막힌 물체 잡기를 위한 작업-모션 계획의 연계

이석준*, 김인철*
*경기대학교 컴퓨터과학과
e-mail:20171102101@kyonggi.ac.kr, kic@kyonggi.ac.kr

Combining Task and Motion Planning for Grasping Obstructed Object

Seok-Jun Lee*, In-Cheol Kim*
*Dept of Computer Science, Kyonggi University

요 약

어질러진 환경에서의 물체 조작은 개방형 과제이다. 어질러진 환경에서는 장애물들에 의한 물체 조작의 실패가 자주 발생한다. 본 논문에서는 어질러진 환경에서 장애물에 의해 막힌 물체 잡기을 위한 작업-모션 계획의 연계 방법을 제안한다. 작업-모션 계획의 연계는 로봇이 실행해야 할 연속된 행위들을 생성하고 이 행위들의 실행 가능성을 확인하는 방법이다. 본 논문에서 제안하는 연계 방법은 1) 실행 가능성 확인을 위해 비-기호적 상태를 포함한 작업 계획 생성, 2) 장애물 치움 위치를 고려한 목표 포즈 생성, 3) 실행 가능성 확인 과정에서 발생한 실패 처리를 위한 재계획 등의 세부 기술들을 포함한다. 본 논문에서 제안하는 연계 방법의 높은 효율성을 확인하기 위해 휴머노이드 로봇을 이용한 실험을 진행하고 그 내용을 소개한다.

1. 서론

최근 로봇의 자동화를 위해 인공지능과 로보틱스가 융합된 연구들이 활발히 진행되고 있다. 그 중에서도 특히, 로봇이 스스로 목표를 달성하기 위한 행위들을 결정하고 실행하기 위한 작업-모션 계획의 연계(Combining Task And Motion Planning, CTAMP)는 지속적으로 연구되어 오고 있는 핵심 연구 분야이다[1, 2, 3, 4]. 작업-모션 계획의 연계는 실행 가능한 작업 계획을 자동으로 얻어내는 것을 목표로 한다. 이를 위해 작업 계획 생성(task planning)[5]으로 실행해야 할 연속된 행위들(sequence of actions)을 얻어내고 모션 계획 생성(motion planning)을 통해 각 행위들의 실행 가능성(feasibility)[6]을 확인하는 것이 일반적인 연계 방법이다. 본 논문에서는 개방형 과제(open problem)인 어질러진 환경(cluttered environment)에서의 물체 조작(object manipulation)을 위한 작업-모션 계획의 연계 방법을 제시하고자 한다. 어질러진 환경에서는 장애물에 의한 모션 계획의 실패가 자주 발생하기 때문에 작업-모션 계획의 연계 과정에서 실패 처리가 매우 중요하다.

작업 계획 생성(task planning)은 주어진 초기 상태로부터 목표 상태에 도달하기 위한 연속된 행위들을 자동으로 생성하기 위한 논리적인 원인 귀속 추론(causal reasoning) 방법이다. 따라서 주어지는 모든 상태와 행위들은 기호적(symbolic)으로 형식화되어 있다. 하지만 이러한 연속된 행위들은 추상화된 기호 지식이기 때문에 실행 가능성(feasibility)을 보장하지 않는다. 한편, 모션 계획 생성은 물리적인 실행 레벨에서 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 장애물 충돌로부터 자유로운 이동 경로(trajectory)를 생성한다. 이 두 유형의 계획 생성 방법을 보다 수준 높게 연계하기 위해서는 1) 목표 포즈(target pose), 이동 경로(trajectory) 등의 비-기호적 상태를 포함한 작업 계획 생성, 2) 모션 제약(motion constraint)을 위한 목표 포즈(target pose) 생성, 3) 모션 계획 생성 실패 시 재계획(replanning) 등의 세부 기술들이 요구된다. 특히, 작업 계획 생성과 모션 계획 생성은 방대한 탐색 공간을 가지기 때문에 연계 과정에서 적용되는 세부 기술들이 이러한 탐색 공간을 줄일 수 있는 방향으로 모색되어야 한다.

