POMDP approach to robotic sorting and manipulation of deformable objects

Pol Monsó Purtí

pmonso@iri.upc.es

Advisors:

Dr. Guillem Alenyà Ribas
galenya@iri.upc.es

Dra. Carme Torras Genís
torras@iri.upc.es

Institut de Robòtica i Informàtica Industrial

Master in Artificial Intelligence (MIA)

Conjointly brought by

Universitat Politècnica de Catalunya (UPC)

Universitat de Barcelona (UB)

Universitat Rovira i Virgili (URV)

September 14, 2011

"Whatever mankind invented to the present day
is ridiculous compared to the life of a beetle."
"The important thing is not to stop questioning."
– *Albert Einstein* (1879-1955)

Abstract

The object manipulation with robots has mainly relied on precise, expensive models and deterministic executions. Given the great complexity of modeling deformable objects accurately, their manipulation remains an open research challenge. This thesis proposes a probabilistic approach to deformable object manipulation based on Partially Observed Markov Decision Processes (POMDP) where the action and perception deficiencies are compensated through interaction planning that efficiently leads to uncertainty reduction. Hence, we will prove that it is possible to achieve proposed goals through a simple, inexpensive set of actions and perceptions. The results of the thesis have been applied to a cloth sorting task in a real case scenario with a depth and color sensor and a robotic arm.

Keywords: POMDP, planning, deformable, robot, human interaction, artificial intelligence, uncertainty .

Abstract

Fins l'actualitat, la manipulació d'objectes deformables mitjançant robots s'ha basat principalment en models complexos i mil·limètrics, acompanyats d'estratègies de manipulació deterministes. Donada la gran complexitat que comporta modelar objectes deformables de manera acurada, la seva manipulació continua irresolta. Aquesta tesi proposa una aproximació estadística a la manipulació d'objectes deformables basada en Processos de Decisió Markovians Parcialment Observables (POMDP), en la qual les deficiències d'acció i manipulació siguin compensades per la planificació d'una interacció que, a la pràctica, porta a la reducció de la incertesa inherent. Així, demostrarem que és possible assolir els objectius mitjançant accions i comportaments simples i computacionalment poc exigents. Els resultats d'aquesta tesi s'han aplicat a la tasca d'ordenar peces de roba en un context real d'un robot amb un sensor de color i profunditat i un braç robòtic.

Paraules clau: POMDP, planificació, deformable, robot, interacció, intel·ligència artificial, incertesa .

Acknowledgements

Thanks to the *Institut de Robòtica i Informàtica Industrial* for bringing the opportunity of working with state of the art robotics in such a enlightening and productive environment. I am specially thankful to the other participants of this project Eduard Serradell, Adrián Peñate and Arnau Ramisa and emphatically grateful for the guidance of Dr. Guillem Alenyà and Dra. Carme Torras, who have encouraged the synergies between coworkers and supervised such ambitious, holistic thesis which would not have been possible otherwise.

I am also very grateful to my family and friends, for their anytime support, specially through the dead ends and pitfalls of research, without whom the determination of modestly and slightly contributing to the world's knowledge would be, above all, pointless.

Contents

1 Introduction

The study of rigid object robotic manipulation has remarkably advanced recently due to the successful mechanization of industrial activities. While the manipulation of rigid objects has been widely assessed, deformable objects are still manually processed due to the fact that handling deformable objects usually requires prior knowledge, visual and tactile information, complex models and/or the interaction of several dexterous manipulators. Achieving proper manipulation on flexible objects is a key factor for robotic and industrial development in fields such as textile and food industry and service robotics.

Unfortunately, the manipulation of objects in real situations raises important issues that prevent the proper execution of the actions theoretically estimated. State of the art perception and manipulation is not robust, lightweight enough to provide reliable model-based, deterministic solutions to real world problems. Indeed, inaccurate sensory perceptions that are affected by noise, carried out in an indeterministic environment constitute a hostile scenario. In this context, planning has to be capable of coping with the perception deficiencies and the environment uncertainty in order to accomplish the tasks assigned, which should be achievable even with deficient perception and manipulation.

This master thesis addresses planning algorithms for manipulation of deformable objects, acting as a bridge between modeling and planning. The high-level task specifications have to be integrated with the low-level geometry extracted from simple physical models. The proposed method also takes into account the uncertainty of the sensors, movements and the general scenario in order to plan through a probabilistic sequence of motion commands. To that end, we explore several techniques of artificial intelligence, learning and planning to suit the particularities present in deformable objects.

The objectives and expected results of this thesis are part of three currently active projects at the Institut de Robòtica i Informàtica Industrial, which are directly related to the topic. First, the national project *PAU - Perception and action under uncertainty*, establishes among its objectives the manipulation planning of planar, deformable objects such as paper or clothing. The appropriate perception processes have to be defined in order to maintain these models and specify the set of possible actions. Secondly, the project *MIPRCV: Consolider-Ingenio: Multimodal interaction in pattern recognition and computer vision*, given the multi-sensorial input integration required for the manipulation of deformable objects. Finally, *APREN: Perceptive models and learning techniques for service robotics*, which complements the techniques developed by the other two projects with learning and planning capabilities, which are the focus of the goals of the conducted Master Thesis.

2 Goal Description

Given the uncertainty and variability of the deformable objects, it is difficult to obtain a general method for handling such objects, specially when manipulation relies on a model-based approach [13]. The goal of this work is to propose a statistical approach to cope with inefficient vision and grasping systems and reach the goal nonetheless. In particular, the task of the method is to sort deformable objects apart one by one. The approach proposed abstractly handles the specific characteristics of deformable objects through general probabilistic models, whose probability is later statistically established.

As first goal of the thesis we provide an abstract representation of the object sorting task independently of the particular domain features. The representation assumes the scenario has a set of

actions to manipulate the objects. The manipulation is understood in the lighter sense: moving the objects around and simple folding and spreading sequences meant to provide novel views of the objects, assisting the planner to correctly estimate the actual number of deformable objects really present.

In fact, the latter claim stands on the idea that object recognition can be highly improved through interaction. We introduce general planning techniques based on simple action models whose specific implementation is domain-dependent in order to successfully sort the objects individually. In fact, the planning remains decoupled from the underlying actions and perceptions, making little assumptions about how the required information is gathered and the actions performed.

As the second goal, we will apply the representation to a real case scenario where a robotic arm has to sort several T-shirts with the help of a three dimensional camera. In that scenario, we will provide the concrete manipulation and robot vision capabilities necessary, based on artificial intelligence techniques (Sections 8 and 9), while taking advantage of the deformable features of the objects. Figure 1 shows the scenario settings, where the robotic arm has grabbed a T-shirt before reaching the destination box.

Figure 1 – Experimental setting

We are conscious that a general approach for object sorting under uncertainty is expected to perform worse than ad-hoc solutions, and the presented methods are just an initial exploration of the feasibility of such approach. On the other hand, little has been researched regarding the application of statistical planning to deformable object manipulation to our knowledge. In the following Section we review briefly the state of the art on the topic.

3 State of the Art on Deformable Object Sorting and Robot Manipulation

As previous sections have pointed out, recent research on deformable object manipulation mainly focuses on defining reliable models of deformable objects to deterministically foresee how to bring the deformable object from one configuration to the required one [13, 34, 3, 31, 19, 2]. Maitin-Shepard, for example, successfully managed to fold towels given the assertion of the correct grasping points, followed by a generic, ad-hoc folding sequence. Matsuno introduces a topological tree of a knotting pair of grippers to reach from one configuration to the other [20]. Balaguer et al. point out the simplicity really required for object manipulation as opposed to complex object models and propose a translation between model-based trajectory points of a specific deformation and the configuration space, followed by using Dijkstra's algorithm for path selection [2].

Building such models is highly dependent on the physical abstraction of every object; a classical example of deformable model used is the mass-spring model (Figure 2). Mass-spring models are made from a lattice of masses and springs obeying mechanical forces through mass, damping and stiffness; inherent properties of the object handled that have to be learned. Therefore, extensive knowledge of the objects handled is required, which is excessive when dealing with simple tasks or relaxed goals, goals especially present in service robotics and human-robot interaction.

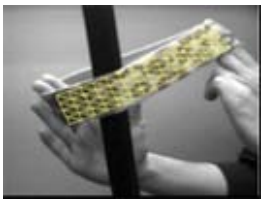


Figure 2 – Virtually superimposed example of a mass-spring model to an elastic tape. The deformation of the object should be precisely and continuously known through the mass-spring model planned manipulation.

Moreover, the decoupling between theoretical models and reality becomes critical in such situations, setting hard constraints upon the domain of applicability and its extension to broader domains. In addition, the required extensive control of the manipulation set demanding error and uncertainty rates that are difficult to meet in practice.

In the following sections we propose a probabilistic approach based on Partially Observable Markov Decision Processes (POMDP) that would relax these constraints. A POMDP-based design and implementation has three major aspects: defining the state space description, obtaining the transition and observation model and computing the policy. The three branches constitute a huge challenge and much research is conducted at the moment to improve the efficiency, adaptability and flexibility of each of these three steps.

In robotics, POMDP formulation is commonly used for map localization and path planning. Kurniawati [16] extends state of the art in motion through roadmaps in both two and three dimensional spaces, introducing sensory uncertainty to the policies with POMDP models. Similarly, partially observable models have been proposed for object tracking in sensory ubiquitous scenarios [9] and also for object pose estimation through most informative viewpoint sequence selection [7].

The formulation, however, is very flexible and can be applied to a broad amount of scenarios. Beyond the classical enumeration of fields made by A. Cassandra in 1998 [5], we can find many other practical examples in the literature; Jung [14] introduces partial observability to dynamic energy management to improve the power usage, given that consumption depends to a huge amount of factors that prevent scalability. A similar problem as the classical Tag, POMDPs can be used for designing band usage policies of the recently proposed cognitive radios [18], where non-licensed users profit the free spectrum bands when licensed users are off-line. Cognitive radios suffer from identity thief attack, as its emission could be deliberatively interrupted by other users simulating the licensed user behavior, hence the uncertainty.

Moreover, biology has also taken advantage of such models to control plant invasions [27] or land planning in endangered species habitats [37]. Also, human-computer interaction has applied the POMDP framework for many different purposes, such as brain-computer interfaces pattern modeling [24], speech processing [15] or experimental sign language communication [23].

The methods proposed are of great interest in the literature, and recent improvements on policy computation speed of POMDP solver's (reviewed at Section 6) give credit to the potential of the planning framework, even under suboptimal policies. The following section provides a brief theoretical insight of the MDP formulation and its extension to partially observable domains.

4 Proposed Methods

As we introduced in the previous sections, we propose a POMDP-based problem definition to cope with uncertainties of deformable object sorting and ease the action and perception constraints. Using a POMDP-based approach has several planning advantages like flexibility and adaptability, although imposes design constraints due to dimensionality issues. On the other hand, establishing hard dimensionality constrains also indirectly encourages generality, which leads to flexible, yet sub-optimal planning policies. Indeed, the available information selection establishes the trade-off between informative state-space and computationally feasible policies.

Using POMDP problem framework requires the development and design of the three major aspects of the POMDP definition: the model design, its probability density function estimation and the policy computation, additionally to a set of mechanical actions and observations that have been developed within the present thesis. The following sections describe our approach to each of the topics; the theoretical background in Sections 5 and 6, the model design in Section 7, Sections 8 and 9 describe the particular manipulation actions and computer vision observations and finally Section 10 explains how the model is estimated.

5 Partially Observable Markov Decision Processes (POMDP) Description

In the following subsections we will introduce the theoretical background of the Partially Observable Markov Decision Processes framework. By expressing the problem that way, POMDP solvers can then obtain the planning policy that will drive the robot actions through the uncertainty to the goal. The following Section describes the mathematical background of the framework, starting with the observable case, namely Markov Decision Process (MDP), followed by the partial observability case.

5.1 Markov Decision Processes

Markov decision processes rely on the Markov Property; the transition between any pair of states s and s' only depends on the state s and the action performed, and not on how state s was reached . If the state representation we have designed fails to hold this property, then the system is not Markovian and the policy obtained by any method that relies on the property won't necessarily behave as expected.

Formally, this property is stated as

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, \ldots, r_1, s_0, a_0\} = Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\}, \quad (1)$$

where s, a, r stand for state, action and reward, and t represents any time/transition step. Altogether, MDP are described by the tuple $< \mathcal{S}, \mathcal{A}, T, R >$. The set of actions is the robot's means of interaction and changing the world, whose representation is established by the robot's internal set of states.

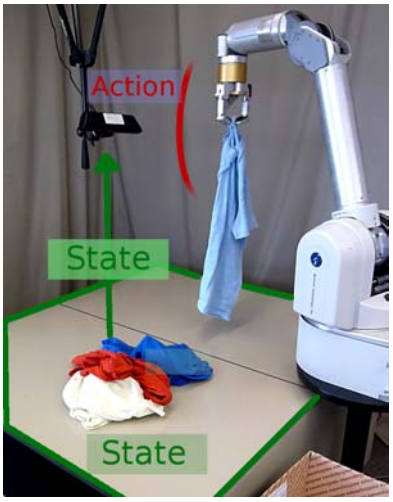


Figure 3 – MDP abstraction of the task. The robot moves visits the states by performing actions following a policy. The best action to be made only depends on the information provided by the current state, but not on the previous history of states.

Defining the state is the most difficult task and it is extremely domain-dependent, since the encoding must have the relevant and informative features of the problem. There are techniques handling the encoding issue by state clustering and classification, using Neural Networks or other learning methods [6]. Those techniques also tackle the dimensionality problem, especially relevant if the transition model must be provided. The issue, however, has not been solved satisfactorily yet.

Once the designers have defined a description of the state, a graph of transitions with its associated probabilities models the entire world. How the agent discovers and traverses this graph to achieve our goals defines different approaches to the solution of the planning problem.

One of the most popular approaches is Reinforcement Learning, which is able to learn the model of the world by an alternate use of exploration and exploitation, the former trying out sub-optimal actions based on an approximate model of the world and the latter following greedy policies, which assumes we have a sufficiently reliable representation of the world. Later, Watkins developed Q-learning[38], a formulation that updates each action-value without storing all the history of trials explicitly and

without requiring a previous model of the environment, independently of the policy followed, in order to avoid the need of representing all the states.

The backup step of a one-step Q-learning,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right], \tag{2}$$

updates the stored value of the state given the new information gathered. α sets the importance we give to the observed reward and therefore sets the learning rate of the algorithm. For example, let $\alpha = 1$, then (2) is rewritten as

$$Q(s_t, a_t) \leftarrow r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \tag{3}$$

which gives all the credit to the new observed reward and the subsequent maximum reward expected from the following states. Note that the expected maximum reward of the whole future history is kept in the subsequent state Q-value, hence the update is extremely fast to compute.

Of course, $\alpha = 1$ is a naive setting for the algorithm for illustrative purposes, as usually $0 < \alpha < 1$ and its exact value is domain-dependent. Notice that its value can also depend on the state-action pair if the domain rewards are asymmetrically relevant for the learning.

5.2 Partial Observability

One of the limitations of the Markov Decision Processes (MDP) representation when it comes to robotics is the assumption that we can observe the state completely, which is unrealistic in many problems due to the noise present in sensors, modest Receiver Operating Characteristic (ROC) curves' optimal points performance of state-of-the-art image processing algorithms, as well as occlusions and particular problem features. The partial observability approach tackles this problem, but with two drawbacks: The dimension increase due to the observations addition and, since the state is not perceived, typically the designers have to code all transitions and associated probability beforehand.

