ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

LEARNING GUIDED PROBABILISTIC PLANNING

M.Sc. THESIS

Melis KAPOTOĞLU

Department of Computer Engineering

Computer Engineering Programme

JUNE 2015

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

LEARNING GUIDED PROBABILISTIC PLANNING

M.Sc. THESIS

Melis KAPOTOĞLU
(504121530)

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Asst. Prof. Dr. Sanem SARIEL

JUNE 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÖGRENME GÜDÜMLÜ OLASILIKSAL PLANLAMA

YÜKSEK LİSANS TEZİ

Melis KAPOTOĞLU
(504121530)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Asst. Prof. Dr. Sanem SARIEL

HAZİRAN 2015

Melis KAPOTOĞLU, a M.Sc. student of ITU Graduate School of Science Engineering and Technology 504121530 successfully defended the thesis entitled **"LEARNING GUIDED PROBABILISTIC PLANNING"**, which she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Asst. Prof. Dr. Sanem SARIEL** Istanbul Technical University	.....
Jury Members :	**Asst. Prof. Dr. Gökhan İNCE** Istanbul Technical University	.....
	Asst. Prof. Dr. Fatih AMASYALI Yıldız Technical University	.....

Date of Submission : **4 May 2015**
Date of Defense : **4 June 2015**

To everyone to whom no thesis has been dedicated before...

FOREWORD

This thesis is supported from my parents, teachers, family, friends, and in essence, all sentient beings. However I feel responsible to dedicate my acknowledgment of gratitude toward especially some of these people. I would like to thank to my advisor, Assistant Prof. Sanem Sarıel for her help during my master education. I sincerely thank to Çagatay Koç, Melodi Deniz Öztürk, Doğan Altan, Sertaç Karapınar, Mustafa Ersen and Berke Kapotoglu and my family for their eternal support, advices and encouragement all the time. This thesis is partly funded by a grant from the Scientific and Technological Research Council of Turkey (TUBITAK), Grant No. 111E286.

June 2015

Melis KAPOTOĞLU
Computer Engineer

TABLE OF CONTENTS

Page

ABBREVIATIONS

MDP	**:** Markov Decision Processes
POMDP	**:** Partially Observable Markov Decision Processes
SARSOP	**:** Successive Approximations of the Reachable Space under Optimal Policies
ILP	**:** Inductive Logic Programming
ROS	**:** Robot Operating System

LIST OF TABLES

Page

LIST OF FIGURES

Page

LEARNING GUIDED PROBABILISTIC PLANNING

SUMMARY

Robots should avoid potential failure situations to improve their performances. The failures which the robot has already experienced in its previous action executions can be used in an adaptive planning strategy to reduce potential failures in the future. Robots need to build and use their experience for achieving this objective. In this thesis, learning and learning-guided planning methods are proposed to address this problem. An experimental learning process using Inductive Logic Programming (ILP) and a probabilistic planning method that uses the experience gained by learning are integrated for improving task execution performance. The solutions are analyzed on a case study with an autonomous mobile robot in a multi-object manipulation domain where the objective is maximizing the number of collected objects while avoiding potential failures using experience. Obtained results indicate that the robot using the adaptive planning strategy ensures safety in task execution and maximizes the probability of success.

As the previous work of author's research group, an ILP-based experiential learning process is used to derive hypotheses associated with probabilities in the presented system. Another prior work is also published which is based on a deterministic planner to use the outcomes of an experimental learning process. Only the contexts of hypotheses which are obtained from the learning process can be used to guide the planner in this work. On the other hand, a probabilistic method is proposed here as a contribution to use the probabilities of hypotheses as well. Here, a Partially Observable Markov Decision Process (POMDP) model is enhanced to create an adaptive policy for the robot to deal with uncertainties. A new point-based algorithm named SARSOP is used as the POMDP planner in this learning-guided planning method. This algorithm differs from other point-based techniques in the way of using heuristic exploration while extracting optimally reachable belief spaces. Therefore, the speed of finding an approximate solution to the planning problem increases.

The selected case study involves mobile manipulation scenarios by an autonomous robot whose goal is to maximize the number of objects transported to a destination while avoiding failures in a given time period. The main contribution of this thesis lies in the POMDP planning formulation developed to use learning outcomes. In the proposed formulation, an object composition is defined to represent encapsulation of qualitative and spatial information about an object. Qualitative information specifies the predetermined attributes (type, color, material, height, width, etc.) and the spatial information represents the semantic location of an object. Different states are created for all combinations of object attributes and semantic locations. The combination sets of all possible object compositions construct the state space of the POMDP formulation. The state definition is expanded with two more states, one of which represents the possible failure cases during action execution. A transition to this state takes place whenever a failure is detected in action execution. The other state specifies that the robot is holding an object. The purpose of separating these states is creating an

abstraction from irrelevant state components to decrease the number of states generated by the POMDP.

The preference orders of objects can be determined by either prioritizing objects based on their locations in order to minimize the travelled distance or primarily targeting objects on which the robot has the best manipulation performance. Therefore, success probabilities for two actions are determined according to special benchmarks. To provide the best manipulation performance, the success probabilities of the action for picking up an object is assigned according to the experience built by the learning phase. In this way, the probability of encountering with a failure is aimed to decrease. As another factor that affects the manipulation timing and performance, the distance between the location of the robot and a corresponding object to be manipulated should be taken into account. Therefore, semantic locations of the objects are also used to decide on the next object to manipulate by determining probabilities of actions to move to objects.

The obtained results show that the probabilistic guidance in planning achieves the best manipulation order of the objects to maximize the transportation success over time in a real world scenario.

ÖGRENME GÜDÜMLÜ OLASILIKSAL PLANLAMA

ÖZET

Fiziksel dünyada eylemlerini yürüten robotlar, çeşitli hata durumlarıyla karşılaşabilirler. Özellikle hataların izole edilmediği dinamik ortamlarda çalışıldığında, hata ile karşılaşmak kaçınılmazdır. Sistemi kararlı hale getirebilmek için, robotun eylem yürütme sırasında karşılaştığı hataları saptayabilmesi ve tecrübe ettiği bu hatalardan daha sonraki plan yürütmelerinde kaçınması gerekmektedir. Geçmiş eylem yürütmelerinde tecrübe edilen hatalar, gelecekte karşılaşılabilecek potansiyel hatalardan kaçınmak için geliştirilen bir adaptif planlama stratejisinde kullanılabilir. Bu amaca ulaşabilmek için, geçmiş yürütmelerdeki gözlemler üzerinde öğrenme algoritması uygulanması gerekir. Bu tez kapsamında, bahsedilen problemin çözümü olarak öğrenme ve öğrenme güdümlü planlama yöntemleri önerilmektedir. Görev yürütme başarımını artırmak için, Tümevarımlı Mantıksal Programlama temelli bir deneyimsel öğrenme metodu ve öğrenme sürecinde edinilmiş deneyimleri kullanan olasılıksal bir planlama yapısı biraraya getirilerek gelişmiş bir robot sistemi oluşturulmuştur. Önerilen sistem, otonom bir gezgin robot ile yürütülen çoklu nesne etkileşim senaryolarında analiz edilmiştir. Burada amaç, kazanılmış tecrübeleri kullanarak karşılaşılması potansiyel olan hatalardan kaçınmak ve böylelikle belirli bir zaman süresince başarı ile toplanmış ve hedefe ulaştırılmış nesne sayısını olabildiğince artırmaktır.

Üzerinde geliştirme yapılan robot sistemi, eylem yürütme sırasında karşılaşılabilecek hataları saptayabilmek için sürekli olarak çalışmasını gözlemleyen bir hata sezme yapısı ve robotun sensörleri aracılığıyla ortamdan topladığı bilgileri işleyen, filtreleyen ve biraraya getirip robotun dünya modelini oluşturan bir sahne yorumlama yapısı içermektedir. Bu sistem bileşenleri robot sisteminin alt katman mimarisini oluşturmaktadır. Bahsedilen robot sistemi üzerinde üst katman sistem bileşenleri olarak geliştirilmiş ve bu tezde yararlanılan önceki çalışmalarda ise, robotun önceki eylem yürütmeleri sonucu edindiği tecrübelerin bir araya getirilerek olasılıksal hipotezlerin oluşturulduğu, Tümevarımlı Mantıksal Programlama'ya dayalı deneyimsel öğrenme süreci önerilmektedir. Aynı zamanda başka bir çalışmada da, deneyimsel öğrenme süreci sonucu elde edilen hipotezlerin kullanıldığı, deterministik planlayıcıya dayalı olarak geliştirilen bir öğrenme güdümlü planlama sistemi önerilmektedir. Öğrenme sonucu elde edilen hipotezler olasılıksal veri sağlasa bile, deterministik planlayıcıyı yönlendirirken, hipotezlerin sadece içerik kısımları kullanılabilmektedir. Ancak bu tezde önerilen olasılıksal sistemde hipotezlerin olasılıkları da planlayıcıya yön vermek amaçlı değerlendirilebilmektedir. Robotun eylem yürütmesi sırasında karşılaşacağı belirsizliklerle başa çıkabilmesini sağlayacak ve adaptif bir plan üretebilecek Kısmi Gözlemlenebilir Markov Karar Süreci (Partially Observable Markov Decision Process, POMDP) modelinden yararlanılmaktadır. Önerilen öğrenme güdümlü planlama sisteminde POMDP planlayıcı olarak, noktaya dayalı yeni bir algoritma olan SARSOP kullanılmaktadır. Bu algoritma aynı yaklaşımı benimseyen diğer tekniklerden, optimal olan erişilebilir inanç uzayına sezgisel bir arama yöntemi kullanarak ulaşması

açısından ayrılmaktadır. Bu şekilde, yaklaşık olarak bulunan çözüme ulaşma hızı arttırılmış olur.

Önerilen sistemle, otonom gezgin bir robot kullanılarak nesne etkileşim senaryoları yürütülmüştür. Amaç belirli bir zaman süresince, robotun karşılaşabileceği hataları azaltarak hedefe ulaştırılan nesne sayısını olabildiğince arttırmaktır. Bu yöntemin asıl katkısı, öğrenme çıktılarını kullanabilecek şekilde geliştirilen POMDP planlama formülasyonudur. Bu formülasyonda, her nesnenin niteliksel ve uzamsal özelliklerini bir araya toplayan nesne kompozisyonu tanımı yapılmaktadır. Nesnelerin niteliksel özellikleri nesnenin tipi, rengi, malzemesi, boyutları gibi önceden tanımlanmış unsurları belirtir. Uzamsal özellikler ise nesnenin alansal/bölgesel konumunu ifade eder. Sistemin görüntü işleme ve yorumlama süreçleri sonucunda elde edilen nesne konumları sürekli değerlerle ifade edildiğinden, POMDP formülasyonunda kullanılabilmeleri için ayrık ifadelere dönüştürülmeleri gerekmektedir. Bu nedenle, deneylerin yürütüldüğü ortam odalara ayrılarak sürekli değer ifade eden konum bilgileri alansal konumlara dönüştürülür. Alansal konuma ek olarak nesne kompozisyonu, robotun o anki konumuna göre belirlenen, nesnenin bağıl konum bilgisini de içerir. Önerilen sistemde, robot ile nesne arasında sadece bir adet konumsal ilişki tanımlanmıştır ve bu konumsal ilişki nesnenin, robotun önünde olup olmadığını belirtmektedir. Yürütme esnasında robotun konumu değiştikçe bu bilgi güncellenir ve herhangi bir dünya durumunda sadece bir nesne robotun önünde olabilir.

Nesnenin niteliksel ve uzamsal özelliklerinin bütün kombinasyonları için farklı nesne komposizyonları üretilir. Olası bütün nesne kompozisyonlarının kombinasyonlarından oluşturulan kümeler, POMDP formülasyonunun durum uzayını oluşturur. Tanımlanan durum uzayına iki farklı dünya durumu daha eklenmektedir. Bunlardan biri, eylem yürütme esnasında oluşabilecek olası hata durumlarını ifade eder. Eylem yürütme sırasında hata oluştuğu saptandığında, bu dünya durumuna geçiş yapılır. Eklenen diğer dünya durumu ise robotun bir nesneyi tutmasını ifade etmektedir. Bu iki dünya durumunu nesne kompozisyonu tanımından ayırmanın amacı, birbiriyle ilgisi olmayan dünya durumu elemanlarını birbirlerinden soyutlayarak, POMDP sisteminde üretilen dünya durumlarının sayısını azaltmaktır.

Bahsedilen çoklu nesne etkileşim senaryosunda, robotun her nesne kompozisyonu tarafından içerilebilecek nesneler üzerinde yürütmesi gereken temel eylemler sırasıyla, nesneye yaklaşma (*moveTo*), nesneyi tutma (*pickUp*) ve tutulan nesneyi hedef konuma götürüp bırakma (*putDown*) eylemleridir. Nesne kompozisyonunun içerdiği nesneyi hedef konuma taşıma ve bırakma işlemleri tek bir eylem içerisine (*putDown*) dahil edilmiştir. Bu sistemde, temel eylemlerin yanısıra ortamdan bilgi toplamak için kullanılabilecek algılamaya yönelik eylemler de tanımlanmıştır. Bunlardan ilki ortamda hiç nesne olmadığında robotun, ortamı gezerek nesne aramasını sağlayan arama (*search*) eylemidir. Diğeri ise robotun sahip olduğu bir bilgiyi güncelleyebilmesi için uygun pozisyona geçerek ortamı algılamasını sağlayan algıla (*monitor*) eylemidir. Eylemler başarılı olarak yürütüldüğünde geçilecek dünya durumu, her eylem yürütüldükten sonra ortamda olması beklenen etkilere göre belirlenir. Robot herhangi bir eylemi yürütme sırasında hata ile karşılaştığında ise, POMDP durum uzayı tanımındaki hata durumlarını belirten dünya durumuna geçilir.

Robot performansını arttırmak için üzerinde işlem yaptığı nesneleri tercih etme sırasını iki farklı yöntem kullanarak belirleyebilir. Bu yöntemlerden biri, robotun nesneye yaklaşırken katettiği mesafeyi en aza indirmek için nesne kompozisyonlarını

konum bazlı olarak önceliklendirmesidir. Diğer bir yaklaşım da robotun en iyi performans gösterdiği nesneleri içeren nesne kompozisyonlarına öncelik vermesidir. Bu yöntemleri uygulayabilmek için tanımlanan iki eylemin (nesneyi tutma eylemi olan *pickUp* ve nesneye yaklaşma eylemi olan *moveTo*) başarım olasılıkları özel kıstaslarla belirlenir. En iyi performans gösterilen nesnelere öncelik tanınmasını sağlamak için, her nesne kompozisyonu için ayrıca tanımlanmış *pickUp* eyleminin başarımı, öğrenme sonucu elde edilen hipotezlerin olasılıklarına göre atanır. Bu yolla, hata ile karşılaşma olasılığını azaltmak hedeflenmektedir. Görevin tamamlanma zamanını ve performansını etkileyen diğer bir faktör olarak, robotun o anki konumu ile nesnelerin konumu arasındaki uzaklık gözönüne alınmalıdır. Bu nedenle, her bir nesne kompozisyonu için ayrıca tanımlanmış *moveTo* eylemine atanan başarım olasılıkları, nesnelerin semantik konum bilgisiyle orantılı olarak hesaplanır. *pickUp* ve *moveTo* eylemleri dışındaki diğer eylemler, her nesne kompozisyonu için aynı şekilde tanımlanmıştır. Ortamın algılanması ve robotun bilgi tabanının güncellenmesi için tasarlanmış eylemlerin (*search* ve *monitor*) başarılı veya başarısız yürütülme durumları yoktur. Bu nedenle bu eylemler yürütüldükten sonra geçiş yapılabilecek dünya durumlarının olasılıkları eşit olarak dağıtılmıştır. *pickUp* ve *moveTo* eylemlerine ek olarak sadece *putDown* eylemi için başarısız sonlanma durumundan bahsedilebilir. Bu eylemin başarısızlık olasılığı ise nesne bazlı olarak değil, sistemin genel nesne bırakma başarımı dikkate alınarak belirlenmektedir.

