ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

ARGUMENTATION-BASED SCENE INTERPRETATION USING DEFEASIBLE LOGIC PROGRAMMING

M.Sc. THESIS

Çağatay KOÇ

Department of Computer Engineering

Computer Engineering Programme

JUNE 2015

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

ARGUMENTATION-BASED SCENE INTERPRETATION USING DEFEASIBLE LOGIC PROGRAMMING

M.Sc. THESIS

Çağatay KOÇ
(504121512)

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Asst. Prof. Sanem SARIEL

JUNE 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

FESHEDİLEBİLİR MANTIKSAL PROGRAMLAMA İLE TARTIŞMA TABANLI SAHNE YORUMLAMA

YÜKSEK LİSANS TEZİ

Çağatay KOÇ
(504121512)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Asst. Prof. Sanem SARIEL

HAZİRAN 2015

Çağatay KOÇ, a M.Sc. student of ITU Graduate School of Science Engineering and Technology 504121512 successfully defended the thesis entitled **"ARGUMENTATION-BASED SCENE INTERPRETATION USING DEFEASIBLE LOGIC PROGRAMMING"**, which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

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

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

FOREWORD

First, I would like to thank my supervisor, Asst. Prof. Sanem Sarıel, for her guidance during this thesis. I would also like to thank Doğan Altan, Mustafa Ersen, Melis Kapotoğlu and Melodi Deniz Öztürk for their support during this thesis. This thesis is partly funded by a grant from the Scientific and Technological Research Council of Turkey (TUBITAK), Grant No. 111E286.

June 2015

Çağatay KOÇ
Computer Engineer

TABLE OF CONTENTS

Page

ABBREVIATIONS

CAD	**:** Computer-aided Design
DeLP	**:** Defeasible Logic Programming
FOL	**:** First Order Logic
FN	**:** False Negative
FP	**:** False Positive
LR	**:** Linear Regression
MSE	**:** Mean Squared Error
NLR	**:** Nonlinear Regression
P-DeLP	**:** Possibilistic Defeasible Logic Programming
ROS	**:** Robot Operating System
SLAM	**:** Simultaneous Localization And Mapping
TP	**:** True Positive
TN	**:** True Negative
UAV	**:** Unmanned Autonomous Vehicle

LIST OF TABLES

Page

LIST OF FIGURES

Page

ARGUMENTATION-BASED SCENE INTERPRETATION USING DEFEASIBLE LOGIC PROGRAMMING

SUMMARY

In an agent system that needs to operate in the real world, the problem of maintaining a consistent world model in the face of unreliable, incomplete and inconsistent sensory data should be solved. In this thesis, an approach is presented that addresses the problem of constructing a consistent world model through inconsistent sensor readings by applying an argumentation-based scene interpretation framework in order to accurately model and represent the observations and beliefs of an agent. The proposed approach is based on temporal and probabilistic defeasible logic programming for reasoning.

When a robot is tasked with an object manipulation action, it initially needs to sense and localize the object though its sensory data. In the proposed system, an RGB-D camera is used as a sensor in order to sense the objects in the scene. The RGB-D camera on the robot provides point cloud data that consists of the distance between points and the camera and the color of each point. The point cloud data is evaluated by using LINE-MOD, LINE-MOD HS and Euclidean Clustering-based object segmentation algorithms. LINE-MOD is used to meet the requirements of online 3D object recognition. Different from most of the object recognition algorithms that run on images, LINE-MOD has an ability to recognize both textured and textureless objects in unstructured environments. Surface normals from depth data are used in order to recognize objects by the LINE-MOD algorithm. LINE-MOD HS is an extended version of the original LINE-MOD algorithm, and it also uses hue and saturation data during the recognizing process. Euclidean Clustering-based object segmentation algorithm segments the objects in the environment and can sense novel objects or objects that other algorithms fail to recognize. It can calculate the width, height and depth of objects using the segmented point clouds. However, these algorithms are subject to failures due to dynamic structure of the environment, noisy data, the viewing angle of the camera to an object or simply due to its incapability on same cases. As a result, the real world may not always match the perception of the robot due to such failures.

The robot that operates in the real world needs to filter these noisy inputs for reaching a consistent decision about the world state. The main problem addressed in this thesis is the problem of dealing with inconsistent observations gathered by the vision algorithms and interpreting them to provide a consistent world model. For solving the stated problem, a scene interpretation system is proposed that uses Defeasible Logic Programming (DeLP). Defeasible Logic is a non-monotonic logic that is capable of presenting logical statements in defeasible form which allows it to infer on contradictory information. In the proposed system, DeLP is extended to have defeasible facts, defeasible rules and observation facts. Background/commonsense knowledge is represented as defeasible facts and defeasible rules, whereas perception

information is represented as observation facts. In the proposed system, the procedure that maps the perception information to the observation facts is also defined.

DeLP based reasoner makes conclusions on any claim by constructing its acceptable argumentation lines. If an argument that supports a given claim cannot be defeated in any acceptable argumentation line, the claim is said to be warranted. Furthermore, it is also possible to query for a predicate to find its warranted claims. In order to construct a consistent world model, all claim of object properties are queried to be warranted. If they can be warranted, these claims are added to the world model. The performance of the proposed approach is evaluated on simulation experiments in the Stage Robot Simulator. It is also shown that the approach is applicable to real world scenarios with an autonomous Pioneer 3-AT robot.

FESHEDİLEBİLİR MANTIKSAL PROGRAMLAMA İLE TARTIŞMA TABANLI SAHNE YORUMLAMA

ÖZET

Robotların gerçek dünyada çalışabilmeleri için bulundukları ortamın farkında olmaları gerekmektedir. Robota bir nesne ile etkileşime geçme görevi verildiğinde, o nesneyi doğru olarak konumlandırabilmesi ve nesnenin nitelikleri hakkında bilgi sahibi olması; nesneyle etkileşime geçebilmesi için ilk adımdır. Robotların bulundukları ortam hakkında bilgi sahibi olmadan bu ortamda karar alıp eylem yürütmeleri mümkün olamaz. Ortam hakkında bilgi sahibi olabilmek için robotların öncelikle ortam için uygun algılayıcılara sahip olmaları gerekir. Bu algılayıcılardan gelen ham verilerin robot tarafından işlenip, anlamlandırılması ve bir bilgi temsili ile saklanması gerekmektedir. Fakat yaygın olarak, algılayıcıdan gelen ham verilerde ya da bu ham verilerin işlenmesi sırasında bazı hatalar oluşabilmektedir. Örneğin, robot bazı nesneleri algılamakta başarısız olabilir ya da bazı nesneleri farklı nesnelere benzetebilir ve bunun sonucunda oluşturulan veriler gürültü ve hatalar içerebilir. Bu sebeplerle robotun algılama sistemi her zaman gerçek dünya ile eşleşmeyebilir. Birbiri ardına alınan algılamalarda ya da farklı algılama yöntemleri ile gelen verilerde de farklılıklar olabilmektedir. Sonuç olarak robotun gelen bu verileri uygun şekilde filtreleyerek gürültülerden arındırması ve dünya durumu üzerine tutarlı bir çıkarım yapabilmesi gerekmektedir. Bu tezde, farklı kaynaklardan gelen verilerin işlenmesi sonucu oluşan bilgilerin Feshedilebilir Mantıksal Programlama ile işlenmesi ve robotun tutarlı bir dünya bilgi temsilinin oluşturulması ve sürdürülmesi hedeflenmiştir. Bu amaçla Feshedilebilir Mantıksal Programlama yöntemi yeni tanımlamalarla geliştirilerek kullanılmıştır. Oluşturulan sistem simulasyon ortamında ve gerçek dünyada test edilmiştir. Sistemin, bir ve birden fazla robot için kullanılabilirliği farklı senaryolar yardımıyla gösterilmiştir. Sahne yorumlama için Feshedilebilir Mantık Programlama kullanılması, bu yöntemin uygun şekilde geliştirilmesi ve bu sistemin gerçek ortamda uygulanması bu tezin katkılarını oluşturmaktadır.

Ortam hakkında bilgi sahibi olabilmek için; robotlar bir çok algılayıcı bulundurmaktadır. Bu algılayıcılar; robotun ortam içerisindeki konumunu bilmesi ve ortamın durumunu anlayıp çıkarsama yapabilmesi için kullanılmaktadırlar. Gerçek dünyada çalışan bir robotun ortamını algılayabilmesi için, algılayıcılarından gelen ham verileri işlemesi gerekmektedir. Bu ham verileri işlemeden bunları kullanmak mümkün olmamaktadır. Bu verilerin işlenmesi için varolan görüntü işleme algoritmalarından faydalanılmıştır. Robotun ortamda bulunan nesneleri algılayabilmesi için RGB-D kamera kullanılmaktadır. Bu kamera ortamı iki boyutlu algılamanın yanında, algıladığı her bir noktanın uzaklığını da çıkarsayabilmektedir. Noktaların sahip olduğu renk ve derinlik bilgisi bütünü nokta bulutu olarak adlandırılmaktadır. Bu nokta bulutu kullanılarak nesnelerin ortamda seçilmesi ve tanınması işlemleri yapılmaktadır. Nesnelerin ortamda tanınabilmeleri için LINE-MOD ve LINE-MOD HS histogram algoritmaları kullanılmaktadır. Bu algoritmalar RGB-D kamera ile elde edilen nokta

bulutlarını kullanarak, nesnelerin yüzey normallerini hesaplamaktadırlar. Robot, daha önceden karşılaştığı nesnelerin yüzey normalleri bilgisini saklamaktadır. Kullanılan algoritma ile nokta bulutundan çıkarılan yüzey normalleri, robotun önceden sakladığı nesnelerin yüzey normalleri bilgileri ile karşılaştırılır ve bulunan nesnelerin gözlem bilgisi oluşturulur. LINE-MOD HS histogram algoritması, yüzey normallerini karşılaştırmasının yanı sıra, renk bilgisini de kullanmaktadır. Bu sayede farklı renklerdeki nesneleri algılayıp tanımlayabilmektedir. Tanınmayan nesnelerin de ortamda algılanabilmesi için bölütleme algoritması kullanılmaktadır. Bu algoritma yardımıyla, RGB-D kamerasından elde edilen nokta bulutu içerisinde nesnelere ait olabilecek nokta kümeleri tespit edilmektedir. Bu nesne bilgilerini içeren nokta kümelerinden, yaklaşık olarak nesne konumu ve nesne büyüklüğü algılanabilmektedir. Tespit edilen bu nesne kümeleri bilgisi algoritmanın çıktısını oluşturmaktadır.

Nesne algılama ve nesne tanıma algoritmaları yardımıyla oluşturulan gözlemler sahne yorumlama modülünün girdisini oluşturmaktadır. Bu gözlemler sürekli olarak alınmakta ve saklanmaktadır. Alınan gözlemler; gözlem nitelik bilgilerini, gözlem zamanını, gözlem güvenilirliğini ve gözlem kaynağını içeren bir veri grubunu oluşturmaktadır. Bu veri grupları tez kapsamında önerilen önişleme yöntemi kullanılarak kümelenir ve bu kümelerdeki bilgilerin bütünününden faydalanılarak gözlem durumları oluşturulur. Bir gözlem durumu; gözlem hakkındaki niteliksel bilgiyi, son gözlem zamanını ve gözlemin doğruluğu bilgisini içerir. Bu bilgiler çıkarsanırken, gözlem durumunu oluşturan gözlem veri grubu kümelerinden faydalanılır. Veri grubundaki son alınan gözlemin zamanı, gözlem durumu zamanı olarak atanırken; gözlem durumu doğruluğu bilgisi veri grubu kümesindeki gözlem sonuçlarından faydalanılarak hesaplanır. Gözlem durumları Feshedilebilir Mantıksal Programlama ile sahne yorumlama yapılması aşamasında kullanılmaktadır. Gözlemlerin yanı sıra, robotun ortam hakkında sahip olduğu bilgiler de Feshedilebilir Mantıksal Programlama dili ile ifade edilip, robota verilir. Böylece robot, algılayıcılarından gelen bilgilerin yanısıra, ortam hakkında da arkaplan bilgisine kullanır. *"labutlar uzun nesnelerdir"* gibi ön bilgiler robota verildiğinde, robot ortam hakkındaki bu bilgiyi kullanarak daha tutarlı çıkarsamalar yapar.

Feshedilebilir Mantıksal Programlama, tartışma tabanlı bir monoton olmayan mantıksal dildir. Robotun algılayıcılarından gelen veriler ve bu veriler üzerinde çalışan algoritmalardan gelen bilgiler hatalı olabilmektedir. Bu hatalar, oluşan gözlemlere de yansır ve robotun bilgi tabanında tutarsızlıklar oluşturabilmektedir. Bir çok mantıksal dil tutarsızlıkları yorumlayamaz iken, Feshedilebilir Mantıksal Programlama dilinde tutarsızlıklara izin verilebilmektedir. Bir çıkarım yapılırken, bu tutarsız mantıksal ifadeler ağırlıkları ve zaman bilgileri kullanılarak karşılaştırılmakta ve tutarlı bir çıkarıma ulaşılması sağlanmaktadır. Böylece robotun ulaştığı dünya bilgi tabanı, tutarsız mantıksal ifadeler içerdiği halde; tutarlı çıkarımlarla en uygun dünya durumu oluşturulabilmektedir. Ayrıca diğer mantıksal dillerde tutarsız ifadeler önemsenmezken, bu dilde bunlar da kullanılabilmekte ve fayda sağlayabilmektedir. Feshedilebilir Mantıksal Programlama, sahne yorumlama sırasında karşılaşılabilen problemlere çözüm getirmekte ve gerçek zamanlı ortamın bilgi temsilinin işlenmesine olanak sağlamaktadır. Önerilen yöntem ROS altyapısı kullanılarak bilgisayar ortamında gerçeklenmiştir. Gerçeklemeler simulasyon ve gerçek dünyada robot üzerinde uygulanmıştır. Bu uygulamalardan alınan sonuçlar raporlanmış ve diğer çalışmalarla karşılaştırılıp başarılı yönleriyle ortaya konulmuştur.

