UROP

Final Report

기계공학과 2018-17070 송찬의

기계공학과 2018-13538 서형석

Topic

Automatic robot task generation in assembly process with learning from observation

I. Introduction

본 연구는 Yu, Finn의 "One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning" 연구의 문제점을 보완하기 위한 연구로 진행된다. 따라서 해당 선행연구의 의의와 핵심 아이디어 및 결과와 그 한계를 분석하였다.

사람과 동물은 특정 종류의 행동을 관측하면 그와 연관된 행동을 한 번 관측하는 것을 통해 해당 행동을 모방할 수 있다. 그러나 기존의 머신 러닝 기반 로봇연구는 카메라 위치의 변경에도 취약하며, 많은 양의 학습 데이터가 필요하다는 단점이 있다. 선행연구는 사람의 동작과 로봇의 동작이 유사하다는 점에서 착안해 메타러닝을 활용하여 이러한 문제점을 해결하기 위해 진행하였다.

메타러닝은 2단계 이상의 학습구조를 가짐으로써, 하위구조에서의 학습결과를 기반으로 상대적으로 적은 데이터로 상위구조를 학습하는 머신 러닝의 분야이다. 본 논문에서는 Model-Agnostic Meta-Learning(MAML)을 활용하였다. 이는 모델과 무관하게 머신 러닝에 활용가능한 알고리즘으로서, 일반화된 모델 Parameter θ를 구한뒤, 이를 세부적인 모델 parameter의 초기값으로 사용하여 최적의 parameter φ를 찾아가는 방식이다.

UROP 기간 중 한 연구는 placing, pushing, pick-and-placing task를 메타러닝을 통해 학습시켰다. 이를 인간의 팔로 동작을 수행한 후, 같은 움직임을 갖는 로봇의 동작을 수행시켜 그 데이터를 학습시켰다. 인간의 동작은 영상으로, 로봇의 동작은 영상과 각 joint의 각으로 표현되는 state, 그리고 gripper의 on/off와 joint의 토크와 velocity로 구성된 action으로 정의되었다. 이후, 학습 데이터를 기반으로 사람에 행동에 대한 로봇의 반응을 관측하고, 이를 신경망 구조를 통해 학습하여 로봇의 동작을 제어하는 방식으로 최적의 model parameter를 얻었다. 그 후, 이 parameter와 단 하나의 사람의 task수행영상을 통해 모델을 학습시켜 사람 행동을 따라하게 하였다.

그 결과 기존의 모델들을 활용했을 때와 대비하여 더 높은 정확도로 해당 임무를 성공적으로 수행할 수 있었다. Placing의 경우 93.8%, pushing의 경우 88.9%, pick and place의 경우 80.0%의 정

확도로 수행하였다. 또한 카메라 위치의 변화에도 해당 학습모델이 robust함을 보이기 위해 parameter학습과 다른 각도에서 촬영한 영상을 기반으로 행동을 모방하도록 실험하였다. 이 경우 역시 70%이상의 정확도를 보여줬다. 또한, MuJoCo physics engine에서 시뮬레이션으로 해당 연구를 구현하였을 때 역시 80%이상의 확률로 task가 성공함을 확인하였다.

그러나 이 연구의 한계는 과제가 매우 단순했다는 점과 그럼에도 불구하고 산업현장에서 활용하기엔 낮은 성공률이 나왔다는 점이다. 또한, 해당 task를 실행하는 로봇 역시 안정적으로 움직이지 못하고 진동이 많이 생기는 움직임을 나타내어 안정성에도 문제가 있음이 확인되었다.

II. Method

본 연구는 메타학습을 위한 데이터를 수집하고, 메타학습 모델을 구현하여 데이터를 학습시키는 과정을 통해 진행되었다. 특히, UROP과제에서는 데이터를 수집하는 과정에 집중하였다.

데이터 수집 전에 데이터 수집을 위한 환경구축을 진행하였다. 이 과정은 크게 로봇에 screwing end effector를 부착하는 과정 및 camera calibration을 진행하는 과정으로 이루어졌다.

