석 사 학 위 논 문
Master's Thesis

로봇의 물체 정리를 위한 자동 작업 경로 생성

Automated task planning
using object arrangement optimization

2017

강 민 철 (姜 旻 澈 Kang, Mincheul)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

석 사 학 위 논 문

로봇의 물체 정리를 위한 자동 작업 경로 생성

2017

강 민 철

한 국 과 학 기 술 원

로봇공학 학제전공

로봇의 물체 정리를 위한 자동 작업 경로 생성

강 민 철

위 논문은 한국과학기술원 석사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2017년 6월 7일

심사위원장 윤 성 의 (인)
심 사 위 원 명 현 (인)
심 사 위 원 조 성 호 (인)

Automated task planning using object arrangement optimization

Mincheul Kang

Advisor: Sung-Eui Yoon

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Master of Science in Robotics

Daejeon, Korea
May 22, 2017

Approved by

[illegible]

Sung-Eui Yoon
Professor of Robotics Program

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

MRE
20154315

강민철. 로봇의 물체 정리를 위한 자동 작업 경로 생성. 로봇공학 학제전공 . 2017년. 20+ii 쪽. 지도교수: 윤성의. (영문 논문)
Mincheul Kang. Automated task planning using object arrangement optimization. Robotics Program . 2017. 20+ii pages. Advisor: Sung-Eui Yoon. (Text in English)

초 록

본 논문은 로봇이 Task and Motion planning 알고리즘을 이용하여 자동으로 물체를 정리하는 방법을 제시한다. 로봇이 자동으로 물체를 정리하기 위해서는 어지럽혀진 물체를 어디에 두어야 하는지를 알아야 한다. 우리는 3D 모델들로 구성된 잘 정리된 씬을 만들어서 목표를 선정하여 준다. 정리된 씬은 물체의 관계, 물체와 인간과의 관계를 고려하여 만든다. 이런 관계들은 미리 잘 정리된 씬으로부터 추출하여 학습한다. 이렇게 만들어진 정리된 씬이 만들어지면, 임의의 어지럽혀진 씬으로부터 정리된 씬으로 도달하기 위해 Task and Motion planning 알고리즘을 사용한다. 또한 이러한 전반적인 접근 방식을 고려하여, 우리는 효율적으로 물체를 정리하기 위한 Priority layer를 제안한다. Priority layer는 전체 실행 시간을 줄이기 위해 물체를 옮기기 위한 비용을 계산하는 그리디 방식으로 처리한다. 이러한 비용을 추정하기 위해 우리는 로봇이 이동할 거리와 로봇이 각 목표 위치에 취할 행동의 수를 추정한다. 우리는 다양한 수의 객체로 5 가지 장면에서 테스트하고 Priority layer를 2 개의 잘 알려진 Task and Motion planner에 적용하였다. 결과적으로, 우리의 방법은 잘 정리된 씬으로부터 PR2 로봇이 물체를 정리할 수 있도록 하였다. 또한 우리가 제안한 Priority layer가 최종적인 시간을 기존의 Task and Motion planner 보다 최대 2배까지 줄이는 것을 보였다.

핵 심 낱 말 테스크와 모션 플래닝, 자동화 로봇, 서비스 로봇

Abstract

We present a method for a robot to automatically arrange objects using task and motion planning. To arrange objects automatically using a robot, we need a target layout of objects, which guides the robot where to place those objects. We synthesize such a target layout by considering hierarchical and spatial relationships between parent and child objects, and pairwise relationships. These relationships are pre-extracted from positive examples as a training step. Once we have the target layout, we can use any task and motion planner to reach the target layout from an arbitrary object layout. Given this overall approach, we also propose a priority layer to arrange objects efficiently. The priority layer estimates costs of moving objects and processes an object in a greedy manner to reduce the overall execution time. For estimating such costs, our method estimates the distance traveled by the robot and the number of actions a robot will take to reach the target position. We tested our method in five different scenes with varying numbers of cluttered objects and applied our method to two well-known task planners with target layouts computed by our layout computation method. As a result, our method enables a PR2 robot to arrange cluttered objects by considering positive examples. Additionally, we found that our priority layer reduces the total running time up to two times over prior task planners used with our computed layouts.

Keywords Task and Motion planning, Autonomous robot, Service robot

Contents

Contents . . . .

List of Tables i

List of Figures ii

Chapter 1. INTRODUCTION 1

Chapter 2. RELATED WORK 3

2.1 Object Arrangements 3

2.2 Task and Motion Planning 3

Chapter 3. Overview 4

3.1 Motivation 4

3.2 Overview of Our Approach 4

Chapter 4. Automatic layout computation 6

4.1 Extraction 6

4.2 Layout Computation 8

Chapter 5. Task and Motion Planning 9

5.1 Definition of domain 9

5.2 Defining a problem in the domain 10

5.3 Priority layer 10

Chapter 6. Experiments and Results 12

6.1 Results of layout computation 12

6.2 Evaluations of the priority layer 12

Chapter 7. Conclusion 16

Bibliography 17

Acknowledgments in Korean 19

Curriculum Vitae in Korean 20

List of Tables

6.1 We measured the total running time, number of actions, and travel distance averaged out from five independent tests for each scene. Experimental results show that our method decreases the number of actions and distance traveled. As a result, the running time of different different planners is reduced by using our method. 15

List of Figures

1.1 Our method starts with a cluttered scene (a), computes a target layout of objects, and arranges cluttered objects (b) by using a virtual PR2 robot with an existing task and motion planner (e.g, HPN [1]). Note that books are well located in the book shelf, objects on the desk are well arranged, and the teddy bear is located at the bed. 2

3.1 These are two examples that can occur in the object arrangement. (a) shows that target locations of objects are intertwined. For example, L2 is the current location of the object A, but is the goal location of the object C. (b) shows that the parent-child relationships of objects are intertwined. For example, the object C is the parent of the object B, i.e., to pick C, we need to move B away first, but in the goal, the parent of the object C is the object B. 5

3.2 This shows an overview of our approach. Our method first constructs an initial, cluttered scene (b) from an input state (a). We then construct the arranged scene (c) as the target layout from the cluttered scene (b) using various relationship obtained from positive examples to (b). Using the information from scenes (b) and (c), our priority layer guides the task and motion planner (TMP) about an arranging order of objects. Finally, a robot arranges those objects using TMP. When all the objects are arranged by the robot, we reach the goal state (d). 5

4.1 (a) The blue line is s_1^p chosen from the parent object p given an object o, and d_1 is the distance of the object center c to s_1^p. θ_1 is the orientation of the object between s_1^p and s_1^o. The orange line is s_2^p, and d_2 is an additional distance of the object center c to s_2^p. (b) to (d) show benefits of utilizing these informations additionally. (b) is the result of the prior work to arrange objects. (c) is the result of encoding and considering specific surface IDs, resulting in better arrangement of objects. In (d), we add the additional distance term (d_2) to make our arrangement closer to positive examples. This yields a more human-friendly arrangement of objects by making it easier for humans to reach frequently used objects such as a cellphone. 7

6.1 This figure shows the number of iterations and running time takes for layout computation, as a function of the number of objects. 13

6.2 This figure shows the result of each planner's execution time. We can see that the execution time of two existing TMP algorithms is reduced robustly by applying our priority layer. 14

6.3 This figure shows the running time breakdown: TMP, Priority layer, Pick, Place and Move. This is reported from the scene containing 13 cluttered objects. The priority layer occupies a very small portion, but it significantly reduces the overall running time. 14

