not count towards the *SP*.

	Class2 Picked	Class3 Picked	Class4 Picked
Normal Values	27	30	37
Weight in class 2	<b>33</b>	29	32
Weight in class 3	25	<b>38</b>	31
Weight in class 4	25	30	<b>39</b>
Sum of items per class	35	42	41

Table 4.3: Extreme Value cases

In order to assess the impact of the items' value on the decision making we also measured the efficiency of our picking algorithm on, we conducted a series of experiments by altering the value of each material -namely  $vx_i$  in eq. 3.4- circularly (one at a time) to a value significantly larger than that of to the remaining two. Each of the configurations is tested on 10 generated sequences with approximately 90 seconds duration, x1 conveyor belt speed and 118 items. In our normal run of the algorithm the values are in the same range as shown in table 3.3, while in the rest of the cases we augmented the corresponding class value by 150. An average of the performance of the decision manager, by means of per class items picked, is shown in table 4.3. In



each of the cases, compared to the normal value run, we observe that the planner picks more of the weighted class that it would. These results strengthen the efficiency of our path planner and its ability to focus in the most valuable items on the conveyor belt.

	Accuracy	Time(min)
seq. 1	0.5685	22.3
seq. 2	0.5672	22.23
seq. 3	0.5721	22.9
seq. 4	0.5683	22.3
<b>Average</b>	<b>0.569</b>	<b>22.4</b>

Table 4.4: Long-term accuracy.

Finally, we assessed the long-term performance of the decision manager. Towards this we created 4 long (approximately 25 minutes duration) videos with dense distribution of items on the conveyor belt, summing to 3000 items per sequence, providing with a better approximation of a real life problem. The evaluation results, by means of accuracy and total time required, are given in table 4.4.

Considering the results, we can understand that our method can efficiently work even under much more difficult conditions. The dataset that we generated was a much ticker dataset with a large number of items. In this point it must be underlined that in real conditions, for example in a factory where the flow of items falling on the belt in huge, most of the times there are more than one robots in order to pick the items. With that in mind, it becomes obvious that the efficiency of our method is more than satisfactory. We achieve to pick more than half the items on the belt, provided the employed balanced reward function.

## 4.2 Comparative Evaluation

In order to further assess the performance of our method, we compare it with that of a proximity-based picker developed in CVRL for the purposes of the ANASA project[2]. The said picker, while also taking into account the time and the speed of the conveyor belt, bases the choice of an item each time only on its distance from the robot. The value of the items is not taken into account. In this way the picker manages to sort a number of items a satisfactory time but the profit in the end of the process is not the optimum that can be achieved.

For comparison purposes, we created a total of 50 simulated sequences, containing varying number of items (i.e. 80, 100, 150, 300 and 400) at two different conveyor belt speeds (x1 and x1.2). In all experiments, the parameters for our method were set to: material values as in table 3.3, while the maximum iterations and the gamma ( $\gamma$ ) factor for the  $q$  calculation in equation 3.5 were set to 50 and 0.9, respectively. An overview of the performance of the two methods, in terms of accuracy, required time

and profit made is given in tables 4.5 and 4.6 for the two different speed configurations, respectively.

Dataset Size	Accuracy		Time		Profit	
	Proximity Picker	Our Method	Proximity Picker	Our Method	Proximity Picker	Our Method
80	73.45%	84.15%	60.8	64.02	14.63	16.96
100	71.26%	79.3%	69.3	73.9	16.78	19.75
150	69.5%	78.52%	88.7	87.5	25.55	28.73
300	62.83%	70.72%	147.9	153.85	47.07	52.61
400	61.7%	68.46%	187.3	195.42	60.92	67.2

Table 4.5: Performance analysis at conveyor belt speed x1.

Dataset Size	Accuracy		Time		Profit	
	Proximity Picker	Our Method	Proximity Picker	Our Method	Proximity Picker	Our Method
80	62.03%	70.19%	43.02	46.69	12.4	14.15
100	59.67%	69.33%	51	57.1	14.66	17.27
150	54.8%	59.01%	71.5	69.98	20.15	21.63
300	53.74%	60.34%	131.74	137.95	40.29	44.92
400	52.43%	59.97%	169.79	178.87	51.76	58.87

Table 4.6: Performance analysis at conveyor belt speed x1.2.