Tanımlanan POMDP formülasyonunun gözlem uzayı da durum uzayı ile aynı olacak şekilde tanımlanmıştır. Böylelikle her durum geçişinde son alınan gözleme uygun olarak bir sonraki durum belirlenmektedir. Gerçeklenen sistemde, robotun kamera, lazer, mikrofon ve sonar sensörlerini kullanarak ortamdan topladığı veriler, sistemin sahne yorumlama bileşeni tarafından çeşitli algoritmalarla işlenerek biraraya getirilir ve filtrelenerek yorumlanır. Sonuçta elde edilen bilgi, ilgili bütün sensörlerden elde edilen veriler kullanılarak oluşturulduğundan, verinin belirsizliği önemli ölçüde azaltılmış olur. Böylelikle, robotun gözlemine olan güvenilirlik arttırılmış olduğundan, önerilen planlama sisteminin gözlem modelinde olasılıksal bir yaklaşım kullanılmamaktadır. Diğer bir deyişle POMDP formülasyonu, sadece eylemlerin sonuçlarındaki kararsızlığı gösterecek şekilde tasarlanmıştır.

Durum ve gözlemlerle ilgili tanımlanan yapılara ek olarak, önerilen POMDP formülasyonunda robotun başarılı eylem yürütmelerine ödül verilmesi, başarısız olduğunda ise penaltı uygulanması sağlanmıştır. Böylelikle robot hata yapma olasılığı olan eylemlerden mümkün olduğunca kaçınacak ve performansını yükseltecektir. Aynı zamanda ortamdan bilgi toplamak amacıyla yürütülen *search* ve *monitor* eylemlerine de penaltı verilmektedir. Çünkü, robotun en kısa sürede olabildiğince çok nesneyi başarılı bir şekilde hedefe ulaştırabilmesi için, zaman kaybettiren bu eylemleri, başka bir alternatifi olmadığı zamanlarda, yani mümkün olduğunca az sayıda yürütmesi gerekmektedir.

Önerilen sistem gerçek robot üzerinde iki farklı senaryoda analiz edilmiştir. Senaryolardan ilkinde, ortamda bulunan belirli bir kategorideki bütün nesneler üzerinde yürütülen *pickUp* eyleminde, nesneler ortamdan uzaklaştırılarak elle hata üretilmiştir. Bir kategorideki nesnelerin hepsi üzerinde hata üretilmesi, olasılıksal bir sistemin test edilmesinde hiçbir anlam ifade etmeyeceğinden, rastgele üretilmiş bir sayı dizisi kullanılarak belirli bir olasılıkta hata oluşturulmuştur. Bu senaryo için öğrenme süreci sonucunda elde edilen hipotez, o kategorideki nesneler üzerinde

yürütülen *pickUp* eyleminin, rastgele sayı dizisi ile belirlenmiş olasılıkta başarısız olduğudur. Bu senaryo hem öğrenme çıktısını kullanarak üretilmiş plan ile hem de öğrenme olmadan elde edilen plan ile test edildiğinde, öğrenme sonucunun robotun performansında sağladığı artış görülebilmektedir. Çünkü robot, başarım olasılığı düşük olan nesnelere gitmektense, öncelikle *pickUp* eylemini daha başarıyla yürüttüğü nesneleri seçmektedir. Eğer daha yüksek performans gösterdiği bütün nesneleri taşıdıysa ve hala zamanı varsa hata yaptığı nesneleri denemeye başlar. Çünkü, düşük de olsa bu nesneler üzerinde başarı ile eylem yürütme olasılığı vardır. Bu problemin çözümünde olasılıksal bir yöntemin kullanılması sayesinde, robot bir nesne üzerinde tutma eylemini yürütürken belirli bir olasılıkla hata ile karşılaşmış olsa bile, o nesne üzerindeki başarısızlık olasılığı 0 değilse o nesneyi tutmaktan tamamen vazgeçmemiş olur.

Yürütülen ikinci senaryoda ise, önceden tanımlanmış ve nesnelerin bulunabileceği alansal konumları ifade eden odalardan birindeki bütün nesneler üzerinde yürütülen *pickUp* eyleminde, rastgele üretilmiş bir olasılıkla hata oluşmasına neden olunmuştur. Üretilen öğrenme çıktısı, robotun hata üretilen odadaki nesneler üzerinde yürüttüğü *pickUp* eyleminde, önceden rastgele olarak belirlenmiş olasılıkla başarısız olduğunu ifade eder. Bu senaryo da öğrenme kullanılarak üretilen plan ile ve öğrenme çıktısı değerlendirilmeden oluşturulmuş plan ile test edilmiştir. Deney sonucunda, robotun hata üretilmiş odadaki nesneler üzerinde eylem yürütmeyi olabildiğince ertelediği gözlenmiştir. Böylelikle robotun performansında yine artış sağlanmıştır.

Sistemin gerçek dünya senaryolarında çalıştırılmasıyla elde edilen sonuçlara göre, planlamanın olasılıksal olarak yönlendirilmesi, en uygun nesne etkileşim sırasının elde edilmesi için mecburidir. Böylelikle nesne taşıma başarımı zamansal olarak eniyilenmiş olur.

1. INTRODUCTION

There exist many factors which affect the performance of a robot by causing failures in its execution. For example, the robot may be forced to stay in a location by a human, although it tries to move forward. In this case, the robot may fail while it localizes itself in the map of the environment. As a result, it fails while reaching its goal. As another example, if an object is heavier than the weight that the robot can carry, a failure occurs in each manipulation trial for this particular object. Generally, the factors which may be the reason of failures can be categorized as follows [1]:

- Internal factors: Internal factors are related to the robot itself.
  - Lack of hardware resources
  - Sensor limitations
  - Actuator/Effector error
- External factors: The performance of the robot may also be affected externally.
  - Human intervention
  - Lack of knowledge or misbeliefs
  - Conflicting or impossible goals

Building an autonomous and cognitive robot system requires to handle failures. While one approach is recovering from failure cases, another approach is preventing them before they occur. These two approaches tries to solve the problem in different timings. While recovering a failure, the robot executes extra actions or change its current action to correct the failure after it is detected. Here, the robot should be prepared for each failure situation to decide on its recovery action. On the other hand, some unrecoverable situations may also occur. In the other approach, the robot may avoid failure cases before they happen. However prevention makes it necessary that the robot should use prior experience on these cases. Therefore the best way is to let the robot

build its own experience and use an adaptive planning strategy. In this way, the robot can avoid situations in which it encounters failures or unrecoverable states without external assistance.

The main objective of this work is to develop an experience-guided planning method for a robot to improve its performance on manipulation tasks. In this method, the robot builds its experience by learning through observations made after each action execution. Proposed method is analyzed on a mobile manipulation case study where the focus is on optimizing the number of manipulated objects in the face of failures. This can be achieved by using the built experience from previous failure cases to guide future planning tasks of the robot.

1.1 The Main Units of the Used Robot System

Creation of an autonomous robot system requires the integration of various system components. All of these components should also work well corporately in addition to their independent performances. In this thesis, used robot system tries to achieve a multi-object manipulation task in real world. Multi-object manipulation is an extensively studied real world application domain for autonomous robots. Although lots of studies are published in this domain, several challenges remain because of the variety of subtasks it includes. The required units to build the autonomous robot proposed in this thesis which achieves its multi-object manipulation task can be listed generally as follows:

- *Sensing the environment*: An autonomous robot needs to have consistent information about the world state to achieve its tasks. Therefore, it senses the environment and processes the data gathered through its sensors to build the world knowledge. However, the data coming from the sensors of a robot contains noise and the way of obtaining a reliable knowledge about the environment requires reducing the noise in the data. Therefore, sensory information gathered from the environment through different sensors of the robot are processed with different algorithms to distinguish relevant information, and the processed data are also fused to provide consistency in the knowledge base of the robot.

- *Motion control*: The robot executes the actions in its plan and also takes observations from the environment during execution. In this way, the robot can be aware of sudden changes occurred in the environment during its action execution, and it can adjust its action parameters according to the current world knowledge to show better performance.

- *Failure detection*: Since the robot is aimed to learn from its previous failures, the robot should detect failures in its action execution. This is achieved by monitoring the execution of the robot runtime. Whenever an unexpected situation is encountered in which the expected effects of the actions are not satisfied or any undesired event has occurred, the robot can detect the anomalie.

- *Learning*: Adapting the plan of the robot according to unexpected changes or unknown situations is a prerequisite to provide robustness. This can be achieved by providing a system that can learn from prior experience and update the plan to avoid undesirable cases.

- *Planning*: A robot uses planners to generate a plan according to its goal and the current world model. In the focus of this work, the robot is aimed to avoid its previous failures as much as possible. An experience-guided planning process is required to achieve the adaptation of the robot to previous failures. After executing the plan which is generated without any guidance from learning, the experience of the robot is built. When the planner is started next time, the built experience can be used for future guidance. While the experience of the robot increases in time, the learning process can produce more reliable outcomes and the robot can show better performance.

The detailed information about the used method and the system components to develop the proposed adaptive planning strategy is given in the following chapters.

1.2 Probabilistic Structure of the Proposed Method

Although it is assumed that the robot has full knowledge of the world state in some isolated problems, actually the observations gathered from the environment include uncertainties because of noisy sensors of the robot and dynamic structure of the

environment, especially in the real world. Therefore one source of uncertainty arises from the partial observability. Additionally, another type of uncertainty is seen in the outcomes of nondeterministic actions which are different from expected. These uncertainties usually result in unpredictable states. Since elimination of uncertainties is impossible, some probabilistic approaches are developed to deal with it.

In this thesis, a real world application is targeted which requires to take the uncertainty into account for a better performance. Therefore, a probabilistic planner is used to generate plans of the robot. Since the main contribution of the thesis is using the outcomes of a learning process to guide the planner for avoiding failures, the learning process should also provide probabilistic outcomes.

1.3 Purpose of Thesis

An autonomous robot should detect failures during its execution, and handle them without external assistance to improve its performance. Development of an experience-based approach is a requirement for avoiding previously encountered failures in future executions. This experience may be used in the planning phase. In this way, the robot can learn from its previous failures and show better performance. In this thesis, such an adaptive planning method is aimed for providing improvement in object manipulation performance of a mobile robot by decreasing the number of encountered failures.

1.4 Hypothesis

The main aim of this work is to develop an experience-guided planning method that improves the performance of a robot building its experience. Multi-object manipulation domain is particularly investigated and optimizing the number of manipulated objects against failures is focused. This can be achieved by using the experience built from previous failure cases on future planning tasks of the robot. The prior work of author's research group involves an experiential learning process [1, 2] using Inductive Logic Programming (ILP) and a deterministic planner that uses the built experience [3]. The previous deterministic planner takes only the contexts of hypotheses into account to provide feedback to the planning. The proposed method in this thesis extends the earlier study with the development of a probabilistic planner that

makes use of probabilities of heuristics framed by learning. The ILP-based learning process derives hypotheses associated with probabilities, and these hypotheses are then used to guide the planning process. The proposed probabilistic method uses a Partially Observable Markov Decision Process (POMDP) model to create an adaptive policy for the robot to deal with uncertainties. In the case study of this thesis, with the use of the probabilistic planner, probabilistic hypotheses are used in determining the order of preferences on objects for manipulation, and the robot initially targets the objects that it believes to be successful in manipulating. Furthermore, the probabilistic planner makes it possible to make instant decisions on actions.

2. BACKGROUND AND RELATED WORK

In the literature, many studies about adaptive planning strategies for improving the performance of the system can be found. In this section, related work about adaptive planning is reviewed after presenting background knowledge on the topics covered in the thesis. In addition, the previous work of author's research group about adaptive planning and the contribution of this thesis over previous studies can also be found in the following subsections.

2.1 Planning

In this work, the performance of a robot is aimed to be improved by using a probabilistic planning method which uses the experience of the robot. Since Markov Decision Processes (MDPs) are simplified versions of Partially Observable Markov Decision Processes (POMDPs) by eliminating some sort of uncertainty in the environment, firstly MDPs are explained to provide a better understanding about POMDPs which are described afterwards.

2.1.1 Markov decision processes

A Markov Decision Process (MDP) is a formulation which is generated to represent the nondeterminism in the action outcomes of an agent. Therefore, the agent in an MDP decides on its next action by taking the uncertain effects of its actions into account. On the other hand, the belief of an agent to be in a state is definite. In other words, perceptual abilities of the agent are considered as perfect [4].

Basic components of a Markov Decision Process can be represented as a tuple $(S, A, T, R, \gamma, s_0)$. Here, S represents a finite state set of the world and A represents a finite action set. T is a state-transition function which determines the probability distribution over all world states after executing each action in each state. Since function T gives probabilities of world states for each state transition, all possible effects of actions can be formulated in the MDP model, and corresponding

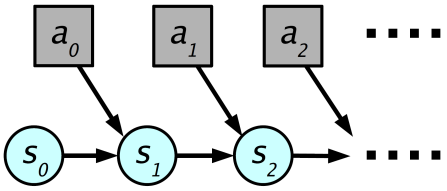

Figure 2.1: A simple illustration of the MDP model [5].

probabilities can be assigned to the next states after executing an action. After executing each action in each state, the agent can be awarded or penalized. Therefore, another function (R) is used for determining the expected immediate reward gained after executing each action in each state in MDP model. The reward may be negative in the cases where the robot is penalized for executing an action in a state [4]. The expected sum of rewards is represented in Equation 2.1 as follows:

$$E\left[\sum_{t=0}^{+\infty}\gamma^t r_t\right] \tag{2.1}$$

where r_t is the reward gained at time step t, and $0 \leq \gamma \leq 1$ is a discount factor which is used for bounding the rewards in the case that the agent is assumed to have infinite lifetime, because in most cases, estimations on the lifetime of an agent would be wrong. Therefore, the total reward is guaranteed to be finite after using a discount factor [4]. As the last component of MDP formulation, s_0 is the start state of the model.

As an important factor in the formulation, the Markov property of the model can be explained like that, the gained reward and state transition probability of a state after executing an action in a state depends only on the previous state, and the last executed action. Although decision of the next action is determined by evaluating all previous states and executed actions to maximize the total reward, this decision is conditionally independent of the action and state history. This is named as *Markovian* property. In Figure 2.1, an MDP model is graphically illustrated which also represents *Markovian* structure. In this graphical model, square nodes represent the actions and circular nodes represent the states of the problem. Dependence relationships between these variables are illustrated with directed edges [5].

MDPs are used to generate policies which correspond to the plan of an agent. A policy π_t provides a mapping between world states and actions. Thus, the agent can decide on the next action that corresponds to the current state in the generated policy.

