Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Fachbereich Informatik
Department of Computer Science

MASTER'S THESIS PROPOSAL

Improving the Manipulation Skills of Service Robots by Refining Action Representations

Author:
Aleksandar Mitrevski

Supervisors
Prof. Dr. Paul G. Plöger
Prof. Dr. Kurt-Ulrich Witt
M. Sc. Anastassia Küstenmacher

June 29, 2015

1 Introduction

Tasks that service robots are expected to execute fall in the domain of problems that are surprisingly complex for automata, even though they are usually taken for granted by us. More specifically, various ostensibly simple actions, ranging from placing solid objects on flat surfaces to manipulating multiple objects for achieving a single goal, require a fair amount of knowledge about the task environment's configuration and the manipulated object's (and occasionally objects') properties if they are to be executed successfully. In order to supply robots with appropriate task-related knowledge, designers usually provide a set of logical rules that cover action preconditions and postconditions in some detail. The rule bases used for this purpose are expected to cover most of the aspects relevant for successfully executing an action and appropriately updating a robot's internal state. Depending on the level of detail that the representation is supposed to possess, different types of logic can be utilised for action specification and reasoning [1].

Rule-based action execution takes two important properties for granted - completeness and predictability. Due to the completeness assumption, robots are made to believe that everything they need to know is contained in the knowledge base and that other aspects are irrelevant for a given task. The predictability assumption is closely related to completeness, implying that different trials of a task are governed by the same set of rules. Both of these assumptions have a direct impact on how robots react to situations that are simply not matching those covered by the rule base, limiting the robots' ability to act in a manner that matches the dynamic properties of the world more closely. Some of the limitations of explicit rule-based task modelling can be overcome by applying learning from demonstration [2]. This paradigm allows robots to mimic the actual behaviour of a human or a simulated teacher, which means that they do not have to rely on a potentially incomplete hand-made rule set, but can instead make use of automatic learning algorithms for extracting relevant task-related details and learning appropriate policy functions [3].

It would have been nice if policies could always guarantee the success of actions; however, their existence is not a sufficient condition for that purpose. In other words, even if a robot is able to generate appropriate action plans, actions can still fail because of external factors that policies cannot take into account. For instance, a robot that needs to place an object on a flat surface might know the actions that it needs to perform in order to get the object there, but might lack knowledge of other important aspects, such as the presence of other objects on the surface. Similarly, a robot that should put a book on a bookshelf might have a perfect knowledge of the manipulative actions that are necessary for achieving its goal, but could be missing the information that other books are occupying the shelf. In both of these cases, additional goal-related knowledge is needed for the purpose of increasing the success likelihood.

Goal-related knowledge might seem like a very general term whose specific requirements depend on the properties of a given task. Even though this intuition is frequently correct, it can be argued that seemingly different tasks can in fact be described by a set of general features that appear over and over again: the placing task that we mentioned previously depends on distances and alignments; putting a power cable in a socket also depends on a specific alignment, even though it is not necessarily described by distances, and so forth. Capturing such common characteristics and trying to allocate them wherever appropriate is an

important aspect that could potentially simplify the process of learning new tasks by limiting the hypothesis space traversed by a learning algorithm.

The two assumptions made by rule-based reasoning systems and the limitations of policy functions are not the only factors that decrease the utility of robots in general everyday scenarios. Given that rule bases and policies are tailored at specific tasks, it can be difficult to make use of them in closely related situations that are labeled as completely different because of a small set of distinctive properties. This implies that a significant portion of the knowledge that could potentially be very useful will remain unused, a fact that could potentially hamper the overall success of a robot on a given task. Unlike rule bases and specialised policies, learning machines are able to interpolate on data, converting the data they have been trained on into an implicit knowledge base that could potentially generalise more easily. As discussed at length in [4], generalisation is the pinnacle of learning systems, but its effective use in action execution is still something to be wished for. Finding a generic set of task features can be seen as an aspect that might be a step in the right direction.

2 Use Cases

In order to elucidate the above discussion, we can consider a few sets of use cases that service robots might be involved in. The use cases are divided into four groups consisting of a few related tasks, a classification that will help us understand why we might want to generalise between variations of them.

1. *Peg-and-hole tasks*: The peg-and-hole problem, which requires putting one object inside another, can be encountered in different forms: stacking cups, putting a pencil in a pen holder, a key in a keyhole, and a power plug in a socket are all variations of it. Some isolated work for such tasks already exists, such as [5–7], but an acknowledgment of the similarity between the tasks and a component that utilises it have not appeared in the literature yet. In other words, the solutions to such problems are generally tailored to one particular problem instance, leading to components with a low degree of reusability and extensibility.

2. *Positioning an object between other objects*: Service robots might also need to execute actions that include not only the target objects, but static objects on the target surface as well. For instance, placing a book or a DVD case on a shelf might include other objects on the surface that will eventually have a direct contact with the manipulated object; placing a cup between two other cups, on the other hand, is a very similar problem, but does not always involve a contact between the cups. All of these task variations require a similar set of actions in order to be executed successfully, but are nevertheless characterised by their own complicating intricacies: the book case is, for instance, conceptually more difficult than the cup case, requiring a very precise alignment with the free space.

3. *Putting sugar in a cup of coffee*: Putting a spoon or cube of sugar is a speciously easy task that can go wrong in various different ways: a cube could be released from a large height, causing some liquid

to spill out, a spoon can be incorrectly turned next to the cup, the action of filling a spoon with sugar might be unsuccessful, and so forth. In order to accomplish the task, a robot would need to know all aspects that might hamper the execution, combining them efficiently in the planning process. Instead of attempting to encode the aspects manually, it could be more effective to allow a generalisation component to learn the intricacies by itself.

4. *Loading a fridge door*: Putting objects on a fridge door is conceptually similar to putting objects on any other surface, but with a slight complicating difference: objects need to be tilted if the task is to be executed successfully. This might not seem like such a big issue, but it does affect how various objects are manipulated. In general, objects cannot be treated equally when it comes to putting them on a fridge door as they usually occupy a different area and require a specific alignment with the target surface.