Robotların gerçek dünyada işlevsel olabilmeleri için sahne yorumlama kabiliyetine sahip olmaları gerekmektedir. Bu tezde, bir sahne yorumlama sistemi Feshedilebilir

Mantıksal Programlama kullanılarak modellenip geliştirilmiştir. Bu sahne yorumlama sistemi, farklı kaynaklardan aldığı bilgileri değerlendirebilmekte ve bunları arkaplan bilgisini de kullanarak işlemektedir. Tutarsız bilgilere de değerlendirebilen bu sistem, ortamın o anki tutarlı olarak sahip olabileceği dünya durumunu çıkarsar. Yapılan deneyler, bu sistemin etkin bir şekilde çalıştığını ve gezgin robotlarda bu sistemin kullanılabilirliğini göstermektedir. Geliştirilen sistem, birden fazla robotun bilgi paylaşımı yapmasını ve tartışarak tutarlı bir sonuca ulaşmasını da sağlayabilecek bir yapıda gerçeklenmiştir.

1. INTRODUCTION

A robot that needs to operate and manipulate objects in a human environment encounters some challenges due to characteristics of human environments [1]. In the human environments, objects and environments are usually human-oriented and consonant with human anatomy, and the robots may came across with people, pets or even other robots. Furthermore presence of the outer factors, makes the environment dynamic that usually results in incomplete and/or inconsistent knowledge elements in the robot's model. Object types, appearances and their placements may also differ in each case. Due to being in a dynamic environment, the robot should also satisfy real-time constraints, and be capable of responding to the potential changes in the environment. The final and also one of the crucial challenges that needs to be overcome is incomplete and noisy sensor data which makes perception less reliable and prune to faults.

A scene interpretation system is needed to extract and maintain consistency in knowledge about the world state to eliminate unreliable sensory information that may mislead the operations of the robot. The robot needs to face with inconsistent observations in such cases. Domain specific properties such as the reliability of the information sources or general domain rules such as *all cups are white* or *boxes are prismatic objects* may be predetermined for improving interpretation results. However, relying only on hand coded information generally fails in coping with unexpected events or failures that may occur at runtime.

Knowledge representation is an important aspect for robots. There are several knowledge representation methods for expressing both observations and commonsense knowledge. As the depth of a robot's knowledge increases, expressing its all elements by existing knowledge representation methods becomes harder. First Order Logic (FOL) is one of the widely used techniques for representing knowledge. However, this technique assumes that the environment is fully observable and deterministic, while in real-world domains, the environment is generally stochastic and partially observable. Probabilistic techniques are widely used for facing with uncertainties and provide

filtering out noise in sensor data. However, these methods do not scale well for the scenarios which need further symbolic reasoning.

1.1 Problem Definition

When a robot is tasked to interact with objects, the first step is correctly localizing the objects and interpreting the relations among them. However, it is common to encounter various errors during object detection and recognition. For example, the robot's vision system may fail in recognizing an object or classifying it correctly. An example vision output is given in Figure 1.1 where recognition errors can be seen. The robot may also encounter novel objects, and it may have no prior information on the objects. Vision algorithms would fail due to the need for templates or CAD (Computer-aided Design) models of an object prior to recognizing it. As a result, the perception of the robot does not always match the real world. When the robot has background knowledge of the environment, for example, in the figure, it can state that there is a door, and there should not be any object on that region, or it may also reason that all objects must have contact to a floor or other surfaces. It is beneficial for the robot to use the background knowledge, which may help the robot to filter noisy data that may originate from the perception algorithms. Two different sensor readings may also be in conflict, due to different factors that may affect their perception. The robot needs to filter all these noises to end up with a consistent decision about the world state. Thus, interpreting the scene accurately is beneficial for efficiency of task execution.

Figure 1.1: A sample object recognition output from the LINE-MOD algorithm.

In this thesis, manipulation scenarios with a mobile robot are focused to show the applicability of a scene interpretation system. It is assumed that there is no apriori information on the objects ($o_i \in O$, where $i = 0, ..., m$) the robot interacts with and their properties. The mobile robot may detect these objects or perceive their properties during its navigation in the environment. As the case study of this thesis, every object has three main properties as *type*, *color* and *size*. Each object has an associated belief on the location information (loc_m). The robot can localize the objects by using its vision algorithms, however, it has a limited range of view. So in order to fully sense the environment, additional sensing actions may be needed which requires additional resources such as time and energy. The robot uses several different perception algorithms in order to gather knowledge from the environment and they may have different capabilities. For instance, some algorithms may just perceive the color of an object, while some other algorithms are capable of classifying the type of an object. The main problem that is addressed in this thesis is the problem of dealing with inconsistent observations related to objects and interpreting them to provide an accurate scene model. Using the knowledge base that is produced by the scene interpretation module, the system should be able to produce the object related predicates. The following predicates are constructed using the gathered knowledge of the environment as seen in the Table 1.1. *object(loc_m)* predicate defines that

Table 1.1: The predicates which are maintained in the knowledge base.

Predicate	Description
object(loc_m)	Object in location loc_m exists in the scene, and it is detected/recognized.
type(loc_m, <type>)	Object located in loc_m belongs to type <type>.
color(loc_m, <color>)	<color> is detected as an object's color located in loc_m.
size(loc_m, <size>)	Object located in loc_m has size <size>.

there exists an object in the scene on the given location loc_m. *type(loc_m, <type>)*, *color(loc_m, <color>)* and *size(loc_m, <size>)* predicates determine the type, color and size properties of an object, respectively. The main objective is to maximise

the correctly classified predicates while minimizing the misclassified and duplicate predicates.

1.2 Purpose of Thesis

In this thesis, the main goal is to implement a scene interpretation system that is able to maintain a consistent world model for autonomous mobile robots. For a robotic system, partial and/or inconsistent observations gathered through unreliable sensors should be filtered and pre-processed in order to reach a consistent knowledge base. Otherwise, the robot fails to interact with objects due to the lack of the knowledge about objects. Therefore, an inference mechanism should be applied in order to evaluate observations by taking into account temporal information and success characteristics. The success criteria for such a mechanism is the number of correctly classified instances for each object predicate.

1.3 Hypothesis

An argumentation-based scene interpretation system is proposed to solve the stated problems by using a Defeasible Logic Programming (DeLP) formulation [2]. DeLP is a logic programming formalism that is able to deal with contradictory information especially for multi-agent systems. The main motivation behind our study is the investigation of this formulation in a single body system which has conflicting sensor readings. In the proposed formulation, partial and/or inconsistent observations gathered through unreliable sensors are represented as observation facts, which are customised versions of defeasible facts. Defeasible facts and defeasible rules are defined to represent background/commonsense knowledge. Domain beliefs(defeasible facts and defeasible rules) and observation facts form the knowledge that is queried in order to maintain a consistent world model. DeLP is extended to have both probabilistic and temporal features. The applicability of the proposed scene interpretation approach is analyzed in both simulated and real-world environments.

2. BACKGROUND

In this chapter, Robot Operating System (ROS) framework and the perception sources used in the proposed system are briefly described. Afterwards, the literature on Defeasible Logic Programming (DeLP) and its applications and Scene Interpretation in robotic systems are summarised.

2.1 ROS Framework

While developing large-scale applications on robots, developers encounter challenges like dealing with the hardware of the robot or implementing supplementary modules beside main parts of the application. For example, implementing navigation of a robot is individually time-consuming but a basis requirement for a large-scale system. Thus, exploiting a library becomes necessity in these situations. ROS is an open-source framework that provides lots of libraries and tools about robot applications. Using ROS, developing software on a robot can be done without considering low level issues like hardware, device drivers, communication or visualization since ROS manages these kinds of details and helps to be able to focus on the main components of the desired system. [3]

In a ROS-based system, all processes that run simultaneously and communicate with each other are defined as nodes. The communication between nodes is handled by ROS, so the developer only needs to know the message interface between nodes. The context of the message and the type of the communication is determined according to the requirements of the system, and the desired messages are incorporated into the system. There are two different approaches, topic and service-based, to handle communications. In the topic based approach, the framework creates a channel that any message can be sent from a node, and any node can listen to that channel. In the server-based approach, the node opens a server with a pre-defined request format, and any node can send a request to inquiry the server node. Communication methods are determined with respect to the structure of the communication between nodes, and more than one message format can be sent between the same pair of nodes.

2.1.1 Localization, mapping and navigation

A mobile robot should estimate its location in the environment at each time step, which is called the localization process. The most important usage of localization is for searching a path to a specific coordinate on the map. An odometer and a laser range finder of the robot are used for determining the location. Change in location is estimated over time by evaluating the data that comes from various sensors. In addition, the laser range finder detects dynamic obstacles around the robot, and the obstacles are avoided during navigation. A probabilistic localization system which is named as Monte Carlo Localization [4] is used. In this approach, the pose of the robot is tracked with particle filtering method.[1]

Robots can also generate their map from scratch or continuously update and improve a predefined map. Grid mapping algorithms are used in order to create a map of an environment. In this thesis, Rao-Blackwellized particle filters are used as an effective method to solve the simultaneous localization and mapping (SLAM) problem in an unknown environment [5]. The robot is manually controlled to explore the environment. The final map that is constructed by the SLAM algorithm is recorded for further usage. The robot uses the recorded map as a static map, and updates its location on the map according to incoming data from the laser range finder. In Figure 2.1, the map of the department's corridor which is generated by the robot can be seen.

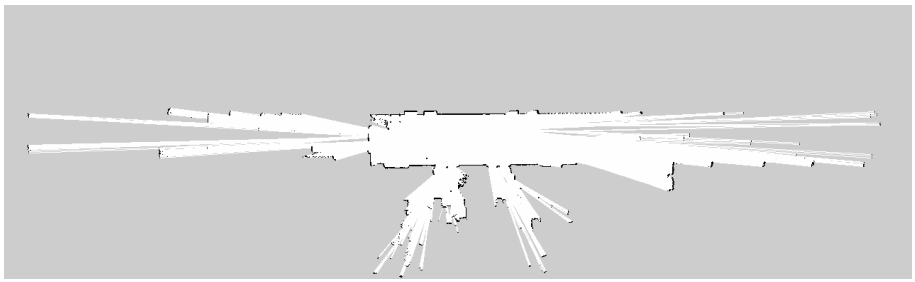


Figure 2.1: An example map of the corridor of Faculty of Computer and Informatics.

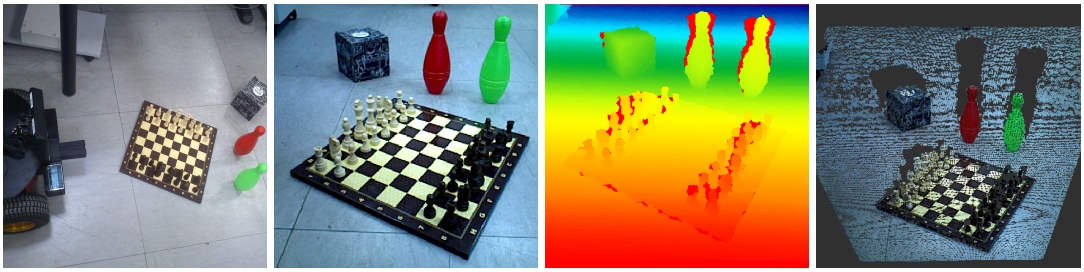
2.2 Perception Sources

A robot may have multiple data sources to maintain its own state and world state knowledge. An RGB-D Camera is used in order to gather data from the environment. Object detection and recognition algorithms are used to recognize objects in the environment.

[1] This algorithm is available as a package in ROS and this package is exploited with proper parameters in our system.

2.2.1 RGB-D camera

In an object manipulation scenario, a robot must be able to sense its environment and recognize the objects that are cluttered in the environment. In order to sense the environment, an Asus Xtion Pro RGB-D camera is used that is placed on top of the robot facing forward. In addition to its RGB sensor which makes it able to sense in 2D environment with color information, it also has a depth sensor that can tag each point with its depth information. In Figure 2.2, samples for a raw image, a depth image and a point cloud that is generated by using the image and depth data can be seen.



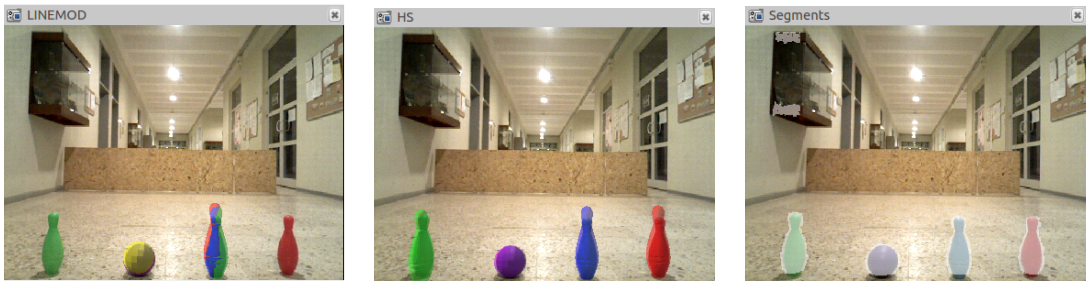
(a) (b) (c) (d)

Figure 2.2: (a) The environment during the sampling process, (b) the raw image, (c) the depth image and (d) the RGB-D point cloud data are shown in the sub-figures.

2.2.2 3D object recognition

Object manipulation scenarios are investigated in this work. For this reason, an efficient object recognition algorithm is needed to recognize and localize objects in the environment runtime. In the proposed system, LINE-MOD algorithm [6] is used for meeting the requirements of onboard 3D object recognition due to its computational efficiency and ability to recognize textureless objects as well as textured ones in unstructured environments. LINE-MOD uses surface normals from depth data in addition to 2D color information. An offline template registering is needed before online object recognition. This algorithm is further extented to incorporate with HS histograms to improve color recognition performance [7]. Although the true positive rate of the latter version is lower, it greatly reduces the false positives [8]. The algorithm starts with loading template information of the objects and tries to find a matching of templates and current scene. Furthermore, this algorithm was extended by taking into account the HS histograms of the objects. By doing so, the algorithm evaluates colour information beside depth information and the objects are recognized

with an extra colour information. The lighting condition of the environment can significantly affect the success ratio of the algorithm due to the variations in the color information, which is a disadvantage of this extension. An example output of LINE-MOD and LINE-MOD with HS on four objects can be seen in Figure 2.3(a) and Figure 2.3(b), respectively. These sources are denoted by s_l and s_{lhs} for LINE-MOD, LINE-MOD with HS sources, respectively.