The formulation of a planning problem can be written in many forms. The form of the formulation is determined according to the requirements of the problem. For example, while the shortest sequence of actions are searched in a problem, the lowest total cost is aimed to reach in another one in the case that different costs are assigned to actions. As a result, a utiliy function should be created such that when the larger score of the function is obtained, the needs of the problem will be satisfied more. Therefore, the optimal plan can be found by maximizing the score of a utility function [5]. The utility function which calculates of the long-term expected sum of immediate rewards that the agent gained after each action execution is named as value function, and a plan can be generated by approximating an optimal value function [5]. The expected sum of rewards gained until the current state s by executing policy π for t time steps is formulated in Equation 2.2.

$$V_{\pi,t}(s) = R\left(s, \pi_t(s)\right) + \gamma \sum_{s' \varepsilon S} T\left(s, \pi_t(s), s'\right) V_{\pi,t-1}(s') \tag{2.2}$$

Here, $T\left(s, \pi_t(s), s'\right)$ represents the probability of being in state s' after executing policy π in state s and $V_{\pi,t-1}(s')$ represents the value of the corresponding transition to state s through $(t-1)$ steps. All possible resulting states are considered while estimating future values of executing actions in a state. If the length of policy is thought as infinite, the expected sum of the future reward with applying the discount factor can be calculated as in Equation 2.3 for starting in state s and executing policy π.

$$V_{\pi}(s) = R\left(s, \pi(s)\right) + \gamma \sum_{s' \varepsilon S} T\left(s, \pi(s), s'\right) V_{\pi}(s') \tag{2.3}$$

In the formulas above, the future rewards are determined for a given policy π. However, a function in the reverse direction can also be generated to obtain a policy for the given value function. A greedy policy π_V for the given value function is defined in Equation 2.4 as follows:

$$\pi_V(s) = \arg\max_a \left[R(s,a) + \gamma \sum_{s' \varepsilon S} T\left(s, a, s'\right) V(s') \right] \tag{2.4}$$

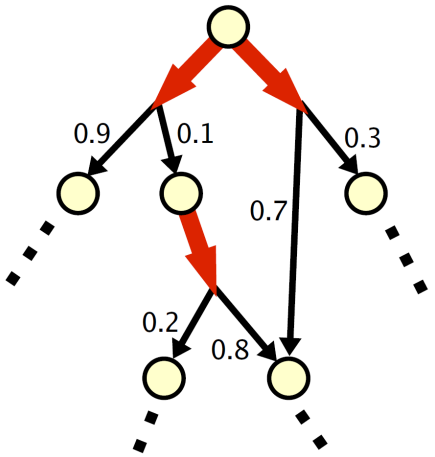

Figure 2.2: A simple search graph example [5].

Here, summation of immediate rewards which maximize the expected sum (the first part of the formula), and the expected discounted value of future states (the second part of the formula) is evaluated for each action to find the best action to execute [4].

2.1.1.1 Graphical representation of MDPs

A graphical representation can be used to illustrate state transitions in an MDP model. Search graphs of an MDP corresponds to an AND/OR graph. Nodes represents the states and state transitions are indicated as edges in the graph. If a probability which is higher than 0 is determined for a transition between two states, these two states are connected with a directed edge. Figure 2.2 is an example search graph. In this graph, available actions are separately illustrated with thick arrows and state transitions occurred after executing an action are illustrated with thin arrows that come after the head of thick arrows corresponding to these actions [5].

As another option, a policy can also be illustrated as a graph. Differently from search graphs, nodes represent actions in policy graphs. However, state transitions after executing the action in a node are represented as arrows like in search graphs. Execution of a policy starts with the root of the graph and after executing the first action at the root, the result of the action specifies the direction to move in the graph. When a leaf node is reached, this means the robot has achieved its goal [5].

2.1.1.2 The robot navigation problem as an example

A simple indoor robot navigation problem can be given as an example [5]. In this example, the robot is placed in a hallway which have walls around and obstacles inside.

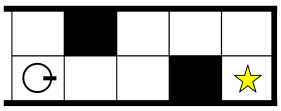

Figure 2.3: The map of the environment in a simple robot navigation problem [5].

The map of the environment and the locations of the robot and the goal can be seen in Figure 2.3. Here, the circle represents the robot and the star represents the goal position. Black grids are the obstacles in the environment.

In this problem, the goal of the robot is navigating to the position of the goal point without any collision with the walls or obstacles in the environment. The robot knows the map of the environment completely. Furthermore, it is assumed that the robot always has the knowledge of both its own position and the position of the goal as a result of MDP structure. Since the environment is divided into grids, all locations can be represented with discrete values. This is a way of approximation of the problem. Because using the continuous-valued locations makes the location space of the problem infinite and increases complexity.

If the described problem is formulated as an MDP problem, a sequence of one-step motion actions is searched to reach the goal without any collision with walls or obstacles. In the case that the robot is assumed to move only forward, the actions can be defined as *east* for moving to the east direction and *north* for moving to the north direction. Each location in the grid environment corresponds to a state in the problem formulation. Since the problem is modelled as an MDP, the policy of the robot can be represented as a sequence of actions. Because the robot knows its state in which it is currently be after executing an action. In MDPs, uncertainties only arise in state transitions. In the state transition model, unexpected outcomes of each action

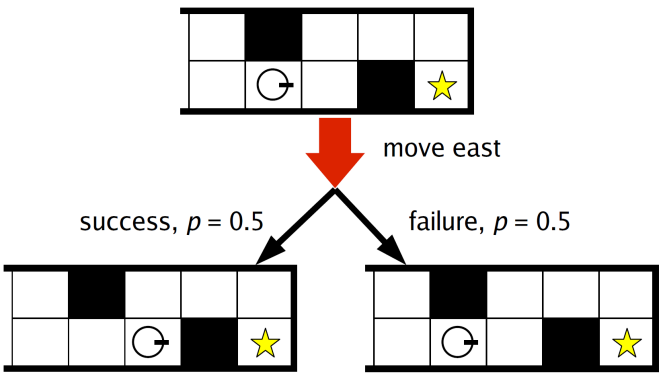

Figure 2.4: Probabilistic state transitions of an action in robot navigation problem [5].

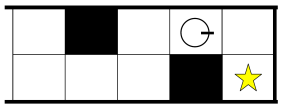

Figure 2.5: An example state of robot navigation problem [5].

is represented probabilistically. For example, actions in this problem can be assumed to have an error of 50%. This means the robot has 50% chance of executing an action successfully. On the other hand, it may find itself in the same cell with the probability of 50% after executing an action. Furthermore, the action fails completely (100%) in the case that an obstacle or a wall stand in front of the robot. If state transitions are determined according to these constraints, the outcomes of the actions cannot be certain and the robot cannot be sure about the effect of an action before it executes the action. After the action is selected and executed, the robot can know the effect of the executed action with certainty. Figure 2.4 illustrates the graphical representation of the state transition of action *east*. Here, two alternative states exist in which the robot can find itself after executing action *east*. One of the alternative states represents the case that the robot fails to move, and the other one represents the successful movement.

A policy graph for the robot navigation problem is generated for the starting state given in Figure 2.5, and only its first three levels are illustrated in Figure 2.6. In this policy, while the robot fails in its movement, it tries the same action again [5].

2.1.2 Partially observable Markov decision processes

As described in the previous subsection, MDPs can only represent uncertainties in the effects of actions. However, observations gathered from the environment also contains uncertainty. Therefore, formulation of an MDP should be extended to handle the partial

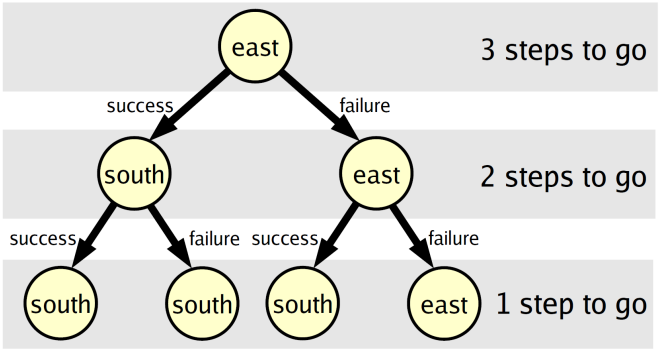

Figure 2.6: A 3-level policy tree for the robot navigation problem [5].

observability. POMDPs represent probabilistic processes that can be used to devise plans for non-deterministic actions in partially observable environments. A standard POMDP framework for an agent is defined by the tuple $(S, A, O, T, Z, R, \gamma, b_0)$. Here, S is the set of states which contains all possible world states the agent might be in, A is the set of actions that the agent can execute, and O is the set of observations that can be gathered through the sensors of the agent. After executing action $a \in A$ in state $s \in S$, the probability of reaching state $s' \in S$ in the next time step is determined by the probabilistic transition model $T(s,a,s') = P(s'|s,a)$. The transition probability denoted with $P(s'|s,a)$ models non-deterministic outcomes of actions. When transitioning to a new state s' after executing action a, the probability of getting the observation $o \in O$ is also defined with a conditional probability function $Z(s',a,o) = P(o|s',a)$ as the sensor model representing uncertainties in sensing. $R(s,a)$ function provides a real-valued immediate reward received after executing action a in state s. The overall aim of POMDP planning is to select actions to maximize the total reward as a cumulation of immediate rewards during the execution of the actions. A discount factor $0 \leq \gamma \leq 1$ is set to bound the effects of future rewards [6].

In short, a POMDP model is an extended version of an MDP model. Representation of observations and the sensor model are additional components. The rest of the POMDP formulation is the same with MDPs. Since the POMDP model includes partial observability, the agent cannot be sure in which state it is. Therefore, possible states the agent might be in after each action execution should be represented in the model. This is called as beliefs in a POMDP model. A belief state represents a world state that the agent believes to be in and involves uncertainty due to partial observability. Each belief state is associated with a probability, and the agent decides on its next action by evaluating all belief states it might exist in. Before generating a plan, the

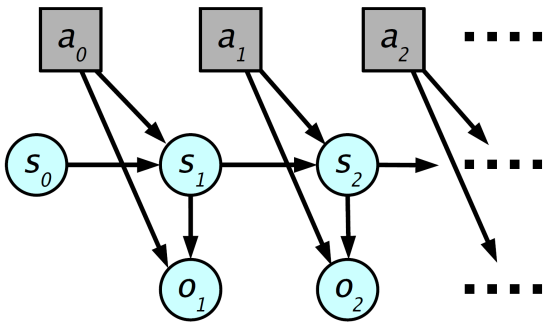

Figure 2.7: A simple illustration of the POMDP model [5].

initial probability distribution over belief states represented as b_0 should be determined. By using these components, a policy which defines the best action for each state to maximize the expected total reward is searched for. The methods used for searching a policy are the same with an MDP model. However lots of state transitions may exist in a POMDP model because of uncertainty in observations. In Figure 2.7, the POMDP model is illustrated in a graphical form. The model extends MDP in the way of using observations while transitioning to the next state. Because the agent tries to find the state it exists in after an action execution by evaluating observations.

After the generalization of MDPs, the computational complexity also increases in POMDPs. The agent evaluates all possible belief states in a world state rather than dealing only with one world state as in MDP. Discretization is a way to decrease the complexity. Although both continuous and discrete representations of POMDPs exist in literature, discrete-state POMDPs are commonly used in robotic domains to reduce computational complexity. Otherwise, complexity grows exponentially with the size of the state space.

Another way to deal with complexity is using approximate methods such as point-based algorithms in which belief states are sampled to obtain an approximate representation of the belief space. Thus, performing on sampled belief states gives more useful results rather than using exact state space which cannot be solved within a reasonable time period [6].

2.1.2.1 POMDP model of the robot navigation problem

The robot navigation example which is modelled as an MDP in one of previous subsections can also be customized as a POMDP. Here, the robot can observe the locations of obstacles in the environment with an error and so it cannot certainly decide on its own location. For example, the robot can be assumed to have a noisy sensor which has 10% error in its reading. The sensor gives an *obstacle* reading when there is an obstacle at the next east grid with reference to the location of robot. The sensor returns a *clear* reading in the case that there is no obstacle at the east grid. However the reading of the sensor is not completely reliable. Since the robot cannot know exactly which state it is in because of partial observability, all possible beliefs

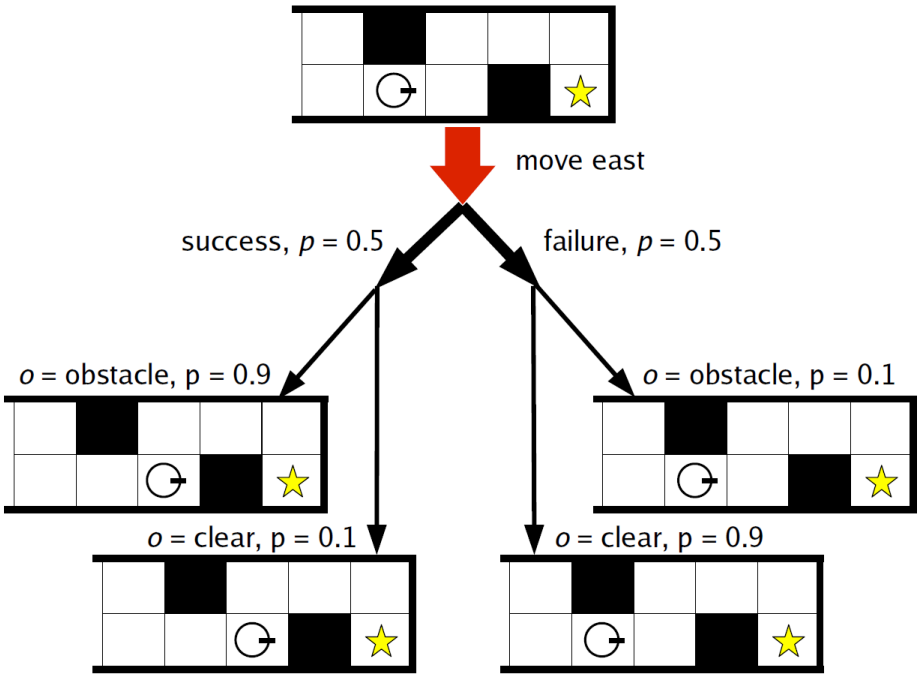

Figure 2.8: Uncertainty in robot navigation problem [5].

should be evaluated as illustrated in Figure 2.8 which shows all possible combinations of observation and action outcomes [5].

2.1.2.2 The policy graph of the tiger problem

To give an example of a fully-generated policy graph, a simple well-known problem named as *tiger problem* can be explained. In this problem, the agent is placed in an environment with two doors. On the other side of one door, there exists a tiger and the agent does not know behind which door the tiger is placed. There are three actions to execute: $left$, *right* and $listen$. Action $left$ is executed to open the door on the left. Similarly, action *right* is for opening the door on the right. The agent can execute action $listen$ to hear the growling of the tiger for gaining information about the position of it. However the information taken after action $listen$ is partially-observable. The agent may suppose that the sound is coming from a door while it is actually coming from the other. Furthermore, opening the door with the tiger behind has a high penalty (because in this case, the tiger will eat the agent) and opening the door without the tiger will be awarded. Execution of action $listen$ is also penalized with a lower cost value. After the agent decides to open a door and gets a reward or a penalty, the problem is reset, and the position of the tiger is determined randomly. Therefore, only actions $left$ and *right* have nondeterministic outcomes. Because the tiger is relocated again randomly

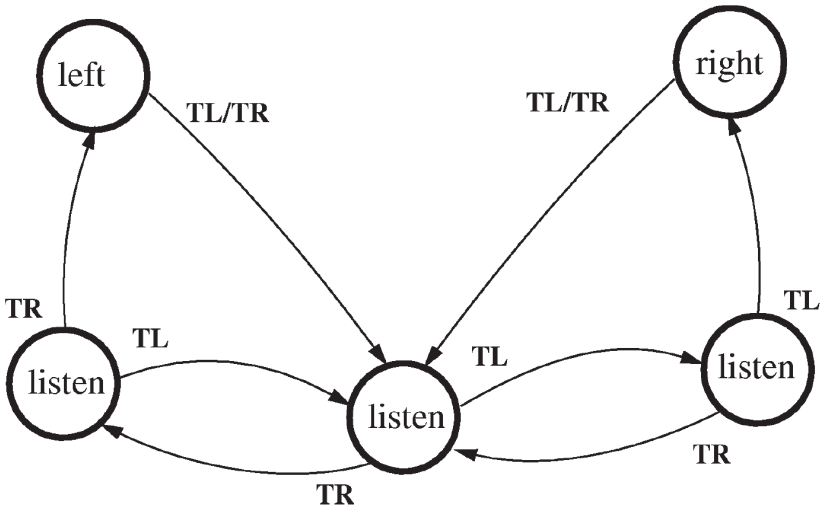

Figure 2.9: Policy graph of tiger problem [4].

after executing these actions. However action *listen* does not affect the world state in any way. Figure 2.9 shows the generated policy graph of tiger problem. Here, TL represents the observation of hearing the sound of tiger on the left, and TR represents the observation of hearing it on the right. The nodes of the graph indicates actions. In the solution, the agent continue executing action *listen* until it consecutively hears the sound of the tiger twice from same door to be sure enough about the position of the tiger [4]. The metric of time to be sure (two observation from the same door in this solution) is determined according to the rewards and probability distributions of state transitions and observations.