Even though all of the above use cases can certainly be considered separately due to their individual complicating properties, we might as well argue that they can be analysed together with the help of generic primitives. Regardless of whether those primitives are purely geometric or physical properties, their appropriate identification and utilisation could potentially ease the process of reliably learning both known and previously unseen tasks.

3 A Thought Experiment: Discarding vs. Learning Task-Related Knowledge

Robots can execute tasks even without a real learning and generalisation element, provided that we incorporate enough of our knowledge into their components and use combinations of it in a clever way [4]. Whether that is a sensible idea is another story. A simple thought experiment could help us understand this aspect: let us suppose that we have a two-dimensional robot whose task is placing a square on a rectangular surface that could potentially be occupied by other square objects. Assuming that we are not using a learning element, we could solve the problem in various different ways, for instance by utilising a large set of execution rules or by employing a sampling-based approach. For the purpose of the discussion, let us assume that we have decided to use a sampling-based method that operates as illustrated in Algorithm 1.

Provided that a solution to the problem exists (in other words, that there is enough room on the surface for our object) and assuming that we generate enough samples, a solution to the problem can always be found. An illustration of this conjecture is provided in Figure 1, which depicts different implementations of the *orderSamplesByUtility* function: 1a depicts a situation in which the function is trying to minimise the distance to the robot, 1b illustrates a case wherein the distance to the robot is minimised and the distances to the other objects are maximised, 1c shows results of a function that tries to maximise the distance to the closest surface edge in addition to minimising the distance to the robot and maximising the distance to the other objects, and 1d represents an even more complex scenario in which object orientations are also taken

Algorithm 1 Pseudo-code of a sampling-based function for placing objects

```
1: function FINDPLACEMENTPOSITION(object, surface, obstacles, numberSamples)
2:     samples ← list()
3:     for i ← 1 to numberSamples do
4:         sample ← generateSample(surface)
5:         samples.append(sample)
6:     end for
7:     targetPosition ← orderSamplesByUtility(samples, object, surface, obstacles)
8:     return targetPosition
9: end function
```

into account. In all of these figures, the surface is represented by an orange rectangle, the sample with the highest utility and the other objects on the surface are depicted as green and red squares respectively, and the robot is plotted as a blue dot.

As these figures demonstrate, even a fairly simple approach can go a long way in solving our two-dimensional variation of the object placement problem, although the provided solution can sometimes be slightly counterintuitive (such as the one in the first plot of Figure 1a's third row, where it would have been more natural to place the object in the large free area next to the top left object). In addition to that, such practically hard-coded solutions can also be evaluated efficiently, allowing a robot to make a decision relatively quickly.

Unfortunately, these observations do not take all important task execution aspects into account. First of all, even if we know how to solve one problem successfully, we might not have a way to reuse the known facts in related situations, unless there is yet another set of rules that takes the scenario similarity into account and models the peculiarities of individual tasks. In addition to that, subtle dynamic changes in the robot's workspace might invalidate certain task assumptions, thereby preventing a robot from successfully executing an action. Finally and most importantly, solutions such as the one used in our example discard a large amount of potentially useful data; in other words, all individual trials of a task are treated independently and the common aspects of different trials are simply ignored.

One way to avoid such a waste of knowledge is to summarise different experimental trials with a function that captures important task-related aspects and then train an algorithm in order to capture the distribution of those functions. One such useful function, which combines the probability of success and the desirability given that the object is placed at a given position on the table, is given in Figure 2. If we attempt to learn such a function and the trials used for training the learning machine are descriptive enough of the analysed task, a success probability function will explicitly capture the most salient properties describing the problem, allowing us to reason about the generated solution at a much higher level.