So, with partial observability, the original MDP tuple $< \mathcal{S}, \mathcal{A}, T, R >$, where

- $\mathcal{S}$ is the discrete state space.
- $\mathcal{A}$ is the action space.
- $T : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$, the state transition function.
- $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, the immediate reward function.

gets extended to $< \mathcal{S}, \mathcal{A}, T, R, \Omega, O >$, where Ω is a set of observations and O the probability distribution of the observations according to the states and actions.

Compared to fully observable models, the POMDP model assumes that the robot is unable to perceive completely the world state, but it rather obtains a partial percept of the real world (Figure 4). Every time the robot performs an action, it retrieves an observation that provides information about

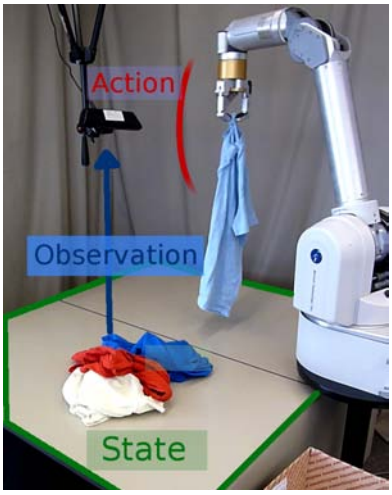


Figure 4 – In a partially observable context, the POMDP task abstraction gathers observations instead of the real state information.

its current context. The approach is statistical, maintaining a probability distribution over the state space, which constitutes the robot's belief of the world state over time.

The biggest issue of partial observability domains is the problem not being Markovian over the state space any more; indeed, now the best action depends on the previous history of observations. Fortunately, it is Markovian over the –continuous– belief space, and a fixed mapping exists for an infinite horizon

$$\pi : \mathcal{B} \mapsto \mathcal{A}. \tag{4}$$

Aström has shown that a properly updated probability distribution over the state space $\mathcal{S}$ is sufficient to summarize all the observable history of a POMDP agent without loss of optimality [1].

Therefore, a POMDP can be cast into a framework of a fully observable MDP where belief states comprise the continuous, but fully observable, MDP state space. A belief state MDP is therefore a quadruple $\langle \mathcal{B}, \mathcal{A}, T^b, R^b \rangle$, where

- $\mathcal{B} = \triangle(\mathcal{S})$ is the continuous state space.
- $\mathcal{A}$ is the action space, which is the same as in the original MDP.
- $T^b : \mathcal{B} \times \mathcal{A} \mapsto \mathcal{B}$ is the belief transition function.

  Given an observation z after applying action a obtained from the current belief b, the probability

$b_z^a(s')$ of the robot reaching state s', obeys the equation

$$\begin{aligned} b_z^a(s') &= Pr(s'|b,a,z) && (5)\\ &= \frac{Pr(s',b,a,z)}{Pr(b,a,z)} && (6)\\ &= \frac{Pr(z|s',b,a)Pr(s'|b,a)Pr(b,a)}{Pr(z|b,a)Pr(b,a)} && (7)\\ &= \frac{Pr(z|s',a)Pr(s'|b,a)}{\sum_{s,s''} Pr(z|b,a,s,s'')Pr(s,s''|b,a)} && (8)\\ &= \frac{Pr(z|s',a)\sum_s Pr(s'|b,a,s)Pr(s|b,a)}{\sum_{s,s''} Pr(z|a,s'')Pr(s''|b,a,s)Pr(s|b,a)} && (9)\\ &= \frac{Pr(z|s',a)\sum_s Pr(s'|a,s)Pr(s|b)}{\sum_{s,s''} Pr(z|a,s'')Pr(s''|a,s)Pr(s,b)}. && (10) \end{aligned}$$

If computed for each state s', we obtain a probability distribution otherwise known as *belief state*.

- $R^b : \mathcal{B} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function

$$R^b(b,a) = \sum_{s \in \mathcal{S}} b(s)R(s,a) \quad (11)$$

To follow the policy that maps from belief states to actions, the agent simply has to execute the action prescribed by the policy, and then update its probability distribution over the system states according to (10).

The infinite horizon optimal value function remains convex, but not necessarily piecewise linear, although it can be approximated arbitrarily close by a piecewise linear and convex function[36]. The optimal policy for infinite horizon problems is then just a stationary mapping from belief space to actions (Equation 4).

Summarizing, first, we implement the POMDP tuple, usually starting from the state description. Then, the robot manipulations and sensor input has to be coded as actions and observations relative to the state description. Next, to complete the POMDP tuple, we compute or estimate the transitions of every state and the probability distribution of the observations related to each action-state pair. Finally, the reward is assigned to the goal state or distributed heuristically to guide the next step: solving the POMDP.

The solving of the POMDP explores the belief space and obtains a policy, the mapping between states and actions, expressed as value functions of each action given each state. Finally, during the execution, the policy matrix is multiplied by the probability distribution across the states at the given time – the belief– to obtain the value of each action. The robot manipulation executes the most valuable action and, upon finalization, the robot vision retrieves an observation to update the belief.

The action-observation-update is iterated until the final goal has been reached with enough certainty, effectively achieved either by exploiting or by reducing the uncertainty of the process.

5.3 A Simple Example

The POMDP description framework is very flexible and can describe a great variety of problems. For example, Pascal et al. supervise and assisted an Alzheimer patient to follow a sequence of actions and adapting to his mistakes [25]. As an illustrative example, we simulated a cook assistant, a robot that handles ingredients to a chef in order to cook a receipt. The robot has to guess which receipt is the cook performing and try to anticipate his moves, taking into account that while offering an ingredient the cook may either take it or not, depending on whether he needs it or not, but also might fail to take it because he is distracted or busy at the moment.

The robot has several receipts but doesn't know a priori which one the cook is elaborating, so the solver must find the correct policy upon what ingredient to give in each state in order to minimize the number of mistakes and obtain the correct receipt as soon as possible.

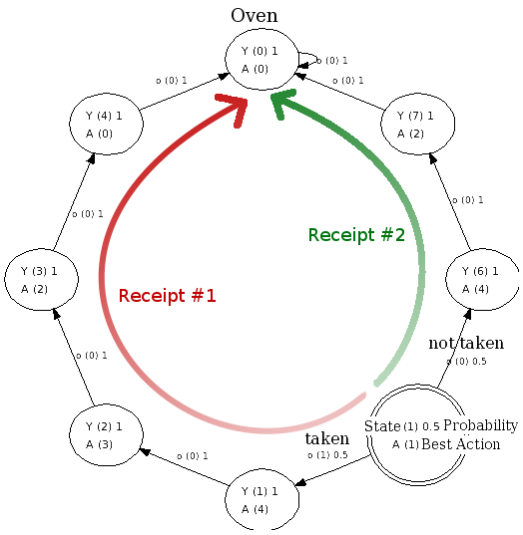


Figure 5 – Example of a computed policy for a cook assistant with two receipts and an action branching of two, where state 0 is the oven and 1 is the start state. Every action is an ingredient and we observe if the cook has taken it (1) or not (0). At the beginning of the execution, the robot does not know which one of the two receipts the cook is cooking. The robot gives one of the two expected ingredients to the cook and thereby, depending on the observations, we assert one receipt or the other.

First, we establish the state definition: a receipt identifier and the step number within the receipt. The robot giving each ingredient to the cook are the actions, and the sole observation is whether the cook takes or not the given ingredient. Now, the transition probabilities consider that the cook might not be ready to take the ingredient, thus a certain probability that after giving the appropriate ingredient to the cook, he does not take it and the current state remains unchanged. Additionally, the cook can make mistakes; accepting the incorrect ingredients, so there's a certain probability of observing the ingredient as taken, while, in fact, it was brought back by the cook as he noticed the mistake made. In this context, the transition and observation probabilities can be simulated assuming the cook's error and readiness rates, completing the POMDP tuple. The reward is given once a receipt reaches the oven, which is the final destination of every receipt.

Once the POMDP description is completed, the solver takes the transition probabilities and starts the simulation of propagating the belief from the starting states over the different possible outcomes of

each action. The members of the starting states set are the first step of any possible receipt, thus the initial belief is an uniform distribution over the set. The rewards are not delivered until the final state is reached, whose probabilistic path across the state space is unknown a priori. For every possible receipt, the solver will keep propagating the belief tree until it reaches the oven, hence finishing the receipt and obtaining the reward. At the end of the process, each state will have an associated value per action, which, combined with the probability density function over the states (the belief) tells the planner which action is the one that provides the greater reward given the belief. In particular, in approximate solvers, every state has an associated action at any time, refining the value over iterations. Finally, the solver provides the computed policy as a reward expectation of each action given a state. For this particular example, Figure 5 shows the most probable part of the policy graph obtained, with the most valuable action at each state, which, as expected, is giving the expected ingredient at the given state. Note that the innovative part of the policy obtained is not the value of

As a result of the POMDP solving, we find that the policy tries to enhance the common sequences between receipts, given that they are a win-win situation, and follows a quick systematic disambiguation process when the observations account for an alternative path. Another interesting behavior is the offering of the most profitable ingredient in case of total uncertainty upon the state. In general, the policy chooses either the ingredient most voted by each receipt or the less voted, in order to either keep advancing through the receipt steps or disambiguate between several receipts.

In a nutshell, the policy either tries to find out which receipt it is to reach the goal faster or proposes the ingredient that will have the most success given that the robot does not know exactly which receipt the chef is cooking. Balancing the two approaches depends on the reward distribution and probabilistic propagation of the beliefs, based on the maximization of the reward.

6 Solver Algorithms

Solving a POMDP is an extremely difficult computational problem that has been addressed mainly from model-free to model-based learning and from exact to approximate approaches. The exact solution emerges from MDP solving algorithms, with the belief state space of a POMDP as a continuous MDP. Many solving algorithms have tackled the dimensionality problem of POMDP: Mohanan Enumeration, Sondik's One-Pass Algorithm, Cheng's Linear Support, Littman Witness algorithm, Zang's Incremental Pruning, ... More recently, point based approximate algorithms have increased the dimensionality limits of approximate solvers [26].

Additionally, building the POMDP model can be a difficult or even impossible task, which is a handicap of model-based POMDPs. Moreover, the partial observability nature of the POMDP problems makes it specially complex to explore and discover the state space, let alone find the relevant, informative variables that should represent the world state space in a low-dimensional, tractable subspace, since the state itself is not observable, but rather an ambiguous function of it is only available.

Nevertheless, history-based algorithms such as Uttile Suffix Memory [21] are capable of discovering sequences of observations and establish an underlying state-space model. Alternatively, the Baum-Welch algorithm applied to Hidden Markov Models assumes that we do not have control over the state transitions and therefore it can be used to obtain the world model rather than moving across it.

The methods proposed are specially relevant in problems where optimal policies only go through a small portion of the state-space, so exploring the entire state space is unnecessary. However, the convergence time and dimensionality required to obtain a reliable model of the world is still unusable in real problems, specially due to the fast growth of the belief propagation.

In fact, the exact solving of the POMDP becomes intractable even for small states spaces, despite that, reaching the goal through sub-optimal paths is satisfactory enough to encourage using the POMDP description for planning under uncertainty. To that purpose, there is a growing interest in obtaining on-line policy iteration POMDPs. Hence the motivation behind several approximate algorithms that have been proposed to achieve an incremental construction of sub-optimal policies that converge to the optimal policy, while keeping track of value function bounds. Shani proposed a modification of the backup Perseus algorithm [32] so it could be run on-line, later, Pineau introduced Point-Based Value Iteration (PBVI) approach, that confronts the POMDP resolution locally by selecting a representative set of beliefs, reducing the computational cost and, therefore, being able to lift the dimensionality boundaries up to thousand states. Their results were encouraging and several PBVI-type algorithms have been proposed recently. For example, Ross uses a rough policy as heuristic function to guide the on-line search algorithm, which prevents solver stalling [29], similarly, Heuristic Search Value Iteration (HSVI) combines the curse of dimensionality and curse of history as a heuristic search criteria, being able to handle state spaces with 10^5 states [35]. These approaches are specially advisable when the gross reward return of the policy is reached fast, followed by a slow convergence to refine the policy, with little impact in practice.

Another example –and the one used in this project– is the APPL approximate solver based on the Successive Approximations of the Reachable Space under Optimal Policies (SARSOP) algorithm [17], a key-point based POMDP algorithm that works on a rough representation of the belief space. As other key-point algorithms that start exploring the reachable states from an initial belief space under arbitrary sequences of actions, the SARSOP algorithm attempts to approximate the optimal sequence of actions of this discovery to retrieve the optimal policy without searching the entire belief space. But this optimal search is, in fact, the solution of the POMDP itself, to which the algorithm approaches iteratively, hence the name Successive Approximations of the Reachable Space under Optimal Policies (SARSOP).

At the moment, the input format used by the algorithm is the original Cassandra representation, but Wei proposes an alternative, more compact, XML-based representation of the POMDP; based on SARSOP and dynamic Bayesian networks, Wei takes advantage of the observable part of the POMDPs, if any, to improve the dimensionality constraints [39]. Their results offer a slight amelioration of the SARSOP algorithm thus, given a proper adaptation to mixed observability, might be introduced in the future.

7 State Representation

The key component of a POMDP model is the definition of the state, whose dimensionality has to stand between the computationally feasible yet retaining enough information to reach the goal successfully. Indeed, its generality may lead to simplistic, irresolutive policies when run in a particular domain, therefore additional information should be added in order to exploit the perception information. However, it has to be kept in mind that, one way or another, either the transitional or observation

probability density functions have to be provided to complete the POMDP tuple. The former assuming that such information is not observable but determined through interaction (*transitional*) and the latter that a correlation exists between observations and the additional information (*observational*).

Being that so, we propose a state space that distinguishes between the common definition of any object sorting problem, and its extension with ad-hoc domain features. Then, provided the set of domain-dependent actions and observations, we can reuse the common definition of the model to solve an object sorting problem particular domain.

In our specific context, the state considers two areas of manipulation and a discarding bucket (Figure 6). The state representation chosen counts up to 3 objects present in each area, the other cases are absorbed by a special `>3` state. Note that the number is decoupled from the kind of objects; plastic, food pieces, solid objects or other, since no domain information has been used at this point.

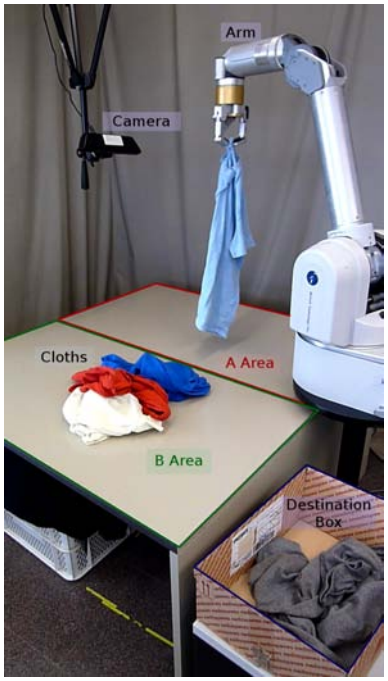


Figure 6 – The spatial distribution of the experimental settings distinguish two working areas and the destination box where the pieces of clothing ought to be placed one at a time.

Intuitively, given the tackled task, the policy followed by the robot is bound to be certain that a single object has been isolated in one of the areas before proceeding with the removal. Since the computed behavior depends on the maximization of the reward obtained across the state transitions, other goals can be easily established by changing the reward distribution and recomputing the policy.

