Journal of Korea Robotics Society (2019) 14(2):104-113
https://doi.org/10.7746/jkros.2019.14.2.104 ISSN: 1975-6291 / eISSN: 2287-3961

복잡 환경에서 가로막힌 물체 잡기를 위한 작업-모션 계획의 연계

Task and Motion Planning for Grasping Obstructed Object in Cluttered Environment

이 석 준† · 김 인 철[1]
Seokjun Lee†, Incheol Kim[1]

Abstract: Object manipulation in cluttered environments remains an open hard problem. In cluttered environments, grasping objects often fails for various reasons. This paper proposes a novel task and motion planning scheme to grasp objects obstructed by other objects in cluttered environments. Task and motion planning (TAMP) aims to generate a sequence of task-level actions where its feasibility is verified in the motion space. The proposed scheme contains an open-loop consisting of three distinct phases: 1) Generation of a task-level skeleton plan with pose references, 2) Instantiation of pose references by motion-level search, and 3) Re-planning task based on the updated state description. By conducting experiments with simulated robots, we show the high efficiency of our scheme.

Keywords: Object Manipulation, Task and Motion Planning, Grasping Obstructed Objects

1. 서 론

최근 자동화된 로봇의 계획과 실행을 위해 인공지능과 로보틱스가 융합된 연구들이 활발히 진행되고 있다. 그 중에서도 특히, 로봇이 스스로 목표를 달성하기 위한 행위들을 결정하고 실행하기 위한 작업-모션 계획의 연계(Task And Motion Planning, TAMP)는 지속적으로 연구되어 오고 있는 핵심 연구 분야이다[1-8]. 작업-모션 계획의 연계는 기하학적 물리 공간에서 모션으로 실현 가능한 작업 계획을 자동으로 수립하는 것을 목표로 한다. 일반적인 작업-모션 계획의 연계는 기호로 정의된 추상화된 행위 모델(symbolic abstract action model)을 토대로 목표(goal)에 도달 가능한 행위들의 연속(sequence of actions)인 하나의 작업 계획을 세우는 작업 계획 생성(task planning)[9-11] 과정과, 실제 물리 공간에서 모션으로 작업 계획을 실현할 수 있을지 그 실행 가능성(feasibility)을 확인하는 모션 계획 생성(motion planning)[12-14] 과정의 결합으로 이루어진다. 본 논문에서는 복잡 환경(cluttered environment)에서 물체 조작(object manipulation)을 위한 로봇 작업-모션 계획의 연계 방법을 제시하고자 한다. 특히, 복잡 환경에서는 조작 대상인 목표 물체(target object)들이 또 다른 물체들에 의해 접근 경로들이 막혀있는 상황이 자주 발생하기 때문에, 이를 효과적으로 다룰 수 있는 작업-모션 계획의 연계 방법을 제시하고자 한다.

작업 계획 생성(task planning)[9-11]은 추상화된 행위 모델들을 이용하여 행위들 간의 원인-결과 관계를 분석함으로써, 작업 초기 상태로부터 목표 상태에 도달할 수 있는 연속된 행위들을 자동으로 생성한다[15]. 일반적으로 작업 계획 생성에서는 실제 물리 세계의 기하학적, 역학적 세부 사항은 생략된 추상화된 기호적 상태 표현과 행위 모델에 의존하여 계획을 생성한다. 따라서 이러한 결과로 얻어지는 작업 계획들은 실제 물리 세계에서의 로봇 모션으로 실행 가능한지 보장할 수 없다는 한계점이 있다. 따라서 이러한 작업 계획들은 실행에 실패할 가능성이 매우 높다. 반면에, 모션 계획 생성(motion planning)은 실제 물리 세계의 기하학적, 역학적 제약을 고려하면서 로봇 팔이나 손을 목표하는 특정 포즈까지 이동할 수 있는 경로(trajectory)를 계획하는 일을 수행한다. 작업 계획 생성은 이산화, 기호화된 추상 공간에서 계획을 생성하는 일이라면, 모션 계획 생성은 연속된 정량 공간(continuous metric

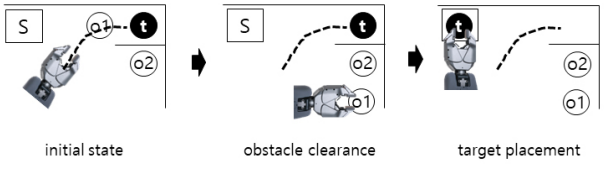
Received : Dec. 13. 2018; Revised : Jan. 17. 2019; Accepted : Jan. 25. 2019
※ This work was supported by the Technology Innovation Program (or Industrial Strategic Technology Development Program (10077538, Development of manipulation technologies in social contexts for human-care service robots) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea)
† Student, Corresponding author: Computer Science, Kyonggi University, Suwon, Korea (20171102101@kyonggi.ac.kr)
1. Full Professor, Computer Science, Kyonggi University, Suwon, Korea (kic@kyonggi.ac.kr)

space)에서 계획을 생성하는 일로 볼 수 있다. 즉, 모션 계획(motion plan)이 하나의 추상 행위를 실제로 물리 세계에서 실현하기 위한 구체적인 계획이라면, 작업 계획(task plan)은 주어진 작업 목표를 달성할 수 있는 여러 복수 개의 추상 행위들의 조합으로 볼 수 있다. 따라서 양자 간에는 서로 특화된 정보와 기능이 존재한다고 볼 수 있다.

이러한 모션 계획 생성의 특성을 작업 계획 생성 단계에서 상호 보완적으로 활용하여 미리 실제 물리 세계에서의 실현 가능성을 보장할 수 있는 작업 계획을 생성함으로써, 실행 실패 가능성이 낮은 작업 계획을 마련하자는 것이 작업-모션 계획 연계의 목표라고 할 수 있다[12-14]. 하지만 두 계획 생성 방법은 각각의 데이터 또는 지식의 표현 레벨과 추론 방식의 격차가 매우 크기 때문에 이 둘을 연계하는 것을 쉽지 않은 일이다. 서로 다른 두 계획 생성 방법을 효과적으로 연계하기 위해서는 1) 올바른 작업 계획 생성을 위한 작업 상태 표현과 행위 모델링, 2) 모션 계획 생성을 통한 작업 계획의 구체화와 실행 가능성 검증, 3) 실행 가능성 검증 실패 시 새로운 작업 계획을 수립하는 재계획(replanning) 등의 세부 기술들이 요구된다. 특히, 작업 계획 생성과 모션 계획 생성은 각각 방대한 탐색 공간을 가지기 때문에 연계 과정에서 적용되는 세부 기술들이 이러한 탐색 공간을 줄일 수 있는 방향으로 모색되어야 한다.

복잡 환경에서 물체 조작 작업을 계획할 때 조작 대상인 목표 물체까지의 모든 접근 경로들이 다른 물체들에 의해 가로막혀 있는 경우, 장애물들을 처리하기 위해서는 새로운 요구사항들이 발생한다. 따라서 새로운 요구 사항들이 연계 방법에 반영되지 않는다면 이 문제를 위한 작업 계획은 성공적으로 수립될 수 없을 것이다. [Fig. 1]은 이러한 상황에서 작업-모션 계획의 연계 예시를 보여준다.

