Sorting objects from a conveyor belt using active perception with a POMDP model

Ady-Daniel Mezei, Levente Tamás, Lucian Buşoniu

***Abstract*— We consider an application where a robot must sort objects traveling on a conveyor belt into different classes. The detector and classifier work on 3D point clouds, but are of course not fully accurate, so they sometimes misclassify objects. We describe this task using a novel model in the formalism of partially observable Markov decision processes. With the objective of finding the correct classes with a small number of observations, we then apply a state-of-the-art POMDP solver to plan a sequence of observations from different viewpoints, as well as the moments when the robot decides the class of the current object (which automatically triggers sorting and moving the conveyor belt). In a first version, observations are carried out only for the object at the end of the conveyor belt, after which we extend the framework to observe multiple objects. The performance with both versions is analyzed in simulations, in which we study the ratio of correct to incorrect classifications and the total number of steps to sort a batch of objects. Real-life experiments with a Baxter robot are then provided with publicly shared code and data at `http://community.clujit.ro/display/TEAM/Active+perception`.**

I. Introduction

Recent advances in the industrial domain (for example Industry 4.0) have opened up a multitude of opportunities for robots, as they are expected to perform the tasks either on their own or along other humans, as coworkers [1], [2], [3], [4]. In such cases, most of the time it is expected from robots to adapt to the environment and to take decisions on their own. In practice however, sensor and motor inaccuracies, noise–affected data, algorithm misbehaviours, improper classifications or object occlusions are just a few of the challenges that the robot has to deal with in the decision making process.

A very useful framework to handle sensing uncertainty in particular is active perception, which closes the loop between the sensing and control modules of the robot by taking control actions with the explicit purpose of obtaining more informative data from the sensors.

In this paper the following active perception problem is considered. A robot is working in a factory and has the task of sorting objects of different shapes that travel on a conveyor belt. The classification of the objects is done using a sensor the robot is equipped with, and due to sensor inaccuracies, multiple scans are required for a proper classification. The robot has a set of poses (viewpoints) from which it scans the conveyor belt. It starts from an initial viewpoint and can move between viewpoints in order to gain more information about the objects.

Our main contribution is a solution of the sorting task in the framework of partially observable Markov decision processes (POMDPs). The step of modeling the problem as a POMDP is essential, and we describe it in detail. Object classes are uncertain so a probability distribution (belief state) over them is maintained, which is updated using the observations. At each step, the robot has the option of either changing the viewpoint to make a new observation, or to decide on the class of the object at the end of the belt, sort it, and advance the belt. A sequence of these actions is planned over a long horizon so as to maximize a cumulative sum of rewards assessing the quality of sorting decisions, using a state-of-the-art planner called DESPOT [5].

A key feature of the method is that it allows the robot to observe multiple positions from the conveyor belt at the same time, from each scan. This allows us to accumulate and propagate information between positions when the conveyor belt advances, so that the robot already starts with a good estimate of the new class at the end of the belt. This reduces the total number of steps needed to sort the objects, while still maintaining a high classification accuracy.

In addition to simulation results, we present also real experiments with a Baxter robot equipped with an ASUS Xtion 3D sensor mounted on one of its wrists. The objects being sorted are light bulbs of different shapes, and their classification is carried out the data coming from the sensor in the form of point clouds. For both simulation and on the physical robot, we report two sets of experimental results. In the first case the robot observes only a single object from the conveyor belt without propagating the belief states, while in the second case the observations are extended to two positions and the propagation strategy is adopted.

Related work: References [6], [7] provide early approaches toward active perception. Since then, the field has grown tremendously, as the availability and range of applications of robotic platforms and sensors has increased.

In active perception, there are two ways to perform detection: passive and active. In the former, the observation viewpoint is changed, leaving the state of the objects unaltered, see e.g. [8], [9], while in the latter, the objects are actively manipulated in order to improve classification, e.g. [10]. In this paper we use the passive approach.

Patten et al. [8] propose an active perception based method to detect objects in a cluttered environment. The solution uses RGB-D data (where D means depth), and by maintaining

The authors are with the Automation Department, Technical University of Cluj-Napoca, Romania ({daniel.mezei, levente.tamas, lucian.busoniu}@aut.utcluj.ro). This work was supported by two grants of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI: project number PN-III-P1-1.1-TE-2016-0670, grant agreement no. 9/2018; and grant number PN-III-P2-2.1-BG-2016-0140; and also by Hungarian Research Fund grant number OTKA K 120367 and MTA Bolyai Scholarship.

class and pose information in an occupancy grid it is able to detect multiple objects at once. The solution chooses, in a greedy manner, from a set of viewpoints the one that contains most information about the scene, handling in this way possible occlusions between objects. Our solution and [8] both take into account class and pose information to choose the next action, the difference being that our approach plans the next actions taking into account a longer horizon, rather than just one step.