2.1.3 Successive approximations of the reachable space under optimal policies

POMDPs provide a probabilistic framework to deal with the uncertain nature of robotic applications. However computational complexity of POMDPs prevents their use in solving large problems. Therefore decreasing the complexity of POMDPs has become another research area, and various techniques are developed to solve POMDPs in a more efficient way. Point-based algorithms present an approximate approach in which the belief space is intended to shrink to obtain a space of sample beliefs which can substitute the whole belief space approximately [6]. Shrinking the belief space can be achieved by applying many different methods. Therefore, several point-based algorithms can be found in the literature which aims to improve the performance of POMDP planners with different approaches.

Successive Approximations of the Reachable Space under Optimal Policies (SARSOP) is a new point-based POMDP algorithm which improves the speed of reaching a solution in a POMDP model [7]. While general point-based algorithms determine reachable belief spaces while sampling belief states, SARSOP deals with extracting optimally reachable belief spaces. Since POMDP algorithms evaluate the belief space to find a solution, the size of the belief space extremely affects the complexity of reaching a solution in POMDPs. Therefore point-based algorithms try to decrease the size of the belief space in a POMDP model by selecting a representative subset of belief states. However difference between point-based methods arise in the selection of belief states. Recently, belief states which are reachable from a given initial belief (b_0) is sampled by applying arbitrary sequences of actions. In SARSOP, this approach is extended, and the set of reachable belief states from b_0 is determined by using *the optimal sequences of actions*. Since the optimal sequences of actions is actually the solution of the problem, an objective set of belief states is extracted approximately. The operation of sampling optimally reachable belief states is performed by using heuristic exploration on the belief space [7]. In heuristic exploration, new belief points are selected or some beliefs are updated by using heuristics while exploring forward in the search tree. Selection of actions and observations are performed according to these heuristics [8].

In this thesis, an implementation of the SARSOP algorithm for solving discrete POMDPs is used as a POMDP planner. Since the selected problem cannot be solved within reasonable time period by using a continuous POMDP because of complexity, a discrete POMDP is selected to use [7]. SARSOP can be used to solve many robotic tasks such as, underwater navigation, grasping, exploration, etc. It generates a policy graph which is similar to the outcome of a general POMDP. However modelling the problem domain properly for the planner of SARSOP framework makes it different. The formulation of the multi-object manipulation task in the SARSOP framework can be found in the following chapters.

2.2 POMDPs In Robotic Applications

The main contribution of this work is the integration of an experiential learning method with an adaptive probabilistic planner. The POMDP framework is selected as the

probabilistic planner to achieve the multi-object manipulation task. In the following subsections, relevant studies in POMDP-based robotic applications and their models are presented. Then, different adaptive planning approaches are described.

2.2.1 Studies about POMDPs in robot systems

There are various studies in which probabilistic planning is used to cope with uncertainties in perception and nondeterminism in action outcomes. Each of these studies is similar to the study presented in this thesis from some aspects. On the other hand, this thesis differs from these works in different dimensions.

POMDPs have been investigated in many studies for a wide variety of robotic tasks ranging from robot navigation [9] to target tracking [10]. To deal with the complexity of POMDPs in these works, different approaches are presented such as decreasing the number of uncertain variables in the problem model [11]. In [12], sensing, information processing and collaboration on multiple mobile robots are handled by using the POMDP framework to achieve reliable and efficient sensing and autonomous operation. They analyze the system in simulation and on physical robots with a scenario that aims to localize target objects in dynamic indoor domains. Each robot in the team locates one or more target objects and shares its beliefs with teammates to provide a robust collaboration. The authors also try to solve the multi-robot communication problem in partially observable environments. The difference of this thesis from [12] is that, their case study depends only on localizing target objects but the objective is extended here by trying to manipulate objects after detecting them in the environment. Their actions do not change the state of the robots because they are based on perception. This approach provides a simplification of the problem.

Also in [13], similar to this study, a failure intervention approach is proposed to provide a safe autonomous robot system. Supervised learning techniques are used to detect potential failures occur in the execution. Internal sensors of the robot are used to provide data to a trained classifier. After detecting a potential failure, a conservative safety response behaviour is triggered to avoid robot from dangerous situations. This goal is achieved by generating an augmented policy which integrates the regular policy of the robot, and a fixed failure intervention policy to be executed only when a failure is detected by the system. The proposed model is built as an MDP model and is analyzed

on a physical robot. Although the main focus of the authors is avoiding unrecoverable failures which is similar to the purpose of this thesis, the way of the solutions and scenarios are very different from each other. They use hand labelled data to train the classifier. However, in this thesis, the experiments are analyzed on a real robot, and the real sensor data gathered from real world experiments are used. Furthermore, the experiments are performed with multi-object manipulation scenarios, but the only unrecoverable failure case they consider in the experiments is the fall of the robot. Therefore, they try to prevent the robot from falling.

2.2.2 Studies about POMDPs in robot manipulation

In the focus of this thesis, POMDPs are used to achieve a multi-object manipulation task. Many studies can be found in the literature in which POMDPs are modelled to be used in robotic manipulation. While the main ideas behind the methods published in these studies and the method proposed in this thesis are similar, the used POMDP framework and the construction of the POMDP model constitute the major differences between these methods. In [14], manipulation task of deformable objects with a robotic hand is modeled as a POMDP to handle uncertainty in percepts and actions. This work does not involve learning. In [15] and [16], manipulation of multiple objects is selected as the main challenge, and an online POMDP planning approach based on particle filtering is proposed to change system dynamics according to action performances. In these works, world state definitions contain grasp success probabilities for objects. The probabilities are updated after each grasping trial.

The work of this thesis differs from [14–16] in the way that the planning process is guided with experience coming from learning and applying a generic method that can be applicable for all types of actions in a planning domain without changing state definitions. While the grasp success probability of an object is calculated as a function of occlusion ratio of that object, and previous grasping success rate on it in these works, an experimental learning method is used here to update the action success probabilities in an offline POMDP planner, and to generate a new plan after the learning phase ends up. Furthermore, the approach of this thesis can be applied for all other actions in a planner domain without changing state definition. However the same operation cannot

be achieved in [15] and [16] without adding new components to state definition to represent success probabilities of new actions.

2.2.3 Studies about adaptive planning

The main focus of this study is on experience-based guidance methods for increasing efficiency by reducing potential failures. For this purpose, adaptive planning algorithms are used commonly to reduce or exclude the possibility of failures. Some studies [17, 18] aim to generate planning operators to improve the performance of the robot. In [17], probabilistic STRIPS-like relational rules are learned by extracting action dynamics that contain preconditions, effects and probability distributions on effects of actions. In this way, deterministic action models are obtained to be used in the future as planning operators. Similarly in [18], probabilistic relational rules are generated and used in planning. These systems use offline learning processes which means randomly generated training sets are used in their learning phases and their systems' performances are analyzed in simulations. In the system of this thesis, however, incrementally built experience gathered from real world experiments are used to build the hypothesis space by the learning process. In addition, guidance in probabilistic planning through the use of these hypotheses is also performed in the real world.

In [19], probabilistic planning is used to select the best sequence of actions to minimize time in robot manipulation tasks. Similar to the approach proposed in this thesis, they apply a learning algorithm to adapt the plan of the robot to the unexpected changes which may occur during its plan execution. The actual effect probabilities of a generic and inaccurate set of rules are learned during execution. Generated rules are used to update the future plans of the robot. The proposed method is experimented on a service robot, and a manipulator cell which aims to clean lentils from a table. They use a model-based planner which is named as PRADA [18] for probabilistic planning in which states are assumed to be fully observable. The experiments show that the length of plans to complete the tasks decreases while the accuracy of the learning rules increases. Since they also use an adaptive planning strategy by integrating probabilistic planning and learning phases, their work is similar to ours. However techniques and

algorithms used to solve the problem and test scenarios are very different from each other.

In another study about adaptive planning [20], configuration parameters of a planner system is discovered by using machine learning techniques. At the beginning, the planner is executed on different planning domains like blocks-world, gripper, logistics, etc. Different configurations are applied while taking results from these preliminary experiments. After obtaining a large dataset, the results are used in machine learning process to extract learning rules about the parameters of planning. The configuration of the planner is changed by the rules coming from machine learning. Their planner allows embedding learning rules inside it, and in this way the configurations are set automatically. Experimental results show that this planner performs better with the configurations of learning rules rather than the one with default parameters. Although they propose an adaptive planning method like the method in this thesis, the authors use a different planner framework and different learning techniques, and they analyze their system on simulation experiments only.

2.3 Previous Work and Contribution of the Thesis

In this thesis, an experience-guided planning method is proposed to improve the performance of a mobile robot in its multi-object manipulation tasks. The objective of the robot is to maximize the number of successfully manipulated objects in the environment. Since many failures may occur because of various reasons, the robot can show better performance after decreasing failures in the execution. Therefore, a learning process is required to build experience from previous failure cases of the robot. After the robot gains enough experience from the learning phase, it can apply the results on its future task executions. In addition to the learning-guided planning method proposed in this thesis, the author's research group has a prior work that focuses on the same purpose [3]. In this work, the same experiential learning framework [1, 2] is used to generate rules which will guide the planner to decrease failures in action execution. The main difference between two studies is the type of planner used in the system. Since they use different planner types, planner guidance methods of them also differs from each other. A forward-chaining temporal planner, named as TLPlan [21] is used to generate plans of the robot in [3]. This planner allows using domain control

knowledge to reduce search space. The learning rules are embedded to the system by converting them to the control knowledge. Therefore, generation of the plan can be modified according to the rules by either selecting a branch in the search tree or pruning it. In this way, the robot can avoid failures and improve its performance on its object manipulation task.

Three types of domain control knowledge are defined in the previous planning system. Search control rules [22] and plan operators [23] are two control knowledge alternatives to guide planning in the proposed method. These types of control knowledge are used in three ways to guide the planning as described below:

- **Derivation of control formulas:** Search control rules are expressed as logical formulas by using Linear Temporal Logic (LTL) which is a way of representing temporal modalities as logical statements. Search control formulas are considered as rejection rules in the planning domain. Therefore, some branches of the search tree can be eliminated according to these learning rules. For example, if the robot fails while picking an object up according to a learning rule, the planner does not select corresponding action on this object.

- **Precondition update:** Updating preconditions of actions defined in the planning domain is another way of rejecting failure cases in plan generation. The effects of precondition update is similar to the previous method. This one also abandons the context of learning rules completely in the planning domain.

- **Adaptive cost assignment:** In this method, the cost of a planning operator is updated according to the weight of the corresponding learning rule. This approach does not eliminate the possibility of selecting related branches in the search tree. It only decreases the weight of the branches that represents failure cases in learning rules.

Analysis of three guidance methods shows that *the precondition update* method performs better than the other two methods in terms of efficiency. Because *the precondition update* method can reduce the search space more than the others. While the number of objects increases, the performance of *the precondition update* can be noticed easily. However, this method prunes the corresponding branch of failure case

completely in the search tree. On the other hand, *the adaptive cost assignment* to plan operators still gives chance to the failure cases by only decreasing the weight of their branch. In this way, the robot may select the action on an object when there is no other alternative although it has encountered with failure beforehand while executing that action on the object. Since probabilities attached to the rules by learning can be used in guidance in the last approach, it is also more efficient than the others in terms of the usage of information provided by learning.

Although the objective and used learning framework are the same in the previous work, the planning system is completely different in this thesis. The probabilistic planning method proposed in this thesis provides to use the computed probabilities for guidance. This can only be performed in one approach (*cost assignment*) of the previous work. Also execution of a policy as a plan is useful for adapting to sudden changes in the environment. In other words, uncertainty in observations can also be handled in a probabilistic planning method by representing the world state of the robot with possible belief states. A comparison with this previous work is provided in the section presenting experiments.

3. BUILDING ACTION EXECUTION EXPERIENCE

The main objective in this study is to develop methods for improving success rates of an autonomous robot's task execution through the use of experience. Particularly, the case study involves a mobile robot with an RGB-D sensor and a gripper manipulating several objects scattered around.

3.1 Object Manipulation Case Study

In the investigated case study, the goal of the robot is to transport as many objects as possible to its destination in a given time period without any harm to its environment. In the scenario, the robot selects the order in which the objects are to be manipulated, and moves these objects to their destinations. The robot needs to determine which object to fetch next to maximize the number of objects transported without a failure. In this context, different optimization criteria can be considered, such as choosing objects according to their locations in order to minimize the distance travelled or primarily targeting the objects on which the robot has the best manipulation performance.

In this study, the robot is initially run to build its experience by an experiential learning process to determine its abilities on object manipulation. Then, this experience is used as a guidance for the robot's next decision. Hypotheses generated by the experiential learning process can be used in planning to decide on the next object to manipulate. The previous study of author's research group uses a deterministic planner for this purpose, which uses only contexts of hypotheses [3]. In this study, a probabilistic planner is proposed that can use probabilities of hypotheses as well.

In the case study, the robot starts its plan by first finding the locations of the objects in the environment (i.e., the object locations are not known apriori). If it cannot detect or recognize any object in its current field of view, it executes action *search* to find any object by exploration. Whenever it finds an object, it stops its movement and executes $moveTo$, $pickUp$, $transport$ actions in sequence, if there is no failure. Whenever

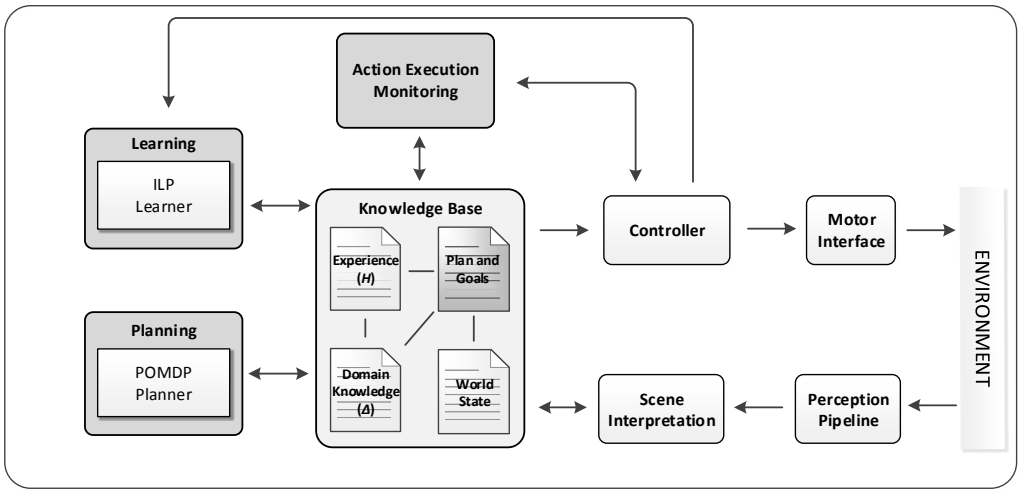

Figure 3.1: System architecture in which the adaptive planning method is used [2].

a failure is detected, the corresponding observation along with its related context is encoded in the knowledge base (KB) of the robot.

Supplementary sensing actions are also designed for detecting failures occurring out of sensor range. For example, for action $pickUp$, when the object is out of view (i.e., when it is close to the robot), the RGB-D sensor cannot detect it. If the robot senses that there is a failure by its pressure sensors inside the gripper, it moves backward to decide on whether the object is in its original form, and another trial would be safe. If the robot can not sense the object, an unknown failure is assumed, and this observation is registered to the KB.