The time difference we see when we compare the two results is due to the different implementations. To be more precise, the proximity picker was implemented in Python and our method was implemented in MATLAB. Also the way time was embedded in each system was different, which also had an effect in the final result. As we can notice though, even with this difference when we observe the over time results in the random picker are outperformed by our method, both in terms of profit and number of items picked.

As we can see in all the above tables our method outperforms the random picker in terms of overall accuracy and profit. We observe an average 10% optimization in overall picking accuracy and in some of the experiments almost 20% rise in the accuracy of class 4 (aluminum) which is the most profitable class. This was accomplished without loses in the overall accuracy and by giving to the valuable class only a small rise in weight compared to the other classes.

Additionally, we can observe that as the number of items on the conveyor grows, the difference between the profit, which was the main target of our method also grows. We have a scalable optimization in profit which as we can understand gives us even greater differences as the number of items approaches the number of items in real world applications.

The accuracy optimization is mainly stable in all the sequences of the dataset, which is expected, as the number of items the robotic arm can pick is restricted by the capabilities of the arm and the speed of the conveyor.

The statistical findings on the performance of the two methods are also graphically demonstrated in the following figures. The left graph in each of the figures depicts the corresponding performance at normal conveyor belt speed, while the right graph depicts the corresponding performance at speed x1.2. In each graph the x-axis depicts the size of the dataset (how many items each dataset has) and the y-axis depicts the corresponding metric (accuracy, time, profit).

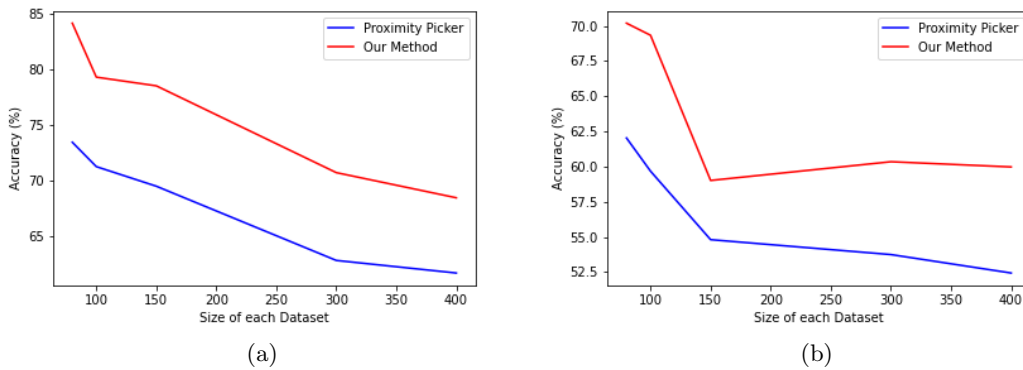


Figure 4.4: Accuracy comparison of the two methods at speeds x1 (a) and x1.2 (b)

Figure 4.4 illustrates the overall accuracy of the methods w.r.t. different dataset sizes. We can see from the figures that in both cases, our method outperforms the proximity picker. We can also observe a stability at the difference between the methods, which is expected as the robotic arm has a limit at performance, it needs some time between each pick and place action, which in both implementations was simulated as a pause of almost a second between them (half second pause after pick action and half second pause after place action).