The setup presented in [11] is similar to ours: the robot uses 3D information to detect objects travelling on a conveyor belt and to manipulate them. One difference from our approach is that the conveyor belt transports the objects continuously and is not stopped to perform object detection before restarting the belt. Besides that, the solution in [11] takes into account only the detection of the objects being transported for their subsequent manipulation and does not treat uncertainty associated with their detection.

In [12], [13], a 3D sensor is moved between multiple viewpoints to perform scans. Class and pose information about the targets are modeled using a POMDP, based on which the next moves are planned. However, the scenario considered there is different and does not have the structure of the conveyor-belt problem considered here, which allows us to additionally use propagation of the belief state to reduce detection time. Moreover, the solutions of [12], [13] handle possible occlusion between the objects from the environment, while in our case the objects are transported in cups on the conveyor belt, in a controlled manner, so the occlusion problem does not arise.

In [14], POMDPs are applied for active navigation and obstacle avoidance. There exist of course other active sensing paradigms that do not employ POMDPs, see e.g. [15], [16].

Next, Section II describes POMDPs in general and for the problem at hand. Section III details the active perception pipeline, hardware, and software. The results are given in Section IV, before the conclusions in Section V.

II. POMDP MODEL OF THE SORTING TASK

A POMDP (partially observable Markov decision process) is defined in general as a tuple (S, A, T, R, Z, O) [17], where S represents a discrete, finite set of states, A is the finite set of actions and $T : S \times A \times S \rightarrow [0, 1]$ is a state transition function attached to the POMDP. When an action a is performed in a state s, the outcome is a new state, s', with the probability $T(s, a, s')$. Moreover, $R : S \times A \rightarrow \mathbb{R}$ is the attached reward function, which is assumed bounded and $r = R(s, a)$ says that the reward r is obtained if in state s action a is executed.

At any step, the underlying state s is not known, instead observations $z \in Z$ provide incomplete information about it. $O : S \times A \times Z \rightarrow [0, 1]$ is the observation function attached to the POMDP. It gives the probability $O(s', a, z)$ of making a certain observation z if s' is obtained after taking a.

Next, we move on to the problem of sorting objects on a conveyor belt. To model this problem as a PODMP, we describe in turn the states, actions, and the transition, reward and observation functions (for the latter, we also explain the set of possible observations). To make things clear, after providing the definition of each object for the general sorting problem, we instantiate it for our specific example problem that we will study in our experiments. It must nevertheless be kept in mind that the framework can be applied to any such sorting problem.

The states of the sorting problem are defined in general as follows:

$$\begin{aligned} & c^j \in C = \{c_1, c_2, c_3, \ldots, c_L\} \\ & p \in P = \{p_1, p_2, p_3, \ldots, p_K\} \\ & s = (c^1, c^2, \ldots, c^H, p) \\ & s \in S = C^H \times P \end{aligned} \tag{1}$$

with c_1, c_2,..., c_L being the classes of objects that travel on the conveyor belt and p_1,p_2,..., p_K being the viewpoints. The states can be divided in two categories: object classes and viewpoints. There are L classes of objects, whose models are known, that are transported on the conveyor belt and K viewpoints from which observations can be taken, both L and K being finite. H represents the number of positions from the conveyor belt that are of interest, and the robot observes all the objects from those positions. The viewpoints are fully observable, owing to access to the internal sensors of the robot arm. The object classes are partially observable (see the observation function definition, later). The state collects the classes for the H positions and the current viewpoint. We use superscript j to denote the position index.

In our particular problem, the observation points were uniformly sampled from the surface of a sphere [18]. In order to avoid occlusion between the objects and the belt, only the upper half of the sphere was used. The sphere had a 70 cm radius and was centered on the end of the conveyor belt. The radius of the sphere has been fined tuned, taking into account the limitations of the RGB-D sensor as well as the geometry of the robot arms. Points outside the workspace of the arm were considered unreachable and eliminated from the sampled list, and a graph was constructed out of the remaining points. Figure 1 shows a set of sampled points that would be used for graph construction. For each point its closest neighbours are computed along the cardinal directions(north, south, east, west), the robot being restricted to travel only through neighbouring points.