This representation approach provides great flexibility upon the particularities of the domain since it can be complemented with domain ad-hoc information, provided that we can obtain a success model per action and the observation probability density function, if applicable. The estimations of these two probability density functions are not trivial and have to be taken into account while designing the description of the state. In our case, the state description presented in this section has been though through to tackle this problem. Section 10 describes the techniques used to build the model

in our particular case, which are also dependent on how the representation defines the observations and actions.

So, as an example in our context, take the wrinkleness of the grasping point as a feature that could be taken into account if correlate to the action success or the number of pieces of clothing. Then, either the wrinkleness is not observable, and therefore, the success rate of the actions would inform us of the wrinkleness of the object, or it is observable, which would help the planning to choose the most valuable action given the correlation action-wrinkleness.

Provided that dimensionality and the model construction issue are taken into account, any other relevant information other than the wrinkleness can extend the state representation in practice. In later sections we will see that indeed a basic and reduced state space can be sufficient to reach a desired goal, while, given the simplicity of the model, accounts for any kind of deformable object sorting.

Once we have defined the state representation, the observations and actions sets establish how we acknowledge and traverse the state space. The former only requires an estimation of the number of objects present in each area, to that purpose, a robot vision probabilistic graphical model is proposed. The latter provides action both to help improve the observations and to isolate the objects taking into account specific features of the deformable objects. Both observations and actions are exhaustively presented in the following sections.

8 Observations

The observations of the POMDP model only assume that the robot vision algorithms are capable of providing an estimate of the number of objects present in the scene. That way, the observations stay decoupled from the underlying observation technology. In particular, the robot vision module implements a probabilistic graphical model for color segmentation based on the Factor Analyzers introduced by Ghahramani[10]. The error distribution figures obtained within the context of our particular problem are shown at Table 2 of Section 12.

8.1 Camera

The camera used in this thesis is the Microsoft Kinect sensor. The Kinect is an horizontal bar connected to a small base with a motorized pivot and is designed to be positioned lengthwise above or below the video display. The device contains an RGB camera and an infrared pattern light projector with a monochrome CMOS receiver which captures video data.

Figure 7 – Microsoft Kinect RGBXYZ Sensor

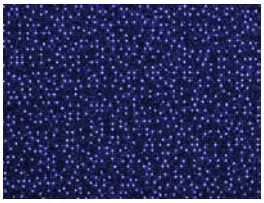


Figure 8 – Infrared, pseudo-random pattern projected by the Kinect. The pattern is then used to obtain the three dimensional information through triangulation.

The light projector emits a hard-coded, pseudo-random pattern (Figure 8) which, compared to the hard-coded pattern provides the 3D information through triangulation.

In 2010, Adafruit Industries offered a bounty of 1000$ for the first to develop an open source driver for the kinect. The bounty was claimed ten days afterwards and the openni driver was ported to ROS briefly after. The ROS Openni driver works with point clouds, a three-dimensional point representation of data broadly spread within the research community [30]. The resolution selected in the current project is 640 x 480 pixels, with slightly smaller point cloud dimensions. The advent of

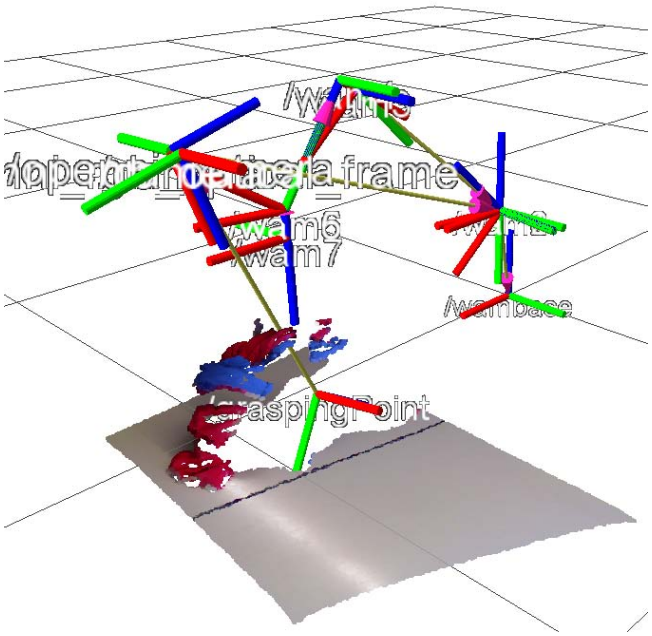
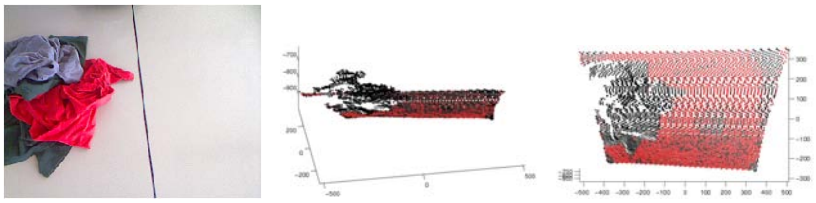


Figure 9 – Cloth imagery in the robot's virtual environment. The three dimensional information provided by the Kinect (left coordinate frame) is shown within the environment with the color information superimposed. The pieces of clothing have been grabbed by the robot, represented as the set of joint frames (right coordinate frame set).

low-cost 3D imagery provides a revolutionary source of computer vision information, since the three-dimensional information inherent of the point clouds (Figure 9) is very informative when it comes to robot manipulation, specially if it is complemented with a calibrated RGB camera.



(a) Original Image **(b)** Top view of the 3D data **(c)** Front view of the 3D data

Figure 10 – Table detection. Points classified as part of the table are depicted in red

8.2 Preprocessing: Background Subtraction

Before performing the segmentation of the image we are going to discard as much irrelevant data as possible. If we obtain the points that correspond to the table we will be able to eliminate them and all the other points that are below the plane that the table defines, taking as reference the camera, leaving only the points that belong to the clothes.

One of the most popular methods to fit a known model into a set of points, given the existence of numerous outliers, is the Random Sample Consensus (RANSAC) algorithm [8]. Using RANSAC we intend to identify the dominant plane, in our case, the table on which the clothes lie. Once the plane that defines the table is found, a distance threshold is applied in the direction of the normal vector of the plane. All the points that fall in this zone will also be subtracted from the image, this is done to assure that no point belonging to the table is left behind. Some points that are part of a piece of clothing will also be eliminated; which has no effect in practice given that the vision goal is to determine the number of pieces of clothing in the scene rather than obtaining an its accurate contour in space.

Our RANSAC implementation uses three points to define the candidate plane, we could use more by applying a least-squares approach but RANSAC converges faster when a minimal set is used. The threshold to consider a point to be inlier is set very low to avoid false positives, we also need a high number of points counted as inliers in order to consider the candidate solution a good one. In Figure 10 we can see the results of using RANSAC to extract the table.

Finally, once the table has been removed, the image is transformed from RGB to L*A*B color space which is more suitable for the color clustering described in the following section as it explicitly splits the space between light and color.

8.3 Clustering

Clustering applies Mixture of Factor Analyzers (MoFA) on the LABXY space. Since not all pixels contain three-dimensional information, MoFA can not be applied to the LABXYZ space straight away.

Factor analysis is, together with Principal Component Analysis (PCA), one of the main tools for informative data projection. The method transforms the data from the original solution space to a more informative space of solutions. Factor analysis is a refinement of the original Gaussian Mixture Models (GMM).

The classical Gaussian Mixture model obeys the equation

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{12}$$

whose graphical representation is shown at Figure 11.

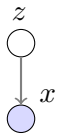


Figure 11 – Graphical representation of a mixture model.

Since the quality of the fit can be evaluated through Log likelihood, which given a Gaussian Mixture Model is

$$\ln p(\boldsymbol{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}, \tag{13}$$

the algorithm consists of a two-step iterative process of evaluation and parameter re-estimation. In particular, following the steps

1. Initialize the Gaussian parameters $\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}$ and mixing coefficients π_k and evaluate the initial value of the log likelihood.
2. Evaluate the responsibilities (inverse conditional probability obtained from the Bayes Theorem)
3. Re-estimate the Gaussian parameters using the current responsibilities.
4. Evaluate the log likelihood and its convergence. Jump to 2.

The generative model for the factor analysis (Figure 12), though, obeys the equation

$$x = \Lambda z + \Psi \tag{14}$$

where Λ are the coefficients of the linear combination of factors that result in the original data vector x and Ψ is a diagonal matrix, whose diagonality represents the factor analysis key assumption: the observed variables are independent given the factors.

Factor analysis also applies dimensionality reduction by expressing a p-dimensional real-valued data vector x as a k-dimensional vector of real-valued factors, z, where k is generally much smaller than p. In our particular case, no dimension reduction is required since the data has only three dimensions, corresponding to each of the L*a*b color space channels.

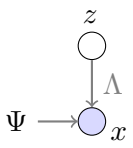


Figure 12 – Factor analysis generative model

In particular, the MoFA uses each factor analyzer to fit a Gaussian to a portion of the data, weighted by the posterior probabilities. These posterior probabilities give the likelihood of each pixel being part of each cluster. In the current implementation, each pixel is assigned to the most likely cluster. A more complex post-processing is also possible with these probabilities, for example, using a Bayesian network to establish whether two clusters are part of the same piece of clothing.

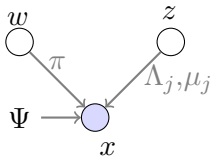


Figure 13 – Mixture of Factor Analyzers generative model

In factor analysis the data is supposed to have zero mean, but in this case each factor analyzer has a different mean, thereby each factor models the data covariance structure in a different part of the input space. Figure 13 shows the generative model of the MoFA, where the mean is taken into account.

Once the MoFA provides the fitting result we remove from the mixture the Gaussians whose contribution is below a certain threshold. The value, namely the *responsibility*, is directly derived from the Bayes Theorem,

$$r_{nk} \equiv \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{15}$$

Now, given that the MoFA's number of Gaussians are not adaptive, the clustering over-fits the image and clusters need to be merged.

8.4 Postprocessing: Cluster Merging

The post-processing algorithm merges the closest clusters based on spatial and euclidean distance within the color space. The filtering is applied iteratively, starting from a strict filter while gradually relaxing the filter after each iteration.

Once the clusters are merged, since the process has a bias towards false positives, clusters below relevant size are discarded, which improves the Receiver Operating Characteristic (ROC) balancing.

Finally, obtaining the number of objects is just a matter of counting the number of clusters. Figure 14 illustrates the three main steps of the algorithm. The performance of the algorithm is depicted at Section 12.

9 Actions

9.1 Robot

The robot used in this thesis is comprised by the Kinect camera, the robotic arm from Barrett Technologies®, Inc. Whole Arm Manipulator (WAM) and the complementary hand BarrettHand™. The WAM arm provides special features specially suitable for manipulation and interaction in human contexts. The characteristics of each device are detailed below.

(a) Original Image **(b)** Clustered Image **(c)** Merged Image

Figure 14 – Image segmentation output images at the three main steps of the proposed algorithm. The final count of objects would provide an observation of three objects on the left and one on the right.

9.1.1 Whole Arm Manipulator (WAM) Robot

The WAM is a manipulator available in two main configurations, 4- and 7-degrees-of-freedom, both with human-like kinematics. IRI's WAM has the seven degrees of freedom configuration, which offers better adaptability and dexterity to grasping constraints. The uniqueness of the WAM arm lies in its backdriven cable drives, similar to human tendons; a design that concentrates the weight at its base and makes the whole arm light enough to have little brake time together with a high acceleration rate and flexibility. This design protects human beings from harm as there is less momentum, which also means fast reaction time on collision. Moreover, its lightness translates into power saving by consuming 28 Watts while enhancing its inherent safety.

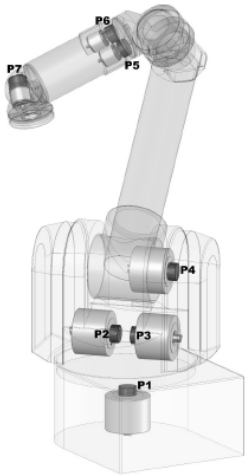


Figure 15 – WAM arm with highlighted *Puck* servomotors

Another advantage of the frictionless backdriven motion is the ability to operate the transposed Jacobian inverse kinematics instead of inverted Jacobian, which allows operation directly in the Cartesian domain [28].

However, there are drawbacks: re-calibration is required once a month due to cable dilation, and accuracy might be worse than for heavier industrial arms. These issues are still unresolved for human interactive robots which have special safety constraints, and the WAM is the best trade-off available at this time.

Intrinsic Safety As described by Brian Rooks [28], the main advantage of the *Puck* servo-controller is the ability to measure the current in the drive motor windings, which can be translated into joint torques and, therefore, externally applied forces. This capacity gives unique features to a manipulating arm, especially concerning safety. If the applied forces over any point of the arm were above the operating joint torque threshold, it would be detected and prevented electronically. This force variation is measured with the power flow within the arm, and requires high precision upon power variations. Given that the steady-state payload is minimized by concentrating the mass at the base, the power consumption is drastically reduced to 28-50 W instead of the usual 1000 W of industrial arms. Usually, the measure of these values are highly distorted by EMI noise, while Barrett Technology®, Inc. surprisingly reduced the EMI effects by bringing the noisy elements together and isolating them with copper-shielding.

On the other hand, the power reduction to a human scale (20-100 W) affects the arm's strength and maximum payload, while it increases dexterity and safety. To put this in perspective, 28 W is roughly what a human arm uses in mild exercise, and this low power is possible in the WAM thanks to the backdriven cables.

9.1.2 BarrettHand

The BH8-series BarrettHand™ (Figure 16) is a self-contained and compact multi-fingered programmable grasper with the dexterity to secure target objects of different sizes, shapes, and orientations. It houses a CPU, software, communications electronics, servo-controllers and 4 brushless motors. Of its three multi-jointed fingers, two have an extra degree of freedom with 180 degrees of synchronous lateral mobility. The BarrettHand™ is designed to overcome the alternative grippers' major limitation: adaptability. Each gripper must be custom designed for each shape and orientation which, unless the host arm will perpetually perform the same task, would need a variable supply of grippers and the ability to switch between them accordingly to targets' shapes and orientations.

Figure 16 – BarrettHand™

The BarrettHand™ matches the functionality of a bank of custom grippers, as its configuration can be changed through its standard ASCII RS232-C serial port using Barrett's Grasping Common Language. Although RS232-C has a slow bandwidth compared to USB or FireWire it provides little latency for small bursts of data, executing and acknowledging commands within milliseconds. The

BarrettHand^{TM}'s firmware resides in the control electronics inside the hand and which is controlled by the WAM's embedded PC.

9.2 Action Definition

The model distinguishes between two sets of actions, disambiguating actions and transitional actions. The former are bound to improve the observations and reduce the uncertainty over the state space and, the latter, to isolate a single object. At the moment, all actions are decoupled from the segmentation algorithm and do not take into account the spatial information provided by the segmentation algorithms, but rely on an independent, standalone algorithm instead. Both sources of information will be taken into account in the future through object filtering to improve the robustness of the actions in order to reduce the grasping variability of the grasping success models (depicted in Table 3 from Section 12). Improving the actions will lead to more effective planning policies, but the thesis does not focus on achieving highly reliable, complex, thoughtful actions, instead, we want to prove that it is indeed possible to reach the goal nonetheless.

Each action has been coded as a scripted set of moves relative to the grasping position, transformed from one frame to the other through the computed hand-eye calibration matrices (Section 9.4) and ROS `tf` tool (Annex A.1).

The disambiguating actions set includes:

- Folding

  Folding sequence of fixed moves includes grabbing a wrinkled point of the object, lifting it, using the friction table against the object while moving fifteen centimeters to the right, moving back while descending in height and finally dropping the object. The sequence is shown at Figure 17.