6.4 This image sequence shows the process of arranging objects from a cluttered state (a) to an arranged state (d). (b) and (c) show intermediate states between (a) and (d). (b) shows the robot is picking up the table clock, and (c) shows the robot is carrying the stapler. 15

Chapter 1. INTRODUCTION

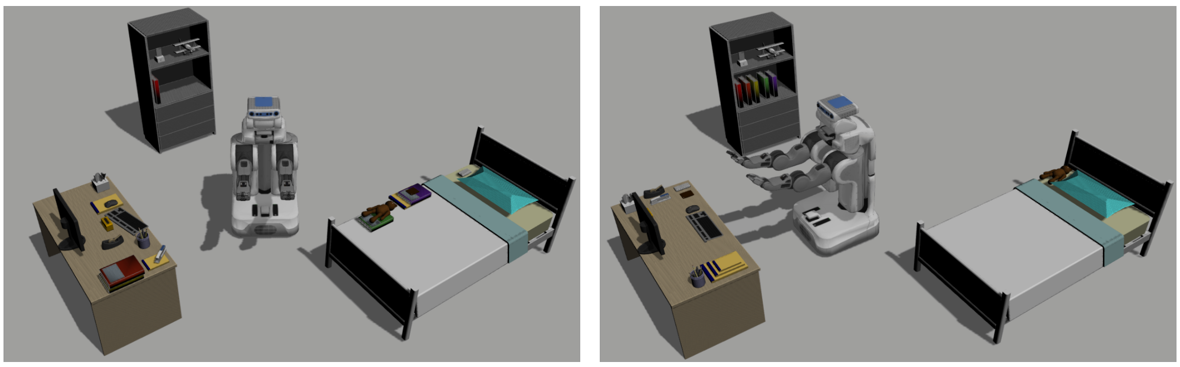
Robots and autonomous devices have helped humans tremendously. We have assigned robots to a variety of tasks ranging from simple tasks - washing cars - to dangerous tasks - handling radioactive materials. Currently, a lot of manufacturing companies use industrial robots to reduce cost and boost productivity. However, domestic robots have not gained much traction due to their incompetence in performing household chores. Recently, with the rise of artificial intelligence and advances of humanoid robots, there has been a huge interest in designing and utilizing domestic robots. Intelligent domestic robots can interact with humans and offload a significant amount of works from humans.

One way that a domestic robot can be intelligent is by finding a series of feasible actions for a given task and performing those actions on its own. Many task and motion planning methods have been proposed to efficiently plan and perform various applications such as cooking food, facilitating industrial logistics, harvesting fruits, and so on. In this paper, we focus on arranging cluttered objects neatly in a room.

To perform a task such as arranging objects, robots require a specific goal or a guided plan from a user. Unfortunately it is very inconvenient for a user to give specific information to a robot on every iteration. To address this issue, we focus on an approach that automatically generates a goal from a non-specific command and plans feasible actions using the goal.

Main contributions. In this paper, we present an integrated approach for automated planning of a robot for object arrangements. Our method consists of two main parts: generating a target layout for object arrangements and using the layout to efficiently plan a series of feasible actions to reach the target layout. To find a target location of each object, we first optimize locations of objects using various relationship between objects, and ergonomic factors extracted from positive examples (Sec. 4). We can then use any task and motion planning to reach the target layout (Sec. 5.1, 5.2). We also propose a priority layer to efficiently perform complex object arrangements by utilizing various relationships extracted from positive examples. The priority layer predicts the cost required to arrange objects to target locations (Sec. 5.3).

To evaluate our method, we tested a room with many objects randomly placed on top of furniture. We applied our layout computation method to two different task and motion planners, and showed that our method automatically arranges objects based on the computed layouts. In addition, we also showed that our priority layer robustly improves the overall execution performance up to two times over prior task planners (Sec. 6).



(a) Cluttered scene (b) Arranged scene

Figure 1.1: Our method starts with a cluttered scene (a), computes a target layout of objects, and arranges cluttered objects (b) by using a virtual PR2 robot with an existing task and motion planner (e.g, HPN [1]). Note that books are well located in the book shelf, objects on the desk are well arranged, and the teddy bear is located at the bed.

Chapter 2. RELATED WORK

In this section, we discuss prior work on object arrangements, followed by integrated task and motion planning algorithms.

2.1 Object Arrangements

Jiang et al. [2] suggested a method for a robot to place objects in stable and semantically preferred area using a supervised learning with point cloud data. On their subsequent work, Jiang et al. [3] considered arranging objects by learning the relationship between human poses and objects so that objects are easily accessible by humans. Abdo et al. [4] proposed a method to tidy up objects on shelves and boxes by applying user preferences. These user preferences were determined by a collaborative filtering technique based on crowd-sourced and mined data.

Our approach applies the work of automatic furniture arrangements [5] to arrange cluttered objects based on the relationship of objects. This work synthesizes an indoor scene by using the relationships of furniture. The relationships are extracted from examples of furnished indoor scenes and ergonomic factors such as visibility and accessibility. Since such 3D examples of scenes are already widely available thanks to the wide usage of 3D modeling tools (e.g., Autodesk 3ds Max), we chose to use this approach to get an arranged scene that gets used in task planning.

2.2 Task and Motion Planning

Task and motion planning (TMP) is an algorithm to get feasible actions based on geometrically planned motions. TMP focuses on scalability and completeness of planning and can be applied to robotic applications that perform complex tasks such as cooking [6, 7] and industrial logistics [8]. Hierarchical Planning in the Now (HPN) [1] reduces search depth and planning time by dividing the plan and execution in a large state space.

Some works proposed a method using an interface layer that connects symbolic task planner to geometric motion planner. The work of De et al. [9, 10] used shared literals to combine two planners and control backtracking based on hierarchical task networks [11]. Srivastava et al. [12] used off-the-shelf task planners, and if an obstacle gets in the way, they simply create an additional task that removes the obstacle. Although these planners can arrange multiple objects accurately, they do not consider the robot's distance or the number of actions.

Recently, Zhang et al. [13] proposed a method to minimize the cost of pick-and-place actions for objects by applying a traveling salesman problem to get feasible actions. They give an arbitrarily large cost to an object that has no feasible actions. This method is effective in tidying up a room by putting away objects in a set of boxes, but it is not suitable for arranging objects. We therefore arrange objects and accelerate the overall execution time by measuring each object's approximate cost in our priority layer that utilizes 3D scene information.

Chapter 3. Overview

We first give motivations of our work, followed by an overview of our approach.

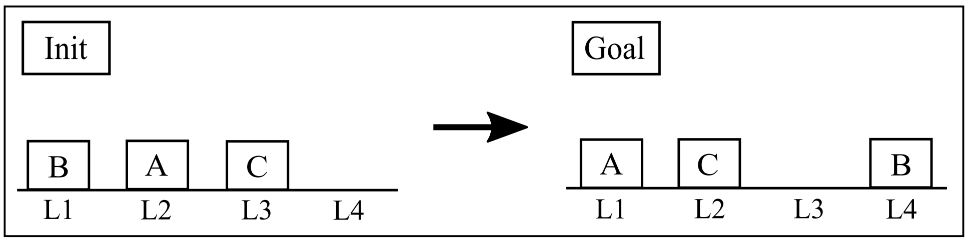
3.1 Motivation

Task and Motion Planning (TMP) automatically computes a sequence of tasks to perform a given task, while considering various geometric constraints (e.g., avoiding collisions). Recent robotic applications use a TMP algorithm to automatically perform a task such as tidying up a table [14] or industrial logistics [8]. These works automatically perform a task through manually specified goal states. Unfortunately, these approaches assume that goals of objects in the task are already given or manually specified, and automatically computing such targets for TMP have been studied less. In addition, most TMP approaches run slower as we have more objects in the task. In our object arrangement tasks, many objects are located on furniture, and their relationships are intertwined between cluttered and arranged states as shown in Fig. 3.1. Existing TMP planners can plan an arrangement of multiple objects, but they can be slow for many objects. This is because it is difficult to estimate the exact cost of reaching the target location of each object in a complicated situation until it is executed.