In the end, we were left with $K = 3$ three viewpoints, denoted by p_1, p_2, p_3. Moreover, $L = 4$ object classes were used, namely: “elongated”, “livarno”, “mushroom”, and “standard”. The number of positions of interest from the conveyor belt is $H = 2$.

Actions are given in general by:

$$\begin{aligned} & A_p \in \{m_1, m_2, \ldots, m_K\} \\ & A_d \in \{d_1, d_2, \ldots, d_L\} \\ & a \in A : A_p \bigcup A_d \end{aligned} \tag{2}$$

Actions are either of the motion type (set A_p) or of the decision type (set A_d). Motion actions are deterministic and each moving the arm to a particular viewpoint. A decision action means that the robot is sufficiently sure about the class identified for the object at the end of the conveyor

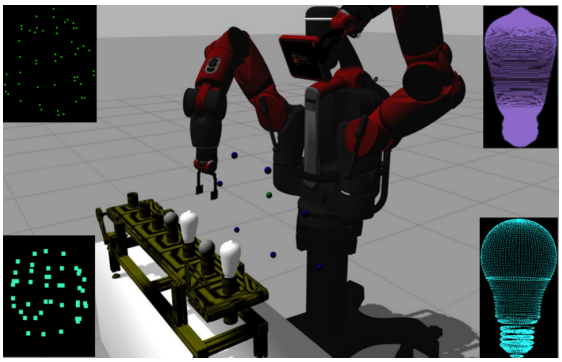

Fig. 1: The point sampling and reachability testing in a simulated environment. Example models for light bulbs are given on the right and preprocessed candidates on the left.

belt to stop the identification process, move the object to its appropriate bin, advance the conveyor belt, and proceed to the next object.

In our particular problem, the motion actions are m_1, m_2, m_3, moving the arm to p_1, p_2, p_3, while the decision actions are: d_1, d_2, d_3, d_4, corresponding to each object class. **The transition function**, T, is made up of the position transition function and the class transition function, the latter one being deterministic. These transition functions are:

$$T_p(p, m, p') = \begin{cases} P(p'|m, p) = 1 & \text{if } p' = p_i \text{ and } m = m_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$T_c^j(c_i'^j, a, c) = \begin{cases} 1, & \text{if } a \notin A_d \text{ and } c'^j = c^j \\ 1, & \text{if } a \in A_d \text{ and } j \leq H-1 \\ & \text{and } c'^j = c^{j+1} \\ \frac{1}{L}, & \text{if } a \in A_d \text{ and } j = H \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

The meaning of this second function is the following. As mentioned, the robot is concerned only with a limited number, H, of positions from the conveyor belt. A motion action leaves the classes unaltered (first branch). However, when a decision action is executed the first object is removed from the conveyor belt, its class is eliminated from the vector of classes and each of the remaining values is shifted with one position (second branch), except the last one which is sampled from a uniform distribution (third branch); see also Figure 2 for an example. Note that in reality, the class will be given by the true, subsequent incoming object, but since the POMDP transition function is time-invariant, this cannot be encoded and we use a uniform distribution over the classes instead. Also note that this transition function is completely defined for any particular instantiation of the problem, including ours (so we do not need to instantiate it further).

The observation function is defined in a factored way, by exploiting the specific structure of the problem. Instead of maintaining a common function O, we factor it across belt positions $j \leq H$. Each such individual observation function o^j provides an observation $z^j \in \{z_1, z_2, \ldots, z_L\}$, where z_i means that the object at position j is observed to have class c_i. We have:

$$o^j(s', a, z^j) = P(z^j|p', c^j), \quad j \leq H \tag{5}$$

where $P(z^j|p', c_i^j)$ is the probability of making observation z^j from the viewpoint p' just reached, when the underlying class of object is c^j. These probabilities are given by the sensor and classification algorithm used, and can be determined experimentally. The overall observation function O is then defined as the joint probability distribution of observations across all the belt positions.

To identify the observation probabilities for our particular problem, we started by performing experiments in which the robot observes the light bulb from the end of the conveyor belt. After that, the number of observed positions was extended to two. We stopped at two positions (counting from the end of the conveyor belt), because from there on the point clouds of the next positions did not offer useful data to be classified.