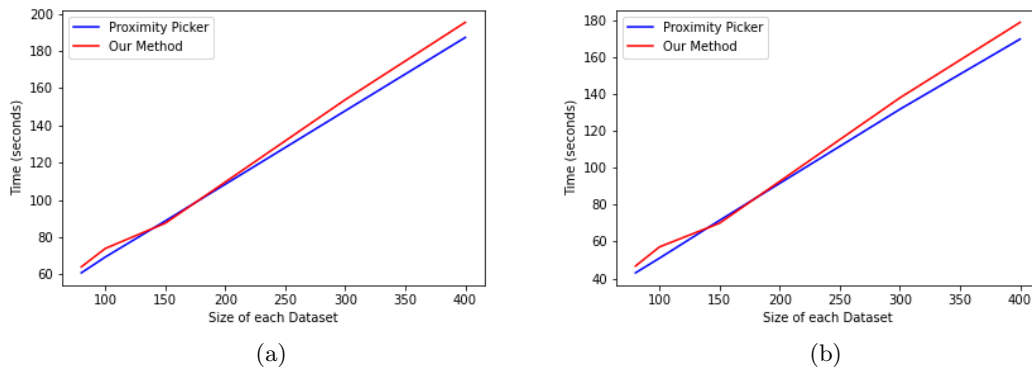


Figure 4.5: Time comparison of the two methods at speeds x1 (a) and x1.2 (b)

In figure 4.5 we demonstrate the time average results for the two methods. We can observe that as explained before due to the different implementations and definition of time in each of them we have a small rise in time in our method, in average a 5 second rise. However, this is practically of minimal importance, given the increase in terms of accuracy and profit.

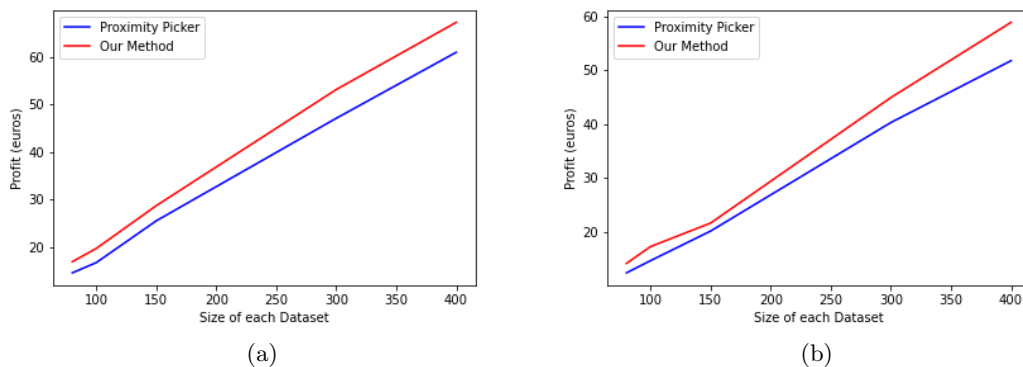


Figure 4.6: Profit comparison of the two methods at speeds x1 (a) and x1.2 (b)

Moreover, on figure 4.6 we graphically demonstrate the profit comparison of the two methods. It becomes obvious by the figures that despite the conveyor speed the profit is always improved. We can also see the scalability of our method as the size of the dataset increases.

We also have to note here that some of the datasets were sparse and some were more dense than others. This parameter also does not affect the effectiveness of our method neither in accuracy nor in profit. The results are consistent in all datasets in spite any parameter changed when we conducted the experiments.

Below we finally demonstrate the overall comparative results concerning the per-class accuracy for the two methods in datasets of different size.

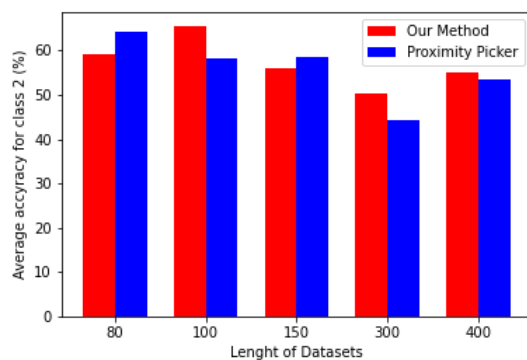


Figure 4.7: Average accuracy of the two methods for class 2 in different dataset sizes