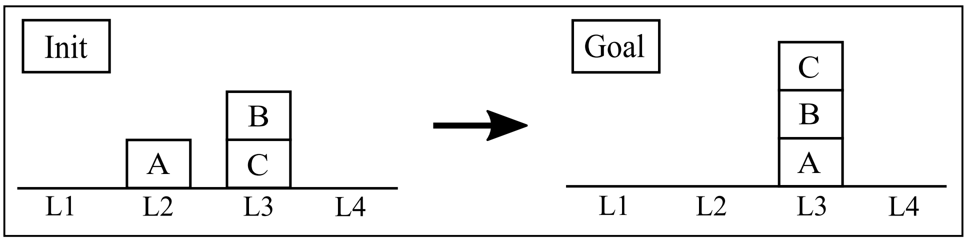
3.2 Overview of Our Approach

To automatically arrange objects, we propose a method that integrates automatic layout computation and TMP. Automatic layout computation synthesizes a target layout of objects contained in the scene. Using the target layout, our work plans feasible actions of the robot to arrange objects using TMP. Moreover, we propose a priority layer that utilizes various relationships between objects, and enables efficient planning between automatic layout and TMP.

Fig. 3.2 shows an overview of our approach. Our approach is divided into three stages: First, we decide locations of objects with automatic optimization of object arrangement. Our work extracts the object relationships, such as spatial, hierarchical and pairwise relationship, from positive examples to a given scene. Using the extracted relationship and ergonomic factors, such as visibility and accessibility, our work optimizes locations of objects into arranged locations. Second, we use the priority layer that not only connects automatic layout and TMP, but also allows for efficient planning by prioritizing objects. The priority layer measures an approximate cost of arranging objects to arrive at their target locations using the information of 3D scene. Third, we use an existing task and motion planner to plan the movement of a robot to arrive at the goal state. In particular, we pick an object that has the lowest cost and move it to its goal state in a greedy way, to efficiently perform the TMP process. Our work repeats this process until all objects are arranged.



(a) Intertwined between objects.



(b) Hierarhically intertwinded case.

Figure 3.1: These are two examples that can occur in the object arrangement. (a) shows that target locations of objects are intertwined. For example, L2 is the current location of the object A, but is the goal location of the object C. (b) shows that the parent-child relationships of objects are intertwined. For example, the object C is the parent of the object B, i.e., to pick C, we need to move B away first, but in the goal, the parent of the object C is the object B.

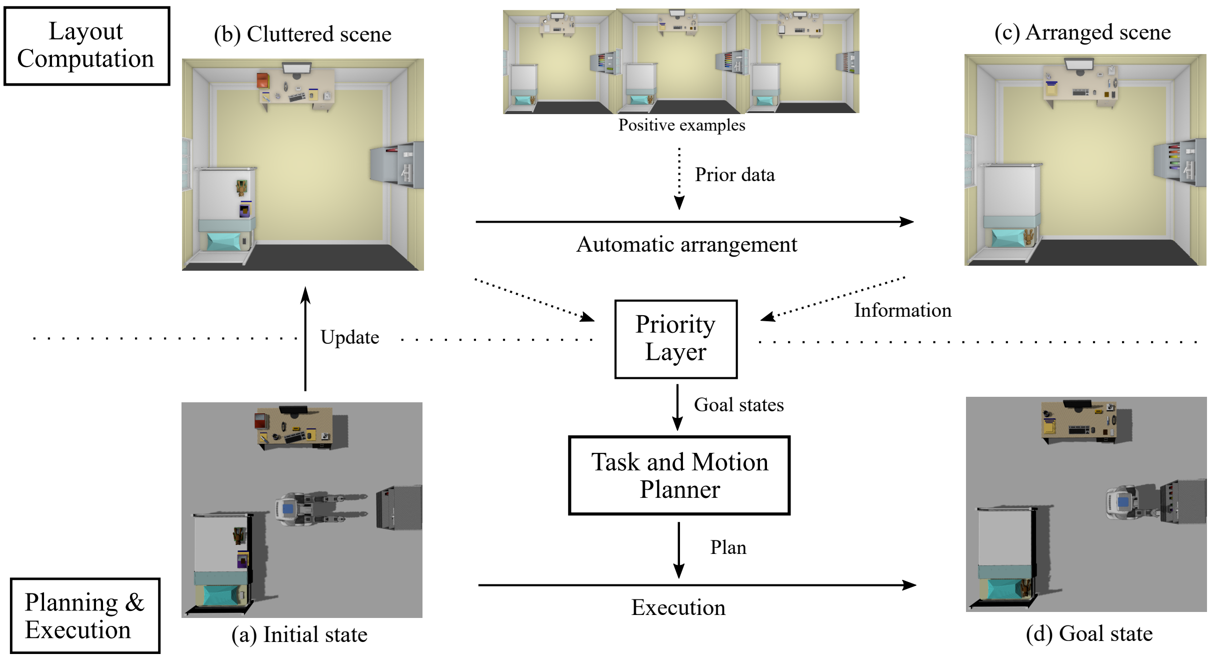


Figure 3.2: This shows an overview of our approach. Our method first constructs an initial, cluttered scene (b) from an input state (a). We then construct the arranged scene (c) as the target layout from the cluttered scene (b) using various relationship obtained from positive examples to (b). Using the information from scenes (b) and (c), our priority layer guides the task and motion planner (TMP) about an arranging order of objects. Finally, a robot arranges those objects using TMP. When all the objects are arranged by the robot, we reach the goal state (d).

Chapter 4. Automatic layout computation

We explain how to get arranged locations of objects from cluttered objects and how to set goal states in task and motion planning. The basic concept of computing the goal states is based on the work of automatic furniture arrangements [5].

4.1 Extraction

The extraction process obtains various relationships that serve as the basis for optimizing locations of objects. Specifically, our work extracts hierarchical, spatial, and pairwise relationship from objects in the training scenes. First, the hierarchical relationship is defined as a parent-child relation of two objects, and a parent object can support a child object. For example, the parent of a piece of furniture can be a room, and the parent of a keyboard can be a desk. We emphasize this relationship because it is important when moving and stacking objects. We also derive the spatial relationship between two objects with the parent-child relationship; this can be also considered as hierarchical-spatial relationship. The spatial relationship indicates where an object is located relative to its parent's location expressed in a two dimensional Cartesian coordinate space.

For computing relative locations of an object o, given its parent object p, we first identify the closest surface, s_1^o, of the object o from its parent object p. We then also find another surface, s_1^p, of the parent object p, from s_1^o of the object o. Once we compute such a pair of surfaces, we then compute relative distance d_1 and angle θ_1, as shown in Fig. 4.1(a).