[Fig. 1]은 로봇이 2차원 공간에서 다른 물체 o1에 의해 막혀 있는 목표 물체 t를 잡아 특정 영역 S에 배치해야 하는 문제를 보인다. 점선은 로봇이 목표 물체 t를 잡으러 가기 위한 하나의 이동 경로를 나타낸다. 이 이동 경로 상에는 물체 o1이 존재하기 때문에 로봇은 물체 o1을 치워야만 목표 물체 t를 잡을 수 있다. 이러한 [Fig. 1]의 상황에서 로봇이 실행 가능한 작업 계획을 성공적으로 수립하기 위해서는 다음 두 가지 질문에 답을 할 수 있어야 한다. 첫 번째 질문은 '어떤 물체가 어떤 경로에서 목표 물체를 가로막고 있는가?'이다. 두 번째 질문은 '장애물을 어느 곳으로 치워야 목표 물체를 잡을 수 있는가?'이다. 첫 번째 질문은 추상된 작업 상태 표현에서는 알 수 없는 접근 경로 상에 물체들 간의 충돌 여부 검사, 두 번째 질문은 작업 초기 상태나 목표 상태에는 구체적으로 주어지지 않은 물체들의 포즈를 결정하는 일을 요구한다. 결국 이 두 가지 질문에 관한 답은 모두 모션 계획 생성 과정을 거쳐 얻을 수 밖에 없다.



[Fig. 1] An example of grasping obstructed object

본 논문에서는 [Fig. 1]과 같이 다른 물체들에 의해 가로막혀 있는 목표 물체를 잡기 위한 작업-모션 계획 간의 연계 방법을 제안한다. 이를 위해서는 작업 계획 생성과 모션 계획 생성 과정에서 발생하는 새로운 요구사항들을 만족시킬 수 있는 연계가 이루어져야 한다. 먼저 작업 계획 생성 관점에서 장애물을 치우기 위한 작업 계획을 생성하기 위해서는 어떤 물체가 장애물인지 또 그 장애물이 어떤 물체를 어떤 경로 상에서 가로막고 있는지에 대한 작업 상태를 제공해주어야 한다. 또한, 장애물 치우기는 일반적인 물체 내려놓기와 달리 더 복잡한 제약조건들을 가지고 있기 때문에 장애물을 치우기 위한 새로운 행위 모델링이 필요하다. 본 논문에서 제안하는 연계 방법은 장애물 치우기를 위한 새로운 작업 상태 서술자들과 행위 모델을 포함한다. 다음으로 모션 계획 생성 관점에서, 장애물 치우기의 실현 가능성을 검증하기 위해 모션 목표인 치움 위치를 제공해주어야 한다. 치움 위치는 단순한 내려놓음 위치보다 더 많은 제약조건들을 가지기 때문에 이를 얻어내기 위한 새로운 알고리즘이 필요하다. 예컨대, 목표 물체의 위치와 목표 물체를 잡기 위한 경로를 고려하고 미래에 치워야 할 다른 장애물들까지 고려해야 하는 등 매우 복잡한 제약조건이 있을 수 있다. 본 논문에서 제안하는 연계 방법은 물체 잡기, 내려놓기 등을 위한 기본적인 목표 포즈뿐만 아니라 장애물을 치우기 위한 목표 위치를 얻어내는 포즈 생성 방법을 포함한다. 한편, 초기에 진행되는 작업 계획 생성은 "장애물이 목표 물체를 가로막고 있다"라는 모션 레벨의 작업 상태들을 모르고 시작한다. 따라서 연계 과정에서 모션 계획 생성을 통해 새롭게 알게 된 모션 레벨의 작업 상태들을 파악하고 이를 작업 계획 생성에 알려줌으로써 올바른 작업 계획이 다시 생성될 수 있도록 유도해야 한다. 본 논문에서 제안하는 연계 방법은 이러한 필연적인 재계획 방법을 포함한다. 본 논문에서 제안하는 연계 방법의 높은 효율성을 확인하기 위해 다수의 물체들이 존재하는 복잡 환경에서 시뮬레이션 로봇을 이용한 실험들을 진행하고 그 내용을 소개한다.

2. 관련 연구

2.1 작업 계획 생성

작업 계획 생성(task planning)[9-11]은 주어진 초기 상태로부터 목표 상태에 도달하기 위한 연속된 행위들을 자동으로

action	*PickUp*
param	*obj gripper*
precond	*empty (gripper)*
effect	*graspedBy (obj, gripper), ¬empty (gripper)*

[Fig. 2] An action specification for pick up

생성하기 위한 원인 귀속 추론(causal reasoning) 방법[15]이다. 따라서 주어지는 모든 상태와 행위들은 기호적(symbolic)으로 표현되고 로봇이 해야 할 연속된 행위들을 논리적으로 추론하고 이해할 수 있다. 작업 계획 생성 문제는 $< A, s_0, g >$로 정의된다. 여기서 A는 행위들의 집합, s_0는 초기 상태, g는 목표 조건을 나타낸다. 위의 [Fig. 2]는 대표적인 작업 계획 생성 언어인 PDDL (Planning Domain Definition Language)[10]의 행위 명세 예시를 보여준다.

행위의 전-조건(precondition), 효과(effect)는 상태 서술자들의 논리곱(conjunctive)으로 구성되어 있다. 목표 조건을 만족시키기 위해 초기 상태에서 전-조건을 만족시키는 행위를 선택하고 그 효과로부터 상태 변화를 일으킨다. 이 과정은 목표 조건을 만족시킬 때까지 반복되고 목표 조건을 만족시키는 연속된 행위들을 찾아간다. 하지만 이러한 연속된 행위들은 추상화된 기호 지식이기 때문에 실행 가능성(feasibility)을 보장하지 않는다. 작업 계획 생성이 가지는 이러한 한계점은 모션 계획 생성(motion planning)으로부터 보완될 수 있다.

2.2 모션 계획 생성

모션 계획 생성[12-14]은 물리적인 실행 관점에서 각각의 행위들에 대한 실행 가능성을 확인할 수 있다. 모션 계획 생성은 운동학(kinematics)과 기하학(geometry)을 기초로 이동 몸체(mobile base), 다관절 팔(multi joint arm) 등의 현재 포즈로부터 목표 포즈까지의 이동 경로(trajectory)를 생성한다. 모션 계획 문제는 $< C, f, p_0, p_t >$로 정의된다. 여기서 C는 로봇의 설정 공간(configuration space), f는 충돌 함수, p_0와 p_t는 각각 초기 포즈와 목표 포즈이다. 모션 계획 생성은 C안에 존재하는 p_0와 p_t에 대해, p_0에서 p_t에 도달하기 위한 경로를 찾아낸다. 이때 f는 경로의 중간 지점(way point)들에서 충돌이 발생하는지 검사한다.

2.3 작업-모션 계획의 연계

이 두 계획 생성 방법을 서로 상호 보완적으로 연계하여 실행 가능한 계획을 얻어내는 것이 작업-모션 계획의 목표라고 할 수 있다. 작업-모션 계획 간의 연계는 크게 두 가지로 나눌 수 있다. 첫 번째는 작업 계획을 먼저 생성하고 각각의 행위에 대해 실행 가능성을 확인하는 방법이고, 두 번째는 작업 계획을 생성하는 과정에서 행위 하나가 결정될 때마다 그 행위의 실행 가능성을 확인하는 방법이다. 첫 번째 방법은 실행 가능성을 확인하기 전에 모든 행위들에서 찾아야 하는 목표 포즈, 경로들 간의 상관관계를 고려할 수 있기 때문에 모션 계획 생성의 검색 공간을 크게 줄일 수 있다는 장점이 있다[2]. 이와 같은 이유로 논문에서 제안하는 연계 방법은 첫 번째 방법을 채택하였다.