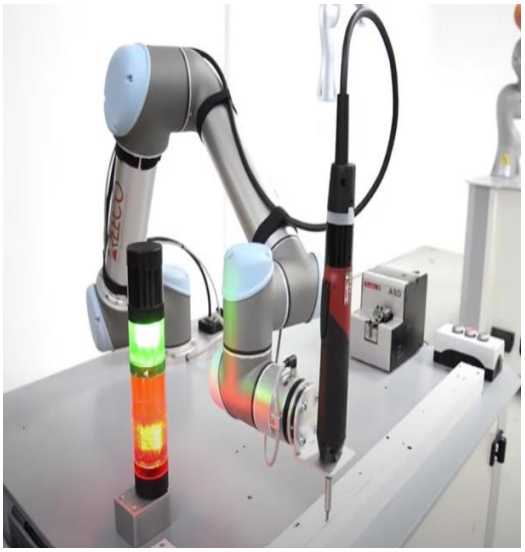


Fig. 1 Screwing motion을 위한 end effector 목표 모델

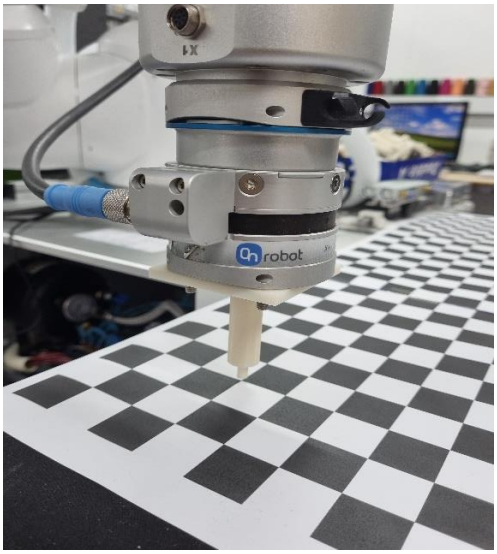


Fig. 2 calibration을 위한 end effector

A. Hardware

Screwing motion을 위하여 그 모션이 가능한 end effector 설계가 우선이 되어야 하였다. 그러기 위해서는 실험실에 있는 Dremel을 이용할 수 있는 screwing end effector가 필요하였는데, Fig.1와 같은 모델을 기반으로 만들었다.

먼저 로봇의 접합부 부분을 설계를 했어야 했는데, 이 접합부 부분을 설계를 완료한다면, 다른 용도의 end effector 설계 또한 가능하였다. 실제로, 설계된 접합부를 이용하여 calibration을 위한 end effector와 screwing motion을 위한 end effector을 제작하였고, calibreation end effector는 정확한 좌표를 측정하는데 이용이 되었으며, Fig.2와 같은 모습으로 이용이 되었다.

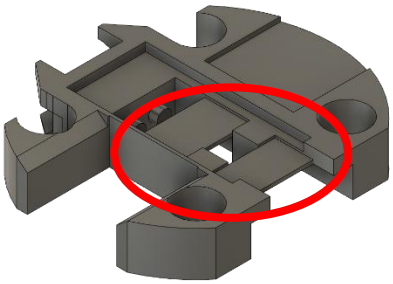


Fig. 3 최종 접합부 모델링

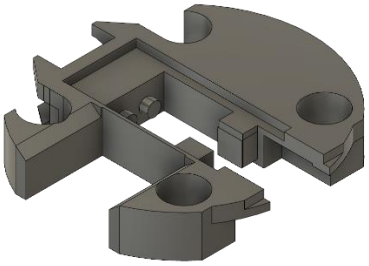


Fig. 4 progress report 때의 유격 고려하지 못한 접합부 모델링

처음 접합부 부분을 완료하였을 때, Dremel의 무게로 인해 유격이 생겨서 흔들거리는 모습을 확인할 수 있었다. 정밀한 움직임을 목표로 하는 것에서, 유격이 허용되면 안 되었기 때문에, 유격을 줄이는 방향으로 수정을 계속 하였다. 유격을 줄이게 만든 접합부 부분은 Fig.3와 같으며, Fig.4에서 유격을 줄이기 위해 고정 스프링이 움직일 수 있는 범위를 제한하는 것을 추가하였다. 또한, 유격을 제거하기 위해 Dremel 쪽으로 무게 중심이 치우친다는 것을 고려하여 Fig.5와 같이 3d 프린터의 무게 중심을 Dremel이 접합되는 반대의 곳에 치우치도록 아치형 구조물을 설계하였다.

앞선 progress report에서 Dremel을 아두이노와 서보 모터를 통하여 세기를 조절하기로 하였는데, 산업용 Dremel이 아닌 사람이 사용하는 Dremel을 사용하기 위해서, 전에 Fig.6에서 볼 수 있는 구조보다는, 로봇이 외부에 설치되어 있는 조절기를 통하여 직접 움직여 조절하는 방식으로 바꾸어 end effector의 크기를 줄여 더 간단한 움직임이 가능하게 되었다.