작업-모션 계획 간의 연계는 크게 두 가지 방법이 있다. 첫 번째는 작업 계획 생성을 모두 마친 후에 행위들의 실행 가능성을 확인하는 방법이고, 두 번째는 작업 계획을 생성하는 과정에서 행위 하나가 결정될 때마다 실행 가능성을 확인하는 방법이다. 본 논문에서 제안하는 연계 방법은 첫 번째 방법에 속한다.

첫 번째 방법을 기초로 한 기존 연구들로는 Srivastava[1], Garret[2]의 연구가 있다. Srivastava는 생성된 연속된 행위들을 순차적으로 확인하고 그 과정에서 장애물로 인해 모션 계획 생성이 실패하면 실패한 상태로부터 작업 계획을 다시 생성하는 연계 방법은 제안하였다. 하지만 Srivastava의 연구에서는 장애물을 처리하기 위한 상태 서술자, 행위 명세, 포즈 생성 방법 등이 부족하여 한 번 치운 장애물을 다시 치워야 하는 등의 효율적이지 못한 계획이 생성된다. Garret은 연속된 행위들의 목표 포즈 생성과 모션 계획 생성의 상관관계를 토대로 이들의 탐색 우선순위를 재정렬하여 전체 탐색 공간을 줄이는 방법을 제안하였다. 하지만 Garret의 연구에서는 실패 처리를 위한 연계 방안은 제시되어 있지 않다.

두 번째 방법을 기초로 한 연구들로는 Bidot[3], Ferrer-Mestres[4]의 연구가 있다. Bidot은 기하 역추적(geometric backtracking)에서 발생하는 방대한 탐색 공간을 줄이기 위한 잡기 포즈(grasp pose), 배치 위치(placement location) 등의 다양한 제약 조건들을 제안하였다. Ferrer-Mestres는 함수형 STRIPS를 이용하여 작업 계획 생성 과정에서 모션 계획이 간삽(interleaving)될 수 있는 새로운 연계 구현 방법을 제안하였다. 하지만 이 두 연구 모두 실패 처리를 위한 연계 방안은 제시되어 있지 않다.

* 이 연구는 2018년도 산업통상자원부 및 산업기술평가관리원(KEIT) 연구비 지원에 의한 연구임('10077538')

본 논문에서는 어질러진 환경에서 막힌 물체 잡기를 위한 작업-모션 계획의 연계 방법을 제안한다. 본 논문에서 제안하는 연계 방법의 세부 기술들로는 먼저, 비-기호적 상태를 포함하는 작업 계획 생성이다. 이 방법은 참조(reference)를 이용하여 비-기호적 상태를 기호적 상태로 변환하여 작업 계획 생성이 가능하도록 한다. 다음으로 장애물의 치움 위치(clearance location)를 결정하는 포즈 생성이다. 결정적인 장애물 치움 위치를 모션 계획 생성기에게 전달한다면 이동 경로를 찾는 무한한 탐색 공간을 매우 작은 공간으로 줄일 수 있다. 이러한 장애물 치움 위치는 비-장애물과는 달리 더 많은 제약 조건들을 고려하여 결정해야 한다. 마지막으로 장애물로 인한 모션 계획 생성 실패 시 재계획하는 방안이다. 이 방법은 모션 계획 생성 실패 시 모션 계획 생성기로부터만 얻어낼 수 있는 실패 원인에 대한 상태 서술자들을 작업 계획 생성기에게 전달하여, 작업 계획 생성기가 실패 처리를 위한 새로운 작업 계획을 생성할 수 있게 한다. 본 논문에서 제안하는 연계 방법의 높은 효율성을 확인하기 위해 어질러진 환경에서 휴머노이드 로봇을 이용한 실험들을 진행하고 그 내용을 소개한다.

2. 관련 연구

작업-모션 계획 간의 연계는 크게 두 가지로 나눌 수 있다. 첫 번째는 작업 계획을 먼저 생성하고 각각의 행위에 대해 모션 계획을 확인하는 방법이고, 두 번째는 작업 계획을 생성하는 과정에서 행위 하나가 결정될 때마다 모션 계획을 확인하는 방법이다. 본 논문에서 제안하는 연계 방법은 첫 번째 방법에 속한다.