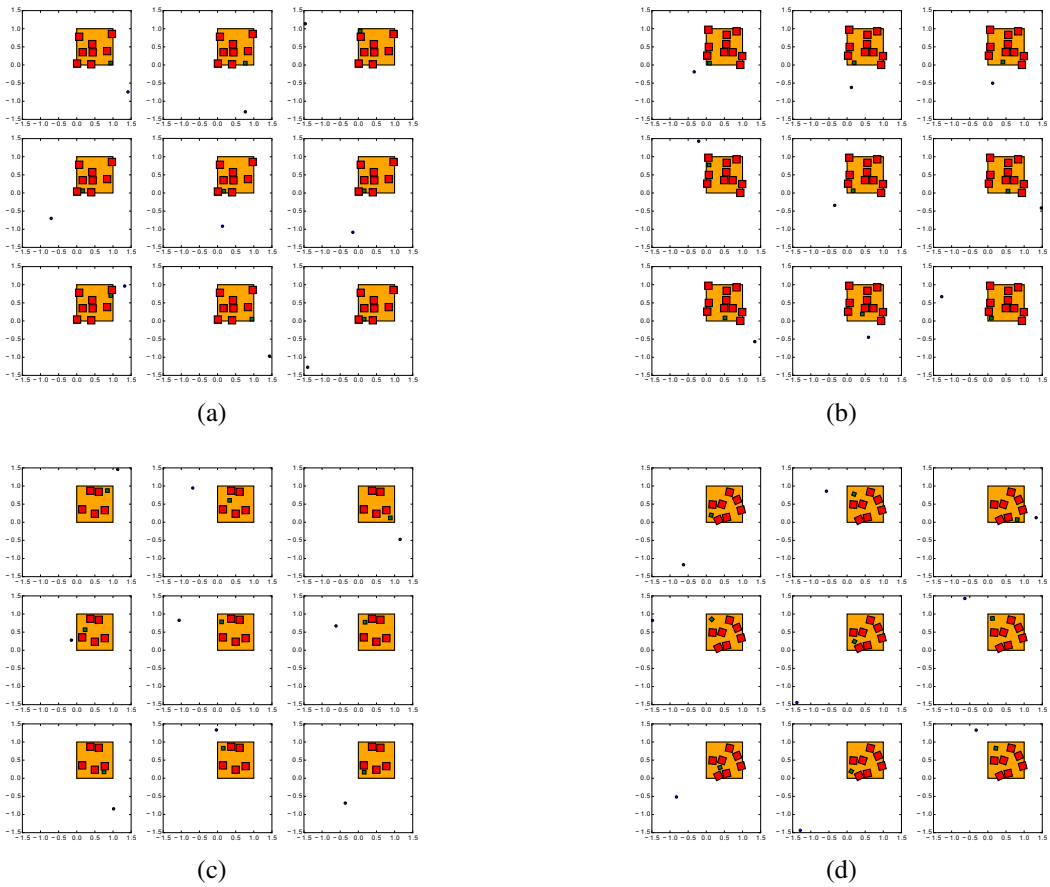


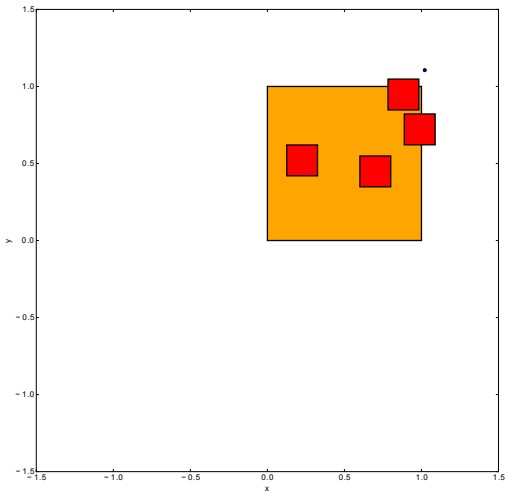
Figure 1: Different implementations of the *orderSamplesByUtility* function. a) Minimising the distance to the robot. b) Maximising the distance to the other objects in addition to a). c) Maximising the distance to the closest surface edge together with b). d) Taking the orientation into account.

4 Problem Statement and Expected Contributions of the Thesis

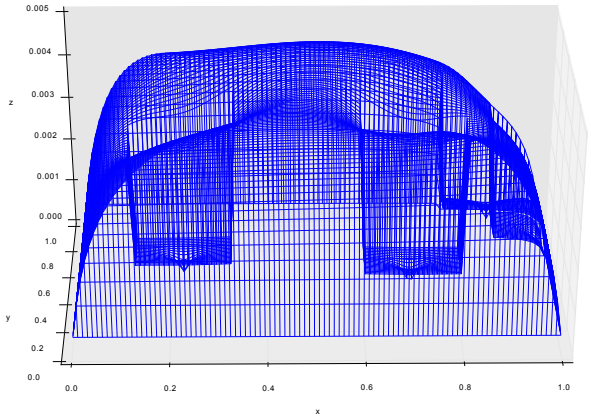
This thesis will investigate the final aspect of the action execution cycle, namely the one concerned with the question about the actual success of an executed action. In that respect, the project will be concerned with the following subproblems:

1. *Creating a self-sufficient set of features that is as general as possible for a wide range of tasks*: The claim here is that even though robots are expected to perform several seemingly unrelated tasks, the execution frequently depends on common factors, such as distances (both linear and angular), vector projections, relations between positions, and so forth. We would therefore like to examine these frequently encountered properties in order to come up with a general feature set whose subsets can describe various common tasks in service robotics.

2. *Training task-specific models by extracting appropriate subsets from the previously assembled feature set*: The power of any generalised task description can be measured by the amount of tasks it can



(a)



(b)

Figure 2: A function combining the probability of success and the desirability for various positions on the target surface. a) Table configuration. b) Success probability function.

successfully represent. Learning task descriptions and examining their generalisation capabilities across task variations can thus help us understand the soundness and completeness of our feature set. In order to achieve this goal, we would like to learn models that quantify the expected success of actions given a task description. Such models should resemble the ones described in section 3.

3. *Generalising tasks for the purpose of reusing trained models in as many related scenarios as possible*: More efficient reasoning depends upon the generalisation capabilities of an agent, among other things. The utility of task-specific models is large as long as the agent executes the same task over and over again, but its value diminishes once a previously unseen problem is encountered. Various problems are quite related to each other, however: placing a bottle on a table is very similar to putting it on the door of a fridge, plugging in a power cable and putting a key into a keyhole depend on similar preconditions, and so on. Utilising the model trained on one task in order to execute a related one is the final and most complicated aspect of the problem that we would like to investigate in the project.

The investigation of these aspects will be governed by the following assumptions:

- We assume that our agent has a selective attention towards the region in space where a task needs to be executed; we call this set of points the *feasibility set* for a given task:

$$F = \{p \in \mathbb{R}^3 | (p_{x_{min}} \leq p_x \leq p_{x_{max}}) \wedge (p_{y_{min}} \leq p_y \leq p_{y_{max}}) \wedge (p_{z_{min}} \leq p_z \leq p_{z_{max}})\} \quad (1)$$

Considering the peg-and-hole use case as a specific example, the feasibility set could be a prism that includes the region around the hole; other objects that might be in the hole belong to F as well. In fact, the name *feasibility set* does not necessarily mean that an action can be successfully executed at

any of the positions in F; on the contrary, the set might also contain positions at which the execution would lead to failure. Nevertheless, F can be considered to be the closest approximation to where the robot should look for feasible execution poses, which is the reason for the set's name.

- Objects are grasped with a controller that we do not have control over. This means that we are working with the object as originally grasped and are not considering regrasping strategies.
- An approximate pose of the manipulated object can be calculated even though the grasp itself might partially block the object.
- We are only considering one action, namely the object releasing action. Other actions, such as pushing objects in order to make free space on a target surface, will not be dealt with.