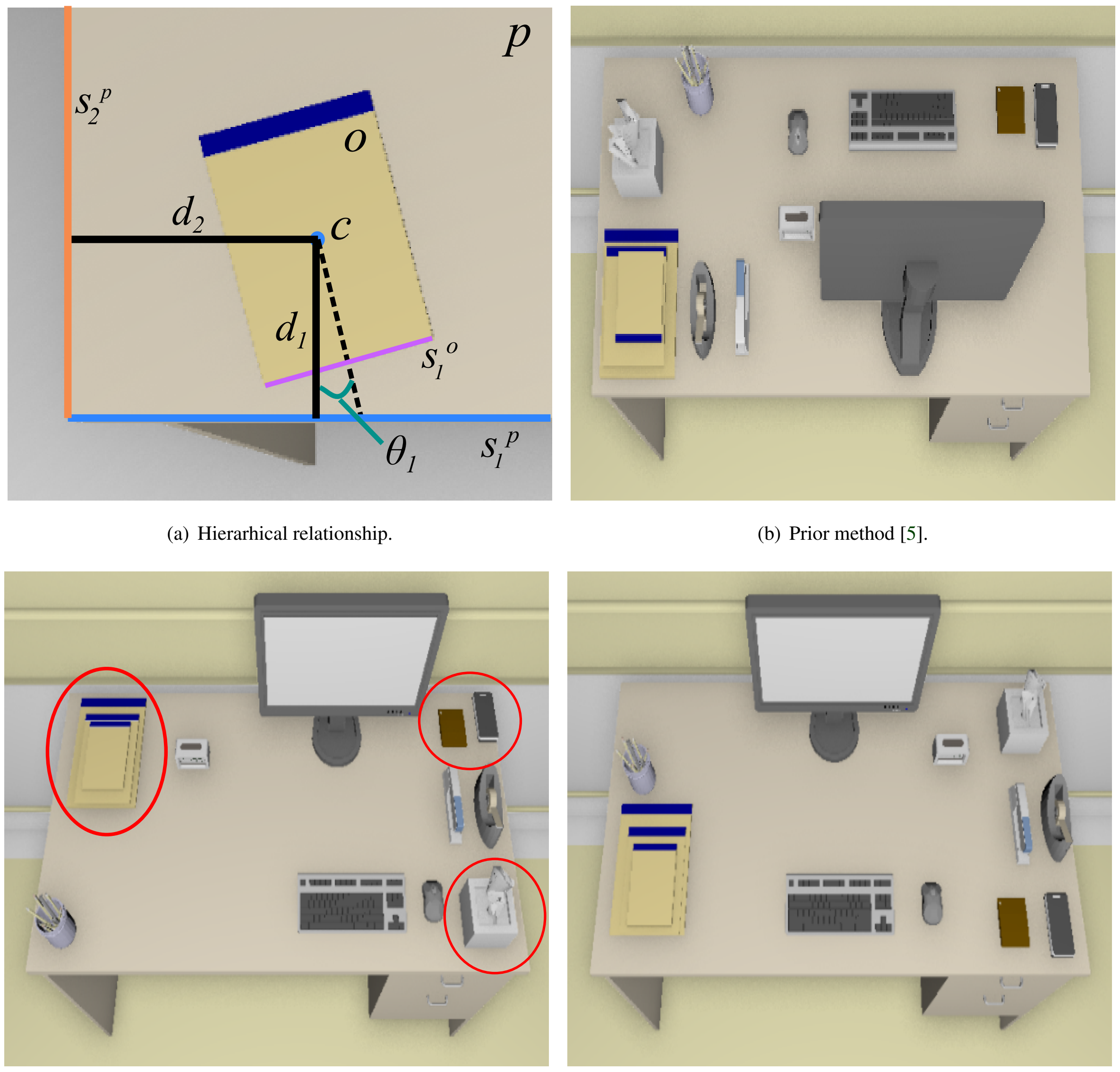
The work of automatic furniture arrangements [5] also uses d_1 and θ_1. When this approach is applied to our application directly, some objects such as monitor and keyboard are located on the opposite side as shown in Fig. 4.1(b). While furniture can play a defined role on any wall, an object often performs its role only in a specific location for the problem of the object arrangement. As a result, we additionally encode surface IDs [1], s_1^p and s_1^o, of objects realizing those particular d_1 and θ_1, unlike the prior method, and we utilize those informations during the layout optimization process.

The effect of having these surface IDs during the layout optimization process is also shown in Fig. 4.1(c). These objects seem to be neatly arranged, but some frequently used objects are still placed in a non-human friendly way; for instance, frequently used objects such as phones or notepads are placed closer to the wall. We want objects to be arranged according to positive examples rather than being arranged in various ways. Therefore, we additionally pick a side, s_2^p, between two or multiple sides connected to the surface s_1^o as the closest side to the object o. We then also compute relative distance, d_2, between s_2^p and the object o; we found that its relative angle is not helpful, given the angle θ_1. When these are added, the result is as shown in the Fig. 4.1(d). Note that the locations of the tissue box and cell phones are swapped for a more human-friendly way, by considering another surfaces s_2^p.

These distances from two different surfaces with the angle can well represent the relative location between the parent and child objects. One may think that we also need to encode the relative spatial location along the height, but we found that it is not necessary for our applications, since objects are stacked together for our application and thus their height can be implicitly encoded.

Finally, for objects that do not have the parent-child relationship, we encode the pairwise relationship representing the relative distance(d_{pair}) and angle(θ_{pair}) between those two objects.

[1] We compute each object to have its oriented box, and assign different IDs to its six different surface of the box.



(a) Hierarhical relationship.

(b) Prior method [5].

(c) Utilizing surface IDs.

(d) Utilizing the additional distance.

Figure 4.1: (a) The blue line is s_1^p chosen from the parent object p given an object o, and d_1 is the distance of the object center c to s_1^p. θ_1 is the orientation of the object between s_1^p and s_1^o. The orange line is s_2^p, and d_2 is an additional distance of the object center c to s_2^p. (b) to (d) show benefits of utilizing these informations additionally. (b) is the result of the prior work to arrange objects. (c) is the result of encoding and considering specific surface IDs, resulting in better arrangement of objects. In (d), we add the additional distance term (d_2) to make our arrangement closer to positive examples. This yields a more human-friendly arrangement of objects by making it easier for humans to reach frequently used objects such as a cellphone.

Our work also considers ergonomic factors of objects such as visibility and accessibility. Some objects such as monitors have a particular surface that should be visible to a human. We assign a viewing frustum to that particular surface to prevent other objects from covering it. Moreover, our work sets an accessible space for each object. We compute an accessible space for the robot's gripper for each object, so that the robot can understand the space that is required to hold an object.

4.2 Layout Computation

We use an optimization process to compute locations of objects using data obtained from the extraction process. Our optimization is based on simulated annealing [5], which is analogous to physical anneal process, to compute layout candidates that have low cost values given our cost functions. The simulated annealing approach has been known to compute the global minimum, under the assumption that its cooling stage is slow enough. Nonetheless, our work does not need the global optimal, but reasonably good solutions with better performance. As a result, we use a fast cooling step.

Specifically, we regard a cluttered level of a scene as temperature, i.e., a cost; more objects are cluttered away from positive examples, higher we give a higher cost. In the extraction process, we extracted various information, d_1, d_2, and θ_1, with particular surfaces and assigned viewing frustum and accessible space. To measure the cost of the current scene, we also extract these information from the current scene and measure how far the current value is away from these pre-extracted values; equations for our cost functions are based on the prior work [5].