첫 번째 방법을 기초로 한 기존 연구들로는 Srivastava[1], Garret[2]의 연구가 있다. Srivastava에서는 Srivastava은 생성된 연속된 행위들을 순차적으로 방문하면서 실행 가능성을 확인하고 그 과정에서 장애물로 인해 모션 계획 생성이 실패하면 작업 계획을 다시 생성하는 연계 방법을 제안하였다. 이 방법은 실패 발생 시 이전에 성공했던 행위들을 보관하고 실패한 행위의 상태로부터 작업 계획을 다시 생성한다. 작업 계획을 다시 생성할 때는 모션 계획 생성기로부터 얻어낸 실패 원인들에 대한 상태 서술자들을 작업 계획 생성기에게 전달한다. 예컨대, 물체1이 물체2를 막고 있다(obstruct object1 object2)라는 상태 서술자가 있다. 따라서 작업 계획 생성기는 행위의 실패 원인에 대한 상태들을 고려하여 다시 장애물 치우기를 포함하는 작업 계획을 생성할 수 있다. 하지만 Srivastava의 연구에서는 장애물을 처리하기 위한 상태 서술자, 행위 명세, 포즈 생성 방법 등이 부족하여 한 번 치운 장애물을 다시 치워야 하는 등의 효율적이지 못한 계획이 생성된다. Garret은 일련의 행위들의 목표 포즈 생성과 모션 계획 생성의 상관관계를 토대로 탐색 우선순위를 재정렬하여 전체 탐색 공간을 줄이는 방법을 제안하였다. 탐색의 우선 순위는 모든 행위의 목표 포즈를 생성하는 것이 가장 우선이고 그런 다음 마지막 행위부터 역으로 모션 계획을 생성해나간다. 하지만 Garret의 연구는 연계 과정에서 실패 처리를 위한 연계 방안은 제시되어 있지 않다.

두 번째 방법을 기초로 한 연구들로는 Bidot[3], Ferrer-Mestres[4]의 연구가 있다. Bidot은 기하 역추적(geometric backtracking)에서 발생하는 방대한 탐색 공간을 줄이기 위한 잡기(grasp), 배치(placement) 등의 다양한 제약 조건들을 제안하였다. Ferrer-Mestres는 함수형 STRIPS를 이용하여 작업 계획 생성 과정에서 모션 계획이 간삽(interleaving)될 수 있는 새로운 연계 구현 방법을 제안하였다. 하지만 이 두 연구 모두 실패 처리를 위한 연계 방안은 제시되어 있지 않다.

3. 참조 기반의 작업 계획 생성

본 논문에서는 작업 계획 생성을 모두 마친 후에 순차적으로 각각의 행위들의 실행 가능성을 확인하기 때문에 작업 계획 생성이 선행되어야 한다. 또한 각각의 행위들에 대해 모션 계획을 확인하기 위해서는 행위의 제약 조건에 목표 포즈, 이동 경로 등과 같은 비-기호적 상태들을 포함하고 있어야 한다. 하지만 작업 계획 생성 알고리즘은 비-기호적 상태를 처리할 수 없기 때문에 포즈와 경로에 대한 참조(reference)를 이용하여 기호적 상태로 변환한다. 참조를 포함하는 행위 명세 예시는 (그림 1)과 같다. (그림 1)의 행위 명세는 PDDL(Planning Domain Description Language)로 작성되었고 PR2와 같은 다관절 휴머노이드 로봇을 대상으로 한다.

```
action    PickUp
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    PutDown
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, tloc)
effect    ¬graspedBy(obj, gripper), at(obj, targetLoc)

action    PutDown_Clear
param     obs, obj, gripper, pose1, pose2, traj, targetLoc
precond   graspedBy(obj, gripper), robotAt(pose1),
          obstacle(obs)
          isClearanceLocation(targetLoc, obs, obj),
          isGrasp(pose2, obj, targetLoc),
          isMotion(traj, pose1, pose2),
          ∀obj'¬obstructs(obj', traj, obj, tloc)
effect    ¬graspedBy(obj, gripper), at(obj, targetLoc),
          ¬obstacle(obs),
          ¬isClearLocation(targetLoc, obs, obj)

action    MoveBase
param     pose1, pose2, tra)
precond   RobotAt(pose1), IsMotionPlan(traj, pose1, pose2)
effect    ¬RobotAt(pose1), RobotAt(pose2)
```