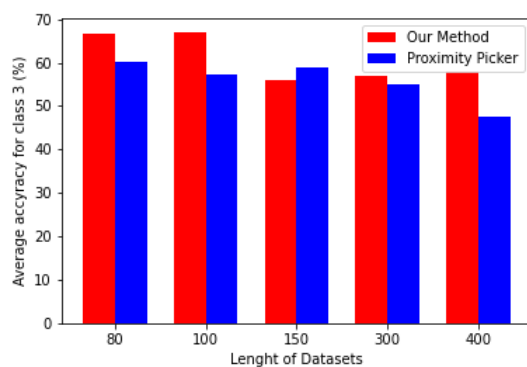


Figure 4.8: Average accuracy of the two methods for class 3 in different dataset sizes

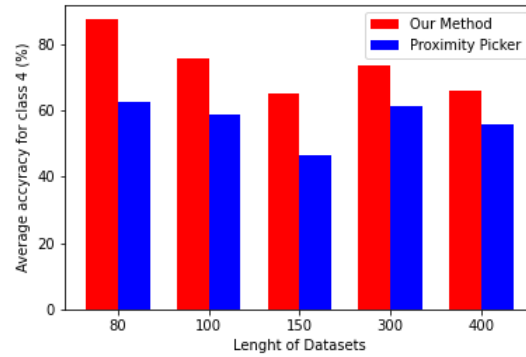


Figure 4.9: Average accuracy of the two methods for class 4 in different dataset sizes

As it is obvious from figure 4.7 to figure 4.8 that with respect to classes 2 and 3, there is no significant difference in the accuracies of the two methods. However, when it comes to class 4 in 4.9 our method outperforms the proximity picker no matter the length of the dataset, the density or the speed of the conveyor, which ultimately shows the success in our implementation as class 4 was in all datasets the most valuable class.

# Chapter 5

## Discussion

In this chapter we conclude with respect to the presented Path Planner framework. The latter is deemed as successful since it met its intended design goals. That being said, and as is the case with most robotics frameworks, there is still space for further improvements and there is also great potential for its use in numerous other fields except recycling. After completing the detailed description of the methodology, assessing and comparing its results experimentally, we will propose a summary of incremental upgrades and additions in the spirit of further improving it.

### 5.1 Our Contribution

The focus of this work has been to create an optimized Path Planner for pick and place processes. Starting with the prediction of the materials that will next appear on the conveyor belt, having only information about the previous throws and features of the materials. For this task we successfully implemented Hidden Markov Model and predicted the materials with great accuracy. The latter made available additional information and greatly facilitated accurate material classification.

As far as it concerns the main part of our work, the Path Planner itself, we demonstrated that it greatly contributes in terms of object selection accuracy and cost (value) of the picked objects. We succeeded in enhancing accuracy by approximately 10%, and the value of the selected objects also increased in all cases. We proved our optimized planner works no matter how small the difference in value is (we experimented in very big differences and very small as it is the case in real world scenarios). In every experiment the Planner worked as expected and gave priority to the most valuable class, always taking into account the limitations of the system, the distance of the materials from the robotic arm and the overall accuracy. We also observed that our method gives us a scalable difference from existing methods, basically in terms of profit, the bigger the number of items on the conveyor grows the bigger the difference in profit grows. This shows great potential of our method in real world applications where the flow of items is very large and continuous.

## 5.2 Future Work

In research and development endeavors, additions and improvements can always be employed to enhance a work. In our case a straightforward enhancement, for the prediction part of our work, would be a bigger dataset which will make the results more robust than they already are, as the framework will be tested in real data in large scale. When we refer to a bigger dataset, we mean in terms of the materials that are predicted, the material's features that are extracted and even the number of samples, in the sense that more samples are always better.

For the Path Planner part, a future task could be to add more parameters in the reward function of the Q-Learning algorithm, such as the energy consumption of the picking robot and also the corrosion of the robot resulting from each pick-and-place action taking into account the distance and the speed it has. In that way, the system can be optimized not only in terms of profit, but also in terms of energy consumption and corrosion. We can understand that such a multifaceted optimized system is highly desirable for processes like the one we are investigating.

Another very interesting future direction is to use the Path Planner we created for more than one picking robots. As it is generally the case, in a factory for sorting processes more than one robots are used. Putting in use in such a complete sorting system our optimized picking method, we could have a clear image about the actual real world benefit that our method provides and its full potential for real-world employment.