(a) (b) (c)

Figure 2.3: Outputs of different vision algorithms from the robot's view: (a) LINE-MOD algorithm (b) LINE-MOD and HS Histograms (c) Segmentation algorithm

2.2.3 Object segmentation

The output coming from Euclidian Clustering-based object segmentation algorithm [9] is used as another perception source (s_c) running on the point cloud extracted from the depth camera. This source is very reliable to segment objects in the environment especially for detecting novel objects or objects that can not be recognized by the recognition algorithm. The width, height and depth data of segmented point clouds can be extracted. A visual output of segmentation can be found in Figure 2.3(c).

2.3 Scene Interpretation

In real world experiments, to successfully perform tasks, robots need to know their environment. However, detecting objects is hard and percepts do not persist over time. In the previous work by the author [10], a scene interpretation system is proposed that is able to create a consistent model of the world and extract spatial relations in the scene by fusing a set of perception sources. In this work, depth-based segmentation, LINE-MOD and HS histograms are used to recognize objects or identifying unknown objects. The spatial relation information for objects are also calculated and maintained to represent world knowledge to have a more detailed information about objects.

Similarly, an another semantic world modeling system namely WIRE that uses probabilistic multiple hypothesis anchoring is proposed to maintain a consistent state estimate by [11]. The system tracks multiple hypotheses and has the ability to incorporate prior knowledge. The schematic overview of the PMHA algorithm can be seen in Figure 2.4. It solves the uncertainty problem by using multiple hypotheses and

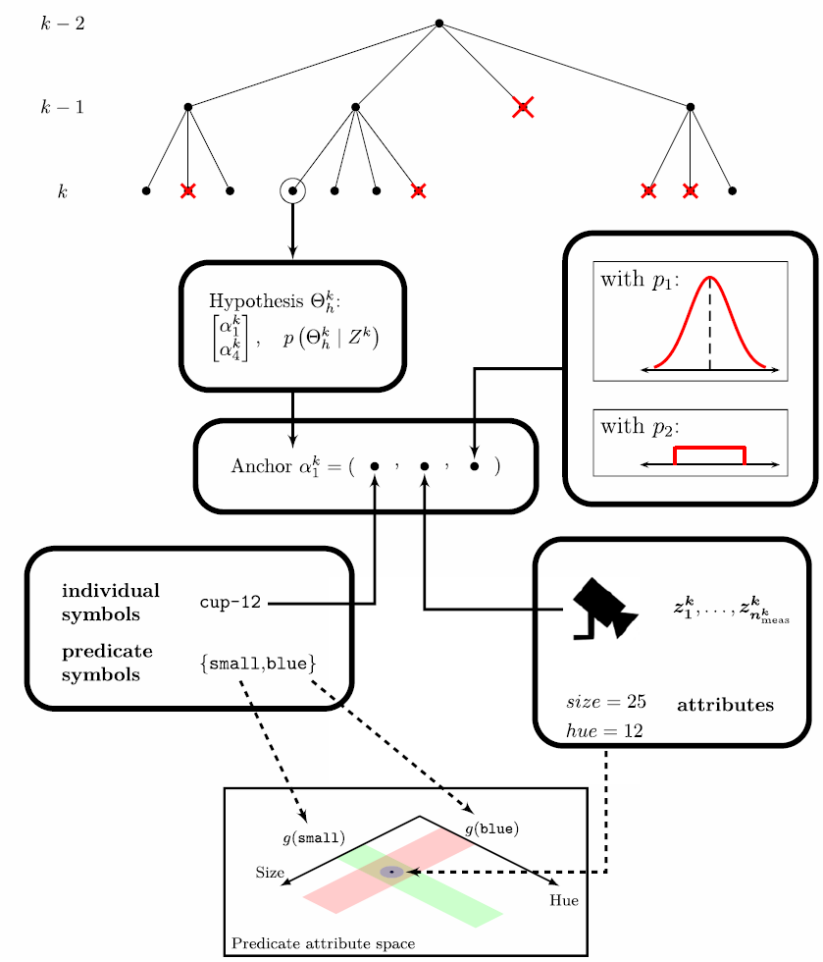


Figure 2.4: Schematic overview of the PMHA algorithm that is used in WIRE [11].

facilitates tracking of various object attributes. It stores object properties, maintains object identities over time and predicts future states. For each object, it maintains a set of attributes which are represented in a probabilistic way. Attributes can be updated based on new observations. Each observation can represent an existing object, a newly observed object or a clutter. Nodes are used as an anchors in a corresponding world state that represent real world object properties. Parent nodes expand with each new observation, and multiple hypotheses created as child nodes of the parents. That process forms a hypothesis tree, and each child node may have different beliefs about the world state. Growth of the tree is managed by filtering the less probable

hypotheses and unrelated nodes. Due to its probabilistic features and usage of multiple hypotheses, it is more robust than other techniques because it is less affected from noise and inconsistency through observations. However, the ability to track multiple hypotheses is also a disadvantage for the system, because it can slow down the entire inference process due to the hypothesis overload based on high quantity of objects. Despite having the ability to incorporate prior knowledge, the prior knowledge is only limited to mathematical models such as object movement models.

In the proposed system, it is possible to use logic-based prior knowledge and apply inference using logical arguments. The formalism is based on DeLP and able to deal with contradictory information especially for multi-agent systems and forms the basis as a knowledge base for argumentation in multi-agent domains.

3. DEFEASIBLE LOGIC

Defeasible logic is a non-monotonic logic initially introduced by Nute [12] in order to formalize defeasible reasoning. A non-monotonic logic is a logic in which consequence relations among logical statements are not monotonic. Different from a monotonic logic, it can reason on conflicting logical statements by using defeat relations among defeasible logical statements. Defeasible Logic consists of five components, namely strict facts, strict rules, defeasible rules, undercutting defeaters and acyclic binary relations among defeasible rules.

- *A strict fact* is represented with α which is an atomic formula. The complement of any formula α is denoted by $\neg\alpha$. For instance, $bird(Tweety)$ is a strict fact.

- *Strict rules* ($\Leftarrow$) are denoted by the following implication: $(\alpha_0 \Leftarrow \alpha_1 \wedge ... \wedge \alpha_n)$ which states that if $\alpha_1 \wedge ... \wedge \alpha_n$ is true, α_0 should also be true. We say that α_0 is the claim of the rule. $bird(X) \Leftarrow chicken(X)$ is an example of a strict rule which states that *a chicken is a bird*. Strict rules can not be defeated.

- *Defeasible rules* ($\leftarrow$) are represented with $(\alpha_0 \leftarrow \alpha_1 \wedge ... \wedge \alpha_n)$ in the same form as strict rules but they have weaker connections. They can be defeated by other defeasible rules. Two rules with contradictory claims (α and $\neg\alpha$) conflict with each other and might defeat each other. Examples of the defeasible rules are $fly(X) \leftarrow bird(X)$ and $\neg fly(X) \leftarrow chicken(X)$ which respectively states that *birds may fly* and *chickens may not fly*.

- *Acyclic binary relations* are defined in order to decide on defeat relations among defeasible rules. They define a priority among the non-strict rules. Following the previous examples, we know that *a chicken is a bird*, *birds may fly* and *chickens may not fly*, and we may both claim that *a chicken is a bird so it may fly* or *a chicken may not fly*. We need a defeat relation among these rules, in order to reason on them. When we give a higher priority to the defeasible rule *a chicken may not fly*, we can conclude that *a chicken may not fly* even in the presence of contradictory defeasible rules. Priorities can be defined in respect to the user needs.

- *Undercutting defeaters* are weaker instantiations of defeasible rules. They are not used as supporting rules in inference due to their weaker claims. Their goal is to prevent making decisions we might otherwise be inclined to.

3.1 Defeasible Logic Programming

DeLP is initially introduced to combine the results of Logic Programming and Defeasible Argumentation [2]. It follows the same characteristics of defeasible logic. The language of DeLP also consists of facts, strict rules, defeasible rules and acyclic binary relations. DeLP allows us to construct a defeasible argumentation inference mechanism in order to query claims as YES, NO, $UNDECIDED$ or $UNKNOWN$. YES and NO respectively state that a claim or its complement is warranted. $UNDECIDED$ denotes neither the claim nor its complement are warranted. The query to the claim returns $UNKNOWN$ if the claim is not in the language.

A Defeasible Logic Program is an infinite set of facts, strict rules and defeasible rules that are used in order to derive a conclusion for queries. In DeLP, defeasible argumentation is applied during the inference process. An argument A_i among a set of arguments ($A_i \in \psi$) relates a consistent support set of facts, strict rules and defeasible rules to a claim, $claim(A_i)$. The support set is always selected as the minimal consistent set that can derive the claim.

Consider the following DeLP example in Table 3.1 that consists of defeasible rules and facts, taken from [2]. Here, predicate $\neg\alpha$ represents the negation of predicate α.

Table 3.1: The DeLP example that consists of defeasible rules and facts.

Defeasible rules and facts			
$a \leftarrow b$	$\neg b \leftarrow e$	$\neg b \leftarrow c \wedge f$	$\neg f \leftarrow i$
$b \leftarrow c$	e	$f \leftarrow g$	i
c	$\neg f \leftarrow g \wedge h$	g	$\neg h \leftarrow k$
$\neg b \leftarrow c \wedge d$	$h \leftarrow j$	k	
d	j		

Argument A_i is a sub-argument of argument A_j if A_j contains all of the supports of A_i. Two arguments, A_i and A_j, conflict with each other if $claim(A_i) = \alpha$ and $claim(A_j) = \neg\alpha$. Argument A_i attacks to argument A_j when A_i is in conflict with a sub-argument of A_j. Using the previous DeLP example, the arguments given in Table 3.2 can be derived.

For instance, argument A_2 attacks to argument A_1 due to the fact that it claims $\neg b$ and sub-argument of A_1 claims b. Note that, only the argument(A_1) that claims a and the arguments that may attack any other argument are listed in the table. Argument A_i is a blocking defeater for argument A_j if argument A_i attacks argument A_j and argument A_i is better than A_j in terms of the defined comparison criteria. If the compared arguments are equal in terms of the comparison criteria, the arguments are blocking defeater for each other.

Table 3.2: The arguments that can be derived from Table 3.1.

A_i	$claim(A_i)$	Support Set of A_i
A_1	a	$(a \leftarrow b), (b \leftarrow c), c$
A_2	$\neg b$	$(\neg b \leftarrow c \wedge d), c, d$
A_3	$\neg b$	$(\neg b \leftarrow c \wedge f), c, (f \leftarrow g), g$
A_4	$\neg b$	$(\neg b \leftarrow e), e$
A_5	$\neg f$	$(\neg f \leftarrow g \wedge h), (h \leftarrow j), g, j$
A_6	$\neg f$	$(\neg f \leftarrow i), i$
A_7	$\neg h$	$(\neg h \leftarrow k), k$

An argumentation line (Λ) is an ordered list of arguments $(A_0, A_1 ... A_n)$ in which each consequent argument is in conflict with the sub-arguments of the previous one. By using these definitions, an acceptable argumentation line exhibits the following properties [2]:

1. Λ is a finite list of arguments.

2. The set of arguments with even indices ($A_0 \cup A_2 \cup A_4 \cup ...$) and the set of arguments with odd indices ($A_1 \cup A_3 \cup A_5 \cup ...$) do not contain any conflicting arguments among them.

3. No argument in Λ is a sub-argument of its previous arguments.

4. If Λ contains a blocking defeater, the next argument must be a proper defeater.

In Defeasible Logic Programming, conclusions are reached on any claim by constructing its acceptable argumentation lines. If an argument that supports the given claim cannot be defeated in any acceptable argumentation line, the claim is said to be warranted.

3.1.1 Warranting through dialectical analysis

For an argument, multiple acceptable argumentation lines may be available, and it would lead to a tree structure which is called as a dialectical tree. In a dialectical tree, each path from the root to a leaf represents a unique acceptable argumentation line. It provides a structure to inquiry each possible argumentation line. To decide whether an argument is defeated, this structure is used by picking the argument as the root of the tree and generating leafs by following the properties of an acceptable argumentation line. In Figure 3.1, a dialectical tree that is derived from the example in Table 3.1 can be seen.

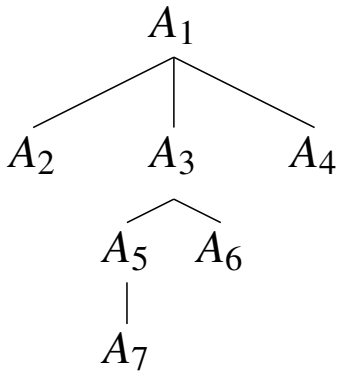


Figure 3.1: Dialectical tree for the example in Table 3.1.

After constructing a dialectical tree for an argument, a marking process is used for determining whether the argument is defeated. The nodes that construct the dialectical tree are recursively marked as defeated (D) or undefeated (U) by using the following the properties [2]:

1. Initially all leaves in a dialectical tree are marked as undefeated.

2. The nodes will be marked as defeated if it has a child node marked as undefeated. If all children are marked as defeated, the parent node will be marked as undefeated.

Figure 3.2 shows the result of the marking procedure for the dialectical tree in Figure 3.1. The defined marking process is a bottom-up approach. If the root of the dialectical tree is marked as undefeated, the root argument is said to be warranted. In order to warrant a claim, initially the arguments that support the claim are constructed. Each argument creates a dialectical tree and marking process is applied. As mentioned before, queries for a DeLP interpreter can be answered as *YES*, *NO*, *UNDECIDED*

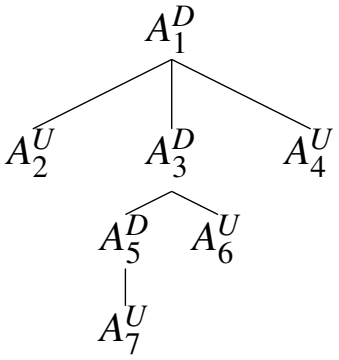


Figure 3.2: Marked dialectical tree for the example in Table 3.1.

or *UNKNOWN*. A claim is said to be warranted if there exists a warranted argument that supports the claim. The query for claim α is answered according to the following rules:

- *YES*, if claim α is warranted
- *NO*, if claim $\neg\alpha$ is warranted
- *UNDECIDED*, if neither α nor $\neg\alpha$ are warranted
- *UNKNOWN*, if α is not in the language

Notice that, it is sufficient to have a single undefeated leaf to defeat a node in a dialectical tree. Therefore, it is unnecessary to explore the whole dialectical tree. If a node already has an undefeated leaf, the other leafs can be skipped because they will not effect the outcome of the marking procedure. Pruning the dialectical tree by preventing unnecessary exploration will result in efficiently computed answers. In Figure 3.3, the marked dialectical tree for the example in Table 3.1 and the pruned dialectical tree are shown. The red nodes indicate the pruned nodes.