(그림 1) 행위 명세 예시

(그림 1)에서는 isGrasp, isPlacementLocation, isClearLocation, isMotion, obstructs 등의 새로운 저수준 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. isClearLocation(clearLoc, obs, obj)는 'clearLoc은 장애물 obs가 obj를 막지 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 goal)로써 제시된다. 각각의 포즈 생성에 대한 내용은 4장에서 좀 더 자세히 설명한다. isMotion(traj, pose1, pose2) 서술자는 'traj는 pose1에 위치한 몸체로부터 잡기 포즈 pose2에 도달 가능한 경로이다'를 의미한다. 여기서 경로 traj는 충돌이 방지된 역운동학(Inverse Kinemarics, IK)을 통해 확인된다. 한편, (그림 1)에서는 두 가지 내려놓기(put down) 행위가 있다. 먼저, PutDown 행위는 비-장애물을 내려놓기 위한 일반적인 내려놓기 행위이고 PutDown_Clear는 장애물을 내려놓기 위한 행위이다. PutDown_Clear는 PutDown 행위를 확장한 행위로서 장애물과 장애물이 막고 있는 목표 물체에 대한 매개변수와 상태 서술자들이 있어 이들을 활용한 목표 포즈 생성과 실행 가능성 확인이 가능하다.

4. 작업-모션 계획 간의 연계

본 논문에서는 어질러진 환경에서 막힌 물체 잡기를 위한 작업-모션 계획 간의 연계 방법을 제안한다. 본 논문에서 제안하는 작업-모션 계획 간의 연계 과정은 크게 1)

작업 계획 생성, 2) 모션 계획 생성, 3) 실패 시 재계획 단계로 나뉜다. 이 중에서 모션 계획 생성 단계는 방대한 탐색 공간을 줄이고 결정적인 모션 계획 목표를 제시하기 위한 포즈 생성을 포함한다. 실패 시 재계획 단계는 모션 계획 생성기로부터 얻어낸 비-기호적 상태들로부터 작업 계획 생성기에 새로운 상태들을 전달하여 실패 처리를 위한 작업 계획 생성이 가능하도록 한다.

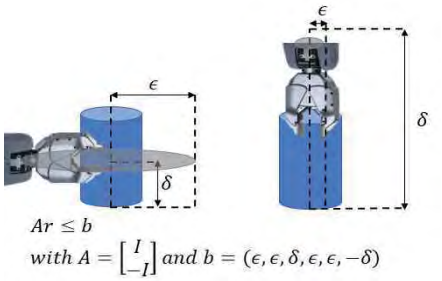
4.1 연계 알고리즘

알고리즘 1은 본 논문에서 제안하는 작업-모션 계획 간의 연계 알고리즘을 나타낸다. 알고리즘의 입력은 초기 상태 I와 목표 조건 G 그리고 행위 명세 A이다. 먼저, 알고리즘은 입력으로부터 작업 계획 p를 생성한다. 그런 다음 작업 계획의 각각의 행위들을 순차적으로 방문하면서 행위들의 모션 계획 $traj$를 생성한다. 모션 계획을 생성하기 위해서는 먼저 목표 포즈 $pose_{target}$를 생성하고 이를 모션 계획 생성의 목표로 전달한다. 모션 계획이 생성되면 행위의 포즈 또는 이동 경로 참조들에 값을 연결하고 행위를 실행 가능한 계획 리스트에 저장한 후 행위 효과에 따라 초기 상태를 갱신한다. 만약 모션 계획 생성이 실패하면 실패에 대한 상태 서술자들을 토대로 초기 상태를 갱신하고 행위 방문을 중단시킨다. 행위 방문이 중단되면 중단된 시점에서 갱신된 초기 상태로부터 다시 작업-모션 계획 간의 연계 과정이 시작된다.