첫 번째 방법을 기초로 한 기존 연구들로는 Srivastava[1]과 Garret[2]의 연구가 있다. Srivastava의 연구에서는 생성된 작업 계획의 행위들을 순차적으로 방문하면서 실행 가능성을 확인한다. 만약 그 과정에서 모션 계획 생성이 실패하면 실패에 대한 새로운 정보로부터 작업 계획을 다시 생성하는 연계 방법을 제안하였다. 이 방법은 실패가 발생하면 이전에 성공했던 행위들을 보관하고 실패한 행위의 시점부터 작업 계획을 다시 생성해나간다. 작업 계획을 다시 생성할 때는 모션 계획기로부터 얻어낸 실패에 대한 정보를 상태 서술자들로 변환하여 작업 계획 생성을 위한 초기 상태를 재구성한다. 예컨대, 물체1이 물체2를 가로막고 있어 물체2를 잡을 수 없다는 사실을 모션 계획기로부터 얻었다면, “물체1이 물체2를 막고 있다(obstruct object1 object2)”라는 상태 서술자를 생성한다. 따라서 작업 계획 생성기는 행위의 장애물에 대한 상태 서술자들을 고려하여 장애물을 치우고 목표 물체를 잡는 작업 계획을 다시 생성할 수 있다. 하지만 Srivastava의 연구에서는 실패 처리를 위한 일반적인 연계 방법에 무게 중심을 두었기 때문에 장애물을 처리하기 위한 상태 서술자, 행위 명세 등의 구체적인 도메인 지식이 부족하고 장애물을 치움 위치를 결정하기 위한 포즈 생성 알고리즘이 존재하지 않아 한 번 치운 장애물을 다시 치워야 하는 등의 효율적이지 못한 계획이 생성된다. Garret은 모든 행위들에서 찾아야하는 목표 포즈, 이동 경로들 간의 상관관계를 토대로 탐색 우선순위를 재정렬하여 전체 탐색 공간을 줄이는 방법을 제안하였다. 탐색의 우선 순위는 목표 포즈, 역운동학 그리고 이동 경로 순이다. 하지만 Garret의 연구는 장애물을 고려하지 않은 일반적인 물체 잡기 및 옮기기에만 초점을 두었고 연계 과정에서 실패 처리를 위한 연계 방안은 제시되어 있지 않다.

두 번째 방법을 기초로 한 연구들로는 Bidot[3], Ferrer-Mestres[4], Kaelbling[5]의 연구가 있다. Bidot은 기하 역추적(geometric backtracking)에서 발생하는 방대한 탐색 공간을 줄이기 위한 잡기 포즈(grasp pose), 배치 포즈(placement pose) 등의 다양한 제약조건들을 제안하였다. 하지만 Bidot의 연구에서는 장애물 치움 포즈(clearance pose) 등과 같이 장애물과 관련된 제약조건을 제시하지는 않았다. Ferrer-Mestres는 함수형 STRIPS를 이용하여 작업 계획 생성 과정에서 모션 계획이 간삽(interleaving)될 수 있는 새로운 연계 구현 방법을 제안하였다.

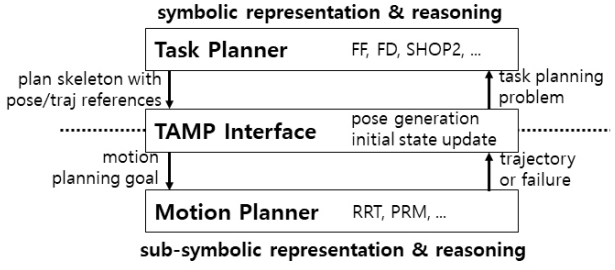
모션 계획 생성은 몸체 이동 포즈, 잡기 포즈, 배치 포즈 등과 관련된 상태 서술자들을 검증할 때 호출된다. 하지만 Ferre-Mestres는 두 번째 방법을 구현하기 위한 하나의 방법론을 제시하였을 뿐 장애물 처리를 위한 연계 방법은 포함하고 있지 않다. Kaelbling의 연구에서는 계층적 작업 네트워크(Hierarchical Task Network)[11] 기반의 작업 계획 생성 도중에 모션 계획 생성이 간삽되는 연계 방법을 제안하였다. 또한, 장애물 치움 위치 생성을 포함하는 다양한 포즈 생성을 위한 기하학 제안 함수(geometric suggestion function)를 제시하였다. 하지만 Kaelbling의 연구에서 제안하는 장애물 치움 위치 생성 방법은 미래에 치워야 할 다른 장애물들을 고려하지 않는다는 등의 한계가 있다. 아울러 이 세 연구 모두 장애물 처리를 위해 필수적인 실패 처리를 위한 연계 방안은 제시되어 있지 않다는 한계가 있다.

두 방법을 모두 적용해본 연구들로는 Lagriffoul[6]의 연구가 있다. Lagriffoul의 연구에서는 작업-모션 계획 간의 연계 과정에서 발생하는 탐색 공간을 탐색 트리로 표현하고 탐색 트리의 깊이를 H라 할 때 깊이 D만큼은 첫 번째 방법으로, 나머지 H-D만큼은 두 번째 방법으로 실행 가능한 계획을 찾는 방법을 제안했다. Lagriffoul는 실험을 통해 최소의 탐색 공간을 가지는 최적의 깊이 D가 존재하는 것을 밝혔다. 하지만 여전히 장애물 처리를 위한 연계 방안은 제시되어 있지 않았다.

3. 작업-모션 계획의 연계

작업 계획 생성과 모션 계획 생성은 이미 많이 연구되어 왔고 잘 구현되어 있는 알고리즘들이 존재한다. 작업-모션 계획 간의 연계 목표는 이들을 잘 연계하여 실행 가능한 작업 계획을 얻어내는 것이다. 두 계획 생성 방법은 각각의 데이터 또는 지식의 표현 레벨과 추론 방식의 격차가 매우 크기 때문에 이들의 중간에서 인터페이스 역할을 해주어야 한다. [Fig. 3]은 작업-모션 계획 간의 연계를 통해 실행 가능한 작업 계획을 생성하는 과정을 보여준다.

[Fig. 3]은 작업-모션 계획 간의 연계를 통해 실행 가능한 작업 계획을 생성하는 과정을 보여준다. [Fig. 3]은 크게 작업 계획 생성기(task planner), 작업-모션 계획 인터페이스(TAMP interface), 모션 계획 생성기(motion planner)로 구성된다. 먼저, 작업-모션 계획 인터페이스는 초기 상태, 목표 조건, 행위 명세를 포함하는 작업 계획 문제를 생성하고 작업 계획 생성기를 호출한다. 작업 계획 생성기는 이를 전달받아 작업 계획을 생성한다. 생성된 작업 계획은 실행 가능성을 검증받아야 하기 때문에 포즈와 이동 경로에 대한 참조들을 포함하고 있다. 참조에 대한 설명은 3.2 절에서 상세히 설명한다. 다음으로 작업-모션 계획 인터페이스는 실행 가능성을 검증하기 위해 작업 계획에 포함된 참조들에 대한 실제 값이 존재하는지 찾아야 된다. 이를 위해 작업-모션 계획 인터페이스는 각 행위의 목표 포즈를 직접 생성하고 모션 계획 생성기를 호출한다. 모션 계획 생성기는 이를 전달받아 목표 포즈까지의 이동 경로를 생성하거나 이동 경로를 생성할 수 없을 경우 그 실패 원인을 분석하여 알려준다. 작업-모션 계획 인터페이스는 이동 경로가 존재할 경우 실행 가능한 계획을 생성하고 실패가 발생한 경우 그 원인으로부터 작업 계획 생성 문제를 다시 생성한 뒤 재계획될 수 있도록 전체 과정을 반복한다.