The examination of the three problems mentioned above will hopefully lead to a better understanding of the complexity involved in properly describing and evaluating solutions to common tasks encountered in service robotics. In particular, the completeness of the feature set - or the lack thereof - should demonstrate whether manual compression of task descriptions can lead to successful task-specific models, also providing information about the potential utility of automatic feature identification procedures as alternative solution approaches; the investigation of the task specific models and their training complexity will be informative with respect to whether it is feasible to learn functions that describe complete task configurations rather than single solutions; finally, the generalisation aspect should shed some light on the practical difficulty of this problem, offering some hints about the abstraction level at which we should analyse tasks in order to uncover their underlying similarities.

Before closing this section, we should mention that the first two problems will form the core of the thesis, while the third one is more far-fetched and there is even some small likelihood that we might not start working on it in this project at all or that it will only take up a small part of the work. Whether we do manage to start the analysis of generalisation will depend on how promising the results produced by the other two segments are.

5 Proposed Approach

5.1 General Description

Our problem description clearly hints to the learning aspect of the proposed work. Given that any learning scenario requires appropriate training data and being aware of the potential costs of performing experiments with real robots, we propose the usage of a simulation environment for generating training data for our models. We are aware that utilising simulations rather than actual robots has its perils, but we have to mention that simulations have proven useful in various situations, such as for example [8, 9]; in addition to that, simulations and their underlying physics engines are fortunately becoming slightly more realistic ([10, 11] are examples of this trend). Our ultimate desire is certainly deploying our work on a real service robot;

however, we suggest the utilisation of a simulation because we would like to have a greater experimentation flexibility while searching for a solution to the tackled problems; moreover, we want to test the possibility of applying the simulated results to an actual robot, thereby examining the discrepancy between our simulated scenarios and their real world counterparts. To that end, we propose using the Unreal Engine 4 [11] as a simulation engine and its PhysX 3.3 physics engine.

It would also be ideal if we could cover a lot of service robotics tasks in our work; unfortunately, we have to make compromises due to the duration of the project and narrow the scope to a manageable level. As a result, our work will only focus on the scenarios mentioned in section 2. So far, we have only created a proof of concept simulation of a scenario dealing with object placement on a flat surface; Figure 3 shows eight tables with different configurations that are simulated in parallel. Once the project commences, we will start simulating the actual use cases that our work is interested in.

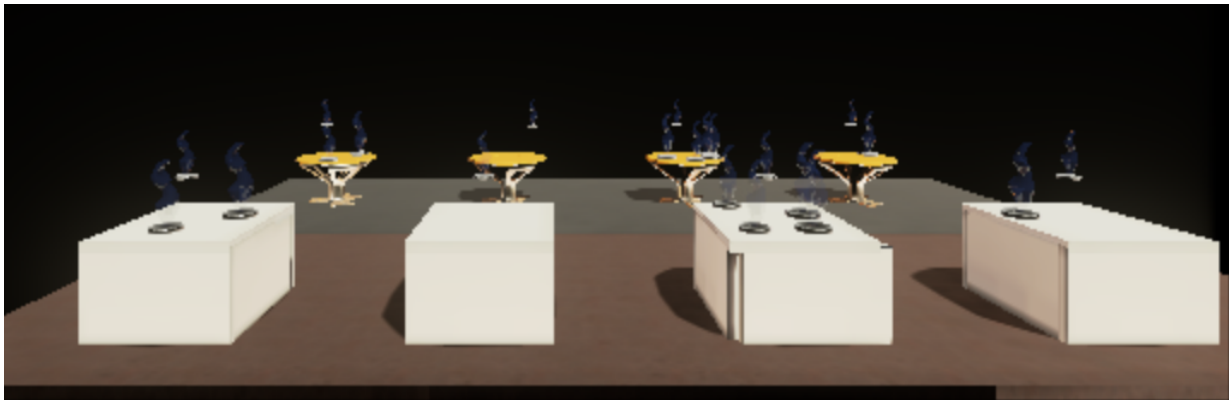


Figure 3: Simulation of the first use case described in section 2.

As mentioned in section 4, the first problem that we are concerned with is that of identifying a set of features whose subsets can be used for describing a range of different tasks. In order to create our feature set, we suggest building our work upon [12], where predefined geometric predicates are used for describing actions and their expected effects in order to avoid external faults. The goal of our approach will be to start with these geometric predicates as initial high-level features and increase the set if tasks require additional information that the predicates do not or cannot capture.

Having a feature set at our disposal does not imply that we know how to prove the completeness of such a set theoretically; we do, however, have a rough idea about a three-step numerical procedure that might be useful for that purpose:

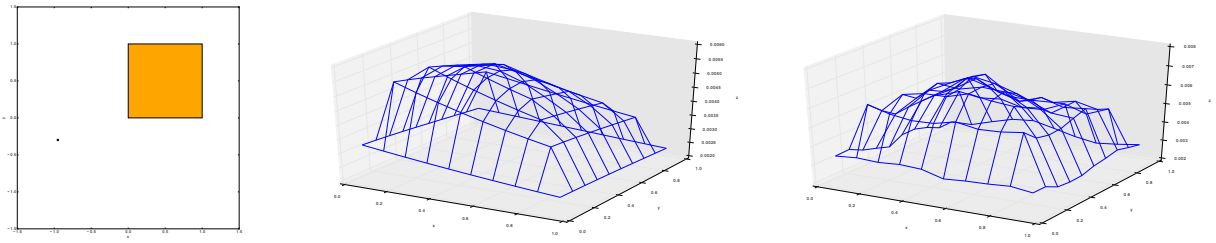
1. Generate a feature set based on our analysis of the tasks.
2. Train models using the feature set and calculate some quality measure (such as the success rate of the calculated solutions).
3. Augment the feature set as long as the quality rate is below some predefined threshold.

Algorithm 2 defines this procedure in more detail. The algorithm's *while* loop can be either automatic (if a feature identification procedure is used) or manual; as mentioned in section 4, we are planning on using a manual procedure here. The same argument holds for the *findFeatures* function. In order to find appropriate features for our tasks, we will need to rely on a feature selection procedure; one possible candidate for feature selection could be [13], although we will have to identify the most appropriate one or even develop our own procedure once our work initiates.