Algorithm 1: CTAMP Alogiritm

Input: Initial State I, Goal Condition G, Action Specification A

```
step s ← 1; plan p ← none; feasiblePlan fp ← none
while feasiblePlan not founded
  p ← generateTaskPlan(I, G, A)
  for action α in plan p
    pose_⋆t ← getCurrentRobotConf()
    pose_target ← generateTargetPoses(α)
    traj, err ← generateMotionPlan(pose_init, pose_target)
    if motion planning succeed
      bindReference(α, pose_⋆t, pose_target, traj)
      fp ← fp + α
      I ← updateInitialState(I, α)
    else
      I ← updateInitialState(I, err)
      break
    end if
    step ← step+1
  end for
end while
```



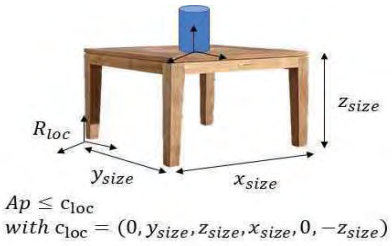
(그림 2) 잡기 포즈 생성을 위한 제약 조건

4.2 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.

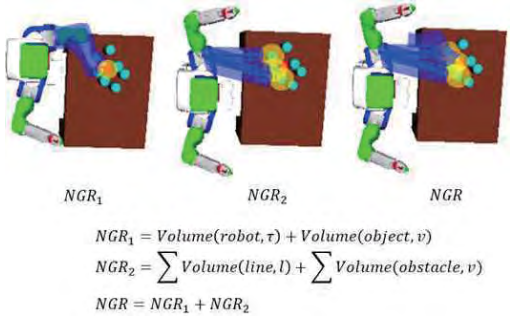
잡기 포즈 생성은 측면 잡기(side grasp)와 상단 잡기(top grasp)로 나뉜다. 각각의 잡기 포즈는 (그림 2)와 같이 두 매개변수 δ, ϵ에 의한 제약 조건을 만족해야 한다. δ는 TCP(Tool Center Point)와 물체의 몸체 간의 거리이다. ϵ는 TCP와 물체의 주축과의 수직거리이다.

배치(placement) 위치 생성은 (그림 3)과 같은 제약 조건을 만족해야 한다. 배치 포즈는 평평한 면 위에 올려놓는 것으로 가정한다. Srivastava의 연구에서는 이 제약 조건만을 이용하여 장애물 치움 위치를 생성한다.



(그림 3) 배치 위치 생성을 위한 제약 조건

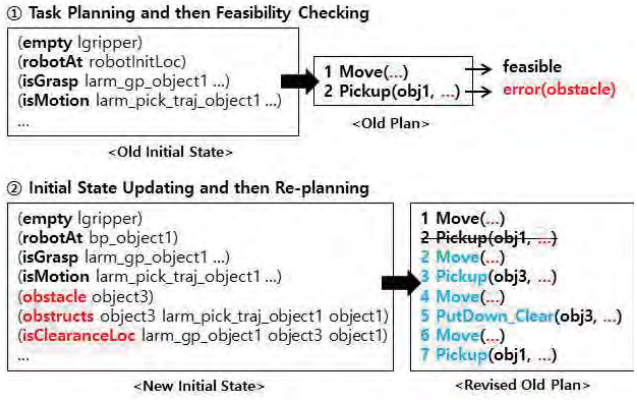
치움(clearance) 위치 생성은 (그림 3)의 제약 조건에 (그림 4)의 NGR(Negative Goal Region) 제약 조건을 추가한다. (그림 4)에서 NGR_1은 로봇의 팔 경로 τ에 의한 볼륨 $Volume(robot, \tau)$과 목표 물체의 확장 영역 ν에 의해 생성된 볼륨 $Volume(object, v)$을 더 값이다. NGR_2은 장애물들과 로봇 팔의 직선 l에 의해 생성된 볼륨들의 합 $\sum Volume(line, l)$과 장애물들의 확장 영역 ν에 의해 생성된 볼륨들의 합 $\sum Volume(obstacle, v)$을 모두 더한 값이다. 최종 NGR은 NGR_1과 NGR_2를 더한 값이다. 장애물의 치움 위치는 최종 NGR과 겹칠 수 없다.