[Fig. 3] Task and motion planning framework

3.1 연계 알고리즘

[Algorithm 1]은 작업-모션 계획 간의 연계 알고리즘을 나타낸다. 알고리즘의 입력은 행위들의 집합 A, 초기 상태 i 그리고 목표 조건 g이다. 먼저, [Algorithm 1]은 입력으로부터 작업 계획 T를 생성한다. 그런 다음 작업 계획의 각 행위들을 순차적으로 방문하면서 행위들의 모션 또는 이동 경로 t를 생성한다. t를 생성하기 위해서는 현재 포즈 p_0와 목표 포즈들의 후보군

[Algorithm 1] TAMP Algorithm

Input: set of actions A, initial state s_0, goal condition g
Output: feasible plan F
$F \leftarrow$ none
step $i \leftarrow 1$
while s_0 is not g
  task plan $T \leftarrow$ callTaskPlanning (A, s_0, g)
  for action α_i **in** T
    initial pose p_0, set of target poses $P \leftarrow$ generatePoses (α_i)
    trajectory t, error $e \leftarrow$ callMotionPlanning (p_0, P)
    if e is none
      bindReferences (α_i, p_0, p_t, t)
      $F \leftarrow F + \alpha_i$
      $s_0 \leftarrow$ updateInitialStateFromAction (s_0, α_i)
    else
      $s_0 \leftarrow$ updateInitialStateFromError (s_0, e)
      break
    end if
    $i \leftarrow i+1$
  end for
end while

집합 P를 생성하고 이를 모션 계획 생성의 입력으로 전달한다. 경로가 생성되면 행위의 참조들에 값을 연결하고 이 행위를 실행 가능한 계획 리스트 F에 저장한다. 행위를 저장한 뒤에는 행위의 효과를 적용하여 초기 상태를 갱신하고 다음 행위를 방문하여 동일한 과정을 반복한다. 만약 장애물에 의해 목표 포즈나 이동 경로 생성이 실패하면 그 원인을 상태 서술자들로 만들어 초기 상태를 갱신하고 행위 방문을 중단 시킨다. 행위 방문이 중단되면 갱신된 초기 상태로부터 다시 작업 계획을 생성하고 행위들을 순차적으로 방문한다. 이때 실행 가능성이 검증된 행위들은 F에 이어서 저장된다.

3.2 참조 기반의 작업 계획 생성

본 논문에서는 작업 계획 생성을 모두 마친 후에 순차적으로 각 행위들의 실행 가능성을 검증해나간다. 따라서 작업-모션 계획 연계 과정 중에 작업 계획 생성이 가장 먼저 선행되어야 한다. 하지만 기존의 작업 계획을 구성하는 매개변수와 상태 서술자들은 모두 기호적으로 표현되기 때문에 실행 가능성을 검증하기 위해서는 모션 레벨의 비-기호적인 매개변수와 상태 서술자들을 포함하는 새로운 유형의 도메인 지식이 필요하다. 하지만 작업 계획 생성 알고리즘은 비-기호적 상태 서술자를 표현하고 처리할 수 없다. 본 논문에서는 이를 위해 참조(reference)를 이용하여 비-기호적인 도메인 지식들을 기호적으로 표현하여 처리한다. 결국 작업 계획 생성 시에는 참조에 값이 존재한다고 가정하여 작업 계획이 생성되도록 유도한 뒤 이 참조들의 실제 값은 모션 계획 생성 과정에서 검증한다. 참조는 목표 포즈를 위한 참조와 경로를 위한 참조가 있다. 참조를 포함하는 행위 명세 예시는 [Fig. 4]와 같다. [Fig. 4]의 행위 명세는 PDDL (Planning Domain Description Language)[10]로 작성되었고 양 팔을 가지는 다관절 휴머노이드 로봇인 PR2를 대상으로 한다.

[Fig. 4]에서는 isGrasp, isPlacementLocation, isClearanceLocation, isMotion, obstructs 등의 기존의 작업 계획 생성 방법들에서 다루지 않던 새로운 상태 서술자들을 확인할 수 있다. isGrasp (pose2, obj, objLoc) 서술자는 ‘poes2는 objLoc에 위치한 물체 obj를 잡을 수 있는 잡기 포즈이다’를 의미한다. isPlacementLocation (targetLoc, obj) 서술자는 ‘targetLoc은 물체 obj를 내려 놓기 위한 목표 위치이다’를 의미한다. isClearanceLocation (clearLoc, obs, tar)는 ‘clearLoc은 장애물 obs가 목표 물체 tar를 가로막지 않는 치움 위치이다’를 의미한다. 잡기 포즈(grasp pose), 배치 위치(placement location), 치움 위치(clearance location) 등은 모두 포즈 생성기로부터 생성되어 모션 계획 생성기에게 모션 계획 목표(motion planning goal)로써 제시된다. 각각의 포즈 생성에 대한 내용은 4.3절에서 좀 더 자세히 설명한다.

action	*PickUp_PR2*
param	*obj gripper, pose1, pose2, traj*
precond	*empty (gripper), robotAt (pose1),* *objectAt (obj, objLoc),* *isGrasp (pose2, obj, objLoc),* *isMotion (traj, pose1, pose2),* *∀obj’ ¬obstructs (obj’, traj, obj1)*
effect	*graspedBy (obj, gripper), ¬empty (gripper),* *∀obj’, traj’ ¬obstructs (obj, traj’, obj’)*
action	*PutDownGeneral_PR2*
param	*obj, gripper, pose1, pose2, traj, targetLoc*
precond	*graspedBy (obj, gripper), robotAt (pose1),* *¬obstacle (obs)* *isPlacementLocation (targetLoc, obj),* *isGrasp (pose2, obj, targetLoc),* *isMotion (traj, pose1, pose2),* *∀obj’¬obstructs (obj’, traj, obj)*
effect	*¬graspedBy (obj, gripper), at (obj, targetLoc)*
action	*PutDownForClearance_PR2*
param	*obs, tar, gripper, pose1, pose2, traj, targetLoc*
precond	*graspedBy (obj, gripper), robotAt (pose1),* *obstacle (obs)* *isClearanceLocation (targetLoc, obs, tar),* *isGrasp (pose2, obs, targetLoc),* *isMotion (traj, pose1, pose2),* *∀obj’¬obstructs (obj’, traj, obs)*
effect	*¬graspedBy (obs, gripper), at (obs, targetLoc),* *¬obstacle (obs),* *¬isClearanceLocation (targetLoc, obs, tar)*
action	*MoveBase_PR2*
param	*pose1, pose2, traj*
precond	*RobotAt (pose1), isMotion (traj, pose1, pose2)*
effect	*¬RobotAt (pose1), RobotAt (pose2)*

[Fig. 4] Action specification for obstacle handling

isMotion (traj, pose1, pose2) 서술자는 ‘traj는 pose1에 위치한 몸체로부터 잡기 포즈 pose2에 도달 가능한 경로이다’를 의미한다. 경로 traj는 충돌이 고려된 역운동학(Inverse Kinemarics, IK)을 통해 확인된다. obstructs (obj‘, traj, obj) 서술자는 ‘obj’은 경로 traj 상에서 obj을 가로막고 있다’를 의미한다. 또 obstacle (obs)은 ‘obs은 장애물이다’를 의미한다. 이 두 서술자들은 모션 계획 생성 과정에서만 알아낼 수 있다는 특징이 있다. 따라서 맨 처음 작업 계획을 생성하기 위한 초기 상태로는 주어지지 않지만 모션 계획 생성 과정에서 장애물로 인한 실패가 발생하면 새롭게 생성된다.