Algorithm 2 Pseudo-code of a procedure for evaluating the feature set quality

```
1:  function CREATEGENERALFEATURESET(tasks, threshold)
2:      quality ← 0
3:      featureSet ← {}
4:      taskQualities ← {}
5:      while quality < threshold do
6:          featureSet ← findFeatures()
7:          for task in tasks do
8:              model ← train(task, featureSet)
9:              taskQualities ← taskQualities ∪ calculateQuality(model)
10:         end for
11:         quality ← calculateQuality(taskQualities)
12:         taskQualities ← {}
13:     end while
14:     return featureSet
15: end function
```

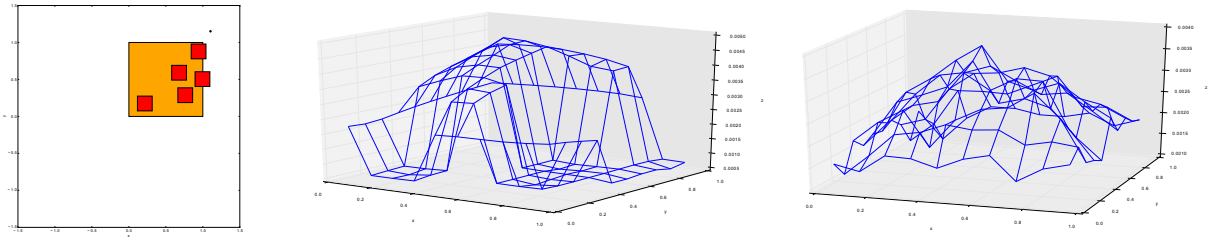
Some initial work regarding the problem of learning a success/utility probability function for the simulated use case has also been done, albeit for a two-dimensional variation of the problem that ignores certain properties of the configurations, such as the orientations of the present objects, and uses a hand-made feature set suited to this particular problem. Figures 4 and 5 show the learned function and its ideal counterpart for two different table configurations. Models of this type are what we are going to aim for in our work.



(a) (b) (c)

Figure 4: Learning configuration utility functions. a) Table configuration. b) Ideal function representation given a manual feature set. c) Learned function.

The ultimate goal of learning success/utility models such as the ones depicted above is improving the representation of corresponding actions, thereby increasing the likelihood that the subsequent execution



(a) (b) (c)

Figure 5: Learning configuration utility functions 2. a) Table configuration. b) Ideal function representation given a manual feature set. c) Learned function.

will be successful. Speaking in terms of the predicates defined in [12], we are ultimately aiming at a more reliable representation of the S_{init} state, whose purpose is describing the state an object should have before an action is being executed.

We have not started working on the generalisation problem yet; as previously mentioned, whether we proceed to it will depend on our success in the other segments of the project. Nevertheless, we do have some ideas as to how we might approach the solution: one way would be to analyse natural language action commands and attempt to compare them on this level; another way could be to first translate natural language to a set of logical rules, as for instance done in [14], and then use the deduced rules for comparing different tasks. These are, however, just ideas at this stage and will need to be developed further when the time comes.

5.2 Theoretical Approach

Given that our work is focused on manipulation, objects are the entities that we are directly concerned with. In order to define the term *object*, we have been considering three different representations that are able to express distinct properties of an object O:

- A parameterised surface (such as a bounding box) or a set of parameterised surfaces that together approximate an object (tangent planes, for instance).
- A point cloud representation, according to which O is given by N three-dimensional points:

$$O = \{p_i | p_i \in \mathbb{R}^3, 1 \leq i \leq N\} \tag{2}$$

- A gradient field, i.e. a discrete set of gradient vectors at N different points on the object's surface:

$$O = \{\overrightarrow{g_i} | \overrightarrow{g_i} = \nabla O_{p_i}, 1 \leq i \leq N\} \tag{3}$$

In order to get the best of all worlds, the most optimal idea might be to combine these representations. For this reason, we are not fixing the definition of an object and our work will use all of the representations as valid definitions (Figure 6).

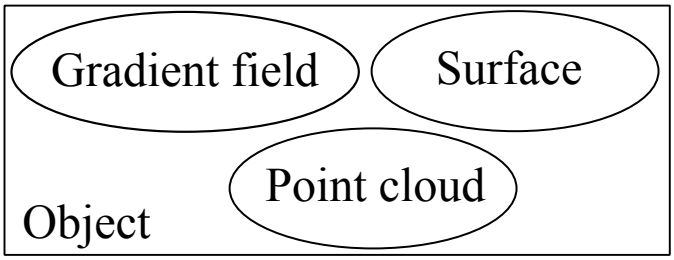


Figure 6: Object representations adopted in our work

It is important to mention that the extent to which any of these object representations can be utilised in practice directly depends on the sensors that a particular robot has at its disposal; nevertheless, the reason for having these different depictions is our desire to remain as general as possible without constraining our work to one particular robot or type of robots.

Following the problem description given in section 4, our first objective is the creation of a predicate/feature set that describes relevant object properties and the relation of an object to its environment. For the purpose of our work, a relation r can be anything from an n-ary predicate to a real-valued function, such that all of them together form a finite set $\mathcal{R}$; this definition of a relation is visually depicted in Figure 7. The purpose of allowing a variety of relations lets us define all sorts of different object proper-

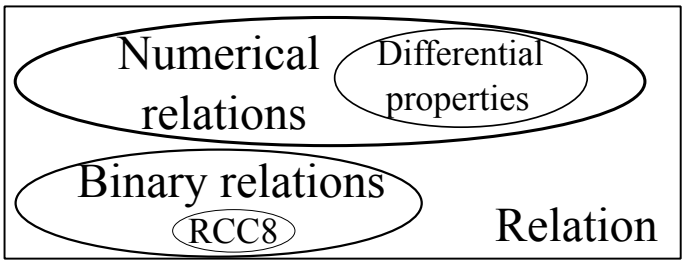


Figure 7: Relation model

ties: binary relations such as $Over(O, O_x)$, numerical ones such as $distance(O, O_x)$, or differential ones ($DirectionalDerivative(O_p, \hat{u})$). As in the case of the object definitions, we would like to remain as general as possible, at least at this stage of the work, which is why we are not limiting the relations to only one type.