3.2 Supplementary System Units

The overall software system of the robot contains various tightly integrated components. The schematic representation of the system architecture can be seen in Figure 3.1. Here, the information gathered from the environment is processed in *Perception Pipeline* with some algorithms to prepare it for *Scene Interpretation*. KB is the data store which includes data about world state, plans and the experience of the robot. All units access to the KB. The generated plan by the planner is executed in the *Controller* unit which manages the actuators of the robot (*Motor Interface*). The required processes for object recognition/segmentation, scene interpretation, action execution monitoring and learning are presented in the following subsections.

Figure 3.2: Outputs of LINE-MOD (left), LINE-MOD&HS (middle) and Segmentation (right) [24].

3.2.1 Scene interpretation

A unit, namely *Scene Interpreter*, is developed to maintain a consistent world model in the *KB* [24]. Information gathered from the environment by various sensors of the robot are processed and fused before being stored into the *KB*. The consistency is provided by integrating outputs from different vision methods. Template-based algorithms, LINE-MOD [25] and LINE-MOD with HS histograms [26], are used for recognizing the objects by processing the data coming from camera sensor. Templates of the objects are provided to *Scene Interpreter* beforehand. Furthermore, some information about objects like category, color, shape, size, etc. are also given as background information. When the robot recognizes an object, it can find extra information about the objects. Moreover, unknown objects (details about them are not located in the background information) are detected with a depth-based segmentation method which uses Euclidean clustering to segment the 3D point cloud data of the objects. In Figure 3.2, example outputs of LINE-MOD, LINE-MOD with HS histograms and segmentation can be seen. Here, the output of the LINE-MOD algorithm gives only the type information of the objects. On the other hand, LINE-MOD with HS histograms can detect the color of the objects as well. Therefore the output of this algorithm also shows the colors of the objects in Figure 3.2. Since the segmentation method can only extract the point clouds of the objects in the scene, the found objects are labelled as unknown. The locations of the objects are determined by fusing the outputs of all these sources. In addition to maintaining the consistency, *Scene Interpreter* applies filtering on noisy data acquired through sensors. The filtering process is performed by updating the information maintained in the *KB* considering temporal and domain-specific information during the evaluation of new observations. Moreover, the appropriate symbolic predicates that represent the world state are also

Table 3.1: List of predicates maintained by *Scene Interpreter*

Predicates	Explanation
obj_i	Object *i* is stored in the *KB*.
$holding$	There exists an object inside gripper peddals of the robot.
$moving$	The robot is moving.
$turning$	The robot is turning.
$closingGripper$	The gripper peddals are being closed.
$liftingUpGripper$	The gripper is moving upward.
$objInGripLoc_i$	Object *i* is between the gripper peddals while the robot is executing action $pickUp$.
$nearObjLoc_i$	Object *i* is close enough to the robot for a suitable grasping.
$tallObj_i$	Object *i* is tall enough to be seen by sonar sensors of the robot.
$objDetectedBySonar_i$	Object *i* in front of the robot is detected with sonar sensors.
$objFallSoundDetected$	The robot detects an unexpected sound in the environment via its microphone.

generated by *Scene Interpreter*. For example, whether an object is held or not is detected by checking the status of the tactile sensors inside the gripper pedals and $holding$ predicate is created for the corresponding object in the *KB*. The predicates that can be created in this system are explained in Table 3.1. Here, several types of sensors are used to create the predicates such as, sonar sensors to detect an object in front of the robot, tactile sensors inside gripper peddals to detect the object in the gripper, a microphone to detect the sound in the environment, and the RGB-D camera to specify the locations of the objects in the scene. Observations acquired through the execution of the plan are pre-processed by using *Scene Interpreter* [24].

3.2.2 Action execution monitoring

Different types of failures may happen during action execution of a robot. These failures should be detected to build experience on them. An *Action Execution Monitoring* system is developed to detect failures that occur while executing an action in the used robot system [27]. Metric temporal formulas defined specifically for each action by using Metric Temporal Logic (MTL) are used to monitor the action execution simultaneously [28]. $\wedge$ (and), $\vee$ (or) and $\neg$ (not) are the boolean connectives, and $\Box_\tau$ (always), U_τ (until), $\Diamond_\tau$ (eventually) and $\bigcirc_\tau$ (next) are the temporal logic operators that can be represented in MTL. Here, τ is a time interval (i.e., $\geq t$, $\leq t$, $< t$ and $> t$ where $t > 0$) used in temporal logic operators. In the case that a formula f is always true for all world states between the specified time period τ, this expression is represented as $\Box_\tau f$. If a formula f will be valid in at least one of future states between the time period τ, $\Diamond_\tau f$ is used. $\bigcirc_\tau f$ expresses that the formula f will become true

in the next state between the time period τ. Finally, to represent that the formula f_1 is true until the formula f_2 becomes true, $f_1 U_\tau f_2$ is used.

Metric temporal formulas for action *moveTo*.

Three metric tamporal formulas are defined for action *moveTo*. Firstly, the robot should have detected the object before it executes action *moveTo* on it. Equation (3.1) is created for checking the presence of object i in the KB. When the robot starts its movement toward object i, it should keep moving or turning until it gets close enough to this object. This is monitored with Equation (3.2) where t_s represents the starting time of the action and $t_{planning}$ represents the required time for path planning. The robot is expected to reach at object i in Equation (3.3) within the time period of reaching the object which is represented with $t_{moveToObj}$.

$$\Box_{\geq t_s}(obj_i) \tag{3.1}$$

$$\Box_{\geq t_{planning}}((moving \vee turning) U_{\geq t_s}(nearObjLoc_i)) \tag{3.2}$$

$$\Diamond_{\geq t_{moveToObj}} nearObjLoc_i \tag{3.3}$$

Metric temporal formulas for action *pickUp*.

Object i on which action *pickUp* is being executed should either exist in the KB or be held in the gripper during the execution of action *pickUp* (Equation (3.4)). After the robot grasps object i, the object should be inside the gripper peddals when the gripper is being lifted (Equation (3.5)). If any unexpected sound is heard within the time period in which the robot is approaching object i to grasp, this sound is interpreted to be caused from the falling of the object (Equation (3.6)). Until the object is adjusted to be in a suitable position to grasp, the robot should continue either moving or turning (Equation (3.7)) and the gripper peddals should not be closed (Equation (3.8)). If object i on which action *pickUp* is being executed is tall enough to be seen by sonar sensors of the robot, it should be detected by sonar sensors after the robot is placed itself into a suitable position to grasp it (Equation (3.9)).

$$\Box_{\geq t_s}(obj_i \vee holding) \tag{3.4}$$

$$\Box_{\geq t_s}((\neg liftingUpGripper) U_{\geq t_s}(holding)) \tag{3.5}$$

$$\Box_{\geq t_s} \neg (objInGripLoc_i \wedge objFallSoundDetected) \quad \textbf{(3.6)}$$

$$(moving \vee turning) U_{\geq t_s} (objInGripLoc_i) \quad \textbf{(3.7)}$$

$$(\neg closingGripper) U_{\geq t_s} (objInGripLoc_i) \quad \textbf{(3.8)}$$

$$(\neg tallObj_i) \vee (\neg objInGripLoc_i \, U_{\geq t_s} \, objDetectedBySonar_i) \quad \textbf{(3.9)}$$

Metric temporal formulas for action *transport*.

Since the robot may drop object *i* down while it executes action *transport* on it, the sound in the environment should also be monitored during this action to detect possible failures (Equation (3.10)). The robot is expected to release object *i* in the gripper after a time period t_{drop} which represents the approximate time to release an object without any failure (Equation (3.11)). If the released object is tall enough to be seen by sonar sensors of the robot, it should be detected by sonar sensors when the robot has released it (Equation (3.12)).

$$\Box_{\geq t_s} \neg (objInGripLoc_i \wedge objFallSoundDetected) \quad \textbf{(3.10)}$$

$$\Box_{\geq t_{drop}} \neg holding \quad \textbf{(3.11)}$$

$$(\neg tallObj_i) \vee (\neg objInGripLoc_i \, U_{\geq t_s} \, objDetectedBySonar_i) \quad \textbf{(3.12)}$$

Metric temporal formulas of actions *moveTo*, *pickUp* and *transport* are controlled by the *Action Execution Monitoring* system and they are evaluated with a goal progression algorithm [29]. Formulas defined for action *pickUp* are used to collect data for building experience in the learning phase. However, the other formulas (defined for actions *moveTo* and *transport*) are also required to provide continuous execution of the robot.

3.2.3 ILP learning

Inductive Logic Programming (ILP) is used in this thesis as an experiential learning framework for autonomous robots [1, 2]. Each detection of an action execution failure contributes to derive hypotheses represented in first-order logic and used to build the experience. The Progol algorithm is used in ILP based-learning [30]. The contexts of hypotheses are constructed from the attributes of observed objects and the relevant facts from the world state. Then, these contexts are mapped to the outcomes (*success*

or *failure*) of corresponding actions. A probability (P) is attached to each hypothesis. An example hypothesis is given in Equation (3.13) where the probability of failure of action $pickUp$ for green cylindrical objects is determined as 0.22. The antecedent part of this rule is represented as the context, and the conclusion is the outcome of the action.

$$category(cylindrical) \wedge color(green) \Rightarrow failure(pickUp)(P:0.22) \quad \textbf{(3.13)}$$

The learning algorithm aims to find the most general set of hypotheses by evaluating the gathered observations. The probability value of each hypothesis is determined by calculating the ratio of positive observations corresponding to failure cases over all observations (negative and positive) covered by that hypothesis. The derived hypotheses and their probabilities are updated or ruled out while new observations arrive. As an example, the learner is assumed to get following observations:

obs_1 :

$$category(obj_1, bowlingPin) \wedge color(obj_1, green) \wedge material(obj_1, plastic) \wedge size(obj_1, medium) \wedge failure(pickUp)$$

obs_2 :

$$category(obj_2, bowlingPin) \wedge color(obj_2, red) \wedge material(obj_2, plastic) \wedge size(obj_2, medium) \wedge failure(pickUp)$$

obs_3 :

$$category(obj_3, ball) \wedge color(obj_3, purple) \wedge material(obj_3, plastic) \wedge size(obj_3, medium) \wedge success(pickUp)$$

where obs_i corresponds to an observation instance taken after executing action $pickUp$, and obj_i corresponds to the object on which the action is executed. After applying ILP learning on this observation set, the hypotheses given in Equation (3.14) and Equation (3.15) are derived.

$$category(bowlingPin) \Rightarrow failure(pickUp) \quad \textbf{(3.14)}$$

$$category(ball) \Rightarrow success(pickUp) \quad \textbf{(3.15)}$$

Probabilities of these hypotheses are assigned as 1 since there are no ambiguities. Furthermore, the learning process can also benefit from background knowledge

whenever it is available while deriving hypotheses. Unifications of hypotheses can be performed more realistically by using background knowledge. Therefore hypotheses can be generalized effectively.

The robot is assumed to get the following observation after deriving hypotheses in Equations (3.14) and (3.15):
obs_4 :

$$category(obj_3, bowlingPin) \wedge color(obj_3, green) \wedge material(obj_3, plastic) \wedge size(obj_3, large) \wedge success(pickUp)$$

The previous hypotheses are generalized and converted to hypotheses in Equations (3.16) and (3.17) after evaluating observation obs_4.

$$category(bowlingPin) \wedge size(medium) \Rightarrow failure(pickUp) \quad \textbf{(3.16)}$$

$$category(ball) \vee size(large) \Rightarrow success(pickUp) \quad \textbf{(3.17)}$$

4. LEARNING GUIDED PROBABILISTIC PLANNING

The POMDP planning formulation for generating policies in multi-object manipulation scenarios is given in the following subsections.

4.1 State Space Definition

An object composition, denoted as oc, represents a structure that encapsulates qualitative and spatial information about an object. The i^{th} object composition formulated as $oc_i = (obj_j^{attr}, obj_j^{loc_m}, rel_j^{loc_r})$ contains attributes and the semantic location of an object obj_j. Since an object may be placed in all possible locations in the environment, more than one object composition exist for an object. Object attributes, denoted as obj_j^{attr}, specify predetermined features (category, color, material, height, width, etc.) of object obj_j. Although the location of an object is valued in continuous space in the KB of the robot, the continuous-valued locations are converted to semantic locations, which are denoted as loc_m for the m^{th} semantic location, to provide discretization in state definition of POMDPs. Therefore, $obj_j^{loc_m}$ indicates that the object obj_j stands in location loc_m. In addition to semantic locations, one more field is attached into oc_i, denoted as $rel_j^{loc_r}$, to represent the relative location of object obj_j to the robot's position loc_r. In the current system, only one spatial relation between an object and the robot is defined as $rel_j^{loc_r} = \{infront, \neg infront\}$. The relation $infront$ indicates that the mentioned object stands in front of the robot. Since the robot moves continuously in the environment, an object may situate in front of the robot according to the robot's location. This relation can be applied to at most one object at any state. This relation can also be used to extract one extra information. Since no variable is included in the proposed state definition for representing the semantic location of the robot, this information can only be extracted from the location of an object that exists in front of the robot. However, the location of the robot is unknown in the states in which no object is in front of it.

For all combinations of object attributes and semantic locations, different object compositions are created. The set of all possible object compositions is indicated as $OC = \{oc_1, oc_2, oc_3, ...\}$. If the maximum number of object compositions that can exist in the world at any moment is determined as N, a state set of a POMDP that represents the current scene of the world is given in Equation (4.1) where s_{scene} is a subset of OC.

$$S_{scene} = \bigcup \{s_{scene} \,|\, s_{scene} \subseteq OC, |s_{scene}| \leq N\} \tag{4.1}$$

State space of a POMDP $S = S_{scene} \cup \{s_{holding}, s_{failure}\}$ is formed as an aggregation of various states. The state definition is expanded with two more states, namely $s_{holding}$ and $s_{failure}$. Differently from state set S_{scene}, each of these states specifies only one condition. The state $s_{holding}$ represents that an object exists in the gripper of the robot during transportation. The state $s_{failure}$ indicates that a failure is encountered in the previous action execution. The purpose of separating these states from the state set S_{scene} is to create an abstraction from the other irrelevant state components in S_{scene}. This abstraction is also useful for reducing the number of states generated by the POMDP. Otherwise, if one of the states $s_{failure}$ and $s_{holding}$ are added as an extra variable to S_{scene} component, the size of the state space increases.

4.2 Actions

The actions considered in this work can be divided into two groups according to their functionalities: main actions and sensing actions. Main actions in this formulation are moving to an object obj_j in object composition oc_i, instantiated for each oc_i as $moveTo(oc_i)$, picking up an object obj_j in object composition oc_i, denoted as $pickUp(oc_i)$ and putting the object in the gripper down after reaching the destination, denoted as $transport$. Action $transport$ is executed as a combined action that includes both transportation of the object to the destination and putting the object down. Note that this action is performed in the same way for all objects held in the gripper, therefore a decision for selecting an object to be transported, and correspondingly an instantiation, is not needed. Even so, it can be easily converted to allow success probabilities special to each object composition by request. On the other hand, decisions on object selection for $moveTo$ and $pickUp$ actions have an impact on task

Table 4.1: Symbolic representation of the state set for the preconditions of actions

Actions	Start States
$moveTo(oc_i)$	$\{s_{scene} \mid oc_i \in s_{scene}, (rel_j^{loc_r} = \neg infront)$ for obj_j in $oc_i\}$
$pickUp(oc_i)$	$\{s_{scene} \mid oc_i \in s_{scene}, (rel_j^{loc_r} = infront)$ for obj_j in $oc_i\}$
$transport$	$\{s_{holding}\}$
$search$	$\{s_{scene} \mid \mid s_{scene} \mid = 0\}$
$monitor$	$\{s_{failure}\}$

execution performance, in terms of minimizing both time to complete and the number of failures. This results in the need for instantiations of *moveTo* and $pickUp$.