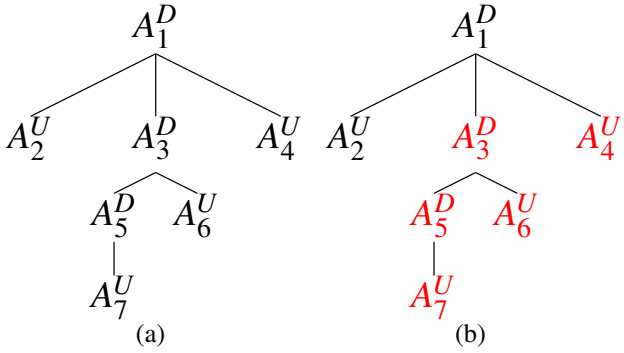


Figure 3.3: (a) The marked dialectical tree for the example in Table 3.1 and (b) the pruned dialectical tree.

3.2 The Applications of Defeasible Logic Programming

DeLP is initially introduced to combine the results of Logic Programming and Defeasible Argumentation [2]. Afterwards, many extensions over the original formalism are proposed in previous studies. In one earlier work, description logic is extended by using defeasible rules [13]. DeLP is also extended by temporal rules to deal with durative facts and the delays between the rules [14]. P-DeLP enhances original DeLP capabilities for qualitative reasoning with possibilistic uncertainity and fuzzy knowledge, and it pairs defeasible beliefs with certainty-weights which defines a lower bound in terms of a necessity measure for the certainty of facts and rules [15, 16]. Pt-DeLP is an extension of temporal defeasible argumentation framework with probabilistic weights [17].

There are several robotic applications of defeasible logic programming. In an earlier work, DeLP is used as a layered framework in a robotic domain with Khepera mobile robots to cope with inconsistent information [18]. In another work, actions of a cleaning service robot are determined by DeLP [19]. DeLP is also proposed to resolve possible collisions in Unmanned Autonomous Vehicles (UAVs) through communications among them [20].

The approach in this thesis differs from earlier works in terms of its use. To the best of our knowledge, this is the first time that defeasible logic programming is used for scene interpretation. Furthermore, defeasible logic programming framework is extended to address probabilistic and temporal analysis of scenes by using observation facts.

4. ARGUMENTATION-BASED SCENE INTERPRETATION

We propose a system to solve the scene interpretation problem by using defeasible logic. The system takes observations as input and produces the values of the requested predicates on objects through internal argumentation. If the system can not reach a consensus on a subject, it does not return conflicting results; instead it states that the query is undecidable. In our system, all commonsense rules and observation facts are represented as defeasible rules and facts, respectively. The problem for the medical domain presented in [21] is similar to the problem that we would like to address in the sense that the incoming data are usually unreliable, incomplete and inconsistent. The same properties also hold for robot domains. For this reason, we believe that this framework improved with our extensions is suitable for solving the scene interpretation problem.

4.1 Sensory Data Sources

In our system, we assume that the robot has a set of perceptual data sources($s_{<source>} \in S$). In our special case study, our robot is equipped with an RGB-D camera, and both 3D object recognition and point cloud segmentation algorithms run in parallel and provide observation information. Note that the outputs of these sources are not necessarily provided synchronously.

We use the same notion about the field of view of the robot as in our previous work [10]. The field of view is defined as a coverage region that the perception algorithms are expected to perceive the objects located in that particular area. This region is continually updated during the movement of the robot. Objects modelled in the knowledge base are continually checked about whether a detection/recognition regarding them should take place with respect to their positions relative to the robot. For more detailed information on how these regions are computed, the reader is referred to [10]. The field of view region sizes of our Pioneer 3-AT robot are presented in Figure 4.1. Assuming that the RGB-D sensor of the robot is on the origin of

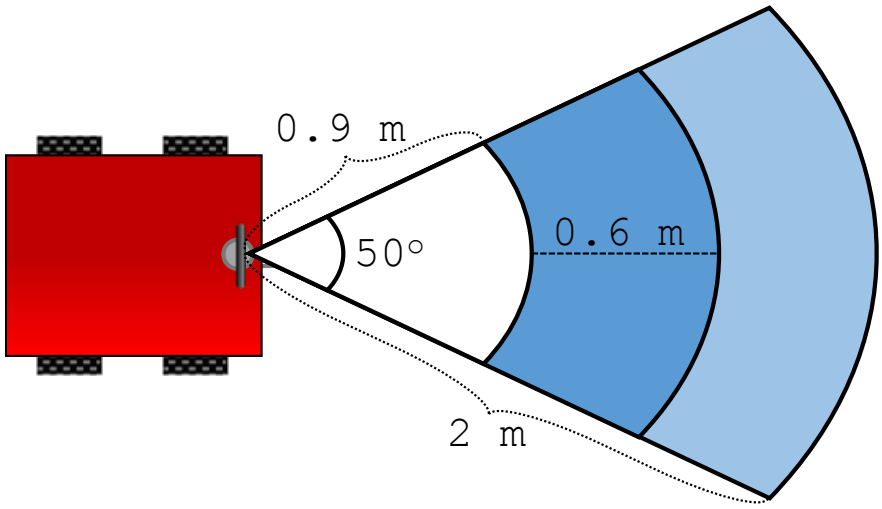


Figure 4.1: The field of the view of the Pioneer 3-AT robot.

the circular slice seen in the figure, the area in dark blue represents the region in which the objects are expected to be detected and recognized consistently with the highest performance. Additionally, recognition expectancy of the objects in the region corresponding to the area in light blue falls linearly starting from the curve between the light and dark blue areas, and reaching zero on the outer curve of the given slice. As a result, object detections and recognitions made in the latter area are considered to be less reliable as the location of the detection/recognition gets further away from the origin. Any gathered information that falls outside of these regions at the time of sensing is considered fully unreliable and are not taken into account.

4.2 Modeling the Scene Using Defeasible Logic Programming

The proposed system is an extended version of the DeLP framework that initially appeared in [2]. Their formalism is adapted and an extension is used that can represent all domain facts and rules as defeasible [21]. Additionally, we define observation beliefs as well as defeasible facts and rules. The method for comparing arguments is also redefined to evaluate both certainty weights and time. Incoming data from different sources serve as input for defeasible logic programming-based interpretation. These are fused with domain beliefs to come up with conclusions.

In this thesis, the notation of the existing DeLP frameworks is followed [2, 15]. In the proposed system, the robot has a set of domain beliefs ($\delta_i \in \Delta$) as a set of defeasible rules or defeasible facts to represent its knowledge about the domain. We adopt the certainty-weighted defeasible rules and defeasible facts in

which certainty-weights serve as lower bounds for the certainty [15]. A defeasible fact is represented with a pair (α, ω) where α denotes a predicate and $\omega \in [0,1]$ denotes the certainty-weight for producing the belief on it. Examples of some defeasible facts used in our system are shown in Table 4.1. δ_1, δ_2, δ_3 and δ_4 define coefficients for reliability of a corresponding source by using defeasible facts. *srcRelForType(<source>)* determines the reliability of <source> by assigning weight to the defeasible fact associated with the type classification success of <source>. *srcRelForColor(<source>)* and *srcRelForSize(<source>)* represent the color and size classification success of the related source, respectively. Different from the other defeasible facts, *srcRelForNotSeen(<source>)* states that <source> is reliable for detecting objects and weight is assigned using the true positive rate.

Table 4.1: Examples of defeasible facts.

δ_m	Defeasible Facts (certainty-weights are not specified.)
δ_1	*srcRelForType(<source>)*
δ_2	*srcRelForColor(<source>)*
δ_3	*srcRelForSize(<source>)*
δ_4	*srcRelForNotSeen(<source>)*

A defeasible rule is denoted by the following pair with implication: $(\alpha_0 \leftarrow \alpha_1 \wedge ... \wedge \alpha_n, \omega)$ which states that if $\alpha_1 \wedge ... \wedge \alpha_n$ is true, α_0 should also be true associated with a given weight ω. In Table 4.2, examples of defeasible rules are given. δ_5, δ_6 and δ_7 are defeasible rules that are used in order to derive object properties from observations. $type(X,Y)$ denotes that object X is of type Y. Similarly, $size(X,Y)$ and $color(X,Y)$ represent corresponding properties of object X. δ_8, δ_9 and δ_{10} states that every value

Table 4.2: Examples of defeasible rules.

δ_m	Defeasible Rules (certainty-weights are not specified.)
δ_5	$type(X,Y) \leftarrow obsObj_{type}(X,Y,Z) \wedge$ srcRelForType(Z)
δ_6	$color(X,Y) \leftarrow obsObj_{color}(X,Y,Z) \wedge$ srcRelForColor(Z)
δ_7	$size(X,Y) \leftarrow obsObj_{size}(X,Y,Z) \wedge$ srcRelForSize(Z)
δ_8	$\neg type(X,Z) \leftarrow type(X,Y) \wedge Y != Z$
δ_9	$\neg color(X,Z) \leftarrow color(X,Y) \wedge Y != Z$
δ_{10}	$\neg size(X,Z) \leftarrow size(X,Y) \wedge Y != Z$
δ_{11}	$object(X) \leftarrow size(X,Y) \vee color(X,Z) \vee type(X,W)$
δ_{12}	$\neg object(X) \leftarrow obsObjNotSeen(X,Y) \wedge srcRelForNotSeen(Y)$
δ_{13}	$size(X, tall) \leftarrow type(X, bowlingPin)$
δ_{14}	$\neg type(X, ball) \leftarrow inRegion(X, inclinedSurface)$

for each object property are mutually exclusive. δ_{11} derives an object predicate, when an object attribute is detected. δ_{12} defines that if an existing object can not be seen in its location (i.e., $objNotSeen$ predicate becomes true.), the system concludes that no object exists in the given location. δ_{13} and δ_{14} are more specific rules that respectively state that for a specific object type (e.g., bowling pin), the object's *size* is predetermined as $tall$, and if an object stands on an inclined surface, the object's $type$ shall not be a $ball$.

An observation belief is defined as a tuple denoted by $\theta_i(\alpha, \omega, \tau)$ among a set of observation beliefs, such that $\theta_i \in \Theta$, where α is a predicate, ω is the certainty-weight proportional to the confidence on the corresponding observation accuracy and τ is the observation time-stamp. Observations are gathered through the sensors of the robot, particularly, the processed RGB-D data by different sources are used. Each observation is tagged by its latest observation time (τ) which distinguishes it from defeasible rules or facts. Observation beliefs are used during the argumentation process. Sample observations are listed in Table 4.3. θ_1 defines the $location$, $type$ and *source* attributes of the perceived object. Similarly, θ_2 and θ_3 indicate *color* and *size* properties of the recognized object. Finally, θ_4 denotes that the object is not recognized even if it is in the field of view of the robot. The details of how these observation beliefs are generated is explained in the next section.

Table 4.3: Observation Beliefs

θ_i	Observation	Description
θ_1	$obsObj_{type}(loc_i,<type>,<source>)$	The type of an object is determined.
θ_2	$obsObj_{color}(loc_i,<color>,<source>)$	The color of an object is detected.
θ_3	$obsObj_{size}(loc_i,<size>,<source>)$	The size of an object is computed.
θ_4	$obsObjNotSeen(loc_i,<source>)$	The object is not seen in its assigned location even though it is in the field of view of the robot.

An argument A_i among a set of arguments ($A_i \in \Psi$) relates a consistent support set from observation beliefs, domain beliefs to a claim, $claim(A_i)$. The support set is always selected as the minimal consistent set that can derive the claim. Each argument is associated with a weight, $weight(A_i)$ computed as the minimum of the certainty-weight of its supports. The time-stamp of an argument (τ) is determined as that of the oldest

observation belief of the support set. It may be left unknown if the support set includes only domain beliefs.

4.3 Constructing Observation Beliefs through Perception

Different sources provide different types of information and they may be in conflict in some cases. They even themselves alone give conflicting measurements in consecutive time steps due to noisy sensory data. A perception outcome is a tuple $\Upsilon(loc_m, \rho, \upsilon, s_{<source>}, \tau, \omega')$ where loc_m denotes an object location that is determined by a perception source. $\rho \in \{type, color, size\}$ denotes the attribute name for the corresponding observation and υ denotes the value of this attribute. $s_{<source>}$ indicates the information source and τ represents the observation time-stamp during sampling. Finally, $\omega' \in [0,1]$ denotes the belief on the accuracy of the observation determined by the corresponding source (e.g., similarity measure for object recognition).

Each new perception outcome is matched to an object location maintained in the knowledge base, and added to the perception outcome set of this object. Then, the observation beliefs are constructed for every unique information type considering time. Multiple occurrences for the same object property with different values may also occur in the perception outcome set. An example set of perception outcomes for a single object (the object's actual attribute values: $type = bowlingPin$, $color = red$ and $size = tall$) are given in Table 4.4. As clearly seen from the table, there are multiple perception outcomes for this object in various time steps from different sources. For

Table 4.4: A list of sample perception outcomes (Υ_i) for an object.