## 5.3 Other Applications of the Framework

Our framework may also find applications in a wide variety of fields. As such, it can be used in various industrial tasks that require selection and separation of items. For example in factories that pack products to get them out for distribution or in units that pack food products, for example selection between which of the tomatoes will be sold and which are rotten or bad for the market. Making the appropriate adjustments each time for the existing system (dimensions of the conveyor belt, robot's specifications, items' classes, conveyor speed, materials' value) this framework can basically be applied in various relevant tasks, in any field that involves classification and simultaneous separation of items on a conveyor belt.

## 5.4 Epilogue

After examining all aspects of the proposed method, making numerous experiments for many different cases and comparing it with an existing method, we can conclude that our method successfully fulfills its goal. The optimization in the collection of the materials was effectively demonstrated. In the heart of our approach lies a reinforcement learning formulation that proved appropriate for such industrial robotic applications. Reinforcement learning in pertinent automation tasks, as shown in this work, caters for



optimization, scalability and the capability of learning any specific application through the direct use in the applications environment.



# Annex: Code

In this section we provide basic parts of the implemented code in three separate parts: (1) Hidden Markov Models, (2) Reward function calculation, and (3)  $q$  matrix calculation.

## PART ONE: HIDDEN MARKOV MODELS

```
T = length(O); % length of each of these sequences
N = 4; %number of states in Markov process
M = length(B); %number of discrete observations
ct(1) = 0;

for i=1:N

    ttmp = prior(i)*B(i,O(1));

    at(i,1) = ttmp;

    ct(1) = ct(1) + ttmp;

end

%scale a0

ct(1) = 1/ct(1);
```

```
for i=1:N
    at(i,1) = at(i,1)*ct(1);
end

% compute at(i)
for t = 2:T
    ct(t) = 0;
    for i = 1:N
        at(i,t)=0;
        for j = 1:N
            at(i,t)=at(i,t) + at(j,t-1)*A(j,i);
        end
        at(i,t) = at(i,t)*B(i,0(t));
        ct(t) = ct(t)+at(i,t);
    end

    %scale at(i)
    ct(t) = 1/ct(t);
    for i = 1:N
        at(i,t) = ct(t)*at(i,t);
    end
end

%The b-pass
```

```
for i=1:N
    b(i,T) = ct(T);
end

%b-pass

for t=T-1:-1:1
    for i=1:N
        b(i,t)=0;
        for j=1:N
            b(i,t)=b(i,t)+A(i,j)*B(j,0(t+1))*b(j,t+1);
        end
        b(i,t)=ct(t)*b(i,t);
    end
end

for t=1:T-1
    tempgt = zeros(N,N);
    for i=1:N
        g(i,t) = 0;
        for j=1:N
            tempgt(i,j)=at(i,t)*A(i,j)*B(j,0(t+1))*b(j,t+1);
            g(i,t) = tempgt(i,j)+g(i,t);
        end
    end
end
```

```
        gt(:,:,t) = tempgt;

end

for i=1:N
    g(i,T) = at(i,T);
end

%Re-estimate A,B and prior
%re-estimate prior
for i=1:N
    prior(i) = g(i,1);
end

%re-estimate A
for i=1:N
    demon = 0;
    for t=1:T-1
        demon = demon + g(i,t);
    end
    for j=1:N
        numer = 0;
        for t=1:T-1
            numer = numer + gt(i,j,t);
        end
        A(i,j) = numer/demon;
```

```
    end
end

%re-estimate B
for i=1:N
    demon = 0;
    for t=1:T
        demon = demon + g(i,t);
    end
    for j=1:M
        numer = 0;
        for t=1:T
            if(O(t) == j)
                numer = numer + g(i,t);
            end
        end
        B(i,j) = numer/demon;
    end
end

MAX = -1;
W = -1;
G = -1;
```

```

for t=1:T
    for i=1:N
        if(g(i,t) > MAX)
            MAX = g(i,t);
            G = i;
            W = t;
        end
    end
    state_in(t) = G;
    MAX = -1;
end