In each type of experiment an observation probability distribution was computed experimentally for every vertex of the graph, where several scans were performed for each true class of light bulb. The bulbs, segmented from each scan, were classified, the results being recorded in tables, the observation probability distribution being computed as the fraction of the experiments in which each class (correct or incorrect) was observed. Table I exemplifies a distribution for the "elongated" object (true class), when observing the end of the belt. It shows how likely the algorithm is to classify the class correctly, and how likely it is to misclassify it as a different class. Table II, show the likelihood of observing the different combinations of bulbs on the last two positions of the belt.

Pr(o) / Point	elongated	livarno	mushroom	standard
1	0.6	0.2	0	0.2
2	0.5	0.3	0.1	0.1
3	0.8	0.1	0.1	0

TABLE I: Observation probability distribution when observing a single bulb, when the underlying object class is "elongated".

The reward function is defined as follows:

$$R(s, a) = \begin{cases} r_{max} & \text{if } a = d_i \text{ and } c^1 = c_i \\ -r_{min} & \text{if } a = d_i \text{ and } c^1 \neq c_i \\ -1, & \text{otherwise} \end{cases} \tag{6}$$

For a decision action that matches the correct class of the object from the end of the conveyor belt, a positive reward is given, while for one that does not, a negative reward is given. For any motion action a penalty of -1 is given, representing the energy consumption associated with arm movement.

In our particular case, r_{max} has been chosen 10, while r_{min} is tunable, and in the experiments we study its effect when it takes different values.

Having defined the elements of the POMDP, we move to defining key elements of its solution, first generally, before

Pr(o) / Point	elongated p1	livarno p1	mushroom p1	standard p1	elongated p2	livarno p2	mushroom p2	standard p2
64	0.3	0.6	0.1	0	0.4	0.1	0.4	0.1
43	0.1	0.8	0	0.1	0.5	0.2	0.2	0.1
87	0.2	0.7	0	0.1	0.2	0.2	0.6	0

TABLE II: Probability distribution for two objects being observed, for a "livarno" bulb on the first position and an "elongated" on the second one.

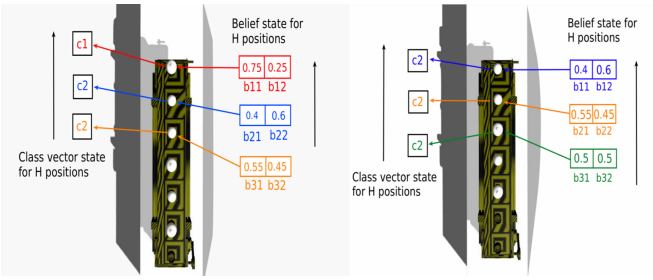

Fig. 2: Belief state and class vector state evolution when propagation occurs, on the left before propagation, while on the right after propagation.

particularizing them to sorting problems. Note that there is no need to instantiate to our particular example; once the POMDP elements has been defined, the objects below follow.

Belief state. Due to lack of access to the internal states, past observations and actions have to be taken into account. These are succinctly represented by a belief state: a probability distribution over all internal states of the POMDP. The belief state is initially state b_0, and it is updated at each step based on the current action a and observation z:

$$b'(s') = \frac{O(s', a, z)}{P(z|s, a)} \sum_s T(s, a, s')b(s) \tag{7}$$

Note that, while $O(s', a, z)$ gives the probability of observing z after reaching state s' as a result of action a, the probability $P(z|s, a)$ is that of observing z after performing a in state s (so one state earlier). The latter probability can be computed from T and O. Denote the set of possible beliefs by B. In the sorting problem, like for the observations, instead of maintaining a joint belief state we factor it across belt positions. So, at each position j, we have a belief state b^j which is a probability distribution over the possible values of the class c^j at that position; b_i^j denotes the individual belief (probability) that c^j is equal to c_i. Once the belt moves, the beliefs across positions are propagated, see Figure 2:

$$\begin{aligned} &\forall i \leq L\,,\; j \leq H \\ &b'^j_i = \begin{cases} b_i^{j+1}, & \text{if } j \leq H-1 \\ \frac{1}{L}, & \text{if } j = H \end{cases} \end{aligned} \tag{8}$$

Note that the belief for the last position is initially uniform, since nothing is known about that object yet.