Υ_i	ρ	υ	$s_{<source>}$	τ	ω'
Υ_1	$type$	$bowlingPin$	s_l	1	0.95
Υ_2	$type$	$ball$	s_l	1	0.85
Υ_3	$type$	$bowlingPin$	s_{lhs}	2	0.9
Υ_4	$color$	red	s_{lhs}	2	0.9
Υ_5	$type$	$bowlingPin$	s_l	2	0.9
Υ_6	$size$	$tall$	s_c	2	0.9
Υ_7	$type$	$bowlingPin$	s_l	3	0.95
Υ_8	$color$	red	s_{lhs}	3	0.85
Υ_9	$type$	$bowlingPin$	s_{lhs}	4	0.95
Υ_{10}	$color$	$blue$	s_{lhs}	4	0.85
Υ_{11}	$type$	$bowlingPin$	s_l	21	0.85
Υ_{12}	$type$	$bowlingPin$	s_l	22	0.95
Υ_{13}	$type$	$ball$	s_l	23	0.85

example, in time-stamp 1, the object is perceived both as a bowling pin and a ball. For dealing with misclassification, recurrences are minimised by applying the following procedure while maintaining the resolution of the data for perception outcomes. Initially, perception outcomes are grouped together considering their time intervals τ_{i-j} and their unique selection of triples that consist of location, property name and source (loc_m, ρ and $s_{<source>}$). Each perception outcome is allowed to be only in a single group. The time interval for a group is determined by the time difference of the consecutive perception outcomes. When the time difference is greater than an empirically defined threshold, the consecutive perception outcomes are placed in a different group. The threshold is determined based on robot experiments to accurately distinguish the perception outcomes gathered during different time intervals. A $Group(\Upsilon)$ function returns a set of groups considering the explained definitions. For example, using the set of perception outcomes from Table 4.4 and a threshold of time interval as 5, $Group(\Upsilon)$ returns the following groups:

$$Group(\Upsilon): \begin{array}{c} \{\Upsilon_1, \Upsilon_2, \Upsilon_5, \Upsilon_7\}_{(loc_i,type,s_l)\tau_1-\tau_3} \\ \{\Upsilon_3, \Upsilon_9\}_{(loc_i,type,s_{lhs})\tau_2-\tau_4} \\ \{\Upsilon_4, \Upsilon_8, \Upsilon_{10}\}_{(loc_i,color,s_{lhs})\tau_2-\tau_4} \\ \{\Upsilon_6\}_{(loc_i,size,s_c)\tau_2-\tau_2} \\ \{\Upsilon_{11}, \Upsilon_{12}, \Upsilon_{13}\}_{(loc_i,type,s_l)\tau_{21}-\tau_{23}} \end{array}$$

After perception outcomes are grouped together, observation beliefs are constructed as follows: Each group of perception outcomes $\{\Upsilon_1, \Upsilon_2, ..., \Upsilon_n\}$ with n number perceptions defines a set of observation beliefs $\theta_j(obsObj_{<property\ name>}(loc_m, <property\ value>, s_{<source>}), \omega_{\theta_j}, \tau_{\theta_j})$ for each unique property value (υ_{Υ_i}) in the perception outcomes. τ_{θ_j} is defined as the time of the last perception outcome in the group. For each observation belief, ω_{θ_j} is calculated with Equation 4.1. Variable a is defined empirically to prevent premature decision making due to small number of observations.

$$\omega_{\theta_j} = \frac{\sum_{i=1}^{n} \begin{cases} \omega'_{\Upsilon_i} & \upsilon_{\Upsilon_i} = <property\ value> \\ 0 \end{cases}}{max(n, a)} \tag{4.1}$$

When we consider $Group(\Upsilon)_1 = \{\Upsilon_1, \Upsilon_2, \Upsilon_5, \Upsilon_7\}_{(loc_i,type,s_l)\tau_1-\tau_3}$ with $\rho = type$ and $s_{<source>} = s_l$ which has three *bowlingPin* values and a single *ball* value, the following two observation beliefs are created $obsObj_{type}(loc_i, bowlingPin, s_l)$

and $obsObj_{type}(loc_i, ball, s_l)$. Each observation belief has $\tau = 3$ due to the latest observation time of the group. $\omega_{bowlingPin}$ and ω_{ball} are calculated using Equation 4.1. After processing all groups in the example, a set of observation beliefs are generated as in Table 4.5 based on the given sample perception outcomes in Table 4.4.

Table 4.5: The constructed observation beliefs for the given perception outcomes from Table 4.4, given that $a = 2$.

θ_i	Observation Belief	ω	τ
θ_1	$obsObj_{type}(loc_i, bowlingPin, s_l)$	0.7	3
θ_2	$obsObj_{type}(loc_i, ball, s_l)$	0.21	3
θ_3	$obsObj_{type}(loc_i, bowlingPin, s_{lhs})$	0.93	4
θ_4	$obsObj_{color}(loc_i, red, s_{lhs})$	0.58	4
θ_5	$obsObj_{color}(loc_i, blue, s_{lhs})$	0.28	4
θ_6	$obsObj_{type}(loc_i, bowlingPin, s_l)$	0.6	23
θ_7	$obsObj_{type}(loc_i, ball, s_l)$	0.28	23
θ_8	$obsObj_{size}(loc_i, tall, s_c)$	0.45	2

4.4 Maintaining Consistency through Argumentation

Partial and/or inconsistent observations gathered through unreliable sensors are represented as observation facts, which are the customised versions of defeasible facts. Defeasible rules are defined in order to reason on observation facts and maintain a consistent world model. Therefore, using observation beliefs (Θ) and domain beliefs (Δ), a set of available arguments (ψ) are derived from $\Theta \cup \Delta$ by an iterative forward chaining algorithm. When a new observation belief is added to the set Θ, or a defeasible fact or rule is added to the set Δ, the algorithm is invoked again in order to derive further arguments. Priorities among the arguments are calculated by using $Defeats(A_i, A_j)$ function. $Defeats(A_i, A_j) = \{Proper, Blocking, none\}$ is a relation function that defines if A_i is a proper or blocking defeater of A_j or neither of the relations exist as in Equation 4.2.

$$\text{Defeats}(A_i, A_j) = \begin{cases} \text{Proper}, & \text{if } (\tau_{A_i} - \tau_{A_j} > b) \vee (\omega_{A_i} - \omega_{A_j} > c \wedge |\tau_{A_i} - \tau_{A_j}| \leq b) \\ \text{Blocking}, & \text{if } |\omega_{A_i} - \omega_{A_j}| \leq c \wedge |\tau_{A_i} - \tau_{A_j}| \leq b \\ \text{none}, & \text{otherwise} \end{cases} \tag{4.2}$$

where, b and c are predetermined thresholds to adjust strictness of comparison. For arguments A_i and A_j, if τ_{A_i} is higher than τ_{A_j} in addition to b, or the time difference between τ_{A_i} and τ_{A_j} is lower than b and $(\omega_{A_i} - \omega_{A_j})$ is greater than c, A_i is a proper defeater of A_j. If the time and weight difference for both A_i and A_j is lower than the thresholds, A_j is a blocking defeater for A_i. Otherwise, there is no existing defeater relation.

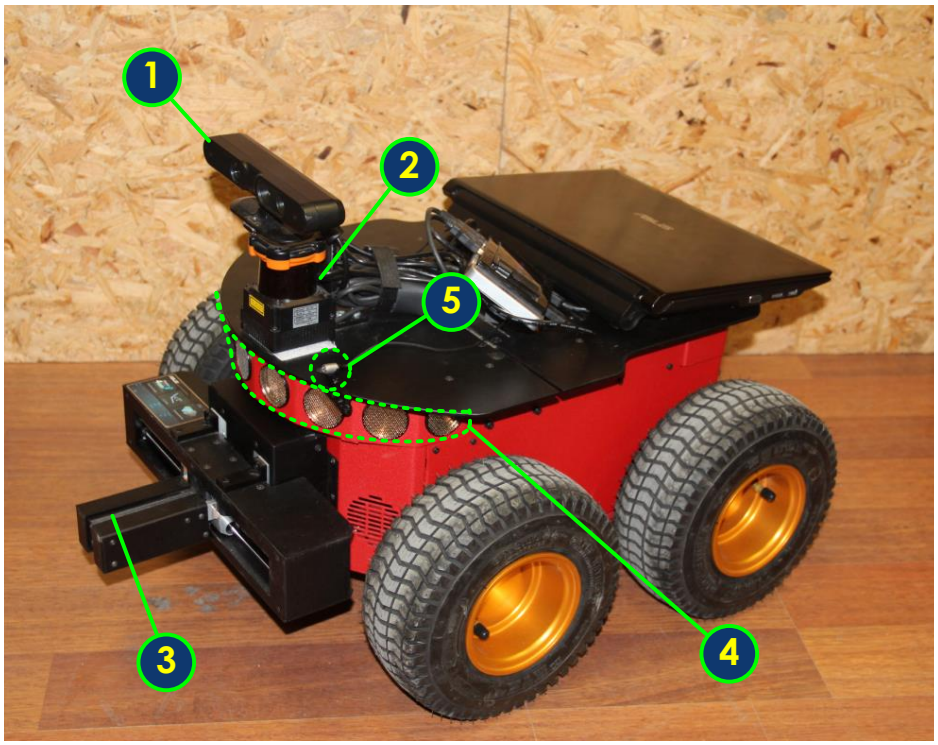
The DeLP inference method makes conclusions on any claim by constructing its arguments and related dialectical trees. If an argument that supports the given claim cannot be defeated by the marking process using the dialectical tree, the claim is said to be warranted. Furthermore, it is also possible to query for a predicate to find its warranted claims. In order to construct a consistent world model, every claim of $object(X)$, $type(X,Y)$, $color(X,Y)$ and $size(X,Y)$ are queried to be warranted. If the queries for the claims return YES at the end of the warrant procedure, these claims represent the world model.

5. EXPERIMENTS RESULTS

In this section, the performance of the system is evaluated on a number of case scenarios in both simulation and real-world environments.

5.1 Experiment Setup

In real-world experiments, Pioneer 3-AT robot is used which is equipped with an Asus Xtion Pro Live RGB-D camera, 2D laser range finder, a sonar ring and a 2-DOF gripper (Figure 5.1). Three plastic bowling pins with different colors (red, green and blue), a green cylinder, a purple ball and a bigger beach ball are used in the experiments as manipulation objects. The robot is also equipped with a laptop (Intel Core i7 CPU 2.67GHz, 8GB RAM) for on-board computing.



① ASUS Xtion PRO RGB-D Camera ③ Tactile Sensors inside the Gripper
② Hokuyo UTM-30LX Laser Rangefinder ④ Forward-facing Sonar Sensors (8 pieces)
⑤ SONY ECMC115 Clip Microphone

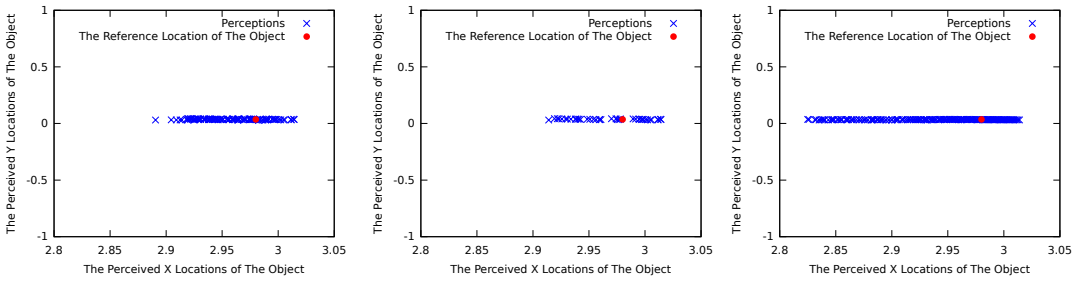
Figure 5.1: Pioneer 3-AT robot and its sensors.

5.1.1 Analysis of perception sources

An RGB-D camera is used to gather visual data from the environment. LINE-MOD, LINE-MOD HS and cluster-based segmentation algorithms, as perception sources, are utilized in order to perceive the world state information. Some preliminary experiments to measure the perception success rate of these sources are done, and the results are presented in the following subsections. The performances of the perception algorithms are evaluated in our robots in the real world.

5.1.1.1 Localization accuracy of perception

Localization accuracy is measured for each algorithm using the Pioneer 3-AT robot. The robot is initially located at (0,0). An object was placed in front of the robot, and the robot slowly moved towards it. The ground truth location of the object is compared with the outcome of the perception algorithms. In Figure 5.2, the perceived object locations are shown for each algorithm. It is noticed that the point cloud data



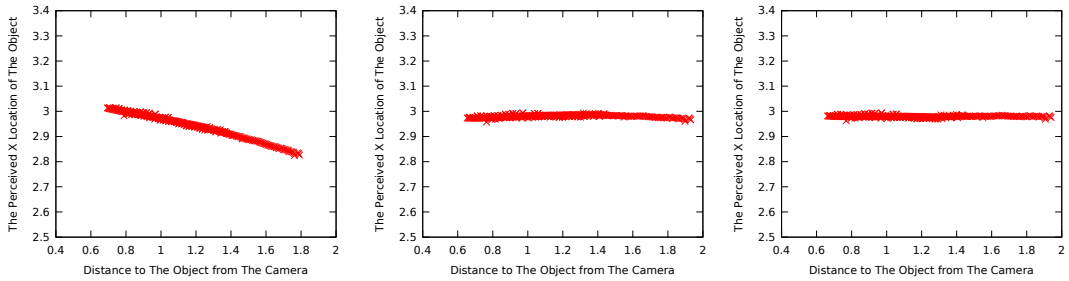
(a) LINE-MOD (b) LINE-MOD HS (c) Segmentation

Figure 5.2: The perceived object location for (a) LINE-MOD, (b) LINE-MOD HS and (c) segmentation algorithms.

gathered from the RGB-D camera affects the location accuracy, due to the error in the depth data. In Figure 5.3(a), outputs of the vision algorithms are fused, and X location of the perceived object is given in relation to the distance to the object from the camera. By applying regression methods to the outcome of the perception algorithms, the localization accuracy of perceptions is significantly improved. In Table 5.1, the mean squared errors (MSE) for the position estimation are given for each algorithm, and the results after the application of the regression algorithms.

The linear regression (LR) function for correcting errors is given in Equation 5.1.

$$Y(dist) = 1.16476 * dist - 0.155 \quad \textbf{(5.1)}$$



(a) (b) (c)

Figure 5.3: (a) The perceived object X location in relation to distance to the object for (a) original, (b) applying linear regression and (c) applying nonlinear regression method.