```

## PART TWO: THE REWARD $w_x$ CALCULATION

```

if (current_position(:) ~= previous_position(:))
for i=1:n
    for j=1:m
        if (reward(i,j)~=0)
            if (STRIPES(i,j)==1)
                reward(i,j) = -Inf;
            elseif (STRIPES(i,j)== 2 || STRIPES(i,j)== 3 ||
                STRIPES(i,j)== 4)
                if (ismember([i j], Cent_ordered, 'rows'))
                    dist = sqrt( (i-current_position(1,1))^2 + (j-
                        current_position(1,2))^2 );
                end
            end
        end
    end
end

```



```

    if (STRIPES(i,j)==2)
        reward(i,j) = 0.22/dist;
    elseif (STRIPES(i,j)==3)
        reward(i,j) = 0.25/dist;
    elseif (STRIPES(i,j)==4)
        reward(i,j) = 0.27/dist;
    end

else

    reward(i,j) = 0;

end

elseif (STRIPES(i,j)==-1)

    reward(i,j) = 0;

end
end
end
end
end

current_position(:) = previous_position(:);

end

```

### PART THREE: $q$ MATRIX CALCULATION

```

[n_actions_r,n_actions_c] = find(reward(:,>0));

T = [n_actions_r,n_actions_c];
if (isempty(T))
    break;
end

% choose an action at random and set it as the next state
next_p(1,1) = n_actions_r(randi([1 length(n_actions_r)]));
[t,~] = find(n_actions_r==next_p(1,1));%?

```

```

next_p(1,2) = n_actions_c(t(1));

if (STRIPES_temp(next_p(1,1), next_p(1,2)) == 2)
    bin_p(1,1) = 7;
    bin_p(1,2) = 1;
elseif (STRIPES_temp(next_p(1,1), next_p(1,2)) == 3)
    bin_p(1,1) = 7;
    bin_p(1,2) = 27;
elseif (STRIPES_temp(next_p(1,1), next_p(1,2)) == 4)
    bin_p(1,1) = 15;
    bin_p(1,2) = 27;
end

current_p(:) = bin_p(:);

[n_actions_r,n_actions_c] = find(reward(:,>0);
% find the maximum q-value i.e, next state with best action
max_q = 0;
for ar=1:length(n_actions_r)
    for ac = 1:length(n_actions_c)

        max_q = max(max_q,q(n_actions_r(ar),n_actions_c(ac)));

    end
end

% Update q-values as per Bellman's equation
q(next_p(1,1),next_p(1,2)) = reward(next_p(1,1),next_p(1,2)
)+gamma*max_q;

reward(next_p(1,1),next_p(1,2)) = 0;

```

# Bibliography

- [1] Kaza, Silpa, Yao, Lisa C., Bhada-Tata, Perinaz, Van Woerden and Frank. *What a Waste 2.0 : A Global Snapshot of Solid Waste Management to 2050*. Washington, D.C. : World Bank Group., 2018.
- [2] ANASA Robotic Sorter. <https://anasasorter.com/>.
- [3] Panos Trahanias. Path Planning, HY475 Autonomous Robotic Navigation. <https://www.csd.uoc.gr/~hy475/?sec=lectures>.
- [4] Sutton, Richard S. and Barto, Andrew G. *Introduction to Reinforcement Learning*. The MIT Press, second edition.
- [5] freeCodeCamp. An introduction to Q-Learning: reinforcement learning. <https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/>.
- [6] McGraw-Hill and Sybil P. Parker. *McGraw-Hill Dictionary of Scientific and Technical Terms*. Addison-Wesley, Reading, Massachusetts, 2003.
- [7] Rinkesh. Various Recycling Facts. <https://www.conserve-energy-future.com/various-recycling-facts.php>.
- [8] A. Huchinson. Recycling by the Numbers. <https://www.popularmechanics.com/science/environment/a3757/4291576/>, 2008.
- [9] E. Papadakis and F. Raptopoulos and M. Koskinopoulou and M. Maniadakis. On the Use of Vacuum Technology for Applied Robotic Systems. *2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, pages 73–77, 2020.
- [10] F. Raptopoulos and M. Koskinopoulou and M. Maniadakis. Robotic Pick-and-Toss Facilitates Urban Waste Sorting \*. *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 1149–1154, 2020.
- [11] R. Sarc and A. Curtis and L. Kandlbauer and K. Khodier and K.E. Lorber and R. Pomberger. Digitalisation and intelligent robotics in value chain of circular economy oriented waste management – A review. *Waste Management*, 95:476–492, 2019.