Policy, value function, and optimal solution. The behavior of the agent (robot) in a POMDP is given by a policy $\pi : B \to A$, which dictates the choice of action for each belief state. The expected return of such a policy from an initial belief state b_0 is called the value function, and is formally defined as:

$$V^\pi(b_0) = E\left\{\sum_{k=0}^{\infty} \gamma^k \sum_{s \in S} b_k(s) R(s, \pi(b_k))\right\} \tag{9}$$

Note that the next belief b_{k+1} obtained as a result of applying action $\pi(b_k)$ in b_k can be computed with (7).

An optimal policy π^* is one that maximizes the value function for any initial belief. For the sorting problem, due to the way in which the reward function (6) is defined, this optimal policy will implicitly strike a balance between classification quality and speed in solving the task (i.e. the number of steps after which the robot makes a decision). Quality is evaluated by the correct decision reward r_max and the incorrect decision penalty r_min, while speed is promoted by the penalty -1 at each step where a decision is not made.

A wide range of algorithms to compute (in general, approximations of) the optimal policy exists [17]. Details of how they work are out of the scope of our paper. In principle any of them is suitable, and the specific algorithm used (DESPOT) is pointed out in next section.

III. Hardware, Software and Application Pipeline

This section presents the active perception pipeline, with its structure and function, as well as relevant hardware and software.

The active perception pipeline has two main modules: a detection module, with multiple submodules, and the planning module. The detection submodule is composed of the acquisition, preprocessing and classification submodules. The pipeline has similarities to the one presented in [19], with differences in the classification algorithm as well in as the presence of the classification and planning submodules.

The objects transported on the conveyor belt (differently shaped light bulbs), are detected from 3D scans performed by the robot. The data is form of point clouds [20], and their acquisition, loading, etc. are handled by the acquisition module. The noisy acquired data, directly influences the percentage of misclassifications. Cleaning, filtering, clustering, and voxelizing the data in advance helps in reducing the percentage of misclassifications.

The input of the classification module is a prepared cloud corresponding to a candidate bulb. The classification process involves a training step, executed before the start of the experiment, implying the computation of Viewpoint Feature Histograms (VFH)[21] for each cloud. The classification is a nearest neighbour search for the closest cloud.

The planning module has the role of finding a good sequence of observations to improve the likelihood of a proper sorting for a candidate light bulb. This module returns

an action to be executed by the robot platform. The POMDP problem is solved using the DESPOT algorithm [5], in an online manner that interleaves the planning and plan execution stages, updating the belief state with the results of the detection module.

Hardware and software. A Baxter research robot has been used for both simulated and real experiments, with an ASUS Xtion 3D sensor mounted on one of its wrists. A conveyor belt, transporting different light bulbs, was placed in front of it. Solvers specific to the robot platform were used for arm motion planning tasks, while the PCL (Point Cloud Library) was used to handle the clouds acquired by the sensor. On the software side, the pipeline was implemented in C++ and Python, in a ROS (Robot Operating System) compatible fashion. Gazebo was used for simulation purposes.

IV. Experimental results and discussion

In this section the results of the experiments are presented and discussed. We performed two experiments, respectively in the single- and multiple-position cases. In both, the robot had to sort ten light bulbs. A light bulb was considered sorted when a decision action was taken, at which moment the last light bulb was taken down and the conveyor belt advanced. The experiment stopped once the ten bulbs were sorted, meaning that ten decision actions were executed. The overall classification percentages for the experiments were computed manually. In the case of the multiple-position experiment, the values from the belief state were propagated. In all the experiments, only bulbs of the "elongated" and "livarno" classes were used for testing (the other two classes used for training, "mushroom" and "standard", were not used).

Simulation results. In Figure 3, the belief state evolution for the single-position case can be seen. The underlying state is chosen randomly by the algorithm (e being "elongated", l being "livarno"). The steps at which decisions are taken are also plotted (d_e for "elongated" decision and d_l for "livarno" decision). Because only the light bulb from the end of the conveyor belt is of importance, there is no belief propagation and after each decision the belief state is reinitialized to uniform values.

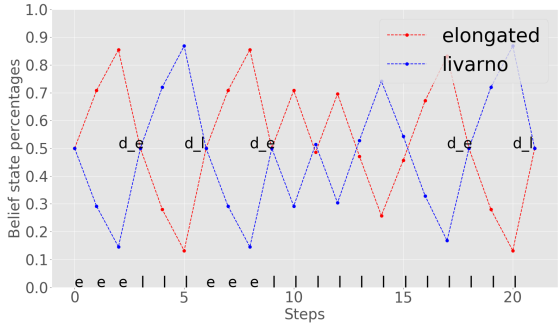

Fig. 3: Belief state evolution for single-position type experiments when sorting light bulbs.