한편, [Fig. 1]에서는 두 가지 내려놓기(put down) 행위가 있다. 먼저, PutDownGeneral_PR2 행위는 비-장애물을 내려놓기 위한 일반적인 내려놓기 행위이고 PutDownForClearance_PR2는 장애물을 내려놓기 위한 행위이다. 따라서 PutDownForClearance_PR2 행위는 PutDownGeneral_PR2 행위보다 더 많은 제약조건을 가진다. [Fig. 3]에서는 PutDownClearance_PR2 행위를 선언함으로써 모션 계획 생성 과정에서 장애물을 치우기 위한 목표 포즈 및 이동 경로 생성을 가능하게 한다.

3.3 포즈 생성

본 논문에서는 잡기 포즈(grasp pose), 배치 위치(placement location), 치움 위치(clearance location)를 생성한다. 생성된 포즈들은 모션 계획 생성기에 결정적인 모션 계획 목표를 제시함으로써 방대한 탐색 공간을 줄여준다. 포즈 생성 방법의 핵심은 연속 공간 상에서 유효한 포즈 값들을 이산화하는 것이다.

잡기 포즈 생성은 측면 잡기(side grasp)와 상단 잡기(top grasp)로 나뉜다. 각각의 잡기 포즈는 [Fig. 5]와 같이 두 매개변수 δ, ϵ에 의한 제약조건을 만족해야 한다. δ는 TCP (Tool Center Point)와 물체의 최하단과의 수직거리이다. ϵ는 TCP와 물체의 주축과의 수직거리다. δ, ϵ는 그리퍼를 닫았을 때 안정적으로 물체가 잡힐 수 있는 위치여야 한다. 본 논문에서는 시행 착오를 통해 미리 알아낸 안정적인 δ, ϵ 값을 이용한다. δ, ϵ 값은 Grasp 계획 알고리즘을 통해 자동으로 알아낼 수도 있다. δ, ϵ가 결정되면 로봇의 그리퍼를 일정 횟수 n만큼 z축을 기준으로 회전시켜 이산화된 잡기 포즈 값을 결정한다. 회전 각도는 $360^\circ/n$으로 결정된다. 예컨대, n이 6일 때 측면 잡기 포즈는 [Fig. 6]과 같이 생성된다. 생성된 측면 잡기 포즈의 수는 n으로 [Fig. 6]에서는 6개가 생성된 것을 확인할 수 있다. 상단 잡기도 동일한 원리이다.

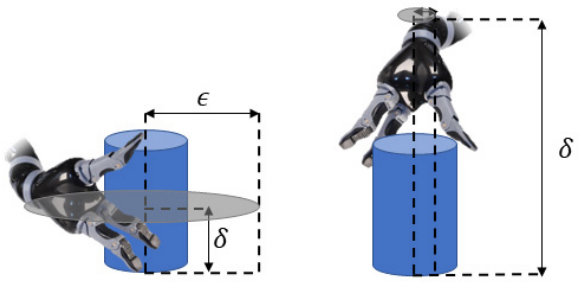
배치(placement) 위치 생성을 위한 제약조건은 [Fig. 7]과 같이 세 매개변수 x_{size}. y_{size}. z_{size}에 의한 제약조건을 만족해야 한다. 이 세 매개변수는 바닥과 수평이 되는 물체의 평평한 면(support plane)의 영역을 표현한다. 따라서 배치 위치는 이 영역 안에 포함되어야 한다. Srivastava의 연구에서는 이 제약조건만을 이용하여 장애물 치움 위치를 생성하였다. 물체를 내려놓기 위한 평평한 면의 x_{size}. y_{size}. z_{size}를 알면 평평한 면을 격자 형태로 나누고 격자의 꼭지점들의 xy좌표로 결정한다. 격자의 수는 위한 매개변수 c와 r로 결정된다. c는 행의 개수, r은 열의 개수이다. 예컨대, c와 r이 각각 4일 때 배치 위치는 [Fig. 8]과 같이 생성된다. 특히, [Fig. 8]에서는 놓인 물체가 바닥으로 떨어지는 것을 방지하기 위해 각 모서리에서 l만큼 여백을 두고 격자를 생성한다. 생성된 치움 위치의 수는 $(c+1)(r+1)$으로 [Fig. 8]에서는 25개가 생성된 것을 확인할 수 있다.

치움(clearance) 위치 생성은 [Fig. 8]과 같이 생성된 배치 위치들을 대상으로 NGR (Negative Goal Region)[16] 제약조건을 적용한다. NGR은 장애물이 놓일 수 없는 위치를 의미한다. NGR에 대한 수식들은 아래와 같다.

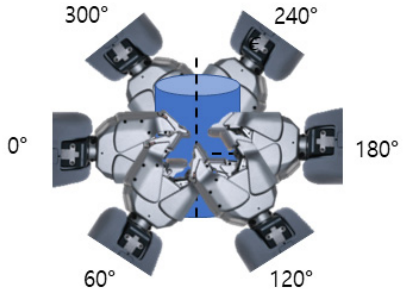
$$NGR_1 = Volume(robot, \tau_{tar}) + Volume(target, v_{tar}) \quad (1)$$

$$NGR_2 = \sum Volume(robot, \tau_{obs}) + \sum Volume(obstacle, v_{obs}) \quad (2)$$

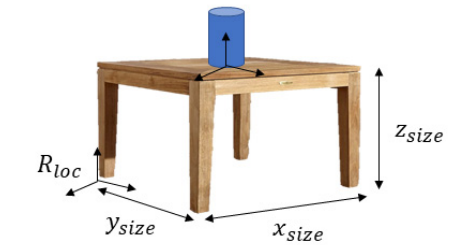
$$NGR = NGR_1 + NGR_2 \quad (3)$$



[Fig. 5] Constraints for grasp pose

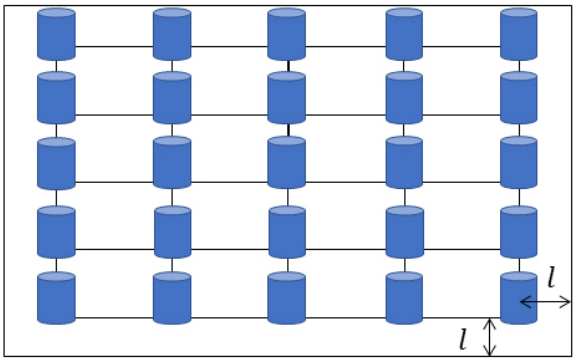


[Fig. 6] Generated grasp poses (n=6)



$Ap \leq c_{loc}$
$with\ c_{loc} = (0, y_{size}, z_{size}, x_{size}, 0, -z_{size})$