- [12] Mokled, E. and Chartouni, Georges and Kassis, Carine and Rizk, Rany. Parallel Robot Integration and Synchronization in a Waste Sorting System. *Mechanisms and Machine Science*, pages 171–187, 05 2019.
- [13] Zhang Z., Wang H., Song H., Zhang S., Zhang J. Industrial Robot Sorting System for Municipal Solid Waste. *ICIRA : Intelligent Robotics and Applications*, pages 342–353, 08 2019.
- [14] C. Zhihong, Z. Hebin, W. Yanbo, L. Binyan and L. Yu. A vision-based robotic grasping system using deep learning for garbage sorting. *2017 36th Chinese Control Conference (CCC)*, pages 11223–11226, 07 2017.
- [15] Simonyan, Karen and Zisserman, Andrew. *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 09 2014.
- [16] L. Li, Y. Zhang, M. Ripperger, J. Nicho, M. Veeraraghavan and A.Fumagalli. Autonomous Object Pick-and-Sort Procedure for Industrial Robotics Application. *International Journal of Semantic Computing Vol. 13, No. 2 (2019) 161–183*.
- [17] Wikipedia. Ros. <http://wiki.ros.org/>.
- [18] Rodney A. Brooks. Planning Collision- Free Motions for Pick-and-Place Operations. *The International Journal of Robotics Research*, 2(4):19–44, 1983.
- [19] Deepak L. Rajnor, A.S Bhide. Automatic Material Handling System Using Pick Place Robotic Arm Image Processing. *IJSRD - International Journal for Scientific Research Development— Vol. 2, Issue 05, 2014 — ISSN (online): 2321-0613*.
- [20] R. Kumar and S. Lal and S. Kumar and P. Chand. Object detection and recognition for a pick and place Robot. *Asia-Pacific World Congress on Computer Science and Engineering*, pages 1–7, 2014.
- [21] Kensuke Harada, Tokuo Tsuji, Kazuyuki Nagata, Natsuki Yamanobe and Hiromu Onda. Validating an object placement planner for robotic pick-and-place tasks. *Robotics and Autonomous Systems*, 62(10):1463 – 1477, 2014.
- [22] Sharma, Neha and Jain, Vibhor and Mishra, Anju. An Analysis Of Convolutional Neural Networks For Image Classification. *Procedia Computer Science*, 132:377–384, 01 2018.
- [23] Liang, Ming and Hu, Xiaolin. Recurrent Convolutional Neural Network for Object Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
- [24] Susanth, G and Livingston, Jenila and Livingston, Agnel. Garbage Waste Segregation Using Deep Learning Techniques. *IOP Conference Series: Materials Science and Engineering*, 1012:012040, 01 2021.