Figure 17 – Folding sequence. Since we are using a single manipulator, the table friction is used to fold the piece of cloth.

- Spread

  Spreading a piece of clothing implies grabbing a wrinkle point of the object and moving it apart to a fixed position (Figure 18).

- Flip

  Grab the object, lift it and flip it vertically (Figure 19). The action is expected to reveal completely occluded pieces of clothing that might hide behind a certain piece of clothing.

Figure 18 – Spreading sequence. Given a grasping point of the cloth, it is displaced to a peripheral point of the working area.

Figure 19 – Flipping sequence

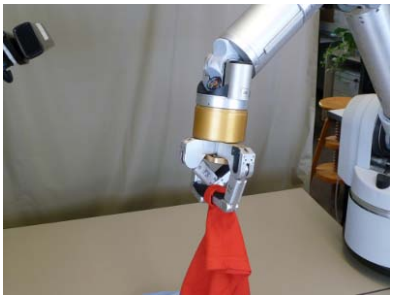
Figure 20 – Shaking sequence

- Shake

  Eventually, pieces of clothing are hidden by or entangled with the grasped piece of clothing, a rapid move from right to left is executed to help separating the pieces of clothing (Figure 20).

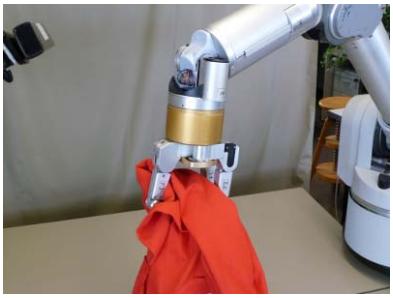
Each action can be executed with three kinds of grasp: peg (Figure 21a), straight (Figure 21c) and isometric (Figure 21e), and at two different heights, always constrained to the height of the table which is hard-coded on the actions for robot safety, although it is also estimated by the vision algorithms and taken into consideration, and therefore the security limit is in fact redundant in practice.

(a) Peg grasp



(b) Taking object with Peg grasp

(c) Straight grasp



(d) Taking object with Straight grasp

(e) Isometric grasp

(f) Taking object with Isometric grasp

Figure 21 – Grasps

The grasping point is obtained through a search for the most wrinkled point. To compute this point, the algorithm obtains the surface normal at each point of the cloud given a fixed size window. The normal vectors are then expressed in spherical coordinates, whose angles are used to build two-dimensional histograms to estimate the probability density function of the normals. Being that so, the entropy of such histogram is in fact a measure of the wrinkleness, since greater normal variability is translated to greater entropy.

The number of bins present in the two-dimensional histogram sets the resolution of the wrinkle scope given a window, and we can control the smoothness of the wrinkle map by adjusting the window size. As a result, the algorithm generates a wrinkleness map of the three-dimensional information (Figure 22).

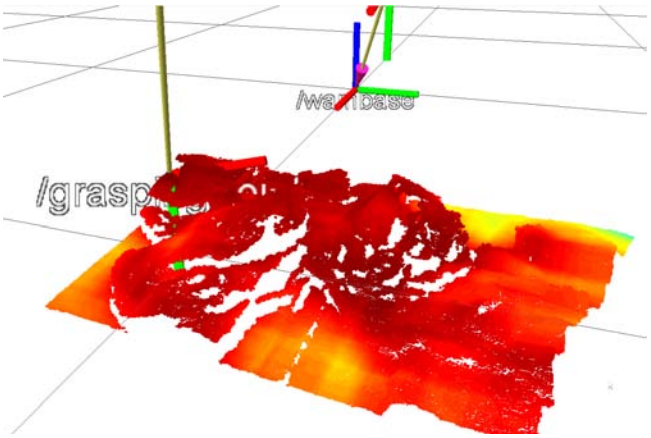


Figure 22 – Unnormalized wrinkle intensity map where the higher red pitch denotes strong wrinkles. We can see the selected grasping point at the top left of the cloth. The missing points are not considered on the histogram voting.

Once we have located the grasping point, additional calibration is required for robot-camera coordination. The theoretical background of the calibration is explained in Section 9.4.

Summarizing, there are more than fifty possible actions if we take into account the different combinations of features and the transitional actions (*change pile* and *remove*). In order to assess which actions are useful in order to reduce the branching effect of the policy computation, the following section proposes an action selection procedure.

9.3 Action Selection

As introduced in Section 3, the policy obtained by the solver given a POMDP model provides an estimation of the infinite horizon reward of each action at any state; which in fact is a measure of the value of each action. Thus, we can use the POMDP evaluation as an implicit action selection, either by removing the actions that are directly invaluable or the ones that, compared to other more popular actions, do not provide a perceivable value increase within the state space regions where they are dominant, hence unnecessarily increasing the POMDP complexity.

Action selection is useful to reduce the dimensionality of the action space, as well as to assess which actions are the most relevant from a task/goal perspective. As we will see in Section 12, in our case the correlation between observations and actions was too weak to be useful for the planning. The confrontation of each value distribution given the actions is shown visually in Figure 29 of that same section.

In practice, the actions' success depends on many variables, indeed, grasping is an unsolved robotic branch of research by itself. Increasing the robustness could improve the probability density functions

over the state space or the correlation amongst the observations. The potential of introducing such techniques in the future falls outside this thesis and are briefly reviewed in Section 13.1.

In our case, the success of the grasping depends, at least, upon where is the chosen wrinkle found, the shape of the cloth (extremely folded opposed to highly spread), the success of the grasping (whether it grasped the whole cloth or just a small wrinkle), altogether with the hand-eye calibration mismatch, commented at Section 9.4, these features introduce great variability on the expected result of the actions, which translates to weak correlation between actions and observations, or the state transitions probability density function.

Hence, after the action selection, the final set of useful actions are part of the transitional set; four grasps with different hand configurations (Figure 21) and grasping heights (high and low). One of the grasps is specially suitable for picking just one piece of cloth, but tends to fail to grasp, while the other is more coarse and tends to grasp several pieces of cloth. The height of the grasp also influences the successful grasping, which is also an indicator that it might help the planning to express it explicitly on the state or observations.

9.4 Hand-eye Calibration

Supplying reliable robot perceptions is the first necessary step to autonomous robots, that is, directing the arm to a target point obtained by image processing algorithms from the images provided by the camera. This purpose is achieved by the called *hand-eye* calibration process.

Specifically, the robot needs the transformation matrices between the camera coordinates and the arm coordinates, as well as the camera's intrinsic matrix required for the transformation from *image pixels* to *camera coordinates* (Figure 23). Given that the relative position between the camera and the base of the arm is fixed, the calibration remains reliable as long as relative positions, orientations and intrinsic mechanics, stay the same.

Figures 23 and 24 show the involvement of such matrices in the arm's operations. Every pixel on the image corresponds to a real-world point in 3D space ($\boldsymbol{p'}$) from a perspective transformation (defined by $\boldsymbol{H}_{intrinsic}$ and a scaling factor s). That is, the projection ($\boldsymbol{m'}$) of 3D points onto the image plane is given by

$$s \cdot \boldsymbol{m'} = \boldsymbol{H}_{intrinsic} \cdot \boldsymbol{p'}. \tag{16}$$

The calibration algorithm can estimate the focal length of the camera and its intrinsic parameters through trigonometry using chessboard patterns with known fixed-width squares. The resulting matrix (18) then lets us calculate the real-world position of the pattern with respect to the camera coordinate frame with

$$\boldsymbol{p}_{px} = \boldsymbol{H}_{intrinsic} \cdot \boldsymbol{p}_{camera}, \tag{17}$$

where $\boldsymbol{p}_{camera} = \begin{pmatrix} X_{cam} \\ Y_{cam} \\ Z_{cam} \end{pmatrix}$ stands for the 3D point in camera coordinates and

$\boldsymbol{p}_{px} = s \begin{pmatrix} x_{px} \\ y_{px} \\ 1 \end{pmatrix}$ is the projected point on the image in pixels scaled by a factor of s and, finally, the

matrix of intrinsic parameters of the camera

$$\boldsymbol{H}_{intrinsic} = \begin{pmatrix} f_{x_{px}} & 0 & c_x \\ 0 & f_{y_{px}} & c_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{18}$$

where c_x and c_y are the coordinates of the center of the image and $f_{x_{px}}$, $f_{y_{px}}$, the focal length in each dimension.

However, rather than knowing the projection given a point in the 3D world, the robot seeks a 3D point matrix from its projection. Hence,

$$\boldsymbol{p}_{camera} = \boldsymbol{H}^{-1}_{\boldsymbol{intrinsic}} \cdot \boldsymbol{p}_{px} \tag{19}$$

$$\boldsymbol{p}_{camera} = \begin{pmatrix} 1/f_{x_{px}} & 0 & -c_x/f_{x_{px}} \\ 0 & 1/f_{y_{px}} & -c_y/f_{y_{px}} \\ 0 & 0 & 1 \end{pmatrix} \cdot s \begin{pmatrix} x_{px} \\ y_{px} \\ 1 \end{pmatrix} \tag{20}$$

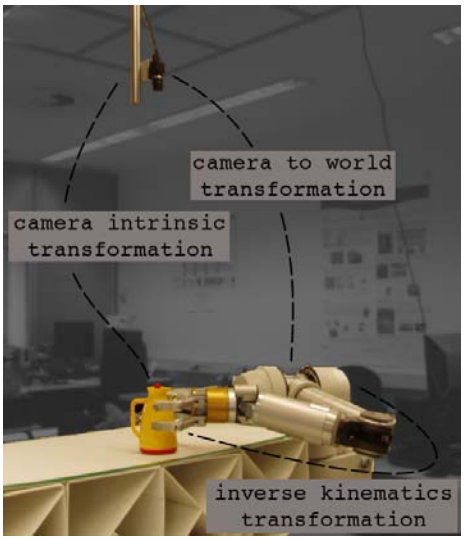


Figure 23 – Object's hand-eye transformation. With the camera intrinsic parameters we obtain the object pose in camera coordinates. The inverse kinematics allow the arm to move in Cartesian coordinates from the world's coordinate frame. In order to grasp the object, we must know the transformation between both coordinate frames.

At this point, the only transformation remaining to complete the loop goes from camera to world coordinates ($\boldsymbol{H}_{cam2world}$). This transformation is obtained by minimization algorithms given a set of correspondences. Once the calibration process has obtained the camera to world transformation, the loop is closed and the hand-eye calibration completed (Figure 23).

In practice, the calibration process required to obtain the above matrices consists of three general steps:

1. Take a set of images of the chessboard pattern from different positions of the arm and store them. Keep in mind that using as many degrees of freedom as possible in the zone of interest improves the accuracy of the transformation matrix.

2. Calculate, from these images, the intrinsic camera matrix and the position and rotation of the chessboard patterns (also known as *pose*) in camera coordinates, expressed as

$$\boldsymbol{O}_{cam} = \begin{pmatrix} r_{11} & r_{21} & r_{31} & d_x \\ r_{12} & r_{22} & r_{32} & d_y \\ r_{13} & r_{23} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{21}$$

   where r_{ij} are the components of the rotation matrix and d_x, d_y, d_z are the components of the translation vector.

   Almost every image processing library has such capabilities, as the intrinsic and distortion parameters are required for any placement in space of image objects. On the other hand, with the *extrinsic* parameters the position of these features can be calculated from a global coordinate frame. Setting a common coordinate frame is a *must* in robotics, as well as to achieve the coordination of multiple cameras, like in stereoscopic vision systems.

3. To compute the transformation matrix that links camera's and arm's coordinate frame.

   Many intrinsic-extrinsic parameters estimators are currently available (Navy, Tsai, dual-quaternion). With intrinsic parameters the algorithm estimates the *image*-to-*camera* coordinate frame transformation matrices and, with the extrinsic, the transformation matrix from *camera*-to-*arm*.

Nevertheless, the pattern can be fixed anywhere on the arm's furthest extremity, and therefore another transformation is required if the pattern has not the same orientation and is not exactly placed on the hand's very end (the *end-effector*). Without applying this transformation, the variation between the pattern and the end-effector point would be *absorbed* by the *camera*-to-*world* transformation matrix, and orientation recognized by the camera would be slightly backwards from the real tool point. More critically, if the pattern was to be on the side of the marker, all positions would be twisted 90º along the Z axis. This last matrix, $\boldsymbol{H_{grid2hand}}$, reverts this effect and is provided by the minimization algorithms in the last phase of the calibration, together with the appropriate *camera*-to-*world* transformation matrix (Figure 24).

At this point, all the needed transformations have been estimated. The camera's transformation describes the pattern's pose in camera coordinates. $\boldsymbol{H_{grid2hand}}$ is the transformation matrix that describes the grid as seen by the hand; therefore, the inverse of this homogeneous matrix is the transformation from the hand to the grid (22).

$$\boldsymbol{H_{hand2grid}} = \boldsymbol{H^{-1}_{grid2hand}} \tag{22}$$

Furthermore, with $\boldsymbol{H_{grid2cam}}$ matrix and $\boldsymbol{H_{cam2world}}$ the hand's position and orientation are finally described in world coordinates. The transformation path followed goes from hand to grid, from grid to camera and from camera to world.

$$\boldsymbol{P_{hand2world}} = \boldsymbol{H_{cam2world}} \cdot \boldsymbol{P_{grid2cam}} \cdot \boldsymbol{H^{-1}_{grid2hand}} \tag{23}$$

where $\boldsymbol{H_{cam2world}}$ and $\boldsymbol{H_{grid2hand}}$ are, respectively, the transformation matrix from camera coordinates to world coordinates and pattern to hand's very end.

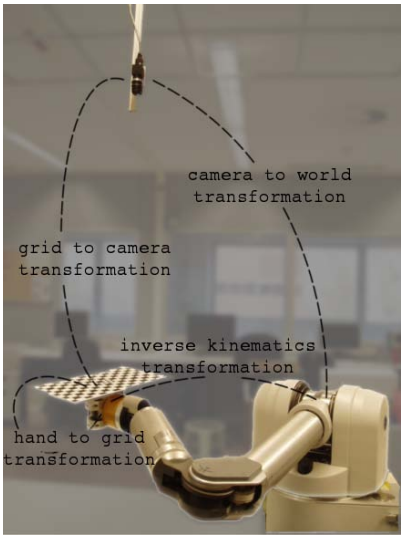


Figure 24 – Calibration matrices. During the calibration, the matrix that links the grid position relative to the hand's has to be estimated. The hand-eye algorithm obtains the hand's pose in every picture in camera coordinates. Matching the poses to the inverse kinematics poses determines the camera to world transformation.

10 Model Construction

Now that we have defined the state, observations and actions of the POMDP, the only remaining information of the tuple are the probabilistic transitions and observations. Traditionally, POMDP models are obtained by manually setting the state space, together with the transition and observation probabilities in virtual problems, and empirically estimating such probabilities in real problems. Recently, there have been some attempts to automatically obtain the state space, and specially the transition and observation probabilities. Hoey proposes a knowledge-based approach using a relational database to dynamically build POMDP models given a task [12, 11]. However, the relational database is bound to the provided encoding of goals, action preconditions, environment states, cognitive model, user and system actions, as well as relevant sensor models, which have to be provided by the user, which is inconvenient.

As we commented in Section 3, other approaches for learning the POMDP model include interpreting it as a Hidden Markov Model [4], statistical memory-based sequencing of observations (such as McCallum's Utile Suffix Memory [21]) or modeling the sensor to obtain the observation distribution [33]. Yet the issue has not been satisfactorily resolved.