Here, *dist* defines the distance to the object from the camera that is calculated using the euclidean distance formula for x and y coordinates. The nonlinear regression (NLR) function is also given in Equation 5.2. The definition of *dist* is the same as the linear regression function. As seen in Table 5.1, using LR or NLR functions reduces the mean squared error and increases position accuracy estimation of the perception algorithms. In the real-world experiments, the nonlinear function in Equation 5.2 is utilized.

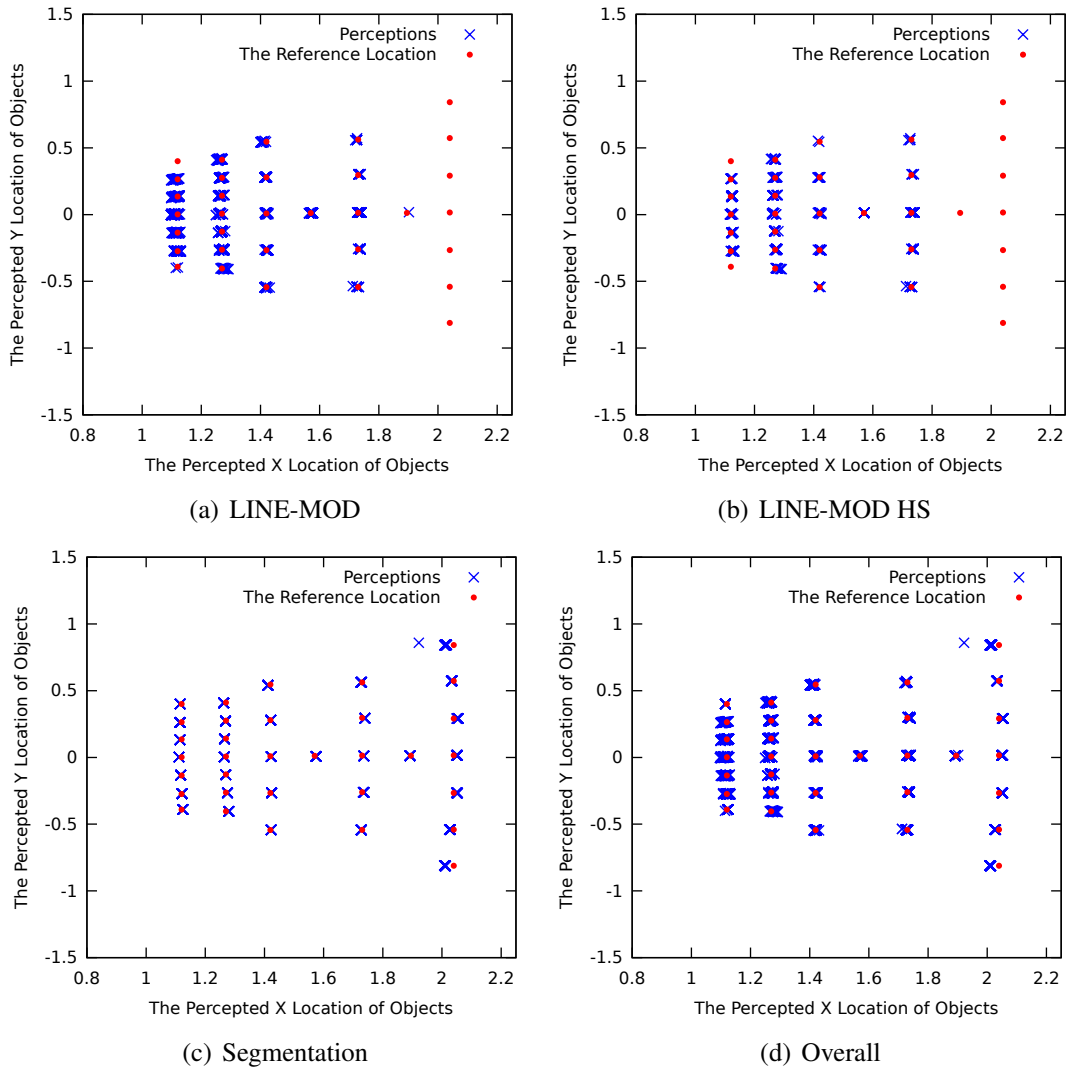
$$Y(dist) = \begin{cases} 1.1243 * dist - 0.130 & dist \leq 1.25 \\ 1.1957 * dist - 0.204 & dist > 1.25 \end{cases} \quad \textbf{(5.2)}$$

Table 5.1: The mean square errors of the perceived locations for each perception algorithm.

Algorithms	MSE for X	MSE for Y	MSE for X and Y
LINE-MOD	0.003207	0.000299	0.003220
LINE-MOD HS	0.005197	0.000505	0.005222
Segmentation	0.005297	0.000205	0.005301
Overall	0.003049	0.000163	0.003054
LINE-MOD w/ LR	0.000504	0.000349	0.000613
LINE-MOD HS w/ LR	0.000813	0.000625	0.001026
Segmentation w/ LR	0.000398	0.000176	0.000435
Overall w/ LR	0.000292	0.000173	0.000339
LINE-MOD w/ NLR	0.000522	0.000340	0.000623
LINE-MOD HS w/ NLR	0.001075	0.000606	0.001235
Segmentation w/ NLR	0.000117	0.000170	0.000206
Overall w/ NLR	0.000237	0.000168	0.000290

The effects of the different angles and distances to the object are also analyzed using 33 object locations. In this experiment, an object is placed to a predefined locations

of which the pattern is given in Figure 5.4, and the performance of the perception algorithms are evaluated. The robot is allowed to perceive each object for 20s. In Figure 5.4, the successful perceptions are shown for LINE-MOD, LINE-MOD HS and segmentation algorithms. The red dots represent real object positions, and the blue crosses represent their perceived locations. Also the overall result, which is the combined result of the perception algorithm outcomes is presented in the figure.



(a) LINE-MOD

(b) LINE-MOD HS

(c) Segmentation

(d) Overall

Figure 5.4: The position estimation of object locations for (a) LINE-MOD, (b) LINE-MOD HS, (c) the segmentation algorithm and also (d) the combination of the estimations.

5.2 Simulation Experiments

The simulation environments are modelled in the Stage robot simulator [22], where 40 objects are randomly cluttered in the environment as in Figure 5.5. In the figure, red and green squares represent objects and robots, respectively. In the simulation

environment, the perception source outcomes are simulated using the results of the preliminary experiments of the perception sources. Five different types (*bowlingPin*, *cylinder*, *box*, *ball* and *bigBall*) of objects are defined with five different colors (*red*, *yellow*, *blue*, *green* and *purple*). The size for *bowlingPin* and *cylinder* types is predetermined as *tall* whereas for *box*, *ball* and *bigBall* *short*. The simulations are run on a system equipped with Intel Core i7 CPU 2.67GHz and 8GB RAM.

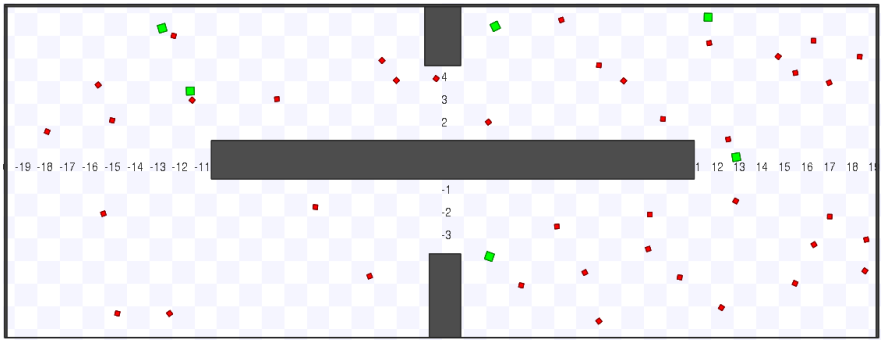


Figure 5.5: The simulation environment.

The robot is allowed to wander in the environment for 10 minutes and to sense the objects through its sensors. This experiment is run 18 times with 6 different initial starting positions for the robot. The simulation experiments are also run on the other world modeling systems for comparison. The results of these experiments are given in Table 5.2, which compares the proposed approach with the previous work of the same

Table 5.2: The results of the simulation experiments with comparison.

		The proposed system	The previous work	WIRE
Object(X)	TP	27.11	22.50	23.83
	FP	3.50	1.89	6.00
	Precision	0.89	0.92	0.80
Type(X,Y)	TP	24.11	17.94	20.78
	FP	3.11	4.56	4.44
	Precision	0.89	0.80	0.82
Color(X,Y)	TP	23.72	22.11	20.78
	FP	3.50	0.39	4.06
	Precision	0.87	0.98	0.84
Size(X,Y)	TP	19.72	11.94	14.56
	FP	7.44	10.56	10.61
	Precision	0.73	0.53	0.58
All Predicates	TP	23.67	18.62	19.99
	FP	4.39	4.35	6.28
	Precision	0.84	0.81	0.76

research group [10] and WIRE [11]. Each column represents a different approach. The average of the final values (at the end of the simulation experiments) for true positives, false positives and precision for object detection are given for each system. The classifications of *type*, *color* and *size* are also given using the same metrics. True positives state that the related predicates are successfully determined, and false positives represent incorrect or duplicate determination of the related predicates. As clearly seen from the table, while the robot can construct its world model consistently with each approach, the proposed approach outperforms the other systems. In Figure 5.6, the total number of true positives in relation to time is given. The proposed system also uses the background knowledge, the size for *bowlingPin* and *cylinder* types is *tall* whereas for *box*, *ball* and *bigBall short*. Due to the background knowledge usage, the estimation of size is significantly more successful than the other method's estimations.

Space and time complexity of the proposed algorithm are also analyzed. In Table 5.3, the number of queries, the number of arguments, the number of node exploration, the total query time and the number of beliefs currently stored in the knowledge are shown in relation to the time passed from the start of the system. In Figure 5.7, the runtime performance of the inference process in DeLP is analyzed. Figure 5.7(a) shows the

Table 5.3: The analysis of the system during runtime.

Time Passed(s)	# of Queries	# of Arguments	# of Node Exploration	Query Time(ms)	# of Facts	# of Observations	# of Rules
100	65	1566	89	6	13	240	69
200	85	2276	117	9	13	346	69
300	89	3287	125	9	13	487	69
400	109	4291	178	29	13	638	69
500	118	5265	168	13	13	777	69
600	147	7566	249	32	13	1128	69
700	155	9527	231	21	13	1426	69
800	164	11568	256	62	13	1737	69
900	168	14293	290	54	13	2147	69
1000	173	16070	969	517	13	2416	69
1100	170	17089	2346	960	13	2567	69
1200	173	17767	289	57	13	2663	69

number of beliefs maintained with respect to time. Each constructed observation belief is added to the knowledge base, and that increases the total number of the beliefs of the robot. The number of beliefs directly affects the number of arguments that can be derived using the beliefs as seen in Figure 5.7(b). The number of the explored nodes during the warrant procedure in relation to the number of queries is given in Figure

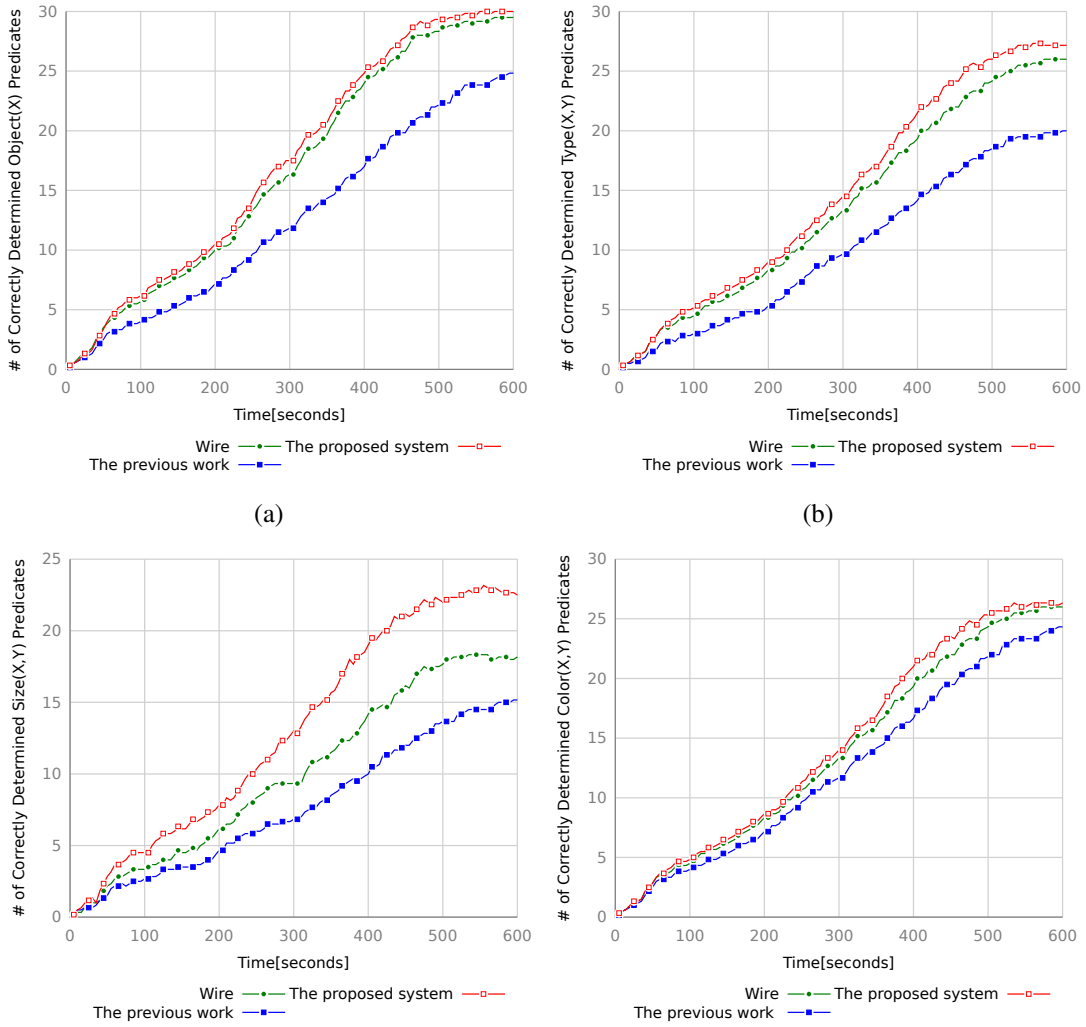


Figure 5.6: The comparison of different approaches in terms of (a) correctly detected object count, correctly classified (b) type, (c) size and (d) color count.