Sensing actions, denoted as *search* and $monitor$, are executed for acquisition of knowledge about the environment at any moment. The robot requires to gather information from the environment when no stored information exists in the KB. Action *search* is the wandering operation to find new objects in these situations. On the other hand, action $monitor$ is executed when the information already available in the KB needs to be updated. The robot senses the object it currently operates on to verify the knowledge on the situation of this object. Monitoring action is designed as sensing the object on which the last main action is operated, by the robot positioning itself to an appropriate location where it can see the object properly. In this way, the knowledge of the robot about the object is updated according to whether the object is sensed again or not.

4.3 Observations

Each observation gathered after an action execution (i.e., after each transition between two world states) is designed similar to the state definition and denoted as $O = O_{scene} \cup \{o_{holding}, o_{failure}\}$ where O_{scene} indicates the combinations of object compositions and $o_{holding}$ indicates that the predicate $holding$ is taken from the environment when there exists an object inside gripper peddals of the robot. The $o_{failure}$ state represents the failure happened in the last action execution before taking the current observation.

4.4 Transition Model

The hypotheses generated by the experiential learning algorithm are to be used in the planning process to decide on the next object to be manipulated. The transition

probabilities of $pickUp(oc_i)$ action to the states which represent the success and failure of action execution are specified using the probabilities attached to the derived hypotheses. If an object composition is matched with the context of a hypothesis, the corresponding $pickUp$ failure probability is set to the probability of the corresponding failure hypothesis. If an object composition is in the scope of multiple hypotheses, the hypothesis with the lowest probability is applied to avoid failures.

Transitions defined in the POMDP system, are explained individually for each described action in the following. Table 4.1 is given for formulating the state sets that represent the preconditions of each action symbolically and Table 4.2 gives the formulation of the state sets that represent the effects (excluding failure cases) of each action symbolically. A failure case in execution of any action cause a transition to state $s_{failure}$.

Action *moveTo* **transitions**.

Action $moveTo(oc_i)$ can be selected as the next action if object composition oc_i exists in the current belief of the robot, and $rel_j^{loc_r}$ in oc_i should be $\neg infront$ before executing $moveTo(oc_i)$ action as in this case executing an extra movement would be redundant. However, there is no limitation about any other objects to be in front of the robot, since the robot may decide on moving to a distant object instead of picking up the object located in front. This decision is based on the evaluation of $pickUp$ success performances regarding all objects in the environment and their distances from the robot. Since the selection of each action is determined by evaluating the total reward which indicates the cumulative sum of the rewards of the current action and the immediate rewards of all actions that may come after this action, the robot decides

Table 4.2: Symbolic representation of the state set for the effects of successfully executed actions

Actions	Successful End States
$moveTo(oc_i)$	$\{s_{scene} \mid oc_i \in s_{scene}, (rel_j^{loc_r} = infront)$ for obj_j in $oc_i\}$
$pickUp(oc_i)$	$\{s_{holding}\}$
$transport$	S_{scene}
$search$	S_{scene}
$monitor$	S_{scene}

to execute either picking up the object in the front or moving to another object with higher $pickUp$ success performance by considering future effects of these two actions.

Transition probabilities to the states that may be encountered after the execution of action $moveTo(oc_i)$, is determined according to the general success performance of the moving operation of the robot and the distance between the robot and the target object in oc_i. The semantic locations of the objects are defined by taking the starting position of the robot as a reference. As described earlier, the location of the robot can be obtained only when any object in the environment exists in front of the robot. Therefore, all predefined semantic locations and distance measures of them are relatively updated according to the robot location in the states where it is known.

After the execution of action $moveTo(oc_i)$, the object in oc_i is expected to be in front of the robot if the action is successfully terminated. The information about other objects found in the environment before the execution of action $moveTo(oc_i)$ is updated with a new observation taken after action execution. The gathered observation is considered to be noisy and the object compositions of the next state after action $moveTo(oc_i)$ is executed may stay the same or change according to the incoming observation. The possibility of noise in the observation is considered for the transitions to address unpredicted changes (e.g., the existing objects in the environment may disappear by external intervention) in scenes. Whenever a failure is encountered in action execution, the next world state becomes $s_{failure}$.

Action $pickUp$ transitions.

The preconditions for action $pickUp(oc_i)$ include that the robot's gripper should be empty (i.e., any state except $s_{holding}$), and the object obj_i in oc_i should be in front of the robot ($rel_j^{loc_r}$). When this action is successfully executed, the effect $s_{holding}$ occurs. If the robot discovers an action failure, the state becomes $s_{failure}$.

The transition probabilities of $pickUp(oc_i)$ action to the states which represent the success and failure of action execution, are determined based on the matchings between the contexts of failure hypotheses and the object composition. Since a hypothesis is based on attributes of objects, grasping success of the objects that have the same set of attributes with the hypothesis is assigned to the probability of it.

If an object is in the scope of multiple hypotheses, the hypothesis with the lowest probability is applied. Whenever an object could not be matched with any hypotheses, the general success rate of the experiments is given as grasping success probability. Success probabilities for these actions are determined based on the matchings between the contexts of failure hypotheses and the object composition.

Action *transport* **transitions**.

Action *transport* entails the transportation of the object in the gripper to the destination and putting it down. Therefore, the precondition of this action is holding an object ($s_{holding}$). At the end of the execution of this action, the robot is expected to put down the object in hand to the destination. Transition probabilities for passing to possible ending states that have successful and failed action effects are assigned empirically-determined values. The object compositions in the environment after a successful execution is determined with the new observation and the transition probabilities to each successful ending state are uniformly distributed. $s_{failure}$ cases are handled in the same way as $pickUp(oc_i)$ and $moveTo(oc_i)$ actions.

Action *search* **transitions**.

If the robot is in the world state that does not have any object composition, it searches the environment in order to find new objects. After the execution of *search* action, the robot may find itself in a world state from the state set S_{scene}. Since this is a sensing action, the aim of which is to provide additional world knowledge, this action does not have a success or failure effect. Therefore, the probability is equally distributed to the states in the state set S_{scene}.

Action *monitor* **transitions**.

When the robot encounters a failure during the execution of a main action, it transfers to the state $s_{failure}$ and needs to refresh its knowledge about the world. In such cases, action *monitor* is executed to provide additional information on the failure case. In this system, the monitoring action is specified as moving to a suitable position to sense the object on which the last manipulation action is operated. After monitoring, the belief state of the robot is updated and the next action is determined according to this

new belief. After the execution of action *monitor*, the robot may find itself in a state with one of the combinations of object compositions. The next action is determined by evaluating the current belief state of the robot. Transition probabilities of *monitor* action are uniformly distributed to all states in the state set S_{scene}.

4.5 Observation Model

Each observation gathered after an action execution (i.e., after each transition between two world states) is designed similar to the state definition. The observation model represents the probabilities of observations gathered after a transition to a new state by executing an action. In this work, each observation is encoded into the *KB* by *Scene Interpreter*. The robot determines its belief state based on an observation which is formulated identical to the state definition. Uncertainties in sensing are handled by *Scene Interpreter* and the output of this process is a set of observations which reduces the complexity of POMDP planning.

4.6 Reward Model

In the investigated case study, the objective is to maximize the number of objects to be transported by the robot. Reducing failures is essential for this purpose, thus a penalty is given when the robot is in state $s_{failure}$. The system is awarded if the intended transition to a new state is achieved by a particular action. Sensing actions also cause the system to be penalized due to their cost of execution.

4.7 Learning Guided Planning for Multi-Object Manipulation

In the multi-object manipulation scenario of this thesis, the goal of the robot is to transfer as many objects as possible to the destination without failure. In this case, the target objects are sequentially selected and moved to the destination. If the goal of the robot is defined as transferring as many objects as possible in a given time period, at each time step, it needs to decide on which object to manipulate next, to maximize the number of objects transferred without a failure. In this context, different factors can be considered, such as choosing objects according to the distances between the locations of the object and the robot in order to decrease the time spent while moving towards

the object, or primarily choosing objects on which the robot has best manipulation performance.

The hypotheses generated by the experimental learning algorithm are used in planning process to decide on the next object to manipulate. The transition probabilities of $pickUp(oc_i)$ action to outcome which represent either success or failure, are specified by using the probabilities of the learned hypotheses. Since a hypothesis is based on a set of attributes of objects, grasping success of the objects that have the same set of attributes with a hypothesis is assigned to the corresponding probability of that hypothesis. If an object is in the scope of multiple hypotheses, the hypothesis with the lowest probability is applied. Whenever an object could not be matched with any hypotheses, the general success rate of the robot's grasping performance is given as the grasping success probability for this object.

5. EXPERIMENTS

Experiments were conducted in the department's corridor of ITU without any special environmental lighting. People are allowed to observe the experiments during which their movements change illumination conditions. The robot maintains its KB during runtime without any human intervention in the face of the challenges of noise in sensory data, partial observability and unexpected situations.

5.1 Experiment Setup

In order to evaluate the performance of the proposed system, real robot experiments were conducted with a Pioneer 3-DX robot which can be seen in Figure 5.1. It is equipped with several sensors to perceive its environment. A Hokuyo UTM-30LX laser rangefinder is mounted on top of the robot facing forward for mapping and localization, and an ASUS Xtion PRO RGB-D camera is placed on top of the laser rangefinder for 3D object recognition and segmentation. The robot has a 2-DOF gripper to manipulate objects.

The system is implemented on ROS (Robot Operating System) framework [31] using an Intel Core i7 laptop with Ubuntu 12.04 for autonomous control of the robot.

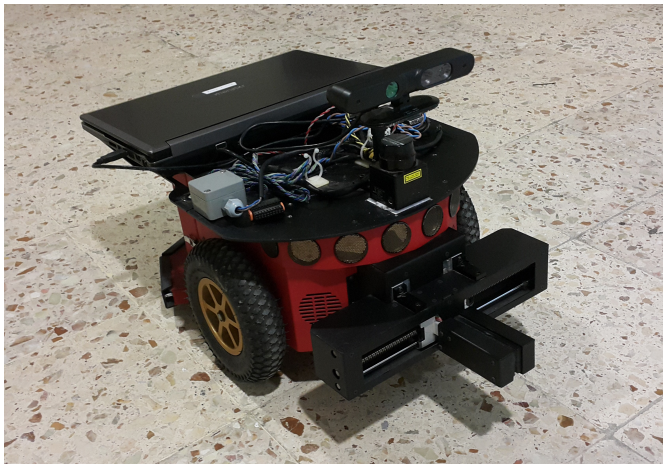

Figure 5.1: Pioneer 3-DX robot with its equipments used in the experiments.

5.2 POMDP System Parameters

In the case study of this thesis, the following object attributes are taken into account: category ($oc_i^{category}$), color (oc_i^{color}), material ($oc_i^{material}$), size (width, oc_i^{width} and height, oc_i^{height}), solidity ($oc_i^{solidity}$) and shape (oc_i^{shape}). Here, the experiments are performed with the attributes represented in the learned hypotheses. Many other attributes can also be added to the system by request. Before executing the POMDP planner to generate a policy, a configuration file which specifies the number of states, actions, observations, the transition model, the observation model and the reward model of the problem formulation should be specified. In this phase, object compositions are created by considering possible color and type attributes provided as background information beforehand to the system. Furthermore, information of possible objects are also given to decrease the number of created object compositions. Because higher number of object compositions will increase the complexity of the POMDP planner by increasing the number of states, actions and observations in the formulation. Additionally, each object is assumed to be unique and to exist only in a single location at any time frame. Thus, a world state cannot have more than one instance of a single object in the experiments. However, the objects can be located in different locations at different world states. The semantic locations are assigned values proportional to their distances from the initial start position of the robot, and these are used to update the success probabilities of *moveTo* actions. As described earlier, the values that represent the reachability of locations are updated with respect to the robot in the states in which the location of the robot is known. Otherwise, default values are used.

5.3 The Scenarios and Experiment Results

Initially, the performance of the robot is tested on a set of five different objects from various colors and categories: two plastic bowling pins whose colors are green and red, a green plastic cylindrical object, a small purple ball and a big red ball. The environment is divided into two regions that represent the semantic locations of objects. The robot, the objects and the possible semantic locations, which can be denoted as $oc_i^{loc} = (room_1, room_2)$, can be seen in Figure 5.2. As preliminary experiments, actions $moveTo(oc_i)$, $pickUp(oc_i)$ and $transport$ are executed ten times

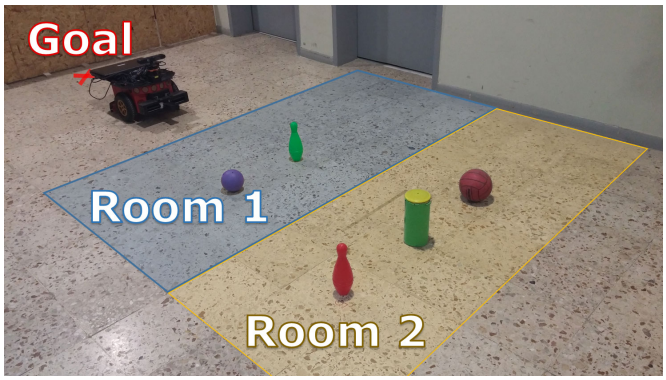

Figure 5.2: The environment: The goal point of the robot is shown with a red cross. $room_1$ and $room_2$ are denoted as blue and yellow areas.

for the object set randomly cluttered in the environment to measure the overall performance of the robot on multi-object manipulation tasks. Preliminary experiments are performed by executing the robot with a predetermined sequence of actions. The robot starts its execution by detecting objects in the environment and moves to them according to their locations. It grasps the object it has moved in the previous action and transports it to the goal position. If a failure occurs in one of these actions, it detects the failure and changes its target object. Here, the aim is to obtain a general success rate of the robot for each action. This information is required to determine the default values of action success performances in the POMDP model. By executing the policy which is generated with default values, the performance of the policies generated with guidance and without guidance can be compared. Therefore the contribution of using learning hypotheses in the POMDP transition model can be revealed. It has been observed in the preliminary experiments that the robot has an overall *pickUp* success rate of 96% and *putDown* success rate of 100%. Since quantifying *moveTo* success rate is not as easy as the measurement of *pickUp* and *putDown* success rates (the successfully manipulated number of objects cannot be used as a criteria for action *moveTo*), an intuitively determined approximate value is used for specifying its success rate. Learning hypotheses are only generated for action *pickUp* in the experiments. Therefore, the initially assigned success parameters for actions *putDown* and *moveTo* remains the same in the experiments with and without learning.

Two different scenarios are investigated to evaluate the performance of the experience-based planning system. Each scenario is repeated three times with human injected failures. The order of action executions in which a failure is injected should be

the same for a reliable comparison between the experiments with and without learning guidance. Therefore, the occurrence times of these failures are determined based on a randomly generated (by using a random number generator) failure probability distribution, and applied on the same order in both scenarios. The experiments are performed with and without applying learning outcomes to compare the results on the same failure distribution. The experiments without learning do not take the failure hypotheses into account, and the robot proceeds to the objects in order based only on their distances. Success probability of $pickUp$ action is determined in the same way for all object compositions according to the general success rate of $pickUp$ action in the preliminary experiments.

In the first scenario, objects with $category(bowlingPin)$ are externally taken away from the environment with a randomly (by using a random number generator) determined probability (67%) at the time of action $pickUp$. For this scenario, ILP generates the hypothesis given in Equation (5.1).

$$category(bowlingPin) \Rightarrow failure(pickUp)(P:0.67) \quad \textbf{(5.1)}$$

The locations of bowling pins are organized in different settings as follows: both located in $room_1$; one of them in $room_1$ and the other one in $room_2$; and both of them in $room_2$ in different experiments. The system's performance is evaluated for all these cases. The aim of locating bowling pins in different locations for each experiment set is to show the performance of the system in terms of selection between the closer object and the object with a higher probability of $pickUp$ success.