In Figure 4, for the same experiment, we present the ratio of positives (that is, correct classifications) to negatives (incorrect classifications), as the incorrect classification penalty r_{min} varies in absolute value from 5 to 500.

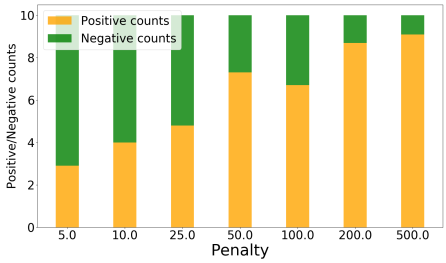

Fig. 4: Positives–negatives counts for single-position type experiments.

Now, we focus our attention on experiments where the robot observes multiple positions at once, propagating belief state values and the vector of classes after each decision. Note that if the belief state of the first position after propagation already has a clear candidate class, then decision actions may follow one after the other, showing the advantage of observing two positions in the same time.

Figure 5 presents the ratio of positives to negatives, for the same range of rewards as above, but now for the multiple-position experiment.

Figure 6 compares the total number of steps required to sort 10 light bulbs, for the single- and multiple-position experiments. Here, the steps counted consist of both the motion and decision actions.

Discussion. First, we examine the effect of the reward values. These have an important impact on the performance of the sorting task, and must be chosen to trade off the quality of classification against the total number of steps required. For a small penalty, the sorting is finished fast, within a few number of steps, however, the percentage of false classifications is high. When the penalty is decreased, as seen in Figures 4 and 5, the rate of the false classifications of sorting decreases, while at the same time increasing the total number of steps needed to carry out the sorting procedure for the whole order of 10 light bulbs.

Next, we compare the case where one position is observed with the multiple-position case, Figures 4 and 5. The evolution of classification quality with the reward value is similar for the two cases. However, a significant difference arises between the total number of steps needed to finish sorting: the multiple-position version requires a significantly smaller number of steps to achieve the same performance as its single-position counterpart, see details in Figures 7, 6.

Experiments on the real robot. The same type of experiments were carried on the real setup as well (refer to Abstract for a video link). The results were similar to the simulated ones, with the actual number of positive classifications being lower due to higher noise corruption. Considering first the impact of the penalty, classification quality and required steps number both grew with the magnitude of the penalty. E.g., for $r_{min} = 5, 50, and 500$, on average the number of positives was 2, 5, and 8 respectively. The number of steps grew from

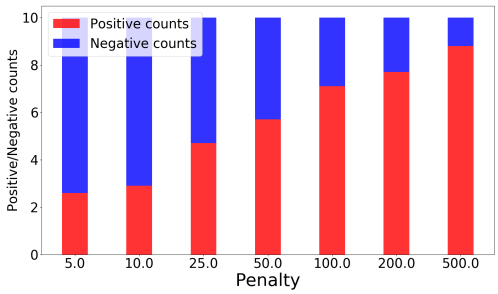

Fig. 5: Positive–negative counts when observing two positions from the conveyor belt.

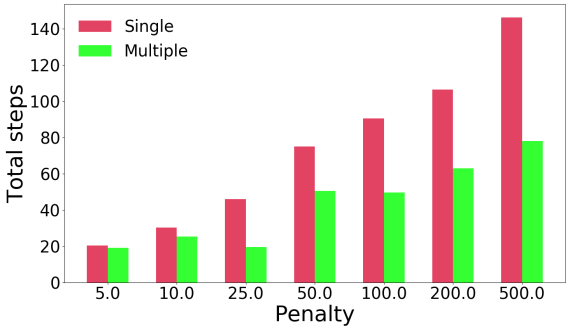

Fig. 6: Comparison between the single- and multiple-position experiments regarding the total number of steps needed to sort 10 light bulbs.

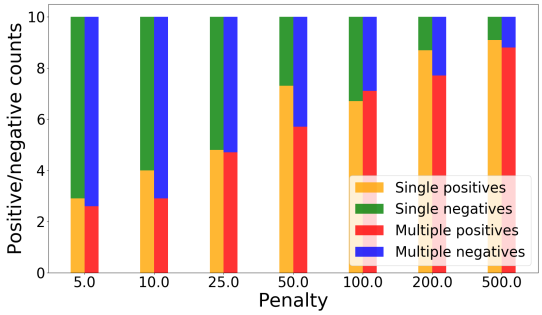

Fig. 7: Comparison between the single and multiple-position experiments regarding the count of positive/negative classifications.