(그림 4) 치움 위치 생성을 위한 제약 조건

4.3 재계획

이 절에서는 재계획이 발생하는 과정을 하나의 예시를 통해 설명한다. (그림 5)는 목표 물체 obj1을 막고 있는 장애물 obj3로 인해 모션 계획 생성이 실패하고 재계획이 발생하는 과정을 보여준다. (그림 5)에서 초기 상태(old initial state)로부터 생성되는 작업 계획은 물체 obj1에 접근한 후에 잡는 행위들로 생성되어 있다. 하지만 장애물 obj3이 있기 때문에 PickUp 행위에 대한 모션 계획 생성이 실패한다. 그러면 초기 상태에 성공했던 행위들의 효과(effect)를 반영하고 모션 계획이 실패한 원인으로부터 새로운 상태 서술자들을 추가한다. 생성된 상태 서술자들은 obj3이 장애물이라는 obstacle 서술자와 obj3이 obj1을 막고 있다는 obstructs 서술자 그리고 obj3이 obj1을 막지 않도록 하는 치움 위치가 존재한다는 isClearanceLoc 서술자이다. 새로운 초기 상태(new initial state)로부터 작업 계획을 다시 생성하고 새로 생성된 작업 계획은 이전의 실패했던 행위를 삭제한 위치에 더해진다.



(그림 5) 재계획 과정 예시

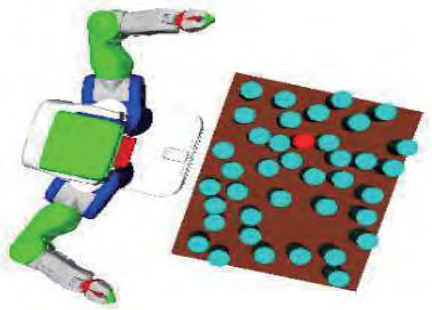
5. 구현 및 실험

5.1 구현

본 논문에서 제안하는 작업-모션 계획의 연계 방법은 Ubuntu 16.02, Python 2.7 환경에서 구현하였다. 작업 계획 생성을 위해서는 전향 탐색(forward search) 알고리즘을 이용하는 FF(fast forward)[5] 공개 라이브러리를 이용하였다. 모션 계획 생성을 위해서는 순차 콘벡스 최적화(sequential convex optimization)를 이용하여 충돌 방지 이동 경로를 계산하는 TrajOpt[6] 공개 라이브러리를 이용하였다. 로봇 가상 환경과 행위 실행은 OpenRave 시뮬레이터와 ROS(Robot Operating System)를 이용하여 구현하였다.

5.2 실험

실험은 3.5Ghz 4core CPU, 8 RAM의 하드웨어 환경에서 진행하였다. 작업-모션 계획의 연계 방법의 성능을 검증하기 위한 로봇 작업 환경은 (그림 6)과 같다. (그림 6)의 작업 환경에서 PR2 로봇은 탁자 위에 무작위로 생성된 물체 중 하나를 집어 올려야한다. 문제를 좀 더 어렵게 만들기 위해 측면 잡기만 허용하고 상단 잡기는 허용하지 않는다.



(그림 6) 실험을 위한 로봇 작업 환경

실험은 <표 1>과 같이 무작위로 생성되는 물체들의 수를 늘려가면서 치운 장애물 수의 평균(# of clearance(Avg.)), 동일한 장애물을 두 번 이상 치운 수의 평균(# of duplication(Avg.)) 그리고 제한 시간 내에 작업 문제를 해결한 횟수의 백분율(solved in 600s(%))을 측정하였다. 이 수치들은 Srivastava[1]의 방법과 함께 측정하여 비교하였다.