The execution trace of the first scenario is given in Figure 5.3 for the configuration given in Figure 5.2. The planning is guided with the failure hypothesis given in Equation 5.1 for the first scenario. In Figure 5.4, an illustration of this scenario is shown. Here, objects are represented as circles which are enumerated respectively for the green bowling pin, the purple ball, the red big ball, the green cylinder and the red bowling pin. The blue region indicates the field of view of the robot where the objects can be recognized. The objects colored gray indicate unseen objects. The circles with bold edges denote the objects with lower probability of $pickUp$ success. The recognized objects are shown in their respective colors. At the beginning, the robot

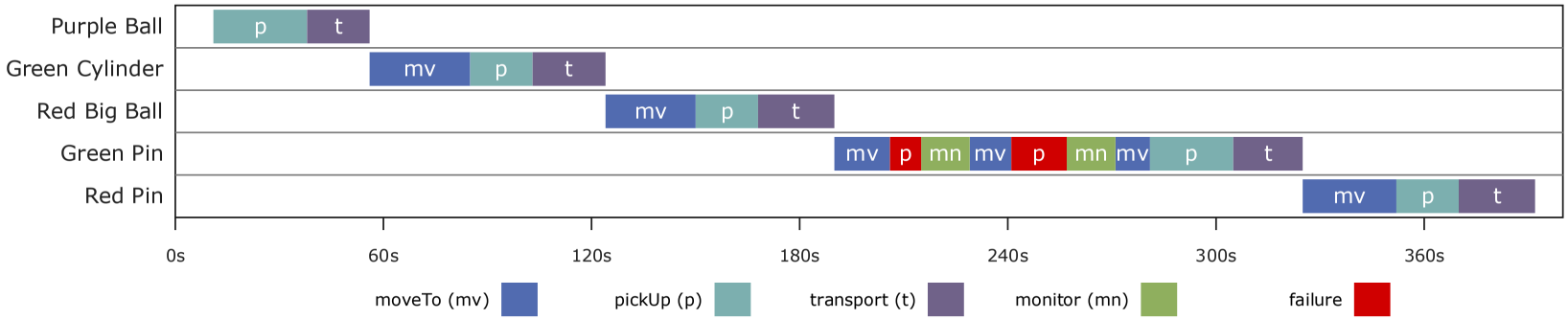

Figure 5.3: Timeline of the learning-guided plan execution with failures in bowling pins.

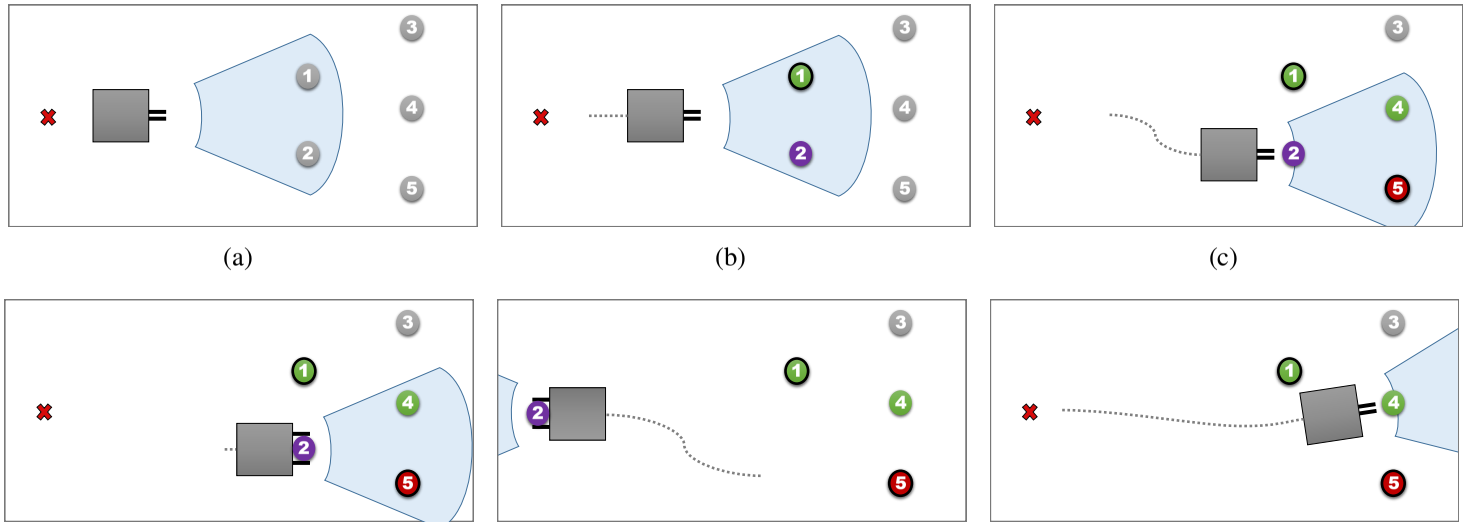

Figure 5.4: Graphical illustration of the first scenario.

starts the execution of its plan without any object in its *KB*, and it executes *search* action to find an object in the environment (Figure 5.4(a)). Note that *search* action is not shown in Figure 5.3 due to clarity. After *search* action, the robot detects the purple ball and the green bowling pin located in $room_1$ (Figure 5.4(b)). According to the success probabilities of action $pickUp$ for different objects that are determined with experience, the robot selects the purple ball to move to, since its $pickUp$ success probability is higher (Figure 5.4(c)). The robot detects the objects that are located in $room_2$ while moving to the purple ball. After transporting the purple ball successfully (Figure 5.4(d) and Figure 5.4(e)), the robot selects the cylindrical object (in $room_2$) with a higher success probability instead of the closer bowling pin (Figure 5.4(f)). Then, the big red ball is moved. Since the remaining objects (the green bowling pin and the red bowling pin) in the environment have the same success probabilities, and there is no other alternative object, if there is enough remaining time, the robot tries to manipulate the closer one (green pin) first. According to the random failure distributions, it is forced to fail two times while picking up these objects. The robot executes *monitor* action to update the information after each failure.

The results of time to complete another trial of the first scneario with and without learning is given as a histogram chart in Figure 5.5. Here, the number of successfully transported objects is shown in relation to time. The red intervals in the histogram

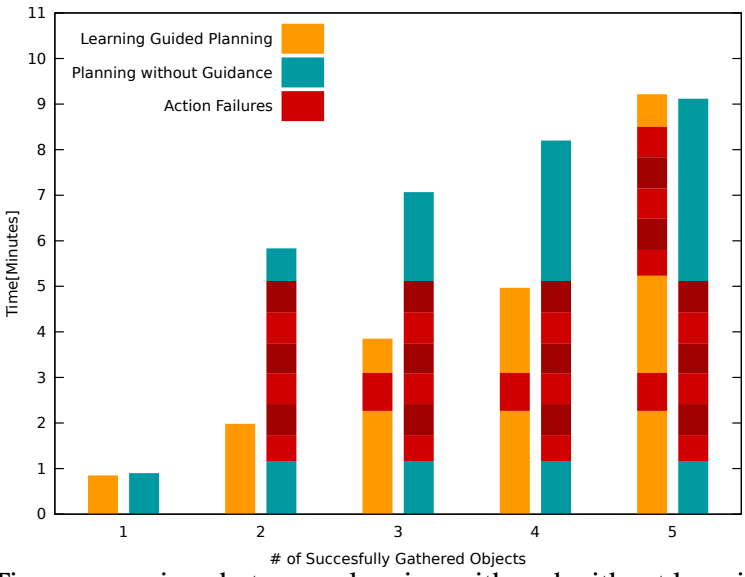

Figure 5.5: Time comparison between planning with and without learning guidance in accordance with number of successfully gathered objects (failures occur during picking up bowling pins).

bars indicate delays due to failures. Two shades of red are used to represent different trials with failure. The first failure is encountered earlier in the plan without guidance. The planner without guidance chooses the closest object to manipulate, since the success probabilities of all the observed objects are believed to be equal. However, the learning-guided planner manipulates the object further away from the robot instead of the closer object with lower success probability. The total required time to manipulate all objects scattered in the environment are approximately the same. However, it is important to note that although the number of manipulation trials in both executions are equivalent at the end of the experiment, the learning-guided planner postpones the manipulation of the objects with lower success probabilities until no other objects are available. As can be seen from the figure, when the robot uses its experience, the potential failures are postponed to a great extent. When the time permits, the robot attempts to move towards objects for which success is not guaranteed. However, only one failure occurs at meantime when the objects with higher success probabilities are still available in the environment. The reason of the robot to select the object with a lower success probability rather than the object with a higher probability is that the robot has not recognized the object yet at this time interval. During the first trial of pick up on the object with a lower success probability, the robot detects the object with the higher probability, and decides to manipulate the new object for its next action execution.

In the second scenario, the robot is forced to encounter failures by human intervention while manipulating objects in $room_1$. The objects are externally removed while the robot is executing $pickUp$ actions in this location. The learning algorithm uses background knowledge given in Equation (5.2) while extracting the hypothesis of the second scenario given in Equation (5.3) where obj_i represents an object, $locX$ represents the x coordinate of an object with respect to the global map, and $location$ represents the semantic location of the specified object. Although the objects in $room_2$ are further away than the objects in $room_1$, the planner selects the objects in $room_2$ since the robot first attempts to transport the objects with higher $pickUp$ success probabilities. However, in the experiments without learning, the planner only takes into account the distances of the objects.

$$
\begin{aligned}
locX(obj_i, X) \wedge 0 \leq X < 1.6 &\Rightarrow location(obj_i, room_1) \\
locX(obj_i, X) \wedge 1.6 \leq X < 2.6 &\Rightarrow location(obj_i, room_2)
\end{aligned}
\tag{5.2}
$$

$$
location(obj_i, room_1) \Rightarrow failure(pickUp)(P : 0.67) \tag{5.3}
$$

An analysis on the results of all experiments for each scenario is performed separately, and the average number of objects transported over time in the first and second scenario with and without learning are shown as histogram charts in Figure 5.6 and Figure 5.7 respectively. Here, the average number of successfully transported objects is shown in relation to time. The red intervals in the histogram bars indicate the average number of objects which the robot fails while manipulating. The blue parts in Figure 5.7 point out the natural failures occurred in the experiments with location failures. As can be seen from the figures, the required time to transport the same number of objects on average is lower with the learning-guided planner except the time of the first and last object' manipulation which exceed the time of the planner without guidance in Figure 5.7. Here, the reason of the unexpected surplus time between the results of the experiments with and without learning is the natural failure indicated with the blue marker. Although a natural failure has occurred at the beginning of the experiment, its negative effect is distinguished only at the first and last steps. Because the number

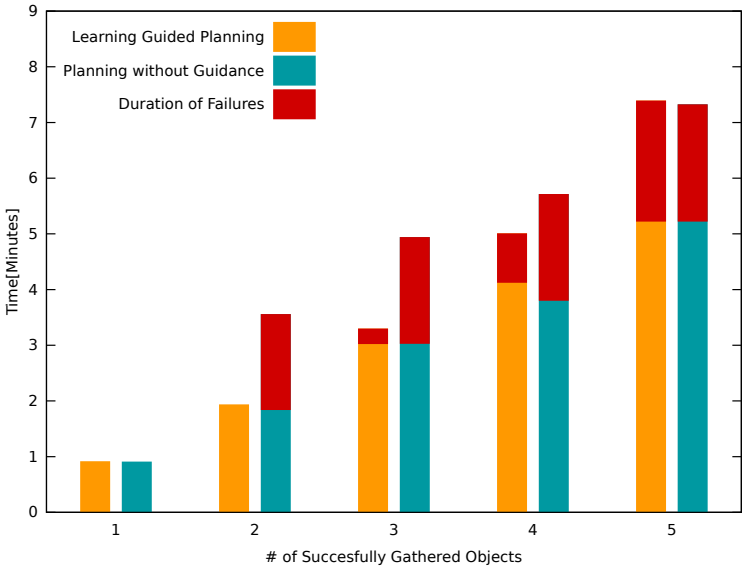

Figure 5.6: Time comparison between planning with and without learning guidance in accordance with number of successfully gathered objects on average (failures in bowling pins).

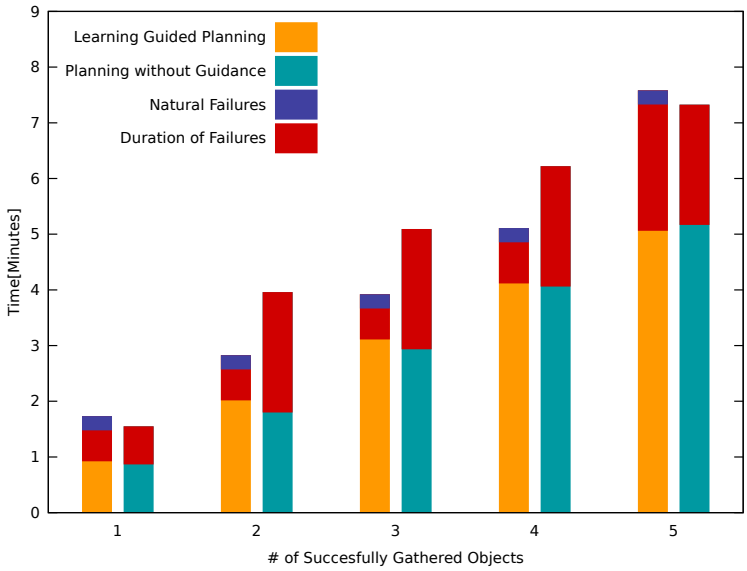

Figure 5.7: Time comparison between planning with and without learning guidance in accordance with the number of successfully gathered objects on average (failures in the objects located in $room_1$).

of failures in the experiments of the learning-guided planner is lower than or equal to (at least) the results of the planner without learning when the same failures occur in both experiments. If another natural failure would have occurred at any other time step (not before the first successful object manipulation), the unexpected surplus in the time period of the learning-guided planner will exist only at the end of the experiments (the last column of learning-guided planner). Because the learning-guided planner postpones the selection of the objects with lower success probabilities until there is no

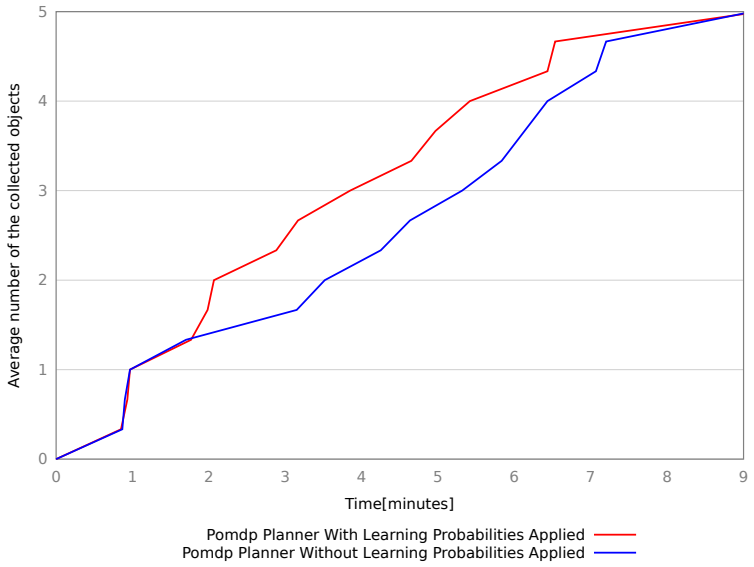

Figure 5.8: The average number of objects transported through time in the first scenario with failures in bowling pins.