최종적으로 완성된 Dremel을 이용한 screwing motion end effector를 통해 실제로 작업하는 모습을 Fig.7에서 볼 수 있다.

Fig. 5 Dremel end effector 모습

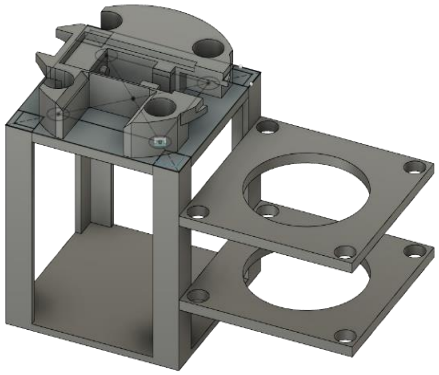


Fig. 6 progress report에서 end effector 모습

Fig. 7 완성된 screwing motion end effector 으로 실제 작업 수행 모습

B. Software

로봇의 데이터 학습을 위해 제공할 데이터 수집을 위한 Camera Calibration이 진행되었다.

기존의 Camera Calibration은 2개의 점의 로봇 팔 좌표와 픽셀 값을 측정하여 x, y좌표를 별개로 선형 근사 하는 방식이었다.

즉, $x_{coord} = a_x x_{pix} + b_x$, $y_{coord} = a_y y_{pix} + b_y$의 모델링을 통해 계수 a_x, b_x, a_y, b_y를 구하는 방식이

다. 그러나 이러한 모델링을 통해 진행한 camera calibration은 정확도가 떨어졌다.

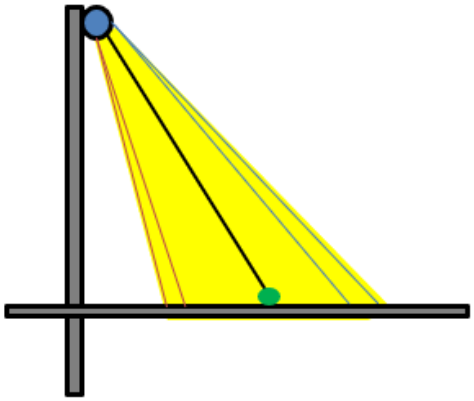


Fig. 8 카메라 왜곡현상

이는 로봇 팔로부터의 시야확보를 위해 설정한 카메라의 설치위치에 의한 왜곡인 것으로 분석되었다. Fig. 8과 같이 카메라의 위치는 촬영지점의 한쪽에 편향된 위치에 존재한다. 이 때문에 카메라의 같은 시야각도 카메라로부터 떨어진 거리에 따라 다른 구간길이를 표현하게 된다.

이를 보정하기 위한 첫번째 방법은 다중선형회귀를 사용한 것이었다. 카메라 사이의 거리가 calibration에 있어 중요요소로 작용한다는 점을 고려하여, x픽셀값 뿐만 아니라 y픽셀값 역시 중요하게 작용할 것이라는 가정을 세웠다. 이를 통해 선형모델링을 하면 다음과 같이 표현된다.

$$x_{coord} = a_x x_{pix} + b_x y_{pix} + c_x,\ y_{coord} = a_y x_{pix} + b_y y_{pix} + c_y \quad (1)$$

한편, 카메라로부터 거리는 카메라의 중앙 픽셀이 나타내는 좌표 값 (Fig. 8에서의 초록 점)으로부터의 위치에 따라 달라진다. 이를 기반으로 Fig. 9dhk 같이 구역을 나누어 해당 구역에 대한 모델링의 보정을 진행하였다.

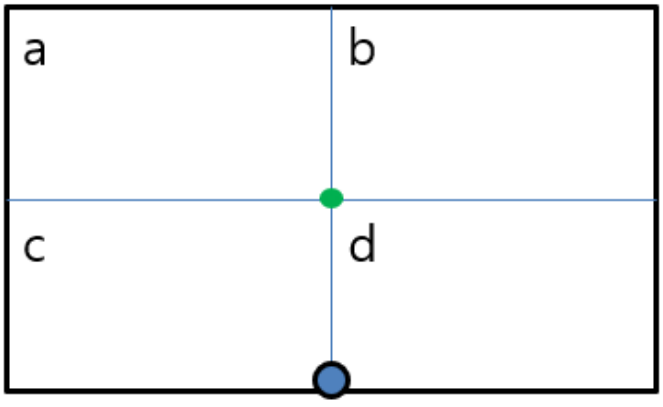


Fig. 9 다중선형회귀 모델링 구역분리

구역에 따른 모델링은 다음과 같이 표현된다.