먼저 <표 1>에서 치운 장애물의 수를 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. 다음으로 동일한 장애물을 2번 이상 치운 수의 평균을 보면, Srivastava의 방법은

<표 1> 비교 실험 결과

Problem	# of Clearance (Avg.)		# of Duplication (Avg.)		Solved in 600s (%)	
	[1]	Ours	[1]	Ours	[1]	Ours
Object-15	3.4	3	0.3	0	100	100
Object-20	6.6	5	0.7	0	94	99
Object-25	7.5	5	1.2	0	90	93
Object-30	8.2	6	1.6	0	84	89
Object-35	10.7	6	2.3	0	67	82
Object-40	11.2	6	3	0	63	77

최소 0.3개에서 최대 3개의 장애물을 중복하여 치운 반면, 본 논문에서 제안한 방법은 장애물을 중복하여 치운 경우가 없다. 마지막으로 제한 시간 내에 작업 문제를 해결한 횟수의 백분율을 보면 Srivastava의 방법보다 본 논문에서 제안한 방법이 더 높게 측정된 것을 확인할 수 있다. 특히, Srivastava의 방법은 물체의 수가 35개 이상일 때 수치가 급격히 떨어지지만 본 논문에서 제안한 방법은 그렇지 않은 것을 확인할 수 있다. 이러한 이유는 물체의 수가 늘어날수록 치운 장애물의 수가 훨씬 더 적어지기 때문이다.

장애물을 치우기 위해서는 로봇의 추가적인 행위 실행이 요구되기 때문에 매우 큰 비용을 발생시킨다. 따라서 이를 줄인 것은 매우 의미 있는 결과이다. 또한 이러한 결과는 기존의 방법보다 더 적은 수의 재계획과 행위들을 가지기 때문에 작업-모션 계획의 연계를 위한 전체 처리 시간을 줄여주는 좋은 효과를 야기한다.

6. 결론

본 논문에서는 어질러진 환경에서 장애물에 의해 막힌 물체 잡기을 위한 작업-모션 계획의 연계 방법을 제안하였다. 본 논문에서 제안하는 연계 방법은 1) 다양한 비-기호적 상태를 포함하는 작업 계획 생성, 2) 장애물 치움 위치를 고려한 목표 포즈 생성, 3) 실행 가능성 확인 과정에서 발생한 실패 처리를 위한 재계획 등의 세부 기술들을 포함한다. 가상 환경에서 휴머노이드 로봇을 이용한 실험을 통해 본 논문에서 제안하는 연계 방법의 높은 효율성을 확인할 수 있었다. 향후 연구로는 복잡한 모양의 물체 잡기, 불확실성을 포함한 작업-모션 계획의 연계와 같이 좀 더 고도화된 연구를 진행할 계획이다.

참고문헌

[1] Siddharth Srivastava, et al. “Combined Task and Motion Planning through An Extensible Planner-Independent Interface Layer.” *Proc. of Robotics and Automation (ICRA)*, pp. 639-646, 2014.

[2] Caelan Reed Garret, et al. “STRIPStream: Integrating Symbolic Planners and Blackbox Samplers.” *arXiv preprint arXiv:1802.08705*, 2018.

[3] Bidot, Julien, et al. “Geometric Backtracking for Combined Task and Motion Planning in Robotic Systems.” *Artificial Intelligence* Vol. 247, pp. 229-265, 2017.

[4] Ferrer-Mestres, Jonathan, Guillem Frances and Hector Geffner. “Combined task and motion planning as classical AI planning.” *arXiv preprint arXiv:1706.06927*, 2017.

[5] Jörg Hoffmann, “FF: The fast-forward planning system.” *Artificial Intelligence magazine*, Vol. 22, No. 3, pp. 57-62, 2001.

[6] John Schulman, et al. “Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization.” *Journal of Robotics: Science and Systems*, Vol. 9, No. 1, pp. 1-10, 2013.