In this thesis, two approaches are used to build the model: first, establishing subsets of the state space that are fully observable, and second, extrapolating simple, empirical action models to our particular problem context.

The first approach has been used to estimate the correlation between the actions and the observation probability. By fixing the number of pieces of clothing beforehand and withdrawing the spatial division, which is symmetrical anyway, the state becomes known and, therefore, we can obtain the observation probability density function through exploration. The only situation that is not taken into account with this approach is the eventual invasion of pieces of clothing outside the region of interest, which would trigger unintended observations in the real case scenario. However, since the

actions that can be modeled that way do not involve moving pieces of clothing from one region to the other, it is unlikely that this situation occurs.

Within this approach, the number of experiments to be conducted in order to obtain a reliable action model depends greatly on the domain, and so do the diversity of initial states that the exploration should start from. A good estimate of the action uncertainties has a direct repercussion on the policy. In fact, since we have constrained the state representation dimensionality, all uncertainties are propagated to the perceived variable – the actual number of objects taken – regardless of the particular source of the error.

Besides, given that it is more important that the models are reliable within the situations the policy is bound to visit, the models are built by conducting fixed policies or randomly executing the actions, rather than manually setting a great diversity of initial situations, which might not be revisited naturally by the established actions. For example, it is unlikely that the actions provided could entangle the pieces of clothing as much as the initial condition of the *entwined* experimental set (Figure 34). On the other hand, if in practice the set of initial states is reduced, it would be advisable to code its particularities within the state description, so the planning takes the domain features into account. In our case, we favor generality rather than specialized actions and states, so this criteria is not applied.

The second approach is applied to state transitions and requires a ground truth in our context. Since the success of the swap actions depends mainly on the success of the grasping, we have confronted the number of pieces taken given a fixed number of pieces of clothing following the same pattern as the other approach, but noting manually the number of pieces of clothing taken. This data has then been translated to a probability density function over the number of pieces of clothing and afterwards to the states space.

In domains with large action spaces, such approach would require a lot of effort to obtain the ground truth. In those cases, the model would either be estimated with the model learning methods introduced in Section 3 or by extrapolating as well, if we take into account that the empty table is fully observable and, therefore, the extrapolation can be extended from the fully observable two-pieces of clothing to the rest assuming a certain degree of mismatch error.

The advantage of the first approach is self-evident; the model is learned without requiring human intervention to obtain the ground truth. The second approach reduces the problem so only the grasping models have to be provided, which simplifies the modeling task if moving to a different sorting context.

10.1 Rewards

The distribution of rewards does not follow any heuristic criteria other than punishing the removal of more than one piece and rewarding the removal of a single piece of clothing or reaching the final state. In the first case, the punishment is of 5000, while on the second and third cases, the rewards are 100 and 1000 respectively. Note that the absolute value is not as important as the proportion between punishment and reward, which is one-to-five for reaching the final state and one-to-fifty for removing one piece of cloth. This proportion tells the solver how much risk can the policy accept. Table 1 summarizes the punishment/reward distribution. Finally, there's a discount value of 5% per step to favor fast policies over slow policies and also the reward is independent of the observation received after reaching the rewarded state.

Action & State	Reward / Punishment
1 Obj. Removed	100
> 1 Obj. Removed	-5000
1 Obj. Removed & final state	1000

Table 1 – Distribution of rewards and punishments. One-to-five for reaching the final state, with a one-to-fifty for removing one piece of cloth. The reward is independent of the final observation.

11 Implementation

Now that we have defined the task, the approach and the theoretical background of each of the elements involved, this section will describe the implementation details of the node structure, robot vision and policy computation, placing special emphasis on the interaction between action, observation, planning, sensory input and robot manipulation.

The communication structure has to provide the interaction with the world to the planner node, after whose actions the robot vision estimates the number of objects at each area so the planner can update the probability distribution amongst the possible states using the value matrix previously computed by the off-line solver.

11.1 Node Structure

The action, observation, planning and sensory input necessary for the sorting task is organized in independent processes, each process is called a *node* within the communication framework context, discussed in Annex A.1. Each node provides services and periodic data publications and retrieves information through a publisher/subscriber structure or client/server hierarchy, provided by the communication framework, together with visualization tools. Figure 25 depicts an schema of the node network structure and the relationship between the nodes.

The devices involved in our experimental settings are encapsulated by nodes and provide access to the underlying machinery and sensors. The data obtained by the cameras is then processed by the different nodes in order to provide comprehensible actions and observations for the planning node.

During an execution, the planner node holds a belief over the state space. Initially, the belief is uniformly distributed across the state labels *none* to *more than three* objects present on the left area. Using the value matrix obtained in the POMDP policy computation, which implicitly defines the policy followed, the action with the maximum projected reward given the current belief is selected and sent to the action node. The action node then asks the object filter to discard the points outside the region of interest and provide a suitable grasping point. The object filtering node is prepared to filter out objects given a selected object label in the future. At the moment, the object filtering node filters out the unselected area of the image, leaving only the area where the grasping will be applied.

As described in Section 9, the wrinkle analysis uses the Point Cloud technology to obtain an array of wrinkleness measures. The maximum of the array is selected, highlighted on the visualization software of the scene and reported back to the actions node. Once the action node obtains the grasping

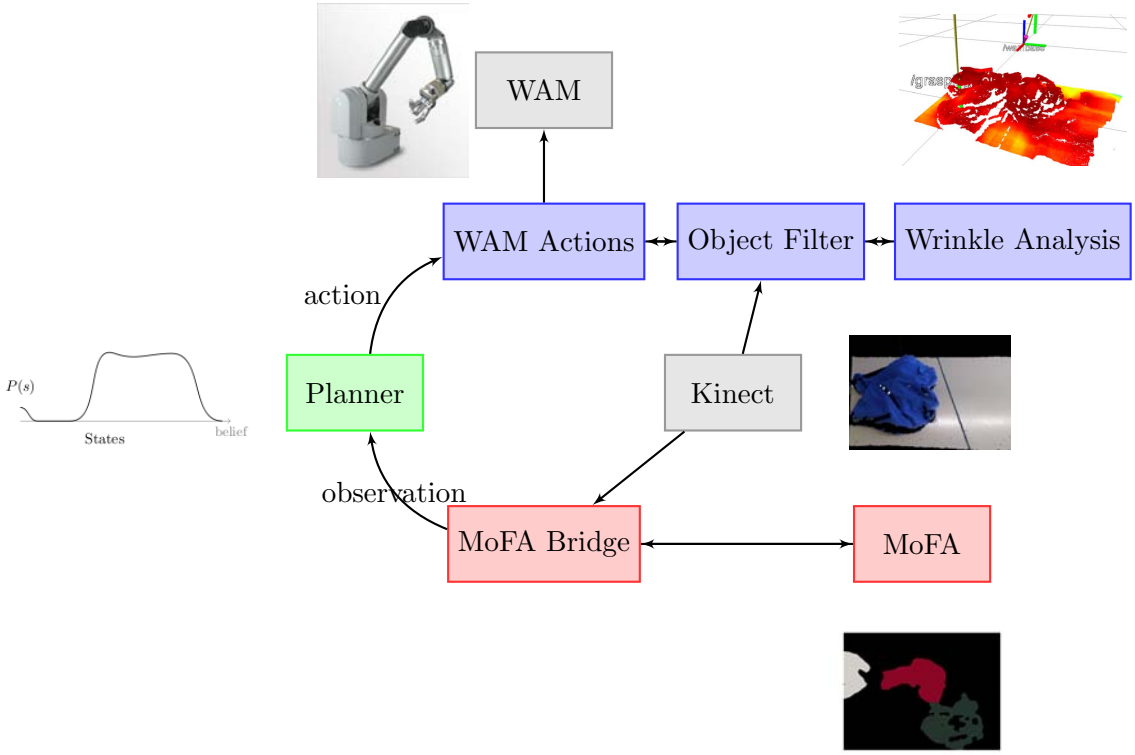


Figure 25 – Node network structure and small excerpts of relevant operations conducted at each node. The planner holds the belief over the real world representation and commands actions to the WAM actions node and retrieves observations from the MoFA node. The action node retrieves the grasping point through the object filter, that filters out the deactivated area of the image, and the wrinkle estimator node. Finally, once the observation is retrieved, the planner's belief is updated and the new optimal action is obtained.

point, it executes the action script of the actions node selected by the planning and the robot node runs the set of moves and grasps that constitute the action.

Once the robot completes the action, the planner retrieves an observation from the other robot vision node, the MoFA bridge, which provides an estimation of the number of objects seen by the camera in each region of the space. The MoFA node translates the point cloud information from the kinect in a suitable format for the image segmentation algorithm, which is implemented in Matlab and whose communication is achieved through a vanilla Berkeley socket. The eight-dimensional information of the point cloud is down sampled by a factor of four in order to increase the speed of the segmentation.

Finally, with the action performed and the observation retrieved, the planner updates its beliefs and retrieves the following best action.

The process is executed in loop until the planner achieves the final state with a certain probability, currently set to 94%. Eventually, if the certainty is not above that threshold, some extra actions are performed to ensure we have indeed reached the final state.

The whole process is logged with the set of images taken, its segmentation and the subset of most probable states of the belief with its associated probability. Since the process is not optimized for speed yet, it might take five minutes per loop, the bottleneck clearly being the robot vision.

11.2 Policy Computation

The policy used by the planning node is computed off-line using the solver presented in Section 6. A complete implementation of a generic problem policy computation involves several steps; once the robot vision and manipulation algorithms were implemented, a data gathering program estimates the basis of the action and observation models, complemented by manually added ground truth for transitional actions (Section 10). Once we have the probabilistic models, the model builder extrapolates to suit the state definition (Section 7) and builds the POMDP tuple. The POMDP tuple follows the Cassandra's syntax [22], which is read by the solver and outputs a policy matrix. Then, the policy matrix is read by the planner node, which allows the planner to compute the most valuable action given a certain belief, updated by the robot vision algorithms implemented (Section 8) retrieved by the actions performed (Section 9).

Figure 26 illustrates the particular implementation of the policy computation process, which is schematically detailed below.

1. POMDP world model building

   This program is a domain-dependent model builder that outputs a Cassandra syntax file[22]. It establishes the POMDP tuple $< \mathcal{S}, \mathcal{A}, T, R, \Omega, O >$ including the state transitions, their associated probability given an action failure ratio and observation error ratio from the grasping and observation subspace models (Tables 2 and 3).

2. POMDP solving

   As commented in Section 6, we use the APPL SARSOP solver, which can read Cassandra files and outputs an XML α-vector history.

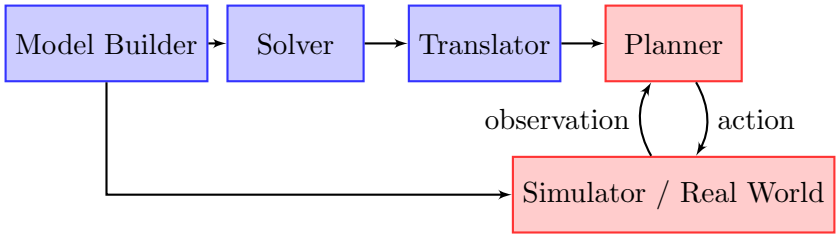


Figure 26 – Flow chart of the implementation of the POMDP. The model is provided to both the simulator and executor, whose relations are described extensively on Figure 37. The Solver computes an approximate policy to maximize the expected reward and is then provided to the POMDP planner properly translated

3. Policy extraction

   APPL Solver outputs the policy as an XML file which is designed for evaluation of the algorithm but is not suitable in practice. In order to use it, the optimal policy is extracted by retrieving the greediest action for each state given the propagated beliefs at the point the approximate solving was halted.

4. POMDP planner node

   The POMDP node is the main POMDP node, where the world model and policy is loaded and saved. It performs actions and retrieves observations in a loop.

5. World simulation node

   The world simulation node starts at some random initial state and uses the world representation computed on the first step to obtain the following state and observation to each action requested.

6. Real World structure

   In practice, the sensory and manipulation functionalities provided by the system are translated to the POMDP codification, which is transparent to the underlying robot and computer vision algorithms provided the observations and actions are coherent with the loaded model. The following section describes the structure designed and implemented and the technology used to retrieve informative sensory input of the real world.

Both the off-line learning of the models and the on-line execution follow a decentralized peer-to-peer connections handled by a centralized connection manager[1].

12 Results and Evaluation

12.1 Underlying Robot Vision and Object Manipulation

12.1.1 MoFA

To build the observation model, the image segmentation used to estimate the number of objects, described on Section 8, was confronted to a small dataset organized by families of number of objects.

[1] See the Annex A.1 for more information

Note that it is not our interest to show the reliability of the segmentation algorithm in general, but rather assess the quality within the planning context. To that purpose, images were obtained through robot manipulation and starting from several initial conditions manually disposed. Table 2 summarizes the results of the observations and represents the model used for the POMDP's observation density function estimation. The datasets are publicly available at `http://www.iri.upc.edu/research/perception/clothplanning`.

Observations		Number of objects present				
		0	1	2	3	> 3
Num. objects seen	0	100	6.5	2.5	2.5	1.25
	1	0.0	89.5	3.75	0.0	1.25
	2	0.0	1.4	86.25	11.5	2.5
	3	0.0	2.6	7.5	70	15
	> 3	0.0	0.0	0.0	16	80

Table 2 – Observations probability density function. The results show that as the number of objects increases, so does the error rate, conditioned by the grouping over three objects. The data is then extrapolated to the observations defined by the POMDP. The results are given in percentages of the number of objects seen over the real number of objects present.

As expected, as we increase the number of objects, more mistakes are made upon image segmentation, dropping from almost 90% success to 70 when there are over 2 pieces of clothing. When there are more than three objects, the error rate decreases due to the grouping made, which collapses the estimation errors of four objects and beyond since the observations node correctly identifies the general category more than three objects.

Also, the algorithm has been tuned to favor false positives rather than false negatives, since it is more critical to mistakenly remove two objects than invest more time into assert the mistaken false positive. Although at the moment there isn't a recovery mechanism for repeatedly misclassified pieces of clothing, the planning tends to give priority to the correctly classified pieces of clothing and leave the wrongly classified pieces of clothing for the end.

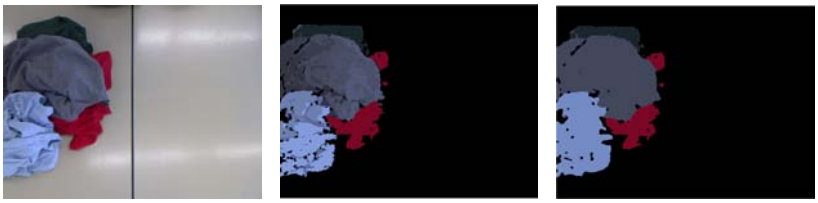
Considering the error rates, interaction helps obtaining better object number estimates in cases like the one shown at Figure 27, where the scene contains a misclassified piece of clothing due to light conditions. However, there are some cloths that are misclassified more often than others, to the point that some are constantly misclassified. This effect dramatically threatens the planning success, since the model does not match at all the handled reality in these situations. In Section 13.1 we propose planning techniques to tackle this local problem.

The MoFA clustering starts from an upper bound number of Gaussians and, therefore, delivers the load of the segmentation decisions to the merging step. Hence, the key step of the current algorithm is the merging of the clusters (Figure 28c), where most errors are encountered, specially when there are many pieces of clothing.

At the moment, we are considering a fairly controlled environment, it is left for future work to assess how the POMDP planning reacts to higher uncertainty on perceptions, in both situations where the model matches the world and when there's an important mismatch with the world and the pre-gathered model.