The cost minimization process based on the simulated annealing works as follows: Most objects in our applications are placed on top of furniture or other objects. Therefore, we first check whether an object, o, in our target scene has its parent object, p , and if so, we place the object o on top of its parent object p by utilizing the extracted hierarchical relationship. Given that hierarchical positioning, we continue to move or rotate the object o until the cost evaluated from our cost function becomes small enough. In this approach, we may get stuck in local minima. To ameliorate this issue, we randomly swap objects and see whether we can reduce the overall cost measured from all the objects in the current scene layout.

Chapter 5. Task and Motion Planning

In this section, we explain how to use the computed layout on TMP using the priority layer. Before applying an arranged scene to task and motion planning, we must first define the state of the world in a symbolic form through the PDDL (Planning Domain Definition Language) [15] format.

PDDL consists of a domain and a problem. A domain describes predicates and a set of robot actions, and a problem sets the initial state and the goal state specifications for objects.

5.1 Definition of domain

For a robot to arrange objects, we first need to define predicates and three basic actions: pick, place, and move. Through these three actions, a robot can pick up an object and move it to the goal position of an object. These are action specifications for a robot to arrange objects:

```
Pick(p, o, $pose_1$, $pose_2$)
  Precon   EmptyGripper, At(o, $pose_2$)
           On(p, o), $\forall$o' $\neg$On(o, o')
           RobotAt($pose_1$), IsMP($pose_1$, $pose_2$)
  Effect   InGripper(o), $\neg$ EmptyGripper,
           At(o, $pose_2$), $\neg$ On(p, o)
Place(p, o, $pose_1$, $pose_2$)
  Precon   InGripper(o), $\neg$ EmptyGripper
           RobotAt($pose_1$), IsMP($pose_1$, $pose_2$)
  Effect   $\neg$ InGripper(o), EmptyGripper
           At(o, $pose_2$), On(p, o)
Move($pose_1$, $pose_2$)
  Precon.   RobotAt($pose_1$), IsMP($pose_1$, $pose_2$)
  effect    $\neg$ RobotAt($pose_1$), RobotAt($pose_2$)
```

Each specification consists of a set of preconditions for an action and a set of effects from taking the action. The predicates used to characterize the object arrangement are:

- *At(o, pose)*: *true* iff the object *o* has *pose*.
- *On(p, o)*: *true* iff the object *o* is on the parent *p*.
- *RobotAt(pose)*: *true* iff the robot has *pose*.
- *IsMP($pose_1$, $pose_2$)*: *true* iff there is a collision-free motion plan from $pose_1$ to $pose_2$.
- *InGripper(o)*: *true* iff the robot is holding the object *o*.

5.2 Defining a problem in the domain

Our method arranges objects by presenting a problem to a defined domain. The problem is constructed by expressing an automatically arranged scene as the goal state. Fortunately, the arranged scene already contains all the information for making symbols to define the goal state.

In the scene, the hierarchical relationship of objects and locations of objects are represented. Specifically, *At(o, pose)* can express the current state and the goal state through the position and orientation of the objects in the 3D scenes. *On(p, o)* is used to define the hierarchical relationship between parent and child objects p and o. In this way, the information of the scene can be used to make predicates in the problem.

5.3 Priority layer

In this section, we explain how to arrange objects efficiently using our priority layer.

We arrange objects using existing TMP planners HPN [1] and the work of Srivastava et al. [12]. Even though they successfully arrange objects to their target locations, we found that these well-known TMP planners have unnecessary actions. For example, HPN creates a sub-plan for each object, reducing the search depth. However, unnecessary actions occur because this method does not consider a cost to reach the goal of the sub-plan.

While considering costs to achieve the goal state from the current state is an effective way of computing optimal actions satisfying the goal, it can take too much time to predict such costs considering the distance traveled and search depth within the planner. Instead of predicting them within a planner, we propose to predict the distance traveled and search depth for each object outside the planner. This is the process of determining the order of arranging objects for efficient planning, instead of the process of creating a sub-plan [1] or planning in off-the-shelf task planner [12].

To improve the overall performance of computing a sequence of actions to move objects to their target locations, we estimate an expected cost of moving each object to its target location. We then pick the object with the lowest cost and compute its planning to reach its goal target object. We iterate this process within the priority layer, until all the objects are placed in their target goals.

The priority layer uses the information such as hierarchical relationships extracted from the 3D scene. Moreover, the 3D scene includes not only the information on the relationship between objects, but also the objects' configurations that facilitate collision checks.

We estimate the execution cost required to move objects in their initial position to the goal position using the scene information, specifically, the distance between a robot and objects, and the accessible space of the object. We use A* path planning to measure the distance required for a robot to approach each object at its current position. In addition, the accessible space uses the information that was used to construct the arrangement layout, to ensure that the robot has enough space to pick the object.

Our work measures the cost of moving an individual object through three processes: a cost of picking, a cost of placing, and the total cost combining two prior costs. First, we measure the cost of picking an object. To pick the object, the robot must be moved to the place where the object is located, and there should be no obstacle. The cost to pick an object, $cost_{pick}$, is defined as follows:

$$cost_{pick} = d_{init} + N_{obs}^{init} * W, \tag{5.1}$$

where d_{init} is the distance from the robot to the object computed by the A* algorithm, and N_{obs}^{init} is the number of obstacles that must be removed to pick the object. Obstacles of an object, o, include objects that overlap in the accessible space of the object o and object's children. W gives the average time of removing an obstacle through our simulation using PR2. It can be changed according to a robot.

Second, we define the cost of placing an object, o, as:

$$cost_{place} = \begin{cases} 0, & \text{if } N_{obs}^{goal} == 0, \\ d_{goal} + d_{gap} + N_{obs}^{goal} * W, & \text{otherwise,} \end{cases} \tag{5.2}$$

where d_{goal} is the distance from a robot to the goal pose of the object o, d_{gap} is the distance from the current pose of the object to its goal, and N_{obs}^{goal} is the number of obstacles that must be removed to place the object. The cost of placing the object is similar to the cost of picking it. However, if there is no obstacle in the target location, we set $cost_{place}$ to zero.

Finally, we check whether the parent of an object is furniture or has reached the goal position. When the parent p of an object o has not reached its target location, we set the cost of o to be infinitely high. The high cost guides object's parent p to be moved to its target location first. The total cost, $cost_{total}$, of each object is then defined as follows:

$$cost_{total} = \begin{cases} cost_{pick} + cost_{place}, & \text{if } p \text{ reached its goal,} \\ inf, & \text{otherwise.} \end{cases} \tag{5.3}$$

We pick the object with the lowest cost for the object arrangement. We then update the current scene and re-evaluate estimated costs with the updated scene information.

Chapter 6. Experiments and Results

We test our method and others on a machine that has 3.60GHz Intel i7-3820 CPU and 16GB RAM. We use Blender, a 3D graphics modeling tool, to represent and visualize objects in 3D scenes. 3D models used in our scenes were obtained from Google 3D warehouse and Fisher et al. [16]. Overall, we used 23 different 3D models to place furniture and objects in a room. We use the PR2 robot for arranging objects within ROS and Gazebo simulator. We also use A* path planning for moving the base of the PR2 robot and MoveIt! [17] for manipulating PR2 arms.

6.1 Results of layout computation

We compute an object layout based on information extracted from positive examples. The prior layout approach [5] prepared about five positive examples to synthesize other varying scenes for computer graphics applications. However, because our application requires a target layout of objects, our computed layouts of objects have to be consistent rather than diverse, and thus our method has additional parameters to specific locations of objects (Sec. 4.1). As a result, positive examples used in our work not only serve the basis for locations of objects, but also reflect user preferences. The habits of arranging objects vary across different persons, and a user can reflect his or her preference based on those positive examples.