$$a: x_{coord} = a_x x_{pix} - b_x y_{pix} + c_{x'} y_{coord} = a_y x_{pix} + b_y y_{pix} + c_y \quad (2)$$

$$b: x_{coord} = a_x x_{pix} + b_x y_{pix} + c_{x'} y_{coord} = a_y x_{pix} + b_y y_{pix} + c_y \quad (3)$$

$$c: x_{coord} = a_x x_{pix} - b_x y_{pix} + c_{x'} y_{coord} = -a_y x_{pix} + b_y y_{pix} + c_y \quad (4)$$

$$d: x_{coord} = a_x x_{pix} + b_x y_{pix} + c_{x'} y_{coord} = -a_y x_{pix} + b_y y_{pix} + c_y \quad (5)$$

9개의 점의 픽셀 값과 로봇 팔 좌표 값을 측정하여 해석학적으로 가장 가깝게 선형근사 되는 계수를 찾을 수 있었고, 이를 기반으로 calibration을 진행하였다. 그 결과 calibration이 큰 폭으로 개선되었다.

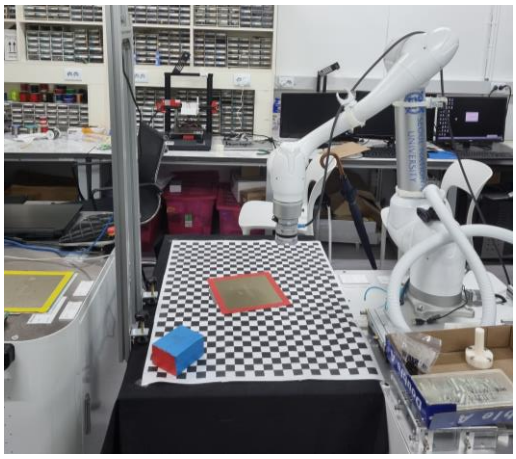


Fig. 10 체커보드를 활용한 Calibration

또한, Fig. 10과 같은 체커보드를 활용하여 더 빠르고 정확한 calibration을 설계하기 위한 알고리즘을 제작하였다. 체커보드의 꼭지점 사이의 좌표거리는 일정하다는 가정 하에 체커보드의 꼭지점들의 좌표는 3개의 점의 좌표만 측정하여도 쉽게 계산이 가능하다. 따라서, 체커보드의 꼭지점에 해당하는 픽셀 값을 빠르게 계산할 수 있다면 수많은 좌표와 픽셀사이의 대응이 가능하고, 이로부터 진행하는 선형 근사는 더 높은 정확도를 가질 것이다.

한편, 체커보드의 꼭지점을 찾는 데에 쓰이는 OpenCV 라이브러리 하수 중 하나인 'findChessboardCorners'는 카메라의 해상도 문제로 인해 적용이 불가능하였다. 따라서, Laplacian edge detection 알고리즘을 활용하여 이미지로부터 체스판의 선을 Fig. 12와 같이 계산하였다. 이로부터 주변과의 밝기 차이 및 상하좌우 픽셀과의 관계를 활용하여 꼭지점을 찾아내는 알고리즘을 코딩을 통해 직접 구현하였다. 이 과정은 Table 1, 2에 상세히 기술하였다.