(a) Original Image **(b)** Clustered Image **(c)** Merged Image

Figure 27 – Image processing segmenting error example. The image fails to segment correctly and hardening the threshold to prevent false positives increases the false negatives rate.



(a) Original Image **(b)** Clustered Image **(c)** Merged Image

Figure 28 – Image processing segmenting example.

12.1.2 Grasping

The robot uses the wrinkle-based grasping presented in Section 9, where the planning combines two relative heights and three hand configurations. The window and resolution of the wrinkle map computation have been manually tuned to a patch radius of 15 with 24 bins for the orientations histogram, a trade-off between speed and quality of the detection. A higher patch radius generates a more robust against spurious, smoother wrinkle map, but it also takes longer computational time. Since the localization of the grasping point is not the focus of the current project, but rather planning over it, the many enhancements possible are left for future work. Nonetheless, Section 13.1 proposes several improvements that will be implemented in the future.

As intended, the current grasping actions are proven biased, either towards grasping more pieces of clothing than expected or towards failing to grab its target, as we can see in the statistics of the set of Table 3. Indeed, the perfect grasping would always grab one piece of clothing regardless of the calibration errors, wrinkle estimation noise or finger closing mechanical errors. Although this is difficult to achieve in a reliable manner, having a great bias per action is something the planning takes advantage of in order to reach the goal faster.

Likewise commented on Section 10, the tables are obtained within the subspace of robot actions given several initial situations, since it is more important that the models are reliable through the interaction rather than subject to the great variability of the initial conditions.

Unfortunately, the resulting wrinkleness distribution is uniformly spread amongst the actions, proving the wrinkleness of the grasping point chosen not correlated enough to be a relevant feature to the given actions. Therefore, the action selection denoted the necessity of designing more specialized cloth handling actions, or include the observation of different cloth features as additional domain information, taking advantage of the split state description proposed. From a different point of view,

the result means the actions were unable to take advantage of the wrinkleness information, which does not mean that the feature does not improve the overall success rate, when unconditioned to the actions. A linear combination of factors is proposed at Section 13.1 to improve the map and achieve more specialized grasps.

Note that the previous consideration is an interesting result since, in general, it is difficult to assess which actions are most useful in a object-sorting problem given the statistical results gathered at Table 3. Hence, the POMDP formulation gives us the further information we need; Figure 29 shows the three-dimensional value distribution over the state space for the transitional actions. It was evident after the policy computation that the isometric-high action was not useful at all, since it was never selected within the minimization of the optimal policy boundaries (not shown in Figure 29). Moreover, we can see which action is the most useful and the correlation and relevance of each one of them in Figure 30.

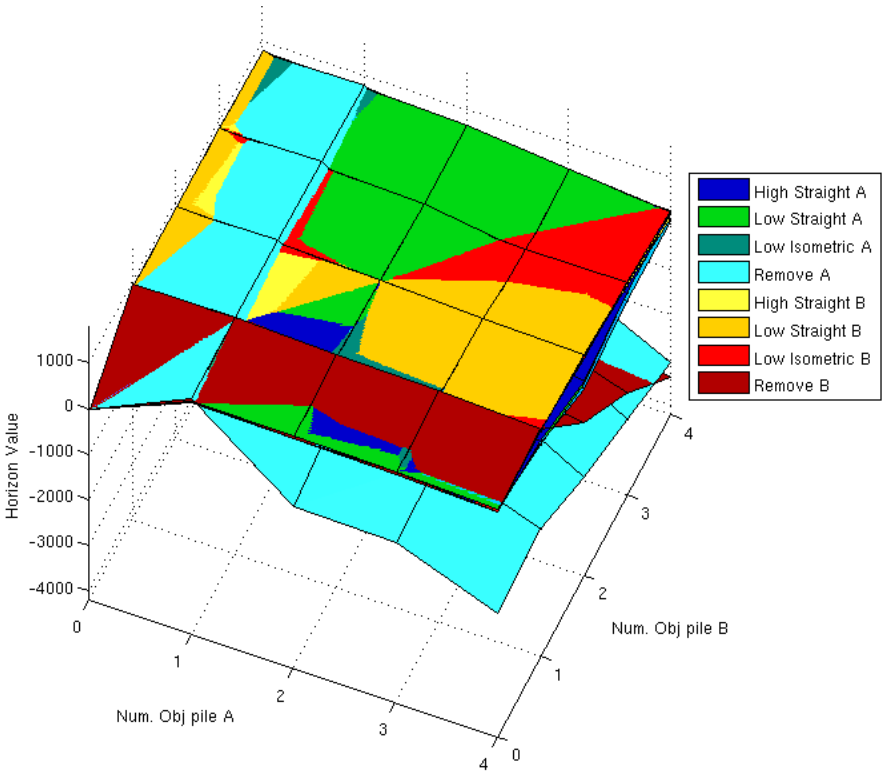


Figure 29 – Given the policy computed values, we can analyze each action expected reward given a state. The graph is not symmetrical because any initial state starts without any object on the area B. The evaluation of the initial conditions impact to the policy obtained is left to further research. Also, we can see that, except for the remove action, the action projected values are very close to one another, which denotes that the actions presented require further specialization.

The exploratory actions were also proven irrelevant for the planning, since the observations were uniformly distributed amongst the actions, contrary to what it was expected for spread actions. However, the poor quality of such actions, highly affected by the grasping point and the diverse scene conditions amongst other factors, made them useless in practice, which allowed us to perform an action selection and reduce the branching factor of the policy computation.

Despite the exploratory actions' results, it remains a future objective to provide the method with resolutive exploratory actions that help intentionally reduce the observation uncertainty, and therefore

Grasping: High Straight		Number of objects present 0	1	2	3	> 3
Num. objects taken	0	100	37	18	21	30
	1	0.0	63	54	47	25
	2	0.0	0.0	27	31	30
	3	0.0	0.0	0.0	0.0	10
	> 3	0.0	0.0	0.0	0.0	5

(a) Straight grasp at hight height

Grasping: Low Straight		Number of objects present 0	1	2	3	> 3
Num. objects taken	0	100	31	5	5	5
	1	0.0	68	60	65	45
	2	0.0	0.0	35	20	35
	3	0.0	0.0	0.0	10	12
	> 3	0.0	0.0	0.0	0.0	5

(b) Straight grasp at low height

Grasping: High Isometric		Number of objects present 0	1	2	3	> 3
Num. objects taken	0	100	20	35	35	29
	1	0.0	80	40	65	29
	2	0.0	0.0	25	5	33
	3	0.0	0.0	0.0	0.0	5
	> 3	0.0	0.0	0.0	0.0	5

(c) Isometric grasp at high height

Grasping: Low Isometric		Number of objects present 0	1	2	3	> 3
Num. objects taken	0	100	7	31	16	18
	1	0.0	93	50	37	44
	2	0.0	0.0	19	42	31
	3	0.0	0.0	0.0	5	5
	> 3	0.0	0.0	0.0	0.0	0.0

(d) Isometric grasp at low height

Table 3 – Grasping probability density function. The probability density functions are then used to build the POMDP transition model. The ideal situation for the planning would be a set of actions that, each one of them respectively, always grasps the exact same number of objects.

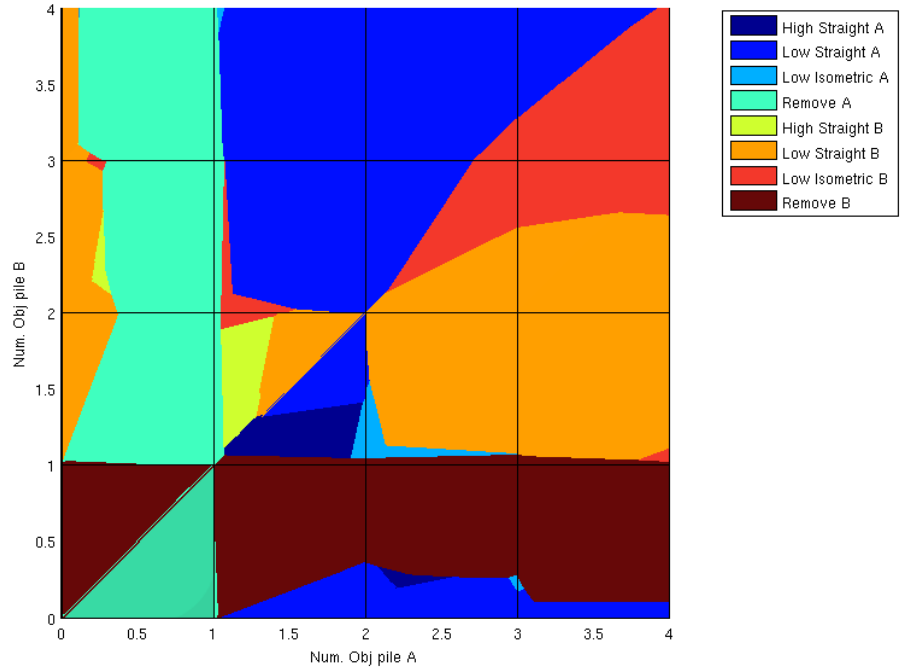


Figure 30 – Top view of the policy action value spatial distribution. The topmost action values will be the ones chosen when the planning has low-certainty beliefs. However, for spread distributions, the addition of distributions might marginally select actions locally below the ones shown, hence the need for a three-dimensional analysis of the distribution.

improve the planning. Meanwhile, the task goal is still achieved through only the transitional actions observation information.

12.2 Solver Results

The grasping and observation models are extrapolated to the whole state space transitions and observations. The obtained POMDP tuple description is then evaluated by the SARSOP solver to approximate the optimal policy. After less than five minutes, the approximate policy achieved an average reward of 853€ within boundaries precision of 0.8%. The computation was manually stop since little improvement was achieved over further propagation of beliefs, and therefore the policy was considered close enough to optimality. Table 4 summarizes the final output of the solver.

Time (s)	Trials	Backups	Lower Bound	Upper Bound	Precision	Alphas	Beliefs
291.5	283	6755	848.952€	857.042€	8.08941€	5226	1365

Table 4 – Output of the solver's policy computation. Given the reward distribution, the average reward is equivalent to roughly eight actions. The difference between lower and upper bound –the *precision*– equals proportionally a 0.8% of the average reward. The solver propagated more than a thousand beliefs leading to more than five thousand alpha planes cutting the belief space.

Due to the reward distribution, which depends on the number of objects present, it is difficult to know the average number of actions required given the average policy reward. We can, however,

compute an approximate value: First, take the positive reward values of Table 1, that is 1000€ for reaching the final state and 100€ per object recovered. The maximum reward without discount factor is 1000€ plus the number of objects $\bar{o}$. With a discount factor of 5% and an average of n actions, the obtained reward roughly obeys the equation

$$r = 0.95^n \cdot 1000 + 0.95^{n/2} \cdot 100 \cdot \bar{o}, \tag{24}$$

where r is the reward, n the number of actions and $\bar{o}$ the average number of objects.

Then, isolating the number of actions in the equation we obtain

$$n = \frac{\ln \frac{r}{1000+0.95^{0.5} \cdot 100 \cdot \bar{o}}}{\ln 0.95}, \tag{25}$$

which, for an average $\bar{o}$ of three objects and a solver's average reward of 853€, corresponds roughly to eight actions.

12.3 Planning Results

In a typical execution of object sorting, with mostly partial occlusions, the planning policy successfully reaches the goal of isolating the objects one by one. The trend of the policy is to encourage stalling rather than mistakenly removing pairs of objects, which is punished by the reward distribution (commented in Section 10.1). The policy is able to acknowledge incorrect beliefs if the observations received through interaction are not coherent with the most probable state.

The value distribution across actions depicted in Figures 31 and 32 provides a lot of information about the quality and suitability of the actions. For example, we can conclude that the isometric grasp is the one that provides better results when there are two or three objects in both piles. Indeed, taking a look at grasping models of Table 3 we can see that it is unlikely that the robot takes two pieces of clothing if there are two, and it is the action most likely to take two pieces of clothing when there are three, thus naturally leaving one. Note, however, that those are unlikely visited situations, since the initial state only considers one pile, and therefore the policy doesn't lead naturally to having many pieces of cloth in both piles, but rather concentrate all cloths except one in one pile.

To assess the planning results, four scenarios with different initial conditions were tested. Each scenario was manually disposed and the robot worked with them in several instances. Some videos of the executions are publicly available at `www.iri.upc.edu/research/perception/clothplanning`.

The notation used to represent the state are two labels, one that describes pile A and another one that describes pile B, whose both possible values are $\{0, 1, 2, 3, > 3\}$. The action notation describes the kind of action, the grasping used, the relative height of the grasping point and the area where the action starts. Specifically, as we have seen in Section 9, the grasp available are **I**sometric, **S**traight or **P**eg, the heights, **H**igh or **L**ow, and the starting area, **A** or **B**. For example, `TSLA` means "**T**ake from area **A** with **S**traight grasp at **L**ow height".

12.3.1 Common Scenario Execution

The sequence of Figure 33 is an illustrative sequence of actions of the common case. The initial belief is uniformly distributed across any number of pieces of clothing on the first pile. Observations are

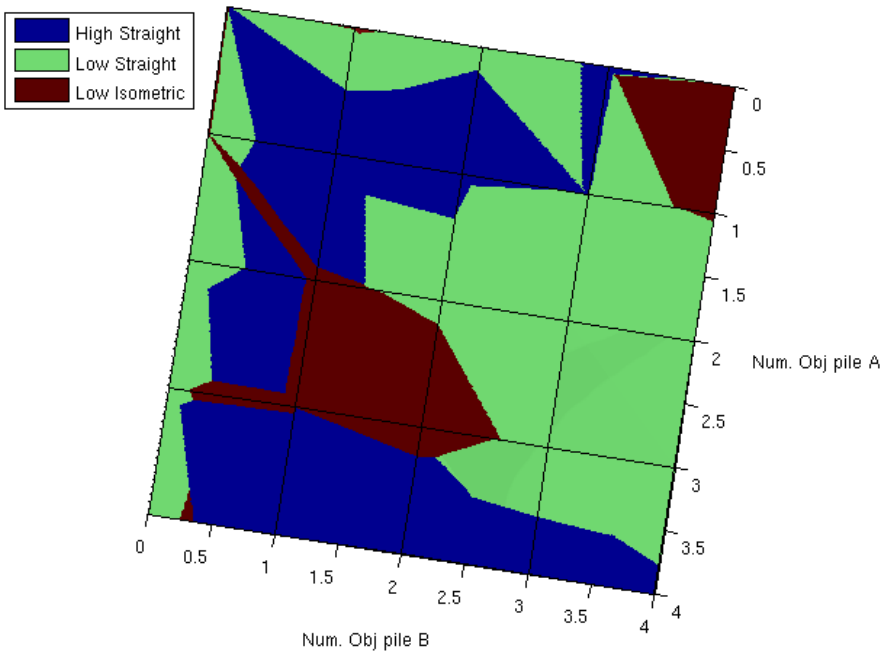


Figure 31 – Policy action value spatial distribution across states. For the sake of visualization only actions starting in area A are shown.