We measured how long our layout computation process takes to get an object arrangement layout as a function of the number of objects. As the number of objects increases, the relationship of objects becomes more complicated, requiring more running time and iteration (Fig. 6.1). Because we randomly rotate and translate an object within the simulated annealing process, finding optimized locations in complex situations requires more iterations and, thus, takes longer running time.

While our layout computation is not real-time (e.g., less than 1 sec. or so), our layout computation enables a robot to intelligently arrange objects based on positive examples. This result is shown in the subsequent section.

6.2 Evaluations of the priority layer

Once we compute target layouts, any task and motion planners can be used to maneuver a robot to arrange objects in such target layouts. Specifically, we used two existing TMP planners: HPN [1] and combined task and motion planner (CTMP) [12] based on FF-planner [18]. In these widely used planners, we evaluated benefits of our priority layer by applying the layer to those two planners.

To compare the efficiency of different methods, we measured the total execution time of computing the goal layout and arranging objects to reach the goal by the PR2 robot in the simulation. We also evaluated the number of actions made by PR2 to reach the goal and the total distance traveled by PR2. In our test scenes, there are a bed, a bookshelf and a desk in the room, and objects are cluttered around those furniture. We experimented with a total of five scenes, each with different numbers of cluttered objects.

Fig. 6.2 shows the total running time of each planner in different tested scenes that have varying numbers of objects. The total running time of each planner includes the planning time of actions, As the number of cluttered objects increases, the running time of HPN and CTMP increases. Given this general trend, we were able to observe meaningful performance improvement by using our priority layer in both tested HPN and CTMP task planners.

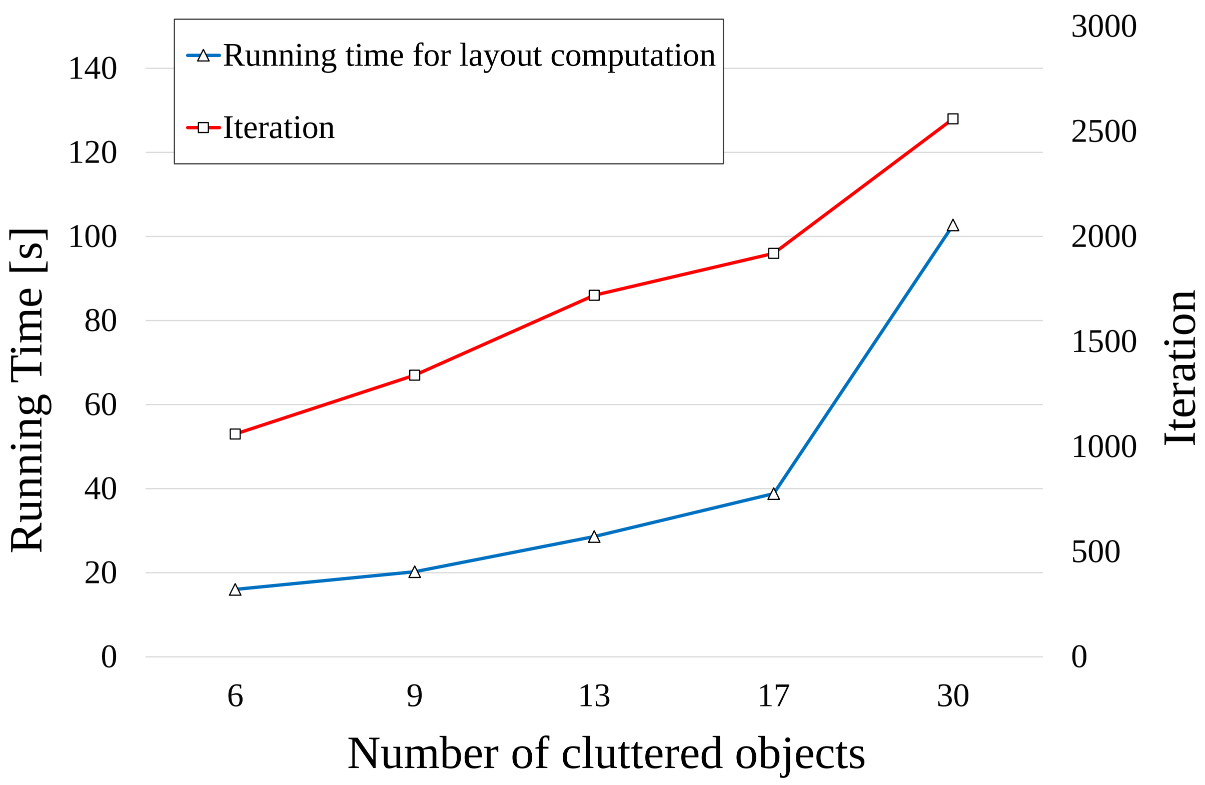


Figure 6.1: This figure shows the number of iterations and running time takes for layout computation, as a function of the number of objects.

This improvement is attributed mainly by reducing the number of actions, thanks to our estimated costs in the priority layer.

Table 6.1 shows the number of actions made and distance traveled by PR2 across different methods. It can be seen that the amount of time is reduced by our method due to the reducing of taken actions and the shortening of the distance traveled. Overall, our method yields 1.6 times and 2.0 times performance improvement on total running speed using HPN and CTMP respectively. As the number of cluttered objects increases, our method shows higher improvement over prior methods.

Fig. 6.3 shows the timing breakdown of the total running time. We can see that it takes a lot of time to execute actions such as move, pick and place with the PR2 robot, while planning (TMP) and our priority layer takes only a minor portion (less then 2.3% of the total time). The priority layer occupies a very small amount that is barely visible in the figure, but it significantly reduces the overall running time, as shown in Fig. 6.2. Fig. 6.4 shows an image sequence showing the process of arranging objects in one of our tested scenes.

Limitations. Our method showed meaningful results. Nonetheless, it has certain drawbacks. Our method selects and moves an object to its target location greedily according to costs estimated in the priority layer. This approach shows higher performance in our tested scenes, but can be stuck in a local minimum and may not reach the final goal, especially in extremely cluttered scenes. To avoid this issue, one can allow probabilistic selection based on the estimated costs within existing TMP planners. Nonetheless, our planning method inherits limitations of an existing TMP planner.