- [25] Lin-Lin Shen and Zhen Ji. Gabor Wavelet Selection and SVM Classification for Object Recognition. *Acta Automatica Sinica*, 35(4):350–355, 2009.
- [26] J. Hornegger and H. Niemann and D. Paulus and G. Schlottke. *Object recognition using hidden Markov models*, volume 16 of *Machine Intelligence and Pattern Recognition*. North-Holland, 1994.
- [27] Özkaya, Umut and Seyfi, Levent. Fine-Tuning Models Comparisons on Garbage Classification for Recyclability. *ISAS*, 12 2018.
- [28] Gary Thung and Mingxiang Yang. Classification of Trash for Recyclability Status. *Corpus Id-27517432*, 2016.
- [29] Ye, N. and Zhang, Y. and Wang, R. Vehicle trajectory prediction based on Hidden Markov Model. *KSII Transactions on Internet and Information Systems*, 10:3150–3170, 07 2016.
- [30] J. Kosecka and F. Li. Vision based topological Markov localization. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 2:1481–1486 Vol.2, 2004.
- [31] Benyacoub, B. and ElBernoussi, S. and Zoglat, A. and El Moudden, Ismail. Classification with hidden markov model. *Applied Mathematical Sciences*, pages 2483–2496, 01 2014.
- [32] Martin, Juliette and Gibrat, Jean-Francois and Rodolphe, Francois. Analysis of an optimal hidden Markov model for secondary structure prediction. *BMC structural biology*, 6:25, 02 2006.
- [33] Laud, Adam. *Theory and Application of Reward Shaping in Reinforcement Learning*. PhD thesis, 04 2011.
- [34] Peters, Jan and Schaal, Stefan. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21:682–, 06 2008.
- [35] Kober, Jens and Peters, Jan. *Reinforcement Learning in Robotics: A Survey*. Springer International Publishing, Cham, 2014.
- [36] Kakade, Sham. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, 01 2003.
- [37] Kearns, Michael and Singh, Satinder. Near-Optimal Reinforcement Learning in Polynomial Time. *Machine Learning*, 49, 01 2002.
- [38] Brafman, Ronen and Tennenholtz, Moshe. R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *The Journal of Machine Learning Research*, 3:953–958, 01 2001.

- [39] Gaskett, Chris. *Q-Learning for Robot Control*. PhD thesis, Research School of Information Sciences and Engineering. Department of Systems Engineering and The Australian National University, 2002.
- [40] Ee Soong Low, Pauline Ong and Kah Chun Cheah. Solving the optimal path planning of a mobile robot using improved Q-learning. *Robotics and Autonomous Systems*, 115:143 – 161, 2019.
- [41] L. Tai and M. Liu. A robot exploration strategy based on Q-learning network. *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 57–62, 2016.
- [42] Millán, J.d.R., Posenato, D. Dedieu, E. Continuous-Action Q-Learning. *Machine Learning*, 49:247–265, 2002.
- [43] Massa, Francisco and Girshick, Ross. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. <https://github.com/facebookresearch/maskrcnn-benchmark>, 2018.
- [44] R.Clavel. Une nouvelle structure de manipulateur parallèle pour la robotique legere. *Automatique-productique informatique industrielle*, pages 401–417, 1989.
- [45] Alfonso Hernandez and Jose Ignacio Ibarreche and Victor Petuya and Oscar Altuzarra. Structural Synthesis of 3-DoF Spatial Fully Parallel Manipulators. *International Journal of Advanced Robotic Systems*, 11(7):101, 2014.
- [46] Mike Wilson. *Implementation of Robot Systems: An Introduction to Robotics, Automation, and Successful Systems Integration in Manufacturing*. Butterworth-Heinemann, 2015.
- [47] W Khalil and E Dombre. *Modeling, Identification and Control of Robots*. Butterworth-Heinemann, 2004.
- [48] Robotics Online Marketing Team. Is a Vacuum Gripper Right for Your Collaborative Robot Application? <https://www.robotics.org/blog-article.cfm/Is-a-Vacuum-Gripper-Right-for-Your-Collaborative-Robot-Application/134>.
- [49] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR Upper Saddle River, NJ United States, 2019.
- [50] Czar Yobero. K-Means Clustering Tutorial. <https://rpubs.com/cyobero/k-means>.
- [51] Watkins C.J.C.H., Dayan P. Q-learning. *Mach Learn* 8, page 279–292, 1992.

- [52] Gregor Klančar and Andrej Zdešar and Sašo Blažič and Igor Škrjanc. *Chapter 7 - Autonomous Guided Vehicles*. Butterworth-Heinemann, 2017.
- [53] Stamp, Mark. A revealing introduction to hidden markov models. *Science*, pages 1-20, 01 2004.
- [54] Bhartendu. Q-learning-example. <https://github.com/matrixBT/Q-learning-example>.