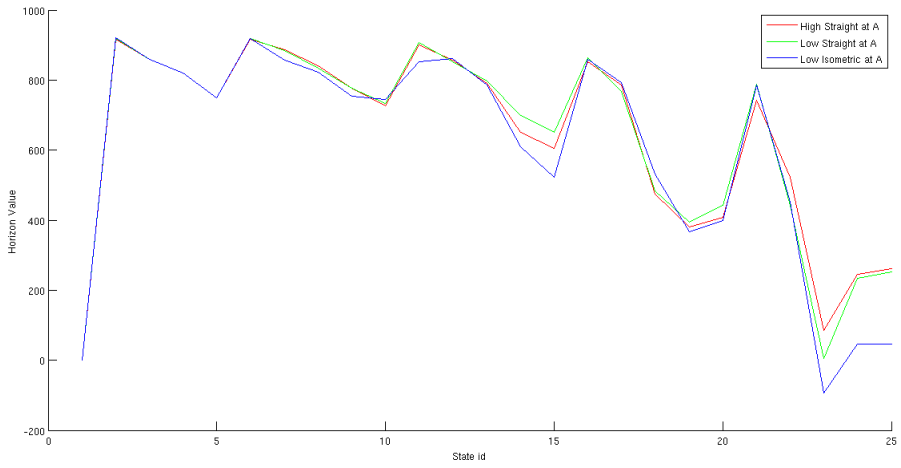
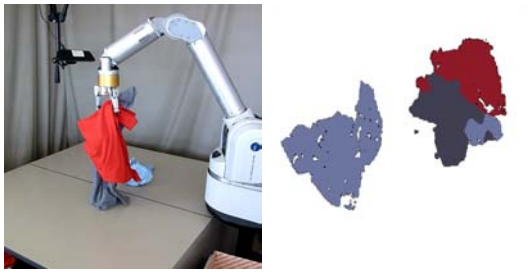


Figure 32 – Side view of the policy action value distribution across states for the transitional actions. The value spread shows the speciality of the actions and provides a graphical value comparison given the policy. The isometric grasp is the one less valued in most situations and that the actions can be further specialized to different situations.

(a) Starting scenario. There are three pieces of clothing.



(b) Belief: 1 3 (70%). Despite the segmentation error on pile B after moving, the isolated Blue is correctly acknowledged and can be removed on the next action.

(c) Remove Blue.

(d) Moving back to area A as an attempt of grabbing just one piece of clothing.

(e) Failed grasp while moving to B.

(f) Successfully moving back to area B as a second attempt of grabbing just one piece of clothing.

(g) The pieces of clothing are successfully separated after the low move.

(h) Remove Red.

(i) Remove Gray.

Figure 33 – Common scenario sequence. After the first action, the robot correctly removes the Blue piece of clothing in spite of the perception errors. Then, the cloths are moved back and forth until they are separated. One of the grasps fails, so the robot retries until it grabs the Grey piece of clothing successfully. Once the two remaining pieces of clothing are separated with high probability ($>90\%$), the robot proceeds to their removal.

retrieved *after* performing an action, so given the departure belief, the initial most valuable action is taking from area A at low height with straight grasping.

The first observation retrieved modulates the current probability distribution to another distribution. In the particular execution depicted in Figure 33, the real initial state has three pieces of clothing. The first moving action grabs two pieces of clothing (Figure 33a), although the vision sees an extra piece of clothing (Figure 33b). With that action and observation, the belief is updated. The overestimate of the number of pieces of clothing on pile B increases the initial probability of having four objects to 70% while lowering the real state of three objects to 12%, which is even less than having two pieces of clothing, with 14% (Table 5). Nevertheless, the planning establishes with more than 95% certainty that there's one piece of clothing on the first pile, regardless of being unsure of the number of pieces on the second area. Hence, when the planner multiplies the updated belief by each action-state value, the resulting most valuable action is to remove the Blue piece of clothing in the following move. after which the robot correctly and successfully executes the action and the isolated piece of clothing is sorted.

Step	Best action value	Consequent Belief (♯A ♯B %Probability)				Figure
0		1 0 25%	2 0 25%	3 0 25%	>3 0 25%	33a
1	TSLA 869€	1 3 70%	1 1 14%	1 2 12%		33b
3	RA 885€	0 2 57%	0 3 37%			33c
4	TSLB 888€	2 0 94%				33d
5	TSLA 903€	2 0 95%				33e
6	TSLA 904€	0 2 99%				33f
7	TSLB 906€	1 1 99%				33g
8	RB 990€	1 0 96%				33h
9	RA 972€	0 0 96%				33i

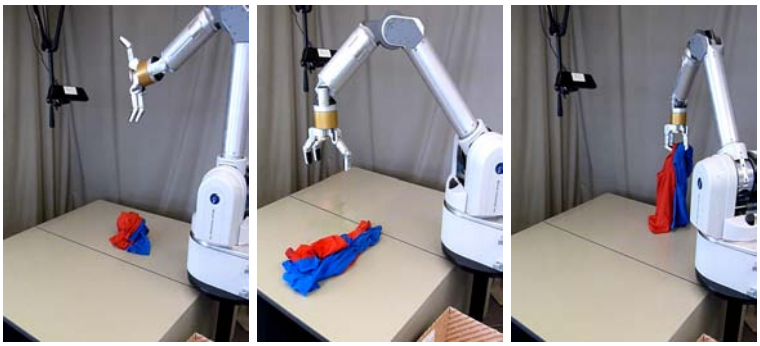
Table 5 – Common scenario belief evolution (belief probabilities under 5% are not displayed). We can see that the state is determined with a certainty over 94% after four actions. The actions acronym corresponds to **T**ake or **R**emove, followed by the grasp **I**sometric, **S**traight or **P**eg, the height with **H**igh or **L**ow, and the starting area **A** or **B**.

With the Blue already sorted, the planning keeps working with the other pieces of clothing, moving them back and forth from step 4 to 6, until it has the strong belief that one of the pieces is isolated in either of the areas (Figure 33g). Note that the planning recognizes mistakes –Step 4 to 5 of Table 5– thanks to the estimate of observations and the statistical models implicitly coded within the policy. The robot is effectively achieving the goal fast in spite of the uncertainty of its perceptions and the failure of some actions.

12.3.2 Entwined

One of the interesting scenarios to confront the planning to is that consisting of two hardly entwined objects. In this context, the robot keeps working with the mixed pieces of clothing until the observations acknowledge the cloths isolated. The Figure 34 sequence shows a sample successful execution. Since the actions are not designed for the task, the sequence of steps tends to be long. In this case, introducing a relevant entanglement measure on the state and specialized actions would vastly improve

the planning. However, setting such actions is not trivial, although general shake and flip actions might be sufficient to separate the two pieces of clothing.



(a) Two objects are highly interlaced, the robot moves them back and forth until they are separated

(b) Finally, the cloths are set apart from each other

Figure 34 – Sequence with entwined objects. The planning works with the clothes until they are isolated.

Note that introducing such actions doesn't necessarily require to reformulate the POMDP design, as the transitional statistics (Table 3) could be set in such context instead of the normal one. However, it would be advisable to denote such characteristic as part of the state, whose modeling could be roughly defined, given image processing techniques to measure the cloth entanglement.

12.3.3 Total Occlusion

In a total occlusion scenario a piece of cloth is covering one or two pieces of clothing, which is particularly threatening since it is impossible for the vision algorithms to assess correctly the number of pieces of clothing present at first glance. Indeed, the planning policy fails if the cloths remain unseen in a sequence of two or more actions, specially if there are two occluded pieces of clothing rather than just one.

In the Figure 35 sequence, the blue piece of cloth is seen at the first observation, which raises the probability of having two pieces of clothing to 80%. Otherwise, since the probability of seeing a piece of clothing when there are two is less than 5%, the planning tends to misjudge the scene. The model mismatch causes the planning to fail often due to a state transition path underestimated probability.

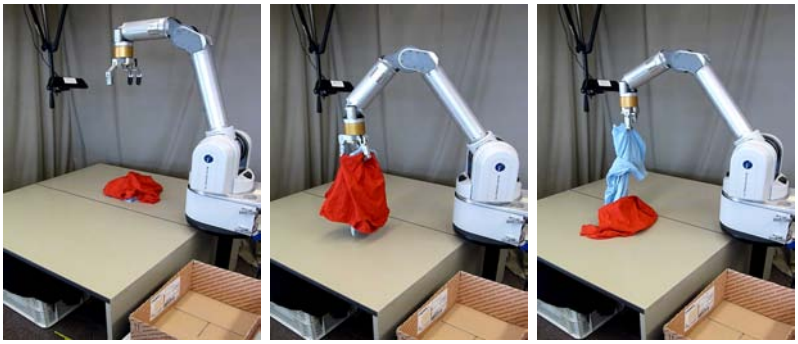


Figure 35 – Relevant excerpt of the totally occluded objects' sequence. The manipulation unveils the hidden object behind the cloth and the belief is updated accordingly. Once the hidden object is found, it's just a matter of removing each cloth.

The issue can be tackled in many ways, for example, it is possible to increase the punishment received to encourage high certainty upon the state distribution to compensate the errors with associated low probability that are threatening for the goal success. Other techniques that could be adopted are introducing the occlusions to the observation model in order to increase the detection rate or including additional information to the object estimation algorithms, such as the estimated volume of the cloths seen. Moreover, it would be also interesting to start off from biased distribution of probabilities over the initial states space, rather than an uniform distribution, which is forced by the solver's syntax used in this project.

12.3.4 Many objects

The last scenario proposed is having many objects to sort. In this context, it is specially complicated to isolate a single piece of clothing due to the uncertainty of the grasping actions, as many actions need to be made to isolate an object and, as it may be expected, in a more cluttered context, the robot grabs many pieces of clothing more often. Figure 36 shows the snapshots of a sample execution.

Merging the segmentation with the wrinkle map could probably decrease the grasping uncertainty on high numbered scenarios, since grasping could focus on a single particular object. We can see on the action distribution (Figure 29), however, how the planning takes advantage of grasps that tend to grasp several pieces of clothing when both piles have too many objects. Thus, the POMDP solver does successfully take advantage of the bias of the actions used.

Indeed, the plans do successfully reach the goal after around seventeen actions, higher than the solver's expected average in the general case, which corresponds roughly to eight actions.

It is interesting, though, how the planning acknowledges pieces of clothing that are partially coupled with one of the two areas, and therefore, conditionally isolated. Figures 36b and 36m take advantage of this eventuality.

(a) Start with five pieces of clothing

(b) Fails to grasp, moving cloths enough so the Blue can be grabbed from the bit trespassing to the B area

(c) Isolates Red

(d) Removes Red

(e) Grasps at area A, taking two pieces of clothing to area B

(f) Isolates Blue taking the Gray

(g) Removes Blue

(h) The robot grasps Gray and Green

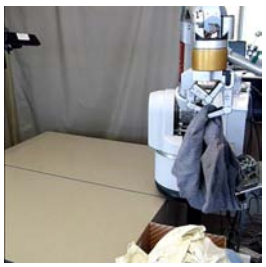
(i) The Gray piece of clothing fell before reaching destination

(j) Removes Green

(k) Fails to grab Gray

(l) Grabs both pieces

(m) Grabs the Yellow piece of clothing from the bit at B area and removes it



(n) Removes the Gray

Figure 36 – Sequence with many objects.

13 Conclusions

The project describes a POMDP design, modeling and implementation for a sorting task, altogether with deformable object manipulation, modeling and image segmentation and several context-based planning executions on a real robot. The key results of the thesis are two-fold: first, providing a task-based action quality measurement, suitable for action selection, and, second, we have shown the appropriateness of the POMDP approach to achieve the robot goals under unreliable actions and perceptions, as a realistic approach.

We have analyzed the development of the planning policies in a context that would not make possible a deterministic approach to object manipulation, and, even though the method presented is a preliminary statistical approach of manipulating deformable objects under uncertainty, we have shown that it is possible to successfully plan with modest action/observation success rates. The approach effectively exploits or reduces the inherent uncertainty through interaction and planning, which relaxes the precision requirements of robot vision and manipulation.

However, we have seen that the supplementary scenarios tested rise locality problems, since the policy can get stuck on local minima, caused by repeated, unrecoverable errors. It is, in fact, reasonable, given that having a more abstract, domain-independent algorithm triggers problems in local situations which are far apart from the underlying abstract models. But it is possible to acknowledge specific problems or to identify loop sequences and recognize stalling situations: by describing such situation within the state, introducing a recovery action or, from a higher-level perspective, complementing the limitations of planning through ad-hoc rules. Nonetheless, it is important that actions are relevant from the planning perspective and to have an objective measure of such importance.

Indeed, for example, the amount of wrinkle was proven to be irrelevant on the planning as soon as the feature was uniformly distributed amongst the actions. Since we would like to take advantage of the particularities of the domain together with the general approach, it still remains an open possibility to identify the relevant domain-dependent additional information that complements and improves the planning method proposed. Keeping in mind that such additional information's density functions have to be accessible or learnable.

Being able to identify the valuable actions is not obvious with the raw information, while it is self-evident once the policy is computed. So, another important result is the usage of the value function computation as a useful tool for task-driven action selection, as well as a measure to assess the importance of each action over the state space. Being that so, we have shown how actions were removed after a simple analysis of the value distribution.

It is, hence, possible to achieve fair planning policies even with inefficient actions immerse in noise, with ambiguous observations, and reach the goal nonetheless. Although the POMDP formulation is very flexible, using it in practice forces low dimensionality, given that the state definition has to be established beforehand, so further research includes the introduction of on-line learning techniques, introducing new actions to enrich the plan and several other proposals. There are many amelioration possibilities, since we have covered all the aspects of a real implementation (actions and observations design and modeling, planning, communication infrastructure, ...), so the extension of this future work is fully addressed in the following section.

13.1 Future Work

This thesis has presented the whole design and implementation of deformable object modeling, robot manipulation and planning over a real robotic platform. Given the holistic perspective of the work, there are many fields of expertise involved whose exploration can improve the results presented.

One of the first focus of the future work will be the specialization of the manipulation actions, with the objective of increasing the correlation between states, observations and actions, which have an important impact on the relevance of the planning. Additionally, reducing the uncertainty of the actions also increases the planning effectiveness. The window-based approach of the grasping point localization algorithm, for example, suffers for its locality, given the simplicity of the descriptors. The localization of the grasping point which is based on a wrinkle analysis cloth model would certainly improve if a linear combination of factors provided a rating over the grasping point. We have empirically shown that the height of the grasping point is a relevant feature over the action outcome, so a grasping point rating could obey the equation

$$r = \boldsymbol{d}^T \cdot \begin{pmatrix} h \\ c \\ w \end{pmatrix} \tag{26}$$

where h stands for height, c for distance to the center of the targeted cloth and w for wrinkleness and $\boldsymbol{d}$ is a weighting vector.

Obviously, any other parameter can be added to the grasping point, without necessarily representing its value within the POMDP tuple. Reducing the uncertainty would affect the transition model of the POMDP, easing the associated uncertainty of the policy.

Also, the current image processing techniques are constrained to color segmentation, which sets a strong constraint on the environment. Given that the planning only needs an estimation of the number of objects, if the robot has to plan on a more general environment, image processing should include a combination of techniques from plain to textured objects to provide the required estimation or even a more informative one.

Additionally, an improvement of the image processing techniques also encourages adding domain-specific information of the state. In this thesis we introduced wrinkleness as a domain feature, although during the model construction the correlation between actions and wrinkleness was not strong enough, it served as an example of how additional domain data can be added to the state representation.

However, adding information to the state is an expensive process both for the policy computation and the modeling. It has to be kept in mind how the model of the world will be obtained, which is not easy on a partially observable domain as we've commented on Section 10.

Other modifications that can be made include improving the state representation. At the moment the observations consider each pile of cloth independently, which means that state inconsistencies can arise when a piece of clothing falls during manipulation, covering both areas. While the actual policy can handle this situation correctly, there is an inconsistency on the number of objects across the execution. Therefore, rather than understanding each zone separately, using the overall object number estimation, which acknowledges the existence of a single piece of clothing, has a repercussion on the belief distribution. Introducing this idea on the observations, which are in fact the result of a

transforming function from states to observations, would forbid the belief of one piece of clothing on area A and another one on area B when in fact the total object count is just one. Note that the actual model of the world does sustain this criterion if the initial states are fixed to three or less objects, since there's no possible transition that leads to more objects than the initial maximum number.