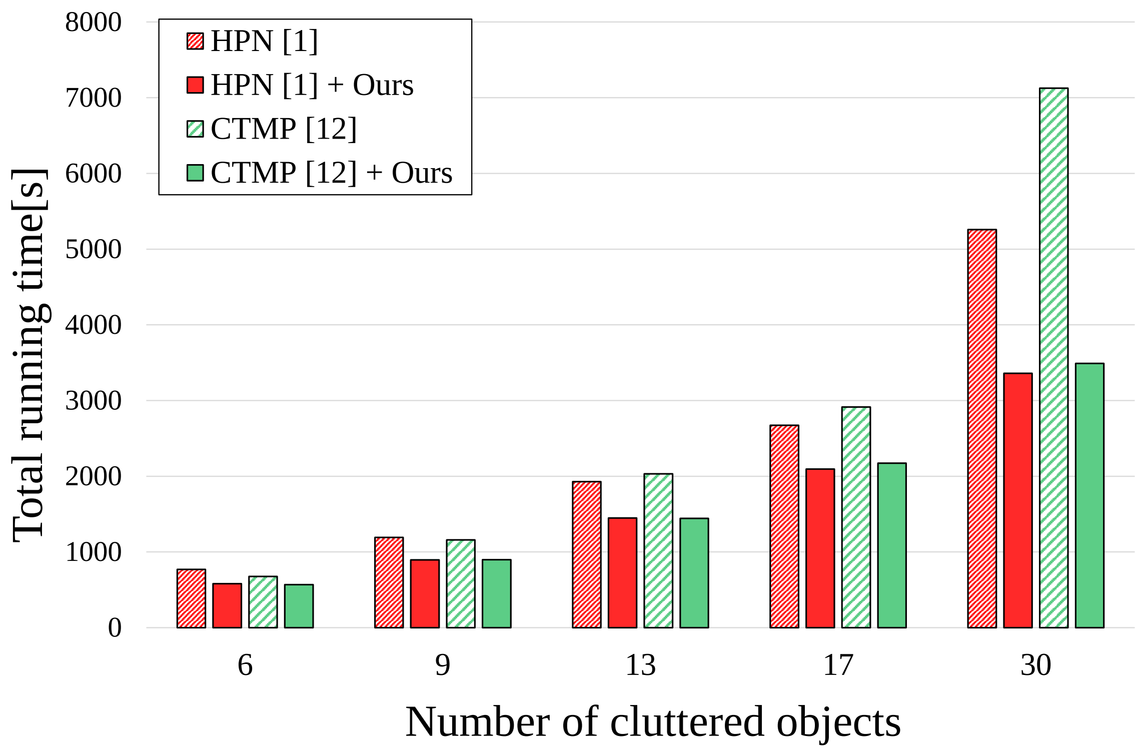


Figure 6.2: This figure shows the result of each planner's execution time. We can see that the execution time of two existing TMP algorithms is reduced robustly by applying our priority layer.

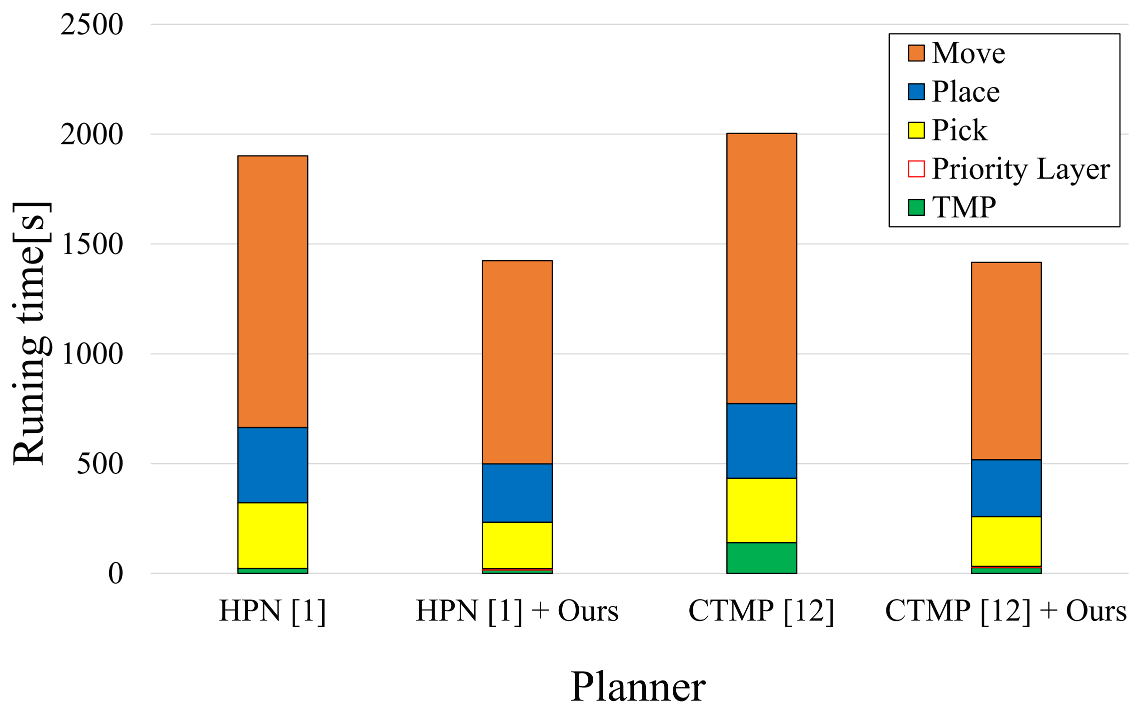
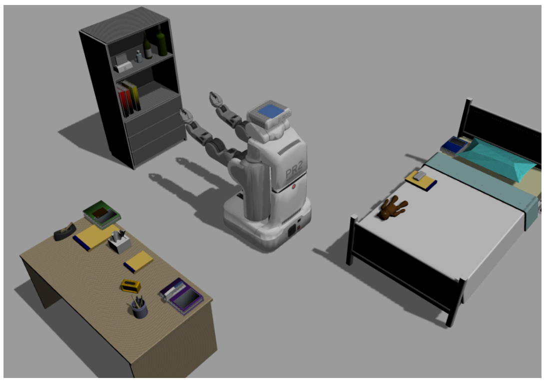
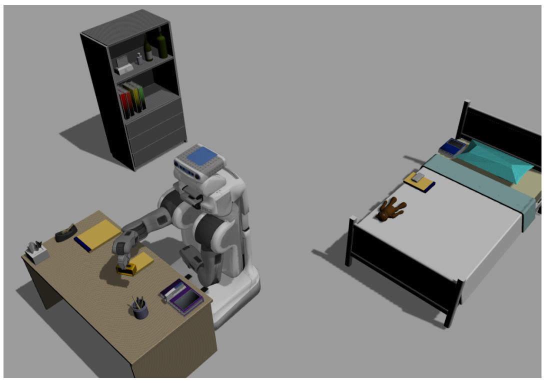


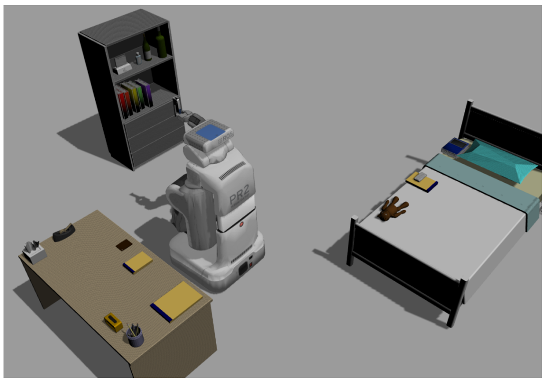
Figure 6.3: This figure shows the running time breakdown: TMP, Priority layer, Pick, Place and Move. This is reported from the scene containing 13 cluttered objects. The priority layer occupies a very small portion, but it significantly reduces the overall running time.



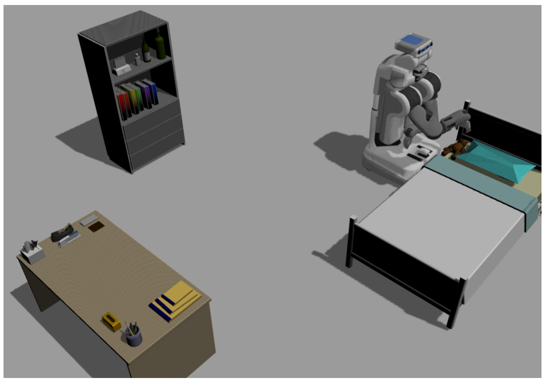
(a) Initial cluttered scene.