Having these definitions and the assumptions mentioned in section 4 at hand, we can define a task T as a function that maps an object and a feasibility set to a pose in space where the robot should act:

$$T(O, F) = \{p | p \in \mathbb{R}^6\} \tag{4}$$

In equation 4, we assume that the object's position is given in Cartesian coordinates and its pose is represented by three orientation angles.

The set of relations consequently allows us to define various properties of the object O with respect to

T, as illustrated in Figure 8. Using equation 4 and the relation set $\mathcal{R}$, our second objective becomes that of

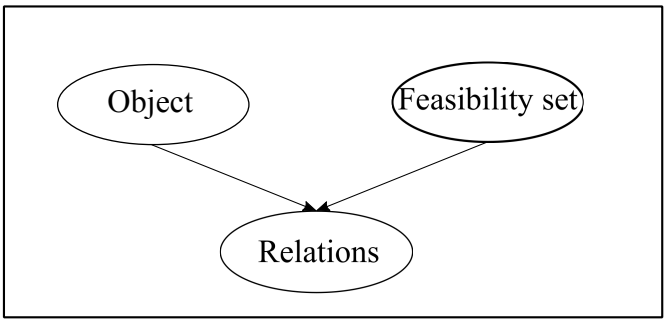


Figure 8: Model of the relations between the manipulated object and the set F

finding a compression function c such that

$$c(\mathcal{R}, T) = \hat{\mathcal{R}},\ \hat{\mathcal{R}} \subseteq \mathcal{R} \wedge u(T, \hat{\mathcal{R}}) = max \tag{5}$$

i.e. we are looking for a subset of the relations in $\mathcal{R}$ that maximises a predefined measure u. We have not yet decided what the actual form of u is going to be, but using a neural network and cross-validation for maximising the success prediction accuracy is one possibility that we are currently considering.

Finding the subset $\hat{\mathcal{R}}$ is what will ultimately help us improve the S_{init} attribute; however, given that the relations in $\mathcal{R}$ do not have a uniform representation, it might not be possible to use $\hat{\mathcal{R}}$ directly. In addition to that, as already mentioned previously, we would like to learn a smooth representation that evaluates the complete feasibility set. In order to accomplish that, the process of finding $\hat{\mathcal{R}}$ will need to be followed by a procedure of identifying a function f that maps the positions in F to a real-valued output signifying the positions' utilities with respect to the given task. One way to achieve this goal would be to use a similar approach as the one in [15], i.e. learn separate evaluation functions for each of the predicates in $\hat{\mathcal{R}}$

$$f_i = (F, \hat{\mathcal{R}}_i) \rightarrow \mathbb{R},\ 1 \leq i \leq |\hat{\mathcal{R}}| \tag{6}$$

and then combine the individual results into one uniform representation

$$e = f(f_1, ..., f_{|\hat{\mathcal{R}}|}) \tag{7}$$

that subsequently allows the agent to discover a position p or a set of positions P where the likelihood of task success is maximised.

Finally, the generalisation problem can be defined as one of finding a function $\hat{f}$ that is derived from f and is used for solving a task that varies in either the manipulated object or the feasibility set. While we are not yet working on a theoretical framework for this aspect, transfer learning [16] seems to be a paradigm that we might pursue further.

We have to close this section by mentioning that the theoretical framework defined here is not a com-

pletely developed one and will most likely need to be adjusted as our work reaches a more mature state.

6 Project Deliverables and Project Plan

As previously described, the thesis will have three core tasks (feature identification, feature selection, and model learning) and one that we would deeply love to tackle (task generalisation in order to utilise previously learned models). In order to prevent a situation in which too much is promised, the project deliverables are divided into two groups: expected and desired results.

The group of expected results consists of those related to feature identification and model learning, such that the project will be considered fairly successful if these are completely achieved at the end. The set of expected results includes:

- The complete code of the simulated environments and any subsequent analysis procedures in order to facilitate data reproducibility.
- A set of features used for describing various tasks in service robotics, including a description of the concepts on which the features are based and the analysis procedure that led to that set.
- Procedures for:
  - generating task-specific success/utility probability functions using the set of generic task features and
  - updating the representation of the S_{init} state defined in [12].

As a concrete example of what the expected results should ultimately represent, we can consider the *peg-and-hole* use case. For simplicity, let us assume that the S_{init} for this particular case is given as

$$S_{init} : above(peg, hole) \wedge withinBoundary(peg, hole) \wedge (dot(peg_{\hat{z}}, \hat{k}) = 1)$$

but that the subsequent action fails and the robot needs to repeat it. Our analysis will perhaps discover that the height from which the object is released also affects the success of the executed action and will therefore update S_{init}, whose updated version will be

$$S^*_{init} : above(peg, hole) \wedge withinBoundary(peg, hole) \wedge (dot(peg_{\hat{z}}, \hat{k}) = 1) \wedge (height(peg, hole) < \delta)$$

for some constant δ. The repeated action will then be performed with an increased knowledge of the task and consequently improved success likelihood.

The collection of desired results represents a superset of the expected results, extending this set by a treatment of the generalisation problem and real world applications of the achieved outcomes:

- An analysis of the task generalisation problem and a potential solution whose feasibility will be tested on the use cases described in section 2.

- Comparison of the results produced by our approach to results dealing with tasks that belong to our list of scenarios (perhaps for instance [17] as a representative of the placing objects task or [5] as an example of putting a power cable in a socket).
- A demonstration showing that the functions learned in a simulated scenario can be applied to a real robot that performs several variations of the tasks.

We genuinely hope that our work can reach not only the expected outcomes, but at least some of the desired results as well. Nevertheless, this result classification is necessary in order to avoid a situation in which we are biting off more than we can chew. Figures 9 and 10 make the categorisation explicit by depicting two different project plans corresponding to the sets of expected and desired outcomes respectively; the difference between these two plans is that the latter is simply more optimistic than the former, allocating less time to the work packages corresponding to the expected outcomes and freeing some space for generalisation and comparison.