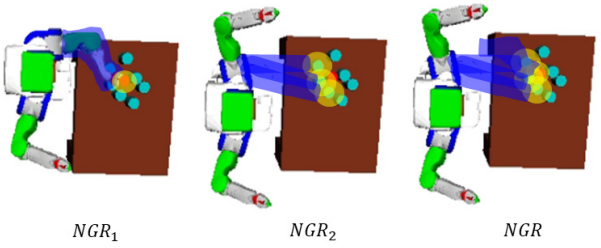
[Fig. 7] Constraints for placement pose



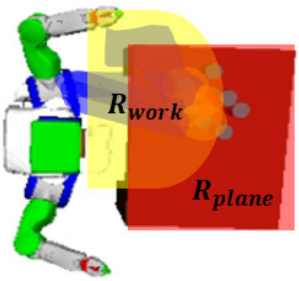
[Fig. 8] Generated placement poses (c=4, r=4)

이 식들에 대한 예시는 [Fig. 9]와 같다. NGR_1은 목표 물체를 잡기 위한 로봇 팔의 경로 τ_{tar}에 대한 부피 $Volume(robot, \tau_{tar})$과 목표 물체의 부피 $Volume(target, v_{tar})$을 더한 값이다. [Fig. 9]에서 로봇 팔의 부피는 파란색으로, 물체의 부피는 노란색으로 표현되어 있다. NGR_2은 각각의 장애물들을 잡기 위한 로봇 팔의 경로 τ_{obs}에 대한 부피 합$\sum Volume(robot, \tau_{obs})$과 모든 장애물들의 부피 합 $\sum Volume(obstacle, v_{obs})$을 모두 더한 값이다. 최종 NGR은 NGR_1과 NGR_2를 더한 값이다. Dogar[16]의 연구에서는 NGR_1만을 최종 NGR로 사용하기 때문에 치워야 할 다른 장애물까지 고려한 올바른 치움 위치를 생성할 수 없다.

최종 NGR이 계산되면 [Fig. 10]과 같이 우선 순위를 가지는 두 영역 안에서 최종 치움 위치를 결정한다. [Fig. 10]에서 R_{work}은 로봇의 작업 영역(work space)을 나타낸다. R_{plane}는 테이블의 영역을 나타낸다. 가장 먼저 로봇은 두 영역이 교차하는 영역 $R_{work} \cap R_{plane}$에서 장애물의 치움 위치를 찾는다. 이 영역에서는 장애물을 치우기 위해 로봇의 몸체를 이동할 필요가 없기 때문에 가장 먼저 잡은 장애물을 가장 먼 곳에 배치되도록 결정한다. 만약 영역 $R_{work} \cap R_{plane}$에서 장애물의 치움 위치를 찾을 수 없다면 테이블의 영역 내에서 로봇의 작업 영역을 제외한 영역 $R_{plane} - R_{work}$에서 치움 위치를 찾는다. 이 영역에서는 장애물을 치우기 위해 로봇의 몸체를 이동시켜야 하기 때문에 가장 먼저 잡은 장애물을 가장 가까운 곳에 배치되도록 결정한다.



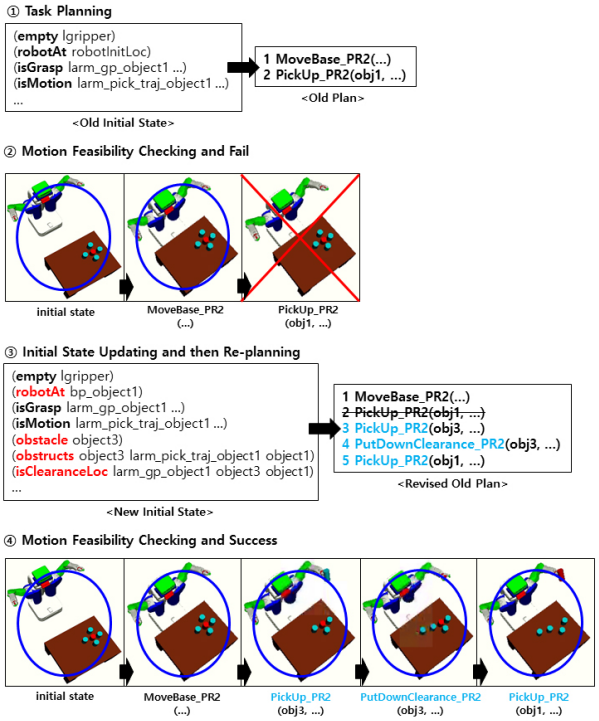
[Fig. 9] Constraints for clearance pose 1: Negative Goal Region (NGR)



[Fig. 10] Constraints for clearance pose 2: workspace and support plane

3.4 재계획

이 절에서는 재계획이 발생하는 과정을 하나의 예시를 통해 설명한다. [Fig. 11]은 맨 처음 생성되었던 작업 계획이 장애물로 인해 모션 계획 생성이 실패된 후 재계획이 발생하는 과정을 보여준다. [Fig. 11]에서 PR2 로봇은 장애물에 대한 정보가 없는 초기 상태(old initial state)로부터 빨간색 물체 obj1에 근접한 위치로 이동 후에 obj1을 잡는 행위들을 생성한다. 하지만 장애물 obj3이 있기 때문에 PickUp_PR2 (obj1, ...) 행위의 모션 실행 가능성 검증이 실패하게 된다. 모션 실행 가능성 검증이 실패하게 되면 모션 실행 가능성 검증 과정에서 얻어낸 장애물 정보로부터 새로운 상태 서술자들을 만들어 초기 상태를 갱신한다. 생성된 상태 서술자들은 obj3이 장애물이라는 obstacle 서술자와 obj3이 obj1을 막고 있다는 obstructs 서술자 그리고 obj3이 obj1을 막지 않도록 하는 치움 위치가 존재한다는 isClearanceLoc 서술자이다. 또한, 앞서 성공했던 MoveBase_PR2(...) 행위의 효과를 반영하여 초기 상태를 갱신한다. 갱신된 상태 서술자는 robotAt 서술자이다. 초기 상태가 갱신되면 새로운 초기 상태(new initial state)로부터 작업 계획을 다시 생성하고 새로 생성된 작업 계획은 이전에 마지막으로 성공했던 MoveBase_PR2(...) 행위 뒤에 이어 붙여진다.



[Fig. 11] An example of replanning process

4. 구현 및 실험

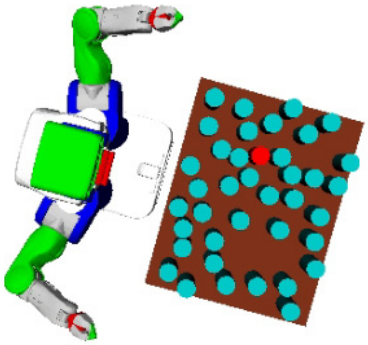
4.1 구현

본 논문에서 제안하는 작업-모션 계획의 연계 방법은 Ubuntu 16.04, Python 2.7 환경에서 구현하였다. 작업 계획 생성을 위해서는 전향 탐색(forward search) 알고리즘을 이용하는 FF (Fast Forward)[17] 공개 라이브러리를 이용하였다. 모션 계획 생성을 위해서는 순차 콘벡스 최적화(sequential convex optimization)를 이용하여 충돌 방지 이동 경로를 계산하는 TrajOpt[18] 공개 라이브러리를 이용하였다. 로봇 가상 환경과 행위 실행은 OpenRAVE 시뮬레이터[19]와 ROS (Robot Operating System)[20]를 이용하여 구현하였다.