(b) Picking up the table clock.



(c) Carrying the stapler.



(d) Final, arranged scene.

Figure 6.4: This image sequence shows the process of arranging objects from a cluttered state (a) to an arranged state (d). (b) and (c) show intermediate states between (a) and (d). (b) shows the robot is picking up the table clock, and (c) shows the robot is carrying the stapler.

Table 6.1: We measured the total running time, number of actions, and travel distance averaged out from five independent tests for each scene. Experimental results show that our method decreases the number of actions and distance traveled. As a result, the running time of different different planners is reduced by using our method.

# of cluttered objects	planner	HPN [1]	HPN [1] + Ours	CTMP [12]	CTMP [12] + Ours
6	Running time [s]	770.88	581.85	677.33	567.85
	Actions	31.2	24.0	28.8	24.0
	Distances [m]	20.63	15.38	17.01	15.37
13	Running time [s]	1929.59	1449.44	2032.29	1444.74
	Actions	80.8	60.0	78.4	60.0
	Distances [m]	55.07	38.48	54.89	38.94
17	Running time [s]	2673.40	2095.39	2913.83	2173.78
	Actions	112.0	84.8	105.6	86.8
	Distances [m]	67.43	57.89	75.84	59.47
30	Running time [s]	5258.73	3359.56	7125.56	3489.31
	Actions	227.2	141.6	193.6	141.6
	Distances [m]	148.57	87.22	132.74	88.57

Chapter 7. Conclusion

In this paper, we have introduced a method to automatically arrange cluttered objects using task and motion planning. To guide a robot where to place objects, we synthesized a target layout automatically by considering hierarchical and pairwise relationships between objects. To reach the layout, our method plans a series of feasible actions using the well-known task and motion planner such as HPN and CTMP. In addition, we proposed a priority layer to arrange objects efficiently by estimating costs of moving objects and processing those objects in a greedy manner, resulting in the performance improvement on the overall running speed to arrange objects.

There are many future directions in automatically arranging objects using a robot. In our work, we have assumed that a robot can capture its environment using a sensor or ideally grab and move objects despite the fact that these are essential parts to let a real robot arrange objects. Recently, many works related to these techniques have been studied in various fields. Therefore, we would like to test our method on a real robot using these techniques for arranging objects.

Bibliography

[1] Leslie Pack Kaelbling and Tomás Lozano-Pérez, "Hierarchical task and motion planning in the now", in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1470–1477.

[2] Yun Jiang, Marcus Lim, Changxi Zheng, and Ashutosh Saxena, "Learning to place new objects in a scene", *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1021–1043, 2012.

[3] Yun Jiang, Marcus Lim, and Ashutosh Saxena, "Learning object arrangements in 3d scenes using human context", in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, John Langford and Joelle Pineau, Eds., New York, NY, USA, 2012, pp. 1543–1550, ACM.

[4] Nichola Abdo, Cyrill Stachniss, Luciano Spinello, and Wolfram Burgard, "Robot, organize my shelves! tidying up objects by predicting user preferences", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1557–1564.

[5] Lap Fai Yu, Sai Kit Yeung, Chi Keung Tang, Demetri Terzopoulos, Tony F Chan, and Stanley J Osher, "Make it home: automatic optimization of furniture arrangement", *ACM Transactions on Graphics (TOG)-Proceedings of ACM SIGGRAPH 2011, v. 30, no. 4, July 2011, article no. 86*, 2011.

[6] Mario Bollini, Jennifer Barry, and Daniela Rus, "Bakebot: Baking cookies with the pr2", in *The PR2 workshop: results, challenges and lessons learned in advancing robots with a common platform, IROS*, 2011.

[7] Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth, "Robotic roommates making pancakes", in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*. IEEE, 2011, pp. 529–536.

[8] Mikkel Rath Pedersen and Volker Krüger, "Automated planning of industrial logistics on a skill-equipped robot", in *Workshop on Task Planning for Intelligent Robots in Service and Manufacturing*, 2015, p. 1.

[9] Lavindra de Silva, Amit Kumar Pandey, and Rachid Alami, "An interface for interleaved symbolic-geometric planning and backtracking", in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 232–239.

[10] Lavindra De Silva, Mamoun Gharbi, Amit Kumar Pandey, and Rachid Alami, "A new approach to combined symbolic-geometric backtracking in the context of human-robot interaction", in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3757–3763.

[11] Kutluhan Erol, James Hendler, and Dana S Nau, "Htn planning: Complexity and expressivity", in *AAAI*, 1994, vol. 94, pp. 1123–1128.

[12] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer", in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 639–646.

[13] Chongjie Zhang and Julie A Shah, "Co-optimizing task and motion planning", in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4750–4756.

[14] Christian Dornhege and Andreas Hertle, “Integrated symbolic planning in the tidyup-robot project.”, in *AAAI Spring Symposium: Designing Intelligent Robots*, 2013.

[15] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins, “Pddl-the planning domain definition language”, 1998.

[16] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan, “Example-based synthesis of 3d object arrangements”, *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, pp. 135, 2012.

[17] Ioan A Sucan and Sachin Chitta, “Moveit!”, *Online at http://moveit. ros. org*, 2013.

[18] Jörg Hoffmann, “Ff: The fast-forward planning system”, *AI magazine*, vol. 22, no. 3, pp. 57, 2001.

Acknowledgments in Korean

석사 과정 동안 이 논문을 작성할 수 있도록 도움 주신 모든 분들께 감사드립니다. 먼저, 부족한 저를 좋은 방향으로 이끌어 주시고 좋은 가르침 주신 윤성의 교수님께 감사드립니다. 저의 지도교수님이 돼주셨기에 제가 연구에 집중하며 성장해 갈 수 있었습니다.

처음 연구실에 왔을 때부터 논문을 작성할 때까지 많은 조언과 도움 주었던 용선군, 항상 친절하게 많은 것을 알려주시는 동혁이형, 고민 있을 때 저를 잘 이해해주시는 인규형, MT 준비뿐만 아니라 로봇 팀에서 많은 도움을 주고 있는 희찬군, 가장 편한 친구이자 모르는 것을 물어보면 좋은 해결책을 제시해주는 명배, 좋은 조언해주시는 랩장 수민이누나, 항상 착실한 태영군, 착하고 아는 것 많은 영기군, 논문 작업할 때 많은 도움을 주었던 재원군, 앞으로 더 알아갈 랩실 막내 재윤군, 모두에게 많은 감사드립니다. 그리고 지금은 고향에서 행복한 생활하고 있는 반한국인 Pio를 비롯하여 졸업하시기 전과 후에도 좋은 조언해주시는 SGLAB 선배님들이신 재필이형, 웅직이형, 윤석이형, 현철이형, 정수형, 재형군, 병윤군께 감사드립니다.

마지막으로 저를 항상 믿어주고 무한한 응원해주시는 부모님과 동생들에게 감사의 인사를 전합니다. 앞으로 더욱 노력하는 모습으로 자랑스러운 아들이 되도록 노력하겠습니다. 사랑합니다.

Curriculum Vitae in Korean

이 름:	강 민 철
생 년 월 일:	1990년 05월 02일
주 소:	대전 유성구 대학로 291 한국과학기술원 전산학부 3443호
전 자 주 소:	kmc0502@kaist.ac.kr

학 력

2006. 3. – 2009. 2.	천안북일고등학교
2009. 3. – 2015. 2.	충남대학교 메카트로닉스 공학과