20, through 52, to 70 for the same three values of the penalty. Secondly, for the impact of the number of observed positions (one versus multiple), fixing e.g. r_{min} to 50, the number of steps decreased from 75 to 42.

V. Conclusions

In this work the task of sorting objects transported using a conveyor belt was handled using an active perception approach. The proposed pipeline uses 3D data to classify the objects and computes the actions using a planner, after describing the problem as a POMDP. The functionality of the pipeline was tested both in simulation and on a real Baxter robot equipped with an Asus 3D sensor. A key point of the approach is that it propagates information across multiple positions of interest from the conveyor belt, thus reducing the total number of steps needed for sorting.

In future works, tracking of the objects could be incorporated in the pipeline to reduce measurement errors for a constantly moving conveyor belt. We will also investigate the inclusion of more informative rewards, such as those based on the expected information gain of each observation.

References

[1] C. Militaru, A.-D. Mezei, and L. Tamas, "Object handling in cluttered indoor environment with a mobile manipulator," in *Automation, Quality and Testing, Robotics (AQTR), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

[2] C. Militaru, A. D. Mezei, and L. Tamas, "Lessons learned from a cobot integration into MES," in *ICRA - Recent Advances in Dynamics for Industrial Applications Workshop*, Singapore, 2017.

[3] L. Tamas and L. Baboly, "Industry 4.0 – mes vertical integration use-case with a cobot," in *ICRA - IC3 Workshop*, Singapore, 2017.

[4] E. Páll, L. Tamás, and L. Buşoniu, "Analysis and a home assistance application of online aems2 planning," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 5013–5019.

[5] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," in *Advances in Neural Information Processing Systems 26*, vol. 2, 2013, pp. 1772–1780.

[6] R. Bajcsy, "Active perception," *The Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, 1988.

[7] J. Aloimonos, I. Weiss, and A. Bandopadhay, "Active vision," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 333 – 356, 1988.

[8] T. Patten, M. Zillich, R. Fitch, M. Vincze, and S. Sukkarieh, "Viewpoint Evaluation for Online 3-D Active Object Classification," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 73–81, 2016.

[9] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4775–4782, 2015.

[10] J. Aleotti, D. Lodi Rizzini, and S. Caselli, "Perception and Grasping of Object Parts from Active Robot Exploration," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 76, no. 3-4, pp. 401–425, 2014.

[11] A. Cowley, B. Cohen, W. Marshall, C. J. Taylor, and M. Likhachev, "Perception and motion planning for pick-and-place of dynamic objects," pp. 816–823, 2013.

[12] N. Atanasov, B. Sankaran, J. Le Ny, G. J. Pappas, and K. Daniilidis, "Nonmyopic view planning for active object classification and pose estimation," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1078–1090, 2014.

[13] R. Eidenberger and J. Scharinger, "Active perception and scene modeling by planning with probabilistic 6D object poses," pp. 1036–1043, 2010.

[14] S. Candido and S. Hutchinson, "Minimum uncertainty robot navigation using information-guided pomdp planning," *2011 IEEE International Conference on Robotics and Automation*, pp. 6102–6108, 2011.

[15] M. Cognetti, P. Salaris, and P. Robuffo Giordano, "Optimal active sensing with process and measurement noise," in *IEEE Int. Conf. on Robotics and Automation, ICRA'18*, Brisbane, Australia, May 2018, pp. 2118–2125.

[16] B. T. Hinson, M. K. Binder, and K. A. Morgansen, "Path planning to optimize wind observability in a planar uniform flow field," 2012.

[17] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, "Online planning algorithms for pomdps," *J. Artif. Int. Res.*, vol. 32, no. 1, pp. 663–704, 2008.

[18] M. Deserno. (2004) How to generate equidistributed points on the surface of a sphere. [Online]. Available: https://www.cmu.edu/biolphys/deserno/pdf/sphere_equi.pdf

[19] A.-D. Mezei and L. Tamas, "Active perception for object manipulation," in *Intelligent Computer Communication and Processing (ICCP), 2016 IEEE 12th International Conference on*. IEEE, 2016, pp. 269–274.

[20] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)."

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