5.7(c). In Figure 5.7(d), the response time per query is analyzed in relation to the number of arguments that can be derived using the current belief space. The system has low query response time, and it is suitable for real-time applications.

5.3 Robot Experiments

In order to analyse the performance of the proposed method, several real-world scenarios are prepared. The main application domain is selected as the mobile manipulation for real-robot experiments. Each scenario is run for two times, and the results for each interpretation approaches are measured. The following scenarios are investigated for the applicability of the system.

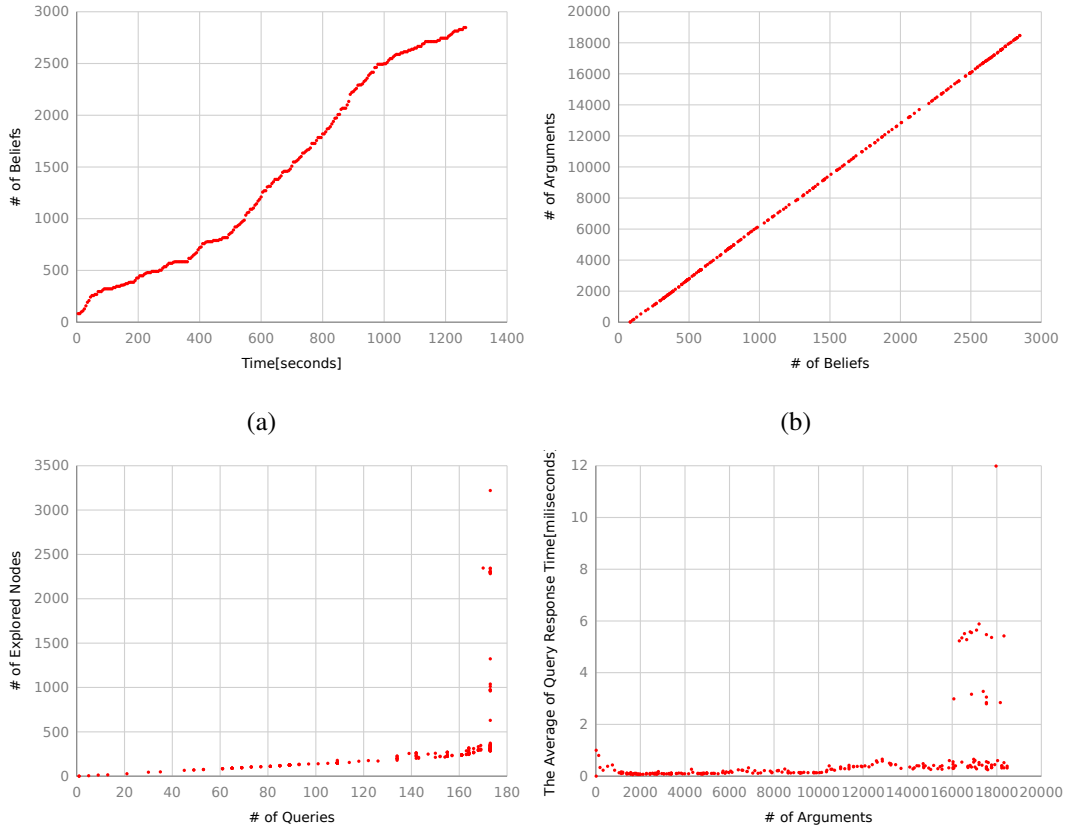
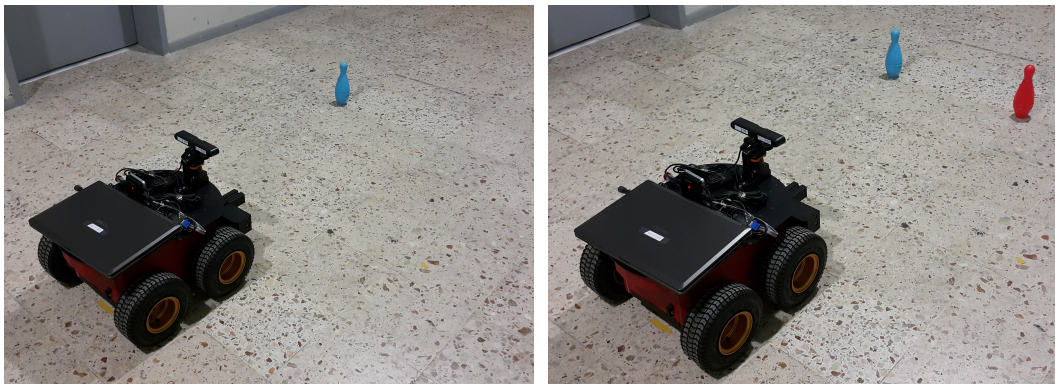


Figure 5.7: The analysis of the inference process in DeLP. (a) The number of maintained beliefs in relation to time, (b) the number of arguments constructed with respect to the number of beliefs, (c) the total number of explored nodes for a given number of queries and (d) the average of query response time in relation to the number of arguments are shown.

5.3.1 Scenario I

In this scenario, a blue bowling pin is initially located in the environment. Afterwards, a red bowling pin is added to the environment while the robot senses the scene as in Figure 5.8. The point of view of the robot before, during and after the object addition is shown in Figure 5.9. In Figure 5.10, the number of true positives for predicates $object(X)$, $type(X,Y)$, $color(X,Y)$ and $size(X,Y)$ that are warranted by the argumentation process are given. The number of the total encountered objects defines how many objects are in the field of view of the robot. As can be seen from this figure, the robot can create the $object(X)$ predicate as soon as it detects an object by segmentation. After the object and its properties are recognized, the values of the corresponding predicates are also assigned to their correct values.



(a) (b)

Figure 5.8: Scenario I: a bowling pin is initially located in the environment (a) , and afterwards another bowling pin is added (b).

(a) (b) (c)

Figure 5.9: Scenario I: The point of view of the robot (a) before, (b) during and (c) after the object removal.

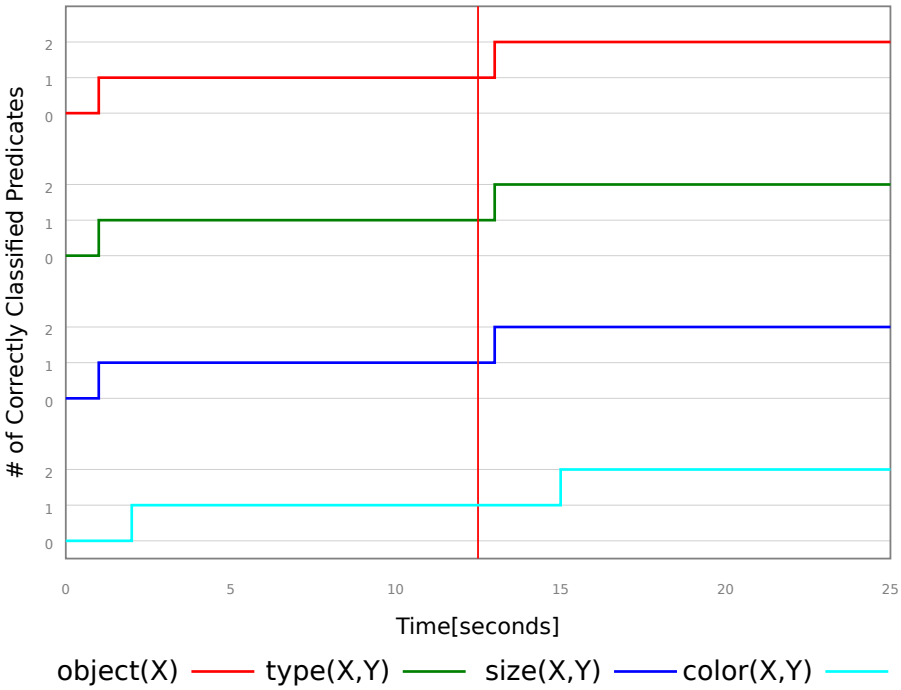


Figure 5.10: Scenario I: a bowling pin is initially located in the environment. Afterwards, another bowling pin is added to the environment which is shown with a red horizontal line as the robot senses the scene.

5.3.2 Scenario II

In this scenario, a red bowling pin is initially located in the environment. Afterwards, the object is taken out of the environment by a human. The changes in the truth values of $object(loc_1)$, $type(loc_1, bowlingPin)$, $color(loc_1, red)$ and $size(loc_1, tall)$ can be seen in Figure 5.11. The time-stamp that the object is taken out is marked with a red horizontal line. As expected, the claim $object(loc_1)$ is no more warranted after several seconds following the disappearance of the object. Note that even the claim $object(loc_1)$ is not warranted any longer, *type*, *size* and *color* properties are still available for the missing object in case of this object can be returned to the scene again.

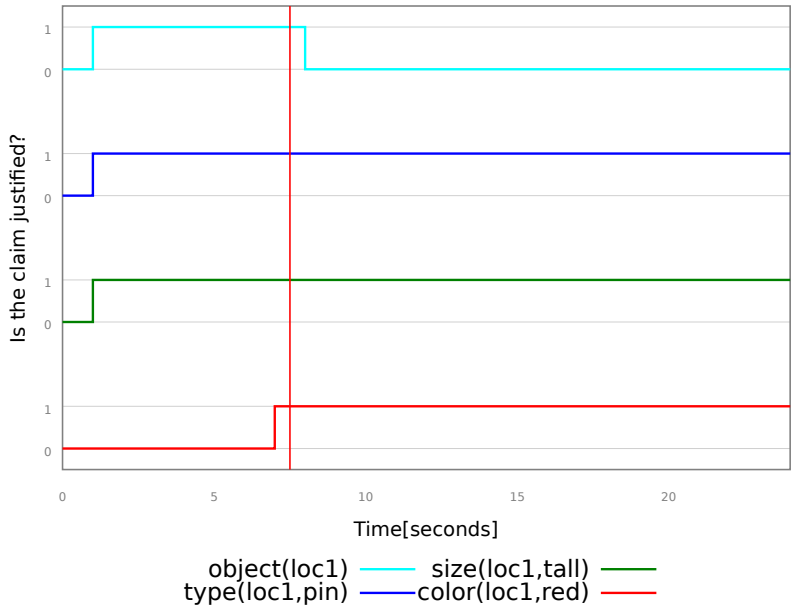


Figure 5.11: Scenario II: a red bowling pin is initially located in the environment. Afterwards, the object is taken out of the environment by a human. The red horizontal line states the time that the object is taken out.

5.3.3 Scenario III

In this scenario, a bowling pin is initially located in the environment. Later on, the location of the object is changed by a human. Predicate $objMovedTo(loc_i, loc_j)$ states that the object initially located at loc_i is moved to the new location loc_j. The following defeasible rule is created for this scenario as follows:

$$objMovedTo(L1,L2) \leftarrow \neg object(L1) \wedge type(L1,X) \wedge size(L1,Y) \wedge color(L1,Z) \wedge type(L2,X) \wedge size(L2,Y) \wedge color(L2,Z) \wedge L1 != L2$$

The given rule defines that if an object is disappeared from the world model, and a new object is added to the model with the same properties, it is assumed to be the same object as the disappearing one. In Figure 5.12, the truth values of the claims $object(loc_1)$, $object(loc_2)$ and $objMovedTo(loc_1, loc_2)$ are given along with the number of true positives for *type*, *color* and *size* properties.

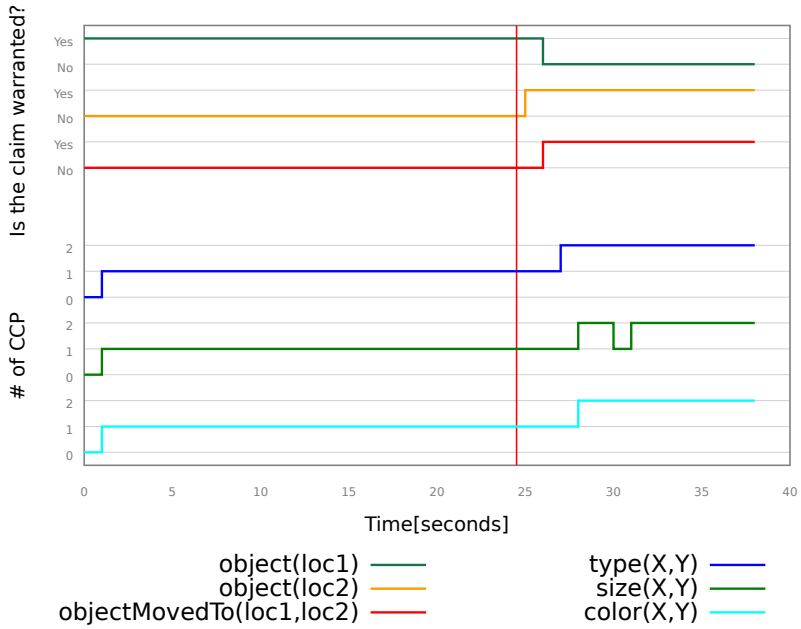


Figure 5.12: Scenario III: a bowling pin is initially located in the environment. Later on, the location of the object is changed by a human. The red horizontal line states the time that the object is moved.