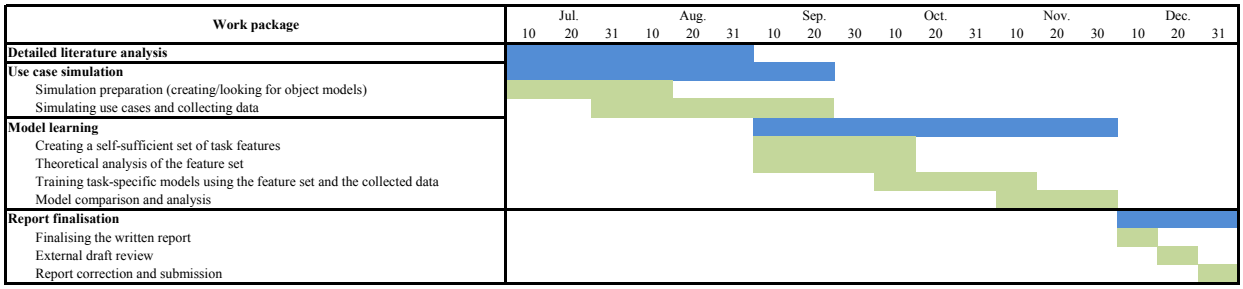


Figure 9: Project plan assuming that only the expected outcomes will be achieved.

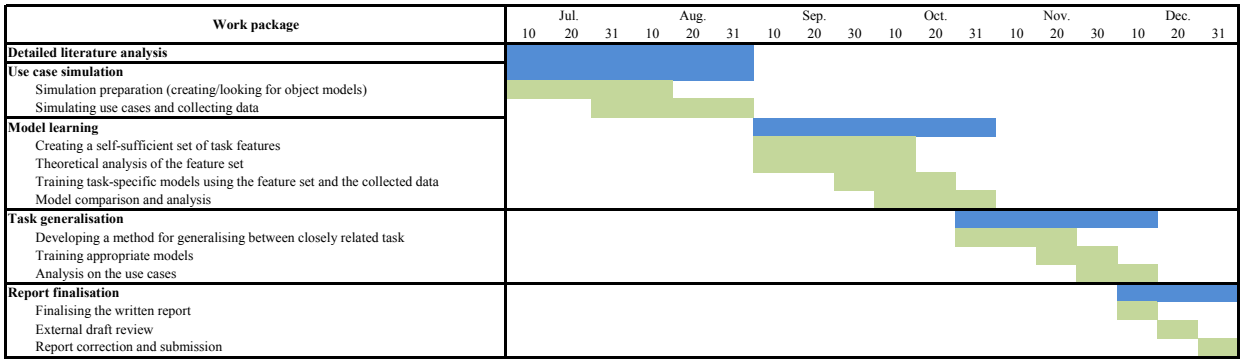


Figure 10: A more optimistic project plan that allocates time for the desired outcomes as well.

7 Related Work

As mentioned in section 5, our feature identification and selection approach will be roughly based on [12] and will mainly aim to improve upon the results presented there. The rest of the work will start from various

problem-specific solutions to different task execution problems and will attempt to bring them together in a somewhat more general solution.

One approach worth mentioning here is [17], which is a simulation-based learning algorithm whose purpose is to discover a stable placing area on a target surface. This algorithm analyses the point cloud of a manipulated object and generates several features that describe the object's stability. In order to reduce the space complexity of the learning model, which in this case is a support vector machine, the authors utilise a shared structure that represents common features between different variations of the placing task. The results of the approach are shown to be more reliable than those obtained by several other solutions, such as a random sampling algorithm, a procedure tailored to flat surfaces, and a solution that always chooses the lowest placing point on the surface. A related paper to this one is [18], wherein several convexity and stability features are applied in order to evaluate potential goal positions on a target surface. The authors in [19] go a step further, such that their algorithm tries to free some space for the manipulated object on the target surface; in other words, they do not take the environment configuration for granted, but allow a robot to modify it based on its needs.

Demonstration learning for manipulation tasks is also a fairly popular training approach in robotics. Active learning is, however, more focused on how a robot needs to perform a task, i.e. what motions are required for executing an action and how sequential actions affect a robot's overall success. Notable examples that belong to this group of approaches are [20], [2], [21], as well as [3], the latter being a survey of applications of learning by demonstration. We believe that our work can be seen as complementary to these, as we are more focused on the actual success of actions, which is just as important as the process of executing the actions themselves.

Attempts to predict the outcomes of various manipulation actions have already appeared in the literature, albeit in the context of learning by demonstration and focused on sequential data analysis [22,23] and dealing only with visual data [24].

We should conclude this section by saying that, as depicted in the project plan, a more detailed literature analysis will be conducted as part of the project, occupying the initial stage of the work. Our preliminary work has mostly been done independently in order to avoid some common cognitive biases that might affect our solution process (such as selective attention, confirmation bias, and appeal to authority [25]) and prevent us from clearly thinking about the problem and its potential solutions. This is, however, not to say that we have completely ignored the existing literature, but merely that we cannot claim a complete mastery of the related work just yet.

References

[1] R. J. Brachman *et al.*, "Knowledge representation and reasoning," 2004. Available: http://www.time.mk/trajkovski/thesis/brachman.pdf.

[2] M. N. Nicolescu and M. J. Mataric, "Natural Methods for Robot Task Learning: Instructive Demonstrations, Generalization and Practice," in *Proc. 2nd Int. Joint Conf. Autonomous Agents and Multiagent Systems*, pp. 241–248, 2003.

[3] B. D. Argall *et al.*, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[4] S. Russell and P. Norvig, "The Structure of Agents," in *Artificial Intelligence: A Modern Approach*, ch. 2, pp. 46–58, Upper Saddle River, NJ: Pearson Education, Inc., 3 ed., 2010.