It is also certainly possible to recognize planning dead ends in situations where the vision or grasping fails repeatedly, hence provoking an immutable state. These situations happen in practice when the current context is highly decoupled from the underlying model to which the policy is computed against. For example, a piece of clothing that gets continuously seen as two pieces of clothing or an unreachable grasping point that gets continuously selected. An immediate solution would be adding randomness on the actions procedure to try to break these situations, although a much more smart approach would recognize the context and would execute a recovery plan that maintains the information previously gathered.

Within the same line of thought, currently when an impossible transition happens the recovery procedure uniformizes the probability over all states and continues planning, issuing a warning. Again, a "surprise" smarter recovery plan would be advisable. Another interesting point is starting from a belief other than the uniform one over the initial state, which certainly affects the policy computation and could provide a recovery start-off.

Finally, further research on on-line POMDP learning techniques to complement the method presented would introduce valuable features to the approach proposed. Reducing or exploring the state-space dynamically would provide a step needed for the easy development and implementation of a POMDP approach to problems, whose main drawbacks at the moment are the complexity and the estimation of the world model.

In that sense, using an over-dimensioned action space together with the POMDP policy computation on a small state space does point out which characteristics are relevant to be coded on the state description/observations. The value distribution analysis provides a measure of the value of each action, as we have applied on the determination of the height, which appeared as an actually relevant feature affecting the transitions between states.

References

[1] Aström, K. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications* (1965), 174–205.

[2] Balaguer, B., and Carpin, S. Motion Planning for Cooperative Manipulators Folding Flexible Planar Objects. In *International Conference on Intelligent Robots and Systems* (2010), pp. 3842–3847.

[3] Bell, M. *Flexible object manipulation.* PhD thesis, Dartmouth College Hanover, New Hampshire, 2010.

[4] Bilmes, J. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Tech. rep., University of Berkeley, 1998.

[5] Cassandra, A. A Survey of POMDP Applications. *Working Notes of AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes* (1998), 17–24.

[6] Dung, L. T., Komeda, T., and Takagi, M. Solving POMDPs with Automatic Discovery of Subgoals. In *Theory and Novel Applications of Machine Learning* (2009), no. February, pp. 229–238.

[7] Eidenberger, R., and Scharinger, J. Active Perception and Scene Modeling by Planning with Probabilistic 6D Object Poses. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2010), pp. 1036–1043.

[8] Fischler, M. A., and Bolles, R. C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM 24*, 6 (1981), 381–395.

[9] Fuemmeler, J. a., and Veeravalli, V. V. Energy Efficient Multi-Object Tracking in Sensor Networks. *IEEE Transactions on Signal Processing 58*, 7 (July 2010), 3742–3750.

[10] Ghahramani, Z., and Hinton, G. E. The EM Algorithm for Mixtures of Factor Analyzers. Tech. rep., University of Toronto, 1997.

[11] Hoey, J., Khan, S., Mihailidis, A., Czarnuch, S., and Jackson, D. Relational Approach to Knowledge Engineering for POMDP-based Assistance Systems with Encoding of a Psychological Model. In *21st International Conference on Automated Planning and Scheduling (ICAPS)* (2011), pp. 77–84.

[12] Hoey, J., Poupart, P., Boutilier, C., and Mihailidis, A. POMDP Models for Assistive Technology. Tech. rep., University of Toronto, 2005.

[13] Jiménez, P. Survey on Model-based Manipulation Planning of Deformable Objects. Tech. rep., Institut de Robòtica i Informàtica Industrial, 2010.

[14] Jung, H., and Pedram, M. Uncertainty-Aware Dynamic Power Management in Partially Observable Domains. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 17*, 7 (July 2009), 929–942.

[15] Kim, D., Kim, J. H., and Kim, K.-e. Robust Performance Evaluation of POMDP-Based Dialogue Systems. *Language 19*, 4 (2011), 1029–1040.

[16] Kurniawati, H., and Hsu, D. Motion planning under uncertainty for robotic tasks with long time horizons. *The International Journal of Robotics Research 30*, 3 (Dec. 2010), 308–323.

[17] Kurniawati, H., Hsu, D., and Lee, W. S. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *Robotics: Science & Systems* (2008).

[18] Li, H., and Han, Z. Dogfight in Spectrum: Jamming and Anti-Jamming in Multichannel Cognitive Radio Systems. In *IEEE Global Telecommunications Conference* (Nov. 2009), IEEE, pp. 1–6.

[19] Maitin-Shepard, J., Cusumano-towner, M., Lei, J., and Abbeel, P. Cloth Grasp Point Detection based on Multiple-View Geometric Cues with Application to Robotic Towel Folding. In *Interpreting* (2010).

[20] Matsuno, T., and Fukuda, T. Manipulation of Flexible Rope Using Topological Model Based on Sensor Information. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Oct. 2006), 2638–2643.

[21] McCallum, A. *Reinforcement learning with selective perception and hidden state*. PhD thesis, University of Rochester, 1996.

[22] Meuleau, N., Kim, K.-e., Kaelbling, L., and Cassandra, A. Solving POMDPs by searching the space of finite policies. In *15th Conference on Uncertainty in Artificial Intelligence* (1999), Citeseer, pp. 417–426.

[23] Ong, S. C. W., Hsu, D., Lee, W. S., and Kurniawati, H. Partially Observable Markov Decision Process (POMDP) Technologies for Sign Language based Human-Computer Interaction. In *5th International Conference on Universal Access in Human-Computer Interaction. Part III: Applications and Services* (2009), pp. 577–586.

[24] Park, J., Kim, K.-e., and Jo, S. A POMDP Approach to P300 Brain-Computer Interfaces. In *15th International Conference on Intelligent User Interfaces* (2010), pp. 1–10.

[25] Pascal, J., Poupart, P., Boutilier, C., and Mihailidis, A. Semi-supervised learning of a POMDP model of Patient-Caregiver Interactions. In *IJCAI Workshop on Modeling Others from Observations* (2005), pp. 101–110.

[26] Pineau, J., Gordon, G., and Thrun, S. Anytime Point-Based Approximations for Large POMDPs. *Journal of Artificial Intelligence Research 27* (2006), 335–380.

[27] Regan, T. J., Chadès, I., and Possingham, H. P. Optimally managing under imperfect detection: a method for plant invasions. *Journal of Applied Ecology 48*, 1 (Feb. 2011), 76–85.

[28] Rooks, B. The harmonious robot. *Industrial Robot: An International Journal 33*, 2 (2006), 125–130.

[29] Ross, S., Pineau, J., Paquet, S., and Chaib-Draa, B. Online Planning Algorithms for POMDPs. *The journal of artificial intelligence research 32*, 2 (July 2008), 663–704.

[30] Rusu, R. B., Cousins, S., Garage, W., and Park, M. 3D is here: Point Cloud Library (PCL). In *International Conference on Robotics and Automation* (2011).

[31] SALLEH, K., SEKI, H., KAMIYA, Y., AND HIKIZU, M. Inchworm robot grippers in clothes manipulation — optimizing the tracing algorithm. *International Conference on Intelligent and Advanced Systems* (Nov. 2007), 1051–1055.

[32] SHANI, G., BRAFMAN, R. I., AND SHIMONY, S. E. Model-Based Online Learning of POMDPs. In *European Conference on Machine Learning* (2005), pp. 353–364.

[33] SHANI, G., BRAFMAN, R. I., AND SHIMONY, S. E. Learning POMDP Models Using Noisy Sensors. *25th International Conference on Machine Learning* (2008), 1–44.

[34] SHIBATA, M., OTA, T., AND HIRAI, S. Wiping motion for deformable object handling. In *IEEE International Conference on Robotics and Automation* (2009), IEEE International Conference on Robotics and Automation-ICRA, pp. 1693–1698.

[35] SMITH, T., AND SIMMONS, R. Point-Based POMDP Algorithms: Improved Analysis and Implementation. In *Int. Conf. on Uncertainty in Artificial Intelligence (UAI)* (2005).

[36] SONDIK, E. J. The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research 26*, 2 (1978), 282–304.

[37] TOMBERLIN, D. Endangered Seabird Habitat Management as a Partially Observable Markov Decision Process. 93–104.

[38] WATKINS, J. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge, 1989.

[39] WEI, P. *Factored POMDPs with Mixed Observability*. PhD thesis, National University of Singapore, 2008.

List of Tables

List of Figures

Appendices

A Annexes

A.1 Robot Operating System (ROS)

ROS is an open-source, meta-operating system that provides hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS is similar in some aspects to other *robot frameworks*, such as Player, YARP, Orocos, CARMEN, Orca, MOOS, or Microsoft Robotics Studio.

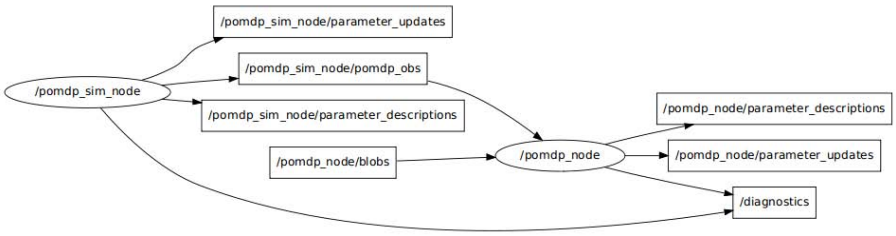


Figure 37 – ROS runtime simulation graph. The simulation node offers action services and observation publishers –which both can be replaced by real manipulator and vision nodes– which are requested by the POMDP node, that stores the internal representation of the POMDP. The parameter server provides the same model to both nodes.

The use of a robotic communication framework simplifies the development phase of robotics. The decoupled design provides flexibility upon the different connection structures, which also helps debugging and development of individual elements within the overall structure. The work can, therefore, focus less on engineering and more on research, which motivates the use of such a framework. Among the alternatives available, at the moment ROS provides a set of valuable tools such as Point Cloud technology and filters, robot coordinate frames integration and many debugging and visualization tools (For example, Figure 38).

The ROS runtime *graph* (Figures 37 and 39) is a peer-to-peer network of processes that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over Services, asynchronous streaming of data over *Topics*, and storage of data on a *Parameter Server*.

In the future, the simulation of such a process can be carried out with the ROS's robot simulation and visualization adopted project Gazebo; which will be useful to tune up the POMDP model and Solver parameters.

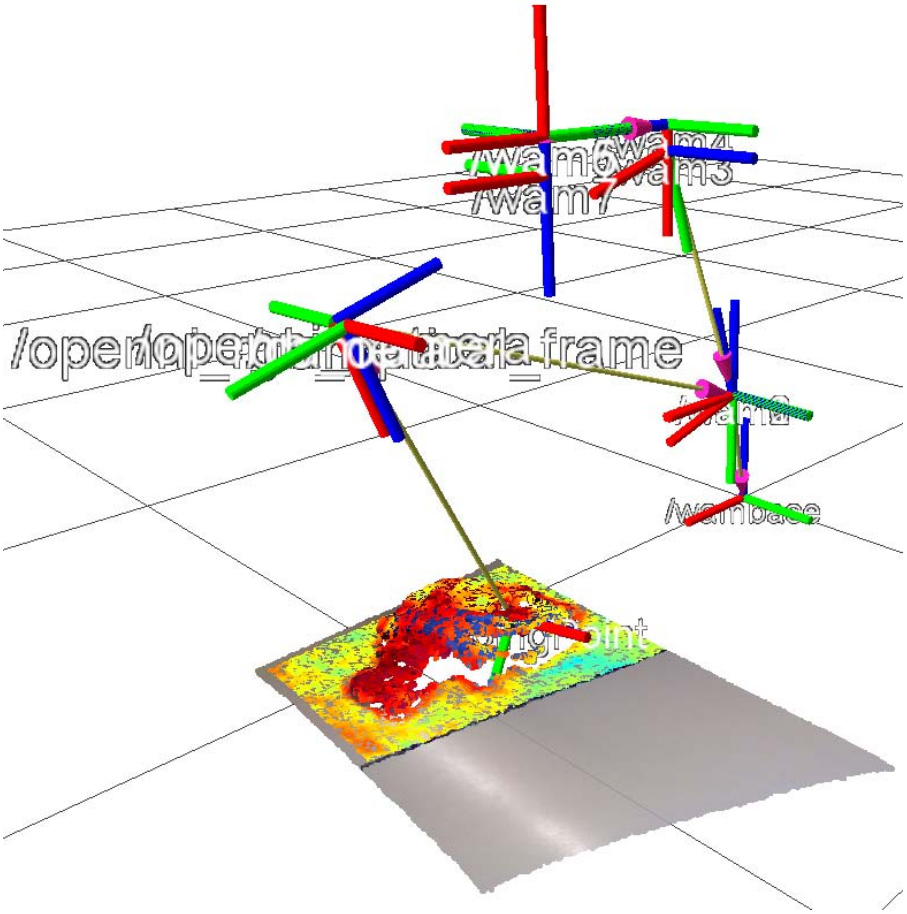


Figure 38 – ROS Robot Visualization environment (RViz). The visualization keeps track of the published robot frames and the the relative position of the camera (based on the hand-eye output). The Point Cloud and superimposed color imagery provided by the Kinect, altogether with the wrinkle map computed, is shown within the environment. The image shows the grasping point frame obtained.

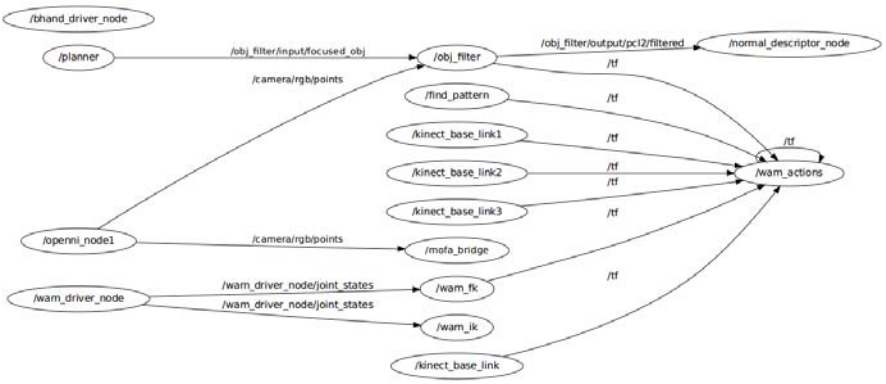


Figure 39 – ROS runtime graph at execution. The planner requests actions to the *wam actions* node asynchronously, which calls the WAM and Barrett Hand driver. The grasping points are obtained through the object filter node and the normal descriptor node, which analyze the wrinkleness of the scene. Finally, the observations are provided by Mixture of Factor Analyzers node which communicates with the Matlab.

A.2 Work Plan

The thesis planning outline has been approximately as follows:

1. A review of the state of the art of methods for modeling together with manipulation of deformable objects. (4 weeks)
2. Problem portability towards the POMDP framework planning. (3 weeks)
3. Design and implementation of a POMDP tuple for the deformable objects manipulation. (9 weeks)
   (a) Implementation of the manipulation actions (3 weeks)
   (b) Implementation of the observation algorithms (2 weeks)
   (c) Policy computation and model estimation (3 week)
4. Establishment of the sets of experiments and planning test (1 week)
5. Results retrieval. (2 weeks)
6. Writing of the memoir. (3 weeks)