5.3.4 Scenario IV

In this scenario, three bowling pins and a ball are used, and these objects are initially cluttered in the environment as in Figure 5.13. The robot wanders in the environment and senses these objects without any apriori information. During wandering in the environment, the objects can enter into or exit from the field of view of the robot as in Figure 5.14. The speed of the robot during the movements is limited to 0.1 m/s. In Figure 5.15, the constructed world model is given in terms of the number of true positives at each time step. The true positive rates of the sensory data sources decrease when the robot is moving. This directly affects the performance of the system. For instance, the falls and rises of the number of correctly classified predicates in the figure originate from the vision failures that occur during the movement of the robot. Our solution to this problem is reducing the reliability of the sources during the movement

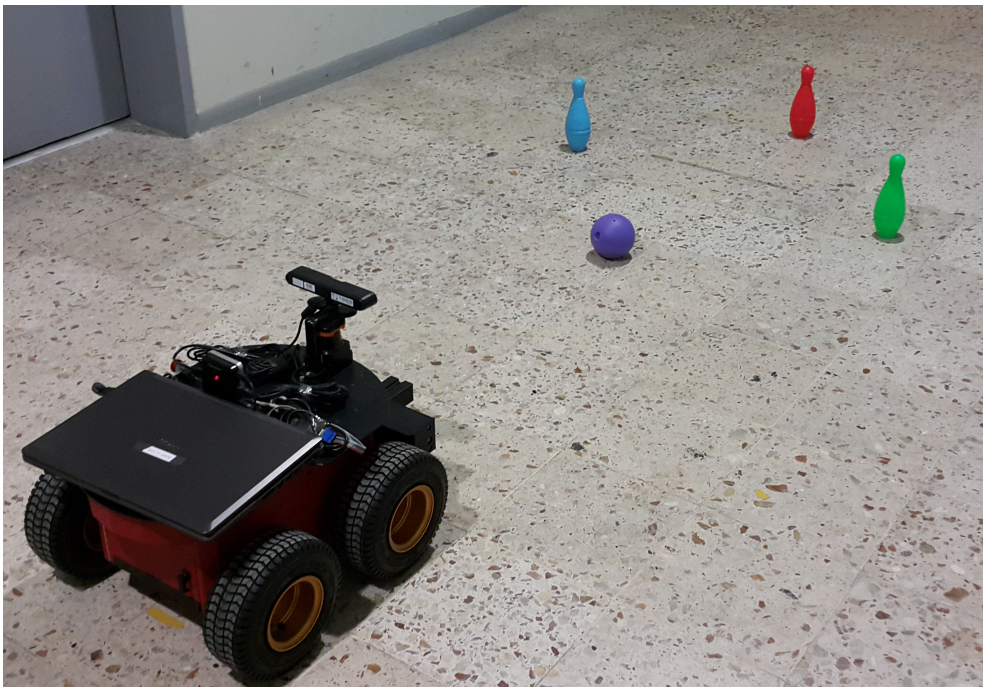
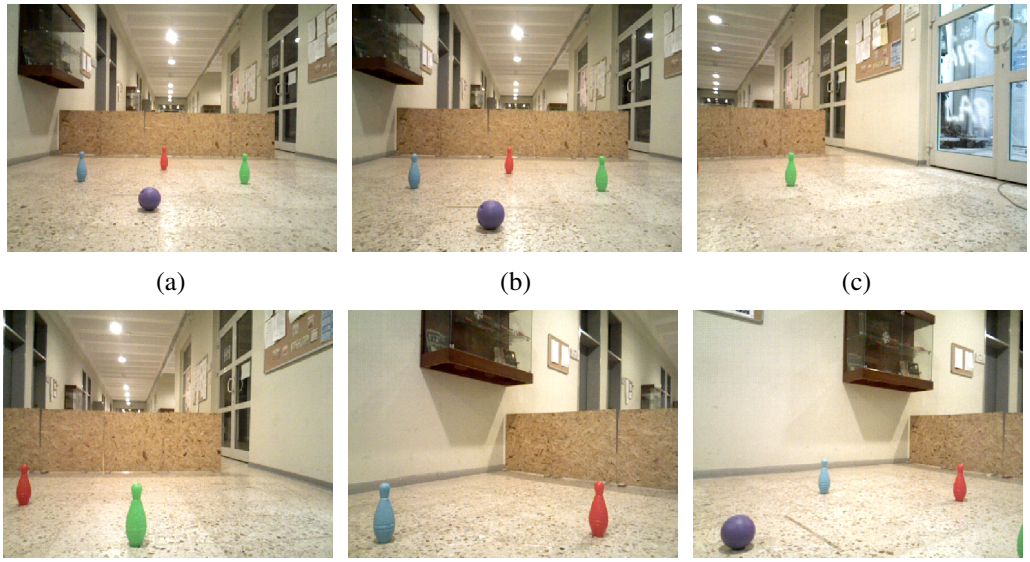


Figure 5.13: Scenario IV: 3 bowling pins and the ball are initially cluttered in the environment.

of the robot or postponing the reasoning process when the sensory data are believed to be too noisy. A more detailed analysis of this case is left as a future work.



(a) (b) (c)

(d) (e) (f)

Figure 5.14: Scenario IV: The point of view of the robot during the wandering in the environment.

The real-world scenarios are also interpreted using the previous work by the same research group and WIRE. In Table 5.4, the comparison of these approach is given in terms of true positives, false positives and false negatives. The true positives represent

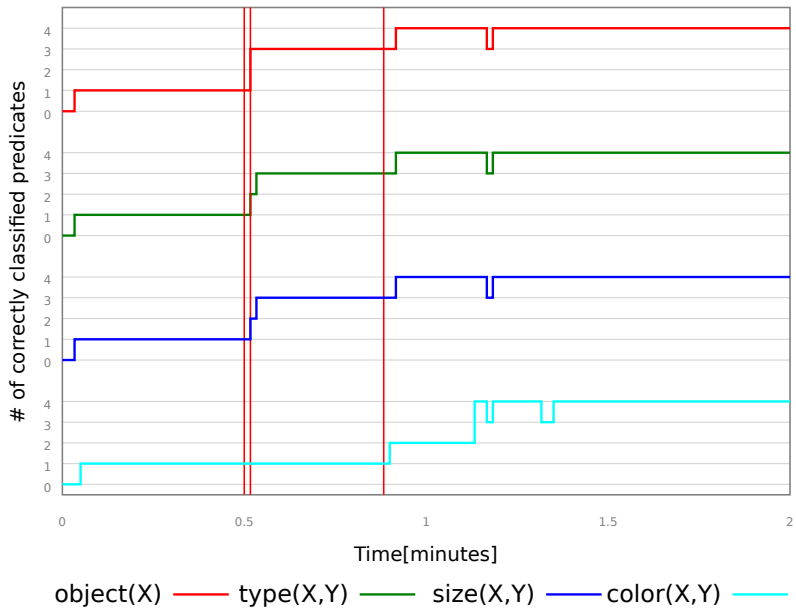


Figure 5.15: Scenario IV: 3 bowling pins and the ball are used, and these objects are initially cluttered in the environment. The robot wanders in the environment and senses the objects. Each red line states that a new object is entered in the field of view of the robot.

Table 5.4: The comparison of different approaches on real-world scenarios.

Scenario #	Method	TP	FP	FN
Scenario I	The proposed method	64	0	24
Scenario I	The previous work	61	0	33
Scenario I	WIRE	64	9	30
Scenario II	The proposed method	15	20	4
Scenario II	The previous work	5	11	14
Scenario II	WIRE	13	34	6
Scenario III	The proposed method	67	5	5
Scenario III	The previous work	48	11	25
Scenario III	WIRE	66	37	7
Scenario IV	The proposed method	768	0	160
Scenario IV	The previous work	627	0	313
Scenario IV	WIRE	737	0	203

the total number of correctly classified objects for 2 runs of the experiment. The false positives denote incorrect or duplicate object detections. The false negatives represent the number of missed objects in the environment. At each time step, measurements from the proposed method, the previous work by the same research group and WIRE are calculated. In the table, the total number of true positives, false positives and false negatives are given for each measurement on time steps. As can be seen from the table, the proposed method outperforms the other approaches in the true positive rates and

the false negative rates. The previous work is only better in terms of the false positives for the second scenario.

6. CONCLUSION AND FUTURE WORK

A cognitive robot needs to maintain both an up-to-date world state model through the incoming data from its sensors and background/commonsense knowledge for reasoning efficiently. If either of these has faulty or missing elements, the robot may not behave rationally in an unknown environment. In order to deal with these kinds of situations, the robot should be able to expand its knowledge by dynamically adding or removing new domain rules and update its domain beliefs.

In this thesis, a DeLP-based argumentation method is proposed to maintain a consistent world model for an autonomous mobile robot system. Defeasible Logic Programming is extended to contain observation facts in addition to defeasible facts and defeasible rules. The performance of the method is evaluated on various case scenarios on both simulations and a real robot with comparisons to the previous work by the research group of the author and an existing world modeling system. In the simulation experiments, the proposed system outperforms the other methods with 18.35% more true positives than the closest approach. Also the precision of the proposed system is 0.04% higher compared to the other approaches in the simulation experiments. A total of 8 hours of real-world experiments are done using a Pioneer 3-AT mobile robot. In addition to simulations, the proposed system also has a higher number of true positives in the real world experiments. Therefore, these results show that an argumentation-based scene interpretation framework ensures consistency to a great extent.

This work can be extended by advancing the comparison method of the arguments so that they can be changed dynamically for different queries in order to obtain alternative results. In particular, comparison shall be made more strict or more relaxed which would result in more accurate or more informative results.

It is also planned to analyze the outcomes of different comparison methods and their effects to the reasoning procedure. Furthermore, Defeasible Logic Programming can be extended to have metric temporal logic, which would increase expressive power of the language.

As another future direction, extending the experimental-setup to include more advanced scenarios and applying the proposed method to multi-robot systems are targeted. The obtained results indicate that the proposed method is promising to be applied to multi-robot scenarios.

REFERENCES

[1] **Kemp, C.C.**, **Edsinger, A. and Torres-Jara, E.** (2007). Challenges for robot manipulation in human environments, *IEEE Robotics and Automation Magazine*, ***14***(1), 20.

[2] **García, A.J. and Simari, G.R.** (2004). Defeasible logic programming: An argumentative approach, *Theory and practice of logic programming*, ***4***(1+ 2), 95–138.

[3] **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.

[4] **Dellaert, F.**, **Fox, D.**, **Burgard, W. and Thrun, S.** (1999). Monte carlo localization for mobile robots, *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, IEEE, pp.1322–1328.

[5] **Grisetti, G.**, **Stachniss, C. and Burgard, W.** (2007). Improved techniques for grid mapping with rao-blackwellized particle filters, *Robotics, IEEE Transactions on*, ***23***(1), 34–46.

[6] **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.

[7] **Ersen, M.**, **Sariel-Talay, S. and Yalcin, H.** (2013). Extracting Spatial Relations Among Objects for Failure Detection, *on Visual and Spatial Cognition*, 13.

[8] **Ersen, M.**, **Ozturk, M.D.**, **Biberci, M.**, **Sariel, S. and Yalcin, H.** (2014). Scene Interpretation for Lifelong Robot Learning, *The 9th International Workshop on Cognitive Robotics (CogRob 2014) held in conjunction with ECAI-2014*, Prague, Czech Republic.

[9] **Trevor, A.J.**, **Gedikli, S.**, **Rusu, R.B. and Christensen, H.I.** (2013). Efficient organized point cloud segmentation with connected components, *Semantic Perception Mapping and Exploration (SPME)*.

[10] **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*.

[11] **Elfring, J.**, **Van Den Dries, S.**, **Van De Molengraft, M. and Steinbuch, M.** (2013). Semantic world modeling using probabilistic multiple hypothesis anchoring, *Robotics and Autonomous Systems*, ***61***(2), 95–105.

[12] **Nute, D.** (1994). Defeasible logic, *Handbook of logic in artificial intelligence and logic programming*, ***3***, 353–395.

[13] **Governatori, G.**, (2004). Defeasible description logics, Rules and Rule Markup Languages for the Semantic Web, Springer, pp.98–112.

[14] **Governatori, G. and Terenziani, P.**, (2007). Temporal extensions to defeasible logic, AI 2007: Advances in Artificial Intelligence, Springer, pp.476–485.

[15] **Chesñevar, C.I.**, **Simari, G.R.**, **Alsinet, T. and Godo, L.** (2004). A logic programming framework for possibilistic argumentation with vague knowledge, *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, AUAI Press, pp.76–84.

[16] **Alsinet, T.**, **Chesnevar, C.I.**, **Godo, L. and Simari, G.R.** (2008). A logic programming framework for possibilistic argumentation: Formalization and logical properties, *Fuzzy Sets and Systems*, ***159***(10), 1208–1228.

[17] **Godo, L.**, **Marchioni, E. and Pardo, P.**, (2012). Extending a temporal defeasible argumentation framework with possibilistic weights, Logics in Artificial Intelligence, Springer, pp.242–254.

[18] **Ferretti, E.**, **Errecalde, M.**, **Garcıa, A. and Simari, G.** (2006). Khedelp: A framework to support defeasible logic programming for the khepera robots, *ISRA06*.

[19] **Ferretti, E.**, **Errecalde, M.**, **García, A.J. and Simari, G.R.**, (2007). An application of defeasible logic programming to decision making in a robotic environment, Logic Programming and Nonmonotonic Reasoning, Springer, pp.297–302.

[20] **Lam, H.P. and Governatori, G.** (2012). Towards a model of UAVs Navigation in urban canyon through Defeasible Logic, *Journal of Logic and Computation*, exr028.

[21] **Black, E. and Hunter, A.** (2009). An inquiry dialogue system, *Autonomous Agents and Multi-Agent Systems*, ***19***(2), 173–209.

[22] **Vaughan, R.** (2008). Massively multi-robot simulation in stage, *Swarm Intelligence*, ***2***(2-4), 189–208.

CURRICULUM VITAE

Name Surname: Çağatay Koç

Place and Date of Birth: Malatya, 6 January 1990

Adress: Istanbul Technical University, Faculty of Computer and Informatics, Research Lab. 3, 34469, Maslak/Istanbul

E-Mail: kocca@itu.edu.tr

B.Sc.: Istanbul Technical University, Department of Computer Engineering, 2012

Professional Experience and Rewards: Research Assistant, Istanbul Technical University, Faculty of Computer and Informatics (February 2013 - ongoing)

PUBLICATIONS/PRESENTATIONS ON THE THESIS

- **Koc C.** and Sariel S. (2015). Argumentation-based Scene Interpretation Using Defeasible Logic Programming, *International Conference on Advanced Robotics, (ICAR)*, 2015 17th.
    - This work is also presented at: *Workshop on Autonomous Robots and Multirobot Systems (ARMS), International Conference on Autonomous Agents & Multiagent Systems*, 2015 Istanbul, Turkey.

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*.

- 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.