[5] B. Mayton *et al.*, "Robot, feed thyself: Plugging in to unmodified electrical outlets by sensing emitted AC electric fields," in *Robotics and Automation (ICRA), 2010 IEEE Int. Conf.*, pp. 715–722, May 2010.

[6] W. Meeussen *et al.*, "Autonomous door opening and plugging in with a personal robot," in *Robotics and Automation (ICRA), 2010 IEEE Int. Conf.*, pp. 729–736, May 2010.

[7] S. R. Chhatpar and M. S. Branicky, "Search strategies for peg-in-hole assemblies with position uncertainty," in *Intelligent Robots and Systems, 2001. Proc. 2001 IEEE/RSJ Int. Conf.*, vol. 3, pp. 1465–1470, 2001.

[8] J. C. Bongard and H. Lipson, "Automated Damage Diagnosis and Recovery for Remote Robotics," in *Proc. ICRA '04 IEEE Int. Conf. Robotics and Automation*, vol. 4, pp. 3545–3550, Apr. 2004.

[9] N. P. Koenig and M. J. Mataric, "Training Wheels for the Robot: Learning from Demonstration Using Simulation," in *Robots Learning Interactively from Human Teachers, Papers from the 2012 AAAI Fall Symposium*, 2012.

[10] R. Diankov, *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, Aug. 2010. Available: http://www.programmingvision.com/rosen_diankov_thesis.pdf.

[11] I. Epic Games, "Unreal Engine 4," 2015. Available: https://www.unrealengine.com/unreal-engine-4.

[12] A. Kuestenmacher *et al.*, "Towards Robust Task Execution for Domestic Service Robots," *Journal of Intelligent & Robotic Systems*, vol. 76, no. 1, pp. 5–33, 2014.

[13] G. Wang, Q. Song, H. Sun, X. Zhang, B. Xu, and Y. Zhou, "A Feature Subset Selection Algorithm Automatic Recommendation Method," *J. Artif. Int. Res.*, vol. 47, pp. 1–34, May 2013.

[14] I. Mani and J. Pustejovsky, "Qualitative Modeling of Spatial Prepositions and Motion Expressions," 2012.

[15] F. Stulp, A. Fedrizzi, and M. Beetz, "Action-related place-based mobile manipulation," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ Int. Conf.*, pp. 3115–3120, Oct. 2009.

[16] D. L. Silver *et al.*, "Lifelong machine learning systems: Beyond learning algorithms," in *AAAI Spring Symposium: Lifelong Machine Learning*, 2013.

[17] Y. Jiang *et al.*, "Learning to Place New Objects in a Scene," *Int. Journal Robotics Research*, vol. 31, pp. 1021–1043, Aug. 2012.

[18] K. Harada *et al.*, "Object placement planner for robotic pick and place tasks," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ Int. Conf.*, pp. 980–985, Oct. 2012.

[19] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push planning for object placement on cluttered table surfaces," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ Int. Conf.*, pp. 4627–4632, Sept. 2011.

[20] J. Kulick *et al.*, "Active Learning for Teaching a Robot Grounded Relational Symbols," in *Proc. 23rd Int. Joint Conf. Artificial Intelligence*, pp. 1451–1457, 2013.

[21] N. S. Pollard and J. K. Hodgins, "Generalizing Demonstrated Manipulation Tasks," in *Algorithmic Foundations of Robotics V*, vol. 7 of *Springer Tracts in Advanced Robotics*, pp. 523–539, 2004.

[22] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *Robotics and Automation (ICRA), 2011 IEEE Int. Conf.*, pp. 3828–3834, May 2011.

[23] A. Haidu *et al.*, "Learning Task Outcome Prediction for Robot Control from Interactive Environments," in *Proc. Int. Conf. Intelligent Robots and Systems*, pp. 4389–4395, Sep. 2014.

[24] H. Nguyen and C. K. Kemp, "Autonomously learning to visually detect where manipulation will succeed," *Autonomous Robots*, vol. 36, no. 1-2, pp. 137–152, 2014.

[25] B. Dowden, "Fallacies," 2015. Available: http://www.iep.utm.edu/fallacy/.

[26] J. Ruiz-del Solar *et al.*, "A Bayesian framework for informed search using convolutions between observation likelihoods and spatial relation masks," in *Advanced Robotics (ICAR), 2013 16th Int. Conf.*, pp. 1–8, Nov. 2013.

[27] Y. Jiang and A. Saxena, "Hallucinating Humans for Learning Robotic Placement of Objects," in *Experimental Robotics*, vol. 88 of *Springer Tracts Advanced Robotics*, pp. 921–937, 2013.

[28] B. Rosman and S. Ramamoorthy, "Learning Spatial Relationships Between Objects," *Int. Journal Robotics Research*, vol. 30, pp. 1328–1342, Sept. 2011.

[29] A. Lodi, S. Martello, and M. Monaci, "Two-dimensional packing problems: A survey," *European Journal Operational Research*, vol. 141, no. 2, pp. 241–252, 2002.

[30] S. Niekum *et al.*, "Learning and Generalization of Complex Tasks from Unstructured Demonstrations," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Oct. 2012.

[31] C. Sammut and D. Hume, "Observation and generalisation in a simulated robot world," in *Proc. 4th Int. Machine Learning Workshop*, pp. 267–273, 1987.

[32] S. Calinon, F. Guenter, and A. Billard, "On Learning, Representing, and Generalizing a Task in a Humanoid Robot," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 37, pp. 286–298, Apr. 2007.

[33] M. Kopicki *et al.*, "Learning dexterous grasps that generalise to novel objects by combining hand and contact models," in *Robotics and Automation (ICRA), 2014 IEEE Int. Conf.*, pp. 5358–5365, May 2014.

[34] A. Edsinger and C. C. Kemp, "Manipulation in Human Environments," in *Humanoid Robots, 2006 6th IEEE-RAS Int. Conf.*, pp. 102–109, Dec. 2006.