4.2 실험

실험은 3.5Ghz 4 cores 4 threads CPU, 8 RAM의 하드웨어 환경에서 진행하였다. 작업-모션 계획의 연계 방법의 성능을 검증하기 위한 로봇 작업 환경은 [Fig. 12]와 같다. [Fig. 12]의 작업 환경에서 PR2 로봇은 탁자 위에 무작위로 생성된 물체 중 빨간 물체와 같이 한 물체를 선택하여 집어 올려야 한다. 선택된 물체는 무작위로 생성된 물체의 수가 늘어날수록 더 많은 장애물들에게 가로막혀 있을 가능성이 높다. 또한, 장애물의 치움 위치가 매우 제한적으로 되기 때문에 장애물의 치움 위치를 잘못 결정할 경우 재차 치워야 하는 문제가 생긴다. 실험은 정성 실험과 정량 실험으로 진행하였다.

먼저, 정성 실험은 [Fig 13]과 같이 동일한 작업 환경, 동일한 목표로부터 생성된 작업 계획들을 비교해보았다. 비교를 위해 Srivastava의 방법으로 작업 계획을 생성하고 본 논문에서 제안한 방법으로 작업 계획을 생성해보았다. 목표는 object36에 의해 가로막혀 있는 object1을 잡는 것이다. 이를 위해 로봇은 object36을 다른 곳으로 치운 후 object1을 잡아야 한다.



[Fig. 12] Task environment with PR2

Srivastava의 방법으로 생성된 작업 계획은 object36을 중복으로 치우는 계획이 생성될 수도 있지만 직관적인 정성 비교를 위해 길이가 가장 짧은 최적의 계획이 생성된 것으로 가정하였다. 따라서, [Fig. 13]에서 두 계획은 길이가 같은 거의 동일한 계획이다. [Fig. 13]에서 두 계획의 차이점은 장애물인 object36을 내려 놓는 4번째 행위이다. 본 논문에서 제안한 방법으로 생성된 계획에서는 PUTDOWNFORCREARANCE_PR2 행위가 이에 해당하고 Srivastava의 방법으로 생성된 계획에서는 PUTDOWN_PR2 행위가 이에 해당한다. 이 두 행위 차이는 행위의 이름과 매개변수에 있다. PUTDOWNFORCREARANCE_PR2 행위에서는 장애물인 object36과 목표 물체인 object1이 행위의 매개변수로 입력되어 있고 PUTDOWN_PR2 행위는 장애물인 아닌지 모르는 물체 object36만 행위의 매개변수로 입력되어 있다. 사소한 차이로 보일 수 있으나 이 차이는 로봇이 물체를 단순히 내려놓는 것이 아니라 잡고 있는 물체가 장애물인지, 이 장애물이 어떤 물체를 막고 있었는지, 이 장애물을 단순히 내려 놓는 것인지 아니면 치우는 것인지 기호적으로 구

Goal: *grasp the object1 (obstructed by object36)*

Task Plan from our method:

1 MOVETO_PR2
(ROBOT_INIT_LOC ROBOT_GP_OBJECT36 ROBOT_MP_OBJECT36)
2 PICKUP_PR2
(OBJECT36 LGRIPPER ROBOT_GP_OBJECT36 LARM_GP_OBJECT36 LARM_MP_OBJECT36)
3 MOVETO_PR2
(ROBOT_GP_OBJECT36 ROBOT_PDP_TABLE6_LOC36 ROBOT_MP_TABLE6_LOC36)
4 PUTDOWNFORCLEARANCE_PR2
(OBJECT36 OBJECT1 LGRIPPER ROBOT_PDP_TABLE6_LOC36 LARM_PDP_TABLE6_LOC36 LARM_MP_TABLE6_LOC36 TABLE6_LOC36)
5 MOVETO_PR2
(ROBOT_PDP_TABLE6_LOC36 ROBOT_GP_OBJECT1 ROBOT_MP_OBJECT1)
6 PICKUP_PR2
(OBJECT1 LGRIPPER ROBOT_GP_OBJECT1 LARM_GP_OBJECT1 LARM_MP_OBJECT1)

Task Plan from Srivastava's method:

1 MOVETO_PR2
(ROBOT_INIT_LOC ROBOT_GP_OBJECT36 ROBOT_MP_OBJECT36)
2 PICKUP_PR2
(OBJECT36 LGRIPPER ROBOT_GP_OBJECT36 LARM_GP_OBJECT36 LARM_MP_OBJECT36)
3 MOVETO_PR2
(ROBOT_GP_OBJECT36 ROBOT_PDP_TABLE6_LOC36 ROBOT_MP_TABLE6_LOC36)
4 PUTDOWN_PR2
(OBJECT36 LGRIPPER ROBOT_PDP_TABLE6_LOC36 LARM_PDP_TABLE6_LOC36 LARM_MP_TABLE6_LOC36 TABLE6_LOC36)
5 MOVETO_PR2
(ROBOT_PDP_TABLE6_LOC36 ROBOT_GP_OBJECT1 ROBOT_MP_OBJECT1)
6 PICKUP_PR2
(OBJECT1 LGRIPPER ROBOT_GP_OBJECT1 LARM_GP_OBJECT1 LARM_MP_OBJECT1)

[Fig. 13] Generated task plan from ours method and Srivastava's method

분할 수 있게 된다. 이러한 작업 계획의 변화는 작업-모션 계획의 나머지 연계 과정에도 막대한 영향을 미친다. 목표 물체의 위치와 목표 물체를 잡기 위한 팔의 경로까지 고려하여 물체의 치움 위치를 결정할 수 있고 또한, 목표 물체를 가로막고 있는 다른 장애물의 위치와 이 장애물들을 잡기 위한 팔의 경로까지 고려하여 치움 위치를 결정할 수 있게 된다. [Fig. 13]의 실험 결과는 기존 연구들에서 다루던 넓은 의미의 물체 집어 옮기기 행위들을 구체화함에 따라 연계 과정에서 더 정확한 목표 포즈들을 생성할 수 있음을 보인다. 따라서 재계획을 포함하는 전체 연계 과정에서 불필요한 치움 행위들을 중복으로 생성하는 것을 방지하기 때문에 생성된 작업 계획의 길이가 짧아지고 그만큼 재계획이 발생되는 수도 줄어든다. 또한, 계획의 길이가 짧아진 만큼 참조의 수도 줄어들기 때문에 포즈 생성, 모션 계획 검증 등의 계산량도 감소하게 된다.

다음으로 정성 실험에서 분석한 내용들을 검증하기 위해 [Table 1]과 같은 정량 실험을 진행하였다. 이 실험은 테이블 위에 무작위로 생성되는 물체들의 수를 늘려가면서 치운 장애물 수의 평균과 동일한 장애물을 두 번 이상 치운 수의 평균을 측정하였다. 이 수치들은 Srivastava[1]의 방법과 함께 측정하여 비교하였다. 한편, 문제를 좀 더 어렵게 만들기 위해 측면 잡기만 허용하고 상단 잡기는 허용하지 않았다.