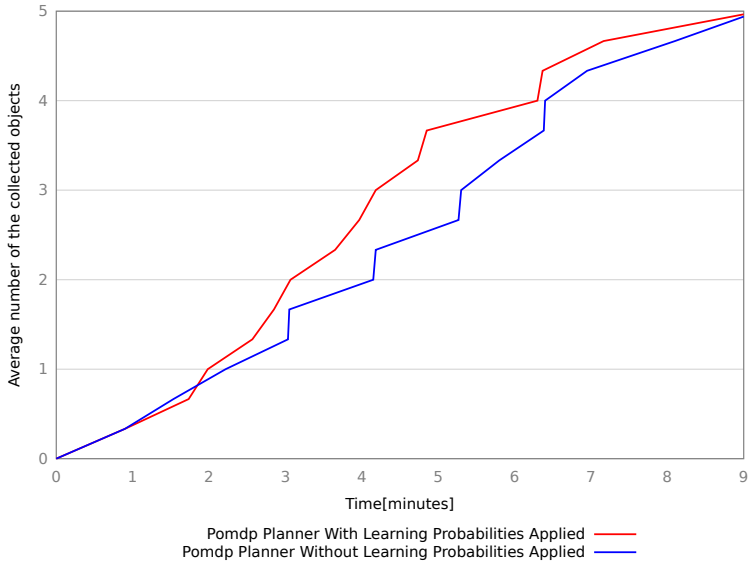

Figure 5.9: The average number of objects transported through time in the second scenario with failures in $room_1$.

alternative object to manipulate. This result can be seen clearly in the results of the first scenario during which no natural failure is encountered (Figure 5.6).

Figure 5.8 and 5.9 show the average number of objects transported over time with and without learning guidance in the first and second scenario, respectively. This result shows that when the probabilistic planner is fed with the learning results, the number of transported objects is higher, since the planner has the option of choosing the objects with higher success probabilities. However, after all these objects are transported, the planner also suggests transporting the remaining objects with lower success probabilities. When the time period given for the manipulation task is limited, the robot always prefers to transport the objects with higher success rates since it uses its experience. It should also be noted that between time steps 1-2 in Figure 5.9, although it seems that POMDP without guidance gives better results, this is due to a natural failure. As the number of objects increase, the advantage of using experience can be realized.

Table 5.1: Performance analysis of POMDP for increasing number of objects

#Objects	#States	#Actions	Time(s)
1	7	7	0.18
2	23	11	0.54
3	83	15	2.62
4	299	19	11.94
5	1055	23	62.77

Table 5.2: Performance analysis of POMDP for increasing number of locations

#Locations	#States	#Actions	Time(s)
1	50	11	0.58
2	299	19	11.94
3	1026	27	65.14

The performance of the planner against the increasing number of objects and the number of locations is also analyzed in Table 5.1 and Table 5.2, respectively. The columns present the number of states, the number of actions and the computation time. In Table 5.1, the performance is analyzed for different number of objects with two semantic locations. Table 5.2 presents the results for four objects as the number of locations changes. It can be seen that as the numbers of objects and locations increase, the numbers of generated states and actions increase as well. Thus, the complexity and computation time of the planner increase exponentially. This is one of the drawbacks of POMDPs. However, this method ensures that the probabilistic hypotheses are used to guide planning in a more realistic way which makes this method suitable for small-sized domains.

5.4 Comparison of a Deterministic Planner and a Probabilistic Planner

In the prior work of author's research group which is previously presented [3], a deterministic planner, namely TLPlan [21], is used on the same application domain. Here, the performance analysis of this previous work is given, and the comparison of it with the proposed method in this thesis is discussed.

Table 5.3: Performance analysis of TLPlan with increasing number of objects

#Objects	#States generated	#States searched	Plan search time(s)
1	11	6	< 1
2	75	29	< 1
3	1155	313	< 1
4	21963	4565	< 1
5	436819	74508	28.540

In Table 5.3, the number of states generated, the number of the states searched and the required time to reach the goal are presented for the run of TLPlan with increasing number of objects. The goal is set to successfully transport objects to their destinations. The locations of the objects are provided to the planner beforehand. In TLPlan, a

Table 5.4: Performance analysis of TLPlan after updating preconditions

#Objects	#States generated	#States searched	Plan search time(s)
1	7	4	< 1
2	42	17	< 1
3	494	140	< 1
4	2369	535	< 1
5	35376	6362	1.670

classical search tree structure is used on which a simple forward chaining search is applied. The search tree is constructed with respect to the definitions of actions in the world model provided to the planner [22]. Since the planner is not guided in Table 5.3, the increase in the number of states generated and in the number of states searched are more than the results of the learning-guided TLPlan which are presented in Table 5.4. Here, *the precondition update* method is used to reduce the search space of the planner by guiding it with the hypothesis coming from learning process. The same scenario is applied in which the bowling pins are taken away from the environment with a probability used in the experiments of the POMDP planner. Therefore, the hypothesis in Equation (5.1) is also used in the guidance of TLPlan. In this way, a new precondition of action $pickUp$ is created which represents that the robot fails in the execution of action $pickUp$ for the bowling pins. Since the probability of the hypothesis cannot be used in *the precondition update* method, the corresponding world states about the hypothesis are totally rejected in the search space of the planner. In this case, an alternative action is required to manipulate bowling pins in the environment. Otherwise, the planner fails to generate a plan that achieves the goal. Therefore a new action is defined, namely *push*, for pushing an object to the destination. Eventually, action *push* is selected instead of action $pickUp$ for the bowling pins in the generated plan.

As a comparison, using a probabilistic planner in such a problem has many advantages. First of all, since the described scenarios are run in a dynamic real-world environment, the uncertainty should be handled. Although uncertainty in observations are not taken into account in the planning method of this thesis, the proposed system can still adapt itself to the sudden changes by evaluating current observation in the selection of the next action. This can be achieved by running a policy instead of a sequential plan. In this way, the probabilistic planner can also generate a policy in the case that

the locations of the objects are not provided to the planner beforehand. However a deterministic planner fails when the locations of the objects are unknown.

The deterministic and probabilistic learning-guided planners can be compared by using the results given in Table 5.1 and Table 5.4 in terms of required time to generate a plan while the number of objects used in the experiments increases. Aside from the limitations of a deterministic planning method explained above, it has the advantage of generating a plan within a more reasonable time period than that of a probabilistic planning method for small size instances.

6. CONCLUSIONS AND RECOMMENDATIONS

Robots need to gain experience, and use this experience to improve their task execution performance to prevent undesired outcomes. In this work, an adaptive probabilistic planning method that uses the outputs of an experimental learning process by an autonomous robot is presented. The selected case study is on multi-object manipulation scenarios where the robot's objective is maximizing the cumulative manipulation performance. In the experiments, the robot collects the contexts of its previous failed action executions while executing its multi-object manipulation task. The built experience is used in an experimental learning process which generates probabilistic hypotheses mapping from execution contexts to success or failure outcomes. The relevant object attributes which are included in the contexts of probabilistic hypotheses are used to feed the probabilistic planner to reduce the number of potential failures in future plans.

A POMDP model is used as a probabilistic planner in this system. POMDP is modelled so as to accept the probabilities coming from learning hypotheses. Since the aim is maximizing the number of successfully manipulated objects over time, the robot should optimize the order of objects to manipulate. In this way, the manipulation performance of the robot can be improved within a time period. Different approaches can be proposed to determine the best order of objects to manipulate. Firstly, the locations of the objects are considered to minimize the distance to the robot for manipulating an object. Secondly, the selection of the objects on which the robot performs better also improves the performance of the robot in its multi-object manipulation task. By using the learning hypotheses which relates the relevant object attributes and action execution outcomes, the robot executes its actions on the objects with higher probabilities of action $pickUp$ success. The results show that the probabilistic guidance in planning achieves the best manipulation order of the objects within the knowledge of the robot to maximize the transportation success over time. However, since only success of action $pickUp$ is considered in the proposed method

to improve the performance of the robot, the evaluation of success rates for the other available actions in addition to action *pickUp* remains as future work.

In the proposed system, a point-based planning framework, named as SARSOP, is used which tries to find approximate solutions to the problems. Although the solution is found approximately, the computational complexity of the planning problem is still high as a result of the variety of the variables in model. As a future work, reducing the computational complexity of this planner can be targeted by abstracting state representations in such domains. In addition, SARSOP does not include symbolic representations. Since the used robot system builds its knowledge base symbolically, modelling of this system in SARSOP complicates the representation of variables and so increases the complexity. Therefore, a symbolic POMDP planner is intended to use in the future.

REFERENCES

[1] **Karapinar, S.**, **Altan, D. and Sariel-Talay, S.** (2012). A robust planning framework for cognitive robots, *Proceedings of the AAAI-12 Workshop on Cognitive Robotics (CogRob)*.

[2] **Karapinar, S. and Sariel, S.** (2015). Cognitive Robots Learning Failure Contexts Through Experimentation, *Proceedings of the 14th International Conference on Autonomous Agents & Multiagent Systems*.

[3] **Yildiz, P.**, **Karapinar, S. and Sariel-Talay, S.** (2013). Learning guided symbolic planning for cognitive robots, *The IEEE International Conference on Robotics and Automation (ICRA), Autonomous Learning Workshop*.

[4] **Kaelbling, L.P.**, **Littman, M.L. and Cassandra, A.R.** (1998). Planning and acting in partially observable stochastic domains, *Artificial intelligence*, ***101***(1), 99–134.

[5] **Smith, T.** (2007). Probabilistic Planning for Robotic Exploration, *Ph.D. thesis*, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

[6] **Smith, T. and Simmons, R.** (2012). Point-based POMDP algorithms: Improved analysis and implementation, *arXiv preprint arXiv:1207.1412*.

[7] **Kurniawati, H.**, **Hsu, D. and Lee, W.S.** (2008). SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces., *Robotics: Science and Systems*, volume2008, Zurich, Switzerland.

[8] **Smith, T. and Simmons, R.** (2004). Heuristic search value iteration for POMDPs, *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, AUAI Press, pp.520–527.

[9] **Roy, N.**, **Gordon, G. and Thrun, S.** (2006). Planning under uncertainty for reliable health care robotics, *Field and Service Robotics*, Springer, pp.417–426.

[10] **Du, Y.**, **Hsu, D.**, **Kurniawati, H.**, **Lee, W.**, **Ong, S. and Png, S.** (2010). A POMDP Approach to Robot Motion Planning under Uncertainty, *Int. Conf. on Automated Planning and Scheduling, Workshop on Solving Real-World POMDP Problems*.

[11] **Png, S.C.O.S.W. and Lee, D.H.W.S.** (2009). POMDPs for Robotic Tasks with Mixed Observability.

[12] **Zhang, S. and Sridharan, M.** (2012). Active visual sensing and collaboration on mobile robots using hierarchical POMDPs, *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent*

Systems-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, pp.181–188.

[13] **Searock, J.**, **Browning, B. and Veloso, M.** (2005). Learning To Prevent Failure State for a Dynamically Balancing Robot, *In AAAI*.

[14] **Monsó, P.**, **Alenyà, G. and Torras, C.** (2012). POMDP approach to robotized clothes separation, *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, pp.1324–1329.

[15] **Pajarinen, J. and Kyrki, V.** (2014). Robotic manipulation of multiple objects as a POMDP, *arXiv preprint arXiv:1402.0649*.

[16] **Pajarinen, J. and Kyrki, V.** (2014). Robotic manipulation in object composition space, *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, IEEE, pp.1–6.

[17] **Pasula, H.**, **Zettlemoyer, L.S. and Kaelbling, L.P.** (2004). Learning Probabilistic Relational Planning Rules., *ICAPS*, pp.73–82.

[18] **Lang, T. and Toussaint, M.** (2010). Planning with noisy probabilistic relational rules, *Journal of Artificial Intelligence Research*, ***39***(1), 1–49.

[19] **Martínez, D.**, **Alenya, G. and Torras, C.** (2015). Planning robot manipulation to clean planar surfaces, *Engineering Applications of Artificial Intelligence*, ***39***, 23–32.

[20] **Vrakas, D.**, **Tsoumakas, G.**, **Bassiliades, N. and Vlahavas, I.P.** (2003). Learning Rules for Adaptive Planning., *ICAPS*, pp.82–91.

[21] **Bacchus, F. and Ady, M.** (2001). Planning with resources and concurrency: A forward chaining approach, *IJCAI*, volume 1, pp.417–424.

[22] **Bacchus, F. and Kabanza, F.** (2000). Using temporal logics to express search control knowledge for planning, *Artificial Intelligence*, ***116***(1), 123–191.

[23] **Wang, X.** (1996). Learning planning operators by observation and practice, *Ph.D. thesis*, Carnegie Mellon University.

[24] **Ozturk, M.D.**, **Ersen, M.**, **Kapotoglu, M.**, **Koc, C.**, **Sariel-Talay, S. and Yalcin, H.** (2014). Scene interpretation for self-aware cognitive robots, *AAAI-14 Spring Symposium on Qualitative Representations for Robots*.

[25] **Hinterstoisser, S.**, **Cagniart, C.**, **Ilic, S.**, **Sturm, P.**, **Navab, N.**, **Fua, P. and Lepetit, V.** (2012). Gradient response maps for real-time detection of textureless objects, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, ***34***(5), 876–888.

[26] **Ersen, M.**, **Talay, S.S. and Yalcin, H.** (2013). Extracting Spatial Relations Among Objects for Failure Detection., *KIK@ KI*, pp.13–20.

[27] **Kapotoglu, M.**, **Koc, C.**, **Sariel, S. and Ince, G.** (2014). Action monitoring in cognitive robots, *Signal Processing and Communications Applications Conference (SIU), 2014 22nd*, IEEE, pp.2154–2157.

[28] **Kvarnström, J.**, **Heintz, F. and Doherty, P.** (2008). A Temporal Logic-Based Planning and Execution Monitoring System., *ICAPS*, pp.198–205.

[29] **Kabanza, F.**, **Barbeau, M. and St-Denis, R.** (1997). Planning control rules for reactive agents, *Artificial Intelligence*, ***95***(1), 67–113.

[30] **Muggleton, S.** (1995). Inverse entailment and Progol, *New generation computing*, ***13***(3-4), 245–286.

[31] **Quigley, M.**, **Conley, K.**, **Gerkey, B.**, **Faust, J.**, **Foote, T.**, **Leibs, J.**, **Wheeler, R. and Ng, A.Y.** (2009). ROS: an open-source Robot Operating System, *ICRA workshop on open source software*, volume 3, p. 5.

CURRICULUM VITAE

Name Surname: Melis Kapotoğlu

Place and Date of Birth: Istanbul - 19/03/1990

Address: Sirinevler Mah. Marasal Fevzi Cakmak Cad. 1. Sok. No:19 Daire: 6 Sirin Apt. B Blok Bahcelievler/Istanbul

E-Mail: kapotoglu@itu.edu.tr

B.Sc.: Computer Engineering - Istanbul Technical University

Professional Experience and Rewards: Eteration Bilisim - Software Engineer (2014 July - present)

List of Publications:

- **Kapotoglu, M.**, Koc, C., Sariel, S., and Ince, G. (2014, April). Action monitoring in cognitive robots, *Signal Processing and Communications Applications Conference (SIU)*, 2014 22nd, IEEE, (pp. 2154-2157).

- Ozturk, M. D., Ersen, M., **Kapotoglu, M.**, Koc, C., Sariel-Talay, S., and Yalcin, H. (2014, March). Scene interpretation for self-aware cognitive robots. *AAAI-14 Spring Symposium on Qualitative Representations for Robots*.

- Karapinar, S., Ersen, M., **Kapotoglu, M.**, Yildiz, P., Sariel-Talay, S., and Yalcin, H. (2013, April). Experimental learning in cognitive robots. *Signal Processing and Communications Applications Conference (SIU)*, 2013 21st (pp. 1-4). IEEE.

- Sariel, S., Yildiz, P., Karapinar, S., Altan, D., and **Kapotoglu, M.** (2015). Robust task Execution through Experience-based Guidance for Cognitive Robots. *International Conference on Advanced Robotics (ICAR)*, 2015 17th.

PUBLICATIONS/PRESENTATIONS ON THE THESIS

- **Kapotoglu, M.**, Koc, C., Sariel, S., and Ince, G. (2015). Failure Avoidance through Learning Guided Probabilistic Planning, *International Conference on Robotics and Automation (ICRA)*, 2015.

- **Kapotoglu, M.**, Koc, C., Sariel, S., and Ince, G. (2015). Robots Avoid Potential Failures through Experience-Based Probabilistic Planning, *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2015 12th.