먼저 [Table 1]에서 치운 장애물 수의 평균을 보면, Srivastava의 방법보다 본 논문에서 제안한 방법이 더 적은 수의 장애물을 치운 것을 확인할 수 있다. 또한, 물체의 수가 늘어날수록 Srivastava의 방법은 치운 장애물의 수가 점점 더 큰 폭으로 늘어나지만 본 논문에서 제안한 방법은 그렇지 않은 것을 확인할 수 있다. 물체가 가장 적은 15개 일 때 Srivastava의 방법은 3.4개, 본 논문의 방법은 3개이고 물체가 가장 많은 40개일 때 Srivastava의 방법은 11.2개, 본 논문의 방법은 6개이다. 다음으로 동일한 장애물을 2번 이상 치운 수의 평균을 보면, Srivastava의 방법은 최소 0.3개에서 최대 3개의 장애물을 중복하여 치운 반면, 본 논문에서 제안한 방법은 장애물을 중복하여 치운 경우가 없다. 이러한 이유는 Srivastava의 방법은 무작위로 물체를 내려놓는 반면 본 논문의 방법은 다음에 잡아야할 목표 물체 또는 장애물을 잡기 위한 팔의 이동 경로를 피하여 물체를 내려놓기 때문이다.

[Table 1] Experiment results: average number of clearance and average number of duplicated clearance

Problem	# of Clearance (Avg.)		# of Duplication (Avg.)	
	Srivastava[1]	Ours	Srivastava[1]	Ours
Object-15	3.4	3	0.3	0
Object-20	6.6	5	0.7	0
Object-25	7.5	5	1.2	0
Object-30	8.2	6	1.6	0
Object-35	10.7	6	2.3	0
Object-40	11.2	6	3	0

장애물을 치우기 위해서는 재계획의 반복과 추가적인 행위의 실행이 요구되기 때문에 매우 큰 비용을 발생시킨다. 따라서 이러한 [Table 1]의 결과는 기존의 방법보다 더 적은 재계획의 반복 수와 더 작은 길이의 행위 시퀀스를 보이기 때문에 작업-모션 계획의 연계를 위한 전체 처리 시간을 줄여주는 좋은 효과를 내포하고 있다.

5. 결 론

본 논문에서는 복잡 환경에서 장애물에 의해 가로막힌 물체 잡기를 위한 작업-모션 계획의 연계 방법을 제안하였다. 본 논문에서 제안하는 연계 방법은 비-기호적 상태를 포함하는 작업 계획 생성, 장애물 치움 위치를 고려한 목표 포즈 생성, 장애물로 인한 실패 발생 시 재계획 등의 세부 기술들을 포함한다. 가상 환경에서 휴머노이드 로봇을 이용한 실험을 통해 본 논문에서 제안하는 연계 방법의 높은 효율성을 확인할 수 있었다.

작업-모션 계획 간의 연계는 단순한 모양, 동일한 크기의 물체들을 집어 옮기거나 쌓는 등의 비교적 간단한 작업 환경에서만 연구가 이루어져 왔다. 실세계에서는 다양한 모양과 변형 가능한 재질의 물체들이 등장하고 요리, 물건 정리 등 다양한 기능을 가진 물체들을 이용한 복잡한 작업들이 요구된다. 또한, 이런 다양한 작업 환경에서의 연구가 부족한 만큼 작업 계획 생성을 위한 상태 서술자들과 행위 명세, 포즈 생성 방법, 실패 시 재계획 등의 연계 방법에 대한 일반화가 잘 이루어지지 않았다. 따라서 향후 연구로는 다양한 유형의 물체, 복잡한 작업, 불확실성 등을 포함한 좀 더 현실에 가까운 문제들을 작업-모션 계획의 연계 문제로 정의하고 해결해나갈 계획이다.

References

[1] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, “Combined Task and Motion Planning through An Extensible Planner-Independent Interface Layer,” *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, pp. 639-646, 2014.

[2] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, “STRIPStream: Integrating Symbolic Planners and Blackbox Samplers,” *arXiv:1802.08705[cs.AI]*, 2018.

[3] J. Bidot, L. Karlsson, F. Lagriffoul, and A. Saffiotti, “Geometric Backtracking for Combined Task and Motion Planning in Robotic Systems,” *Artificial Intelligence*, vol. 247, pp. 229-265, Jun., 2017.

[4] J. Ferrer-Mestres, G. Frances, and H. Geffner, "Combined task and motion planning as classical AI planning," *arXiv:1706.06927 [cs.RO]*, 2017.

[5] L. P. Kaelbling and T. Lozano-Perez. "Hierarchical Task and Motion Planning in Now," *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, DOI: 10.1109/ICRA.2011.5980391.

[6] F. Lagriffoul, L. Karlsson, J. Bidot, and R. Saffiotti, "Combining Task and Motion Planning is Not Always a Good Idea," *RSS Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*, 2013.

[7] J. Choi and E. Amir. "Combining Planning and Motion Planning," *2009 IEEE International Conference on Robotics and Automation. Kobe, Japan*, pp. 238-244, 2009.

[8] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez. "Constructing Symbolic Representations for High-Level Planning," *Association for the Advancement of Artificial Intelligence (AAAI)*, Québec, Canada, pp. 1932-1938, 2014.

[9] R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189-208, 1971.

[10] M. Ghallab, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL−The Planning Domain Definition Language," 1998.

[11] K.Erol, J.A. Hendler, and D. S. Nau. "HTN Planning: Complexity and Expressivity," *Twelfth National Conference on Artificial Intelligence (AAAI)*, pp. 1123-1128, 1994.

[12] J.-C. Latombe, "Robot Motion Planning," *The Springer International Series in Engineering and Computer Science*, Springer, Boston, MA, DOI: 10.1007/978-1-4615-4022-9.

[13] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," *TR 98-11, Computer Science Dept., Iowa State University*, Oct., 1998.

[14] L. E. Kavraki, J.-C. Svestka, and M. H. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, Aug., 1996.

[15] N. McCain and H. Turner, "Causal Theories of Action and Change," Fourteenth National Conference on Artificial Intelligence (AAAI), Providence, RI, USA, pp. 460-465, 1997.

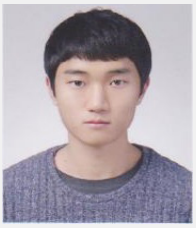
[16] M. R. Dogar and S. S. Srinivasa. "A Framework for Push-Grasping in Clutter," *Robotics: Science and Systems (RSS)*, 2011, DOI: 10.15607/RSS.2011.VII.009.

[17] J. Hoffmann, "FF: The Fast-Forward Planning System," *Artificial Intelligence Magazine*, vol. 22, no. 3, pp. 57-62, 2001.

[18] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization." *Robotics: Science and Systems (RSS)*, vol. 9, no. 1, pp. 1-10, 2013.

[19] R. Diankov and J. J. Kuffner, "OpenRAVE: A Planning Architecture for Autonomous Robotics," *Robotics Institute, Carnegie Mellon University*, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34 79, 2008.

[20] M. Quigley, B. Gerkeyy, K. Conleyy, J. Fausty, T. Footey, J. Leibsz, E. Bergery, R. Wheelery, and A. Ng, "ROS: An Open-Source Robot Operating System," *ICRA Workshop on Open Source Software*, 2009.



이 석 준

2015 경기대학교 컴퓨터과학과(학사)
2017 경기대학교 컴퓨터과학과(석사)
2017~현재 경기대학교 컴퓨터과학과(박사)

관심분야: 인공지능, 지능로봇, 지식 표현 및 추론

김 인 철

1985 서울대학교 수학과(학사)
1987 서울대학교 전산과학과(석사)
1995 서울대학교 전산과학과(이학박사)
1996~현재 경기대학교 컴퓨터과학과 교수

관심분야: 인공지능, 지능로봇, 지식 표현 및 추론