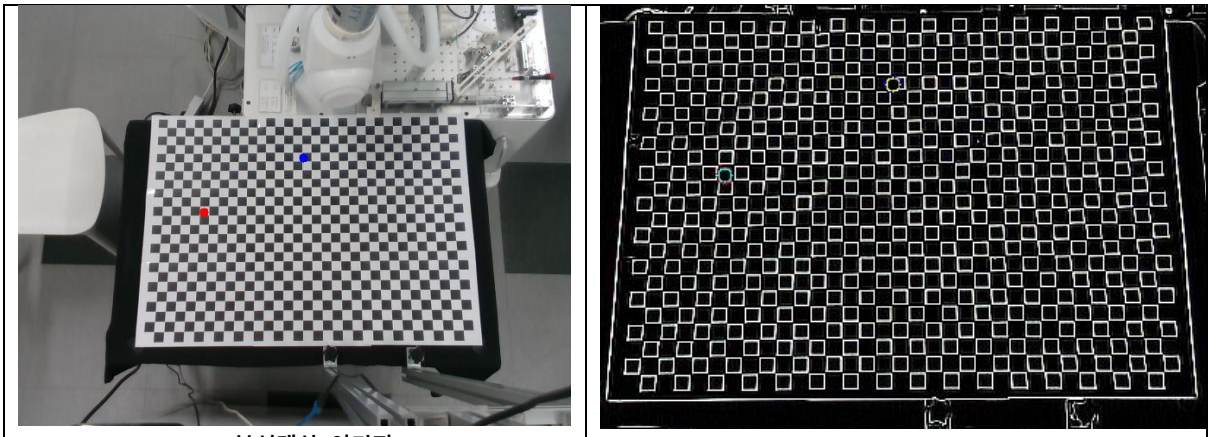
알고리즘1. 체커보드 꼭짓점 인식 알고리즘
1. 이미지에서 체커보드를 포함하는 더 작은 영역으로 이미지 잘라내기
2. Laplacian Edge Detection 알고리즘으로 체커보드의 모서리 따기
3. 이미지 흑백변환하기
4. 체커보드 위치에 해당하는 영역의 픽셀을 반복문으로 훑으며 우측하단 꼭짓점 찾기 꼭짓점기준: 해당픽셀과 왼쪽, 위에 위치한 픽셀의 밝기가 170 이상이며 우측과 하단에 위치한 픽셀의 밝기가 130 이하 혹은 우측하단 픽셀의 밝기가 50 이하
5. 인식되지 않은 꼭짓점 보정 보정방법: 4에서 인식한 꼭짓점 중 같은 행에 해당하는 이웃한 꼭짓점의 간격이 비정상적으로 먼 경우 그 사이꼭짓점이 인식누락 된 것으로 간주하여 두 꼭짓점의 평균값을 추가로 꼭짓점으로 인식
6. 5에서 보정되지 않은 꼭짓점을 이미지를 통해 대조하여 수기로 보정
7. 꼭짓점의 좌표를 같은 행에 위치한 꼭짓점끼리 묶어 데이터 저장 이웃한 꼭짓점과 y축 픽셀값 차이가 1 이하이면 같은 행으로 판정, 2 이상인 경우 꼭짓점 인식오류로 판정

Table 1. 체커보드 꼭짓점 인식 알고리즘

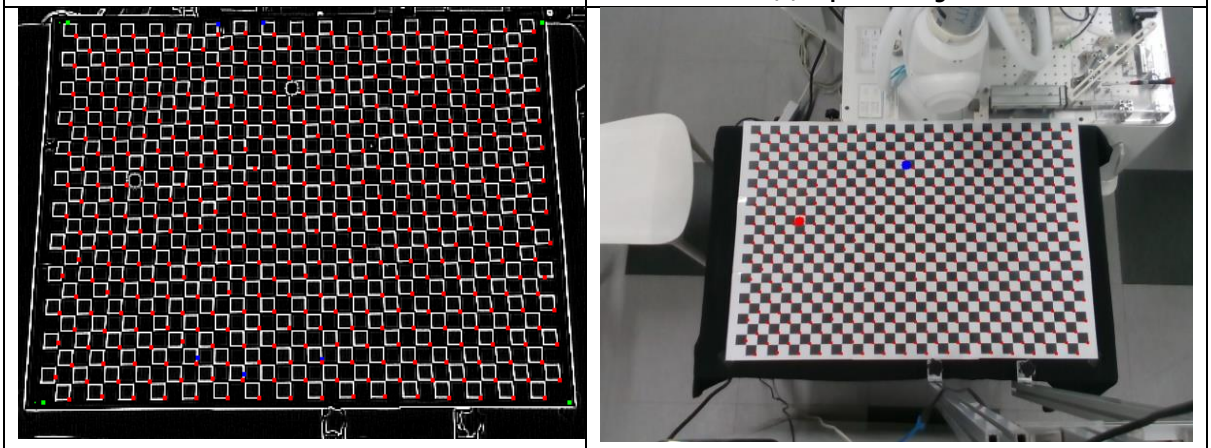
알고리즘2. 픽셀-좌표 대응 알고리즘
1. 체커보드 위 세 꼭짓점의 로봇팔 좌표 측정
2. 1에서의 측정값을 기반으로 체커보드의 가로변, 세로변 각각에 대한 좌표벡터 계산
3. 2에서의 계산값을 기반으로 체커보드의 모든 꼭짓점의 좌표 계산
4. 알고리즘 1을 통해 측정한 꼭짓점의 픽셀 값을 3에서 계산한 좌푯값과 대응
5. 같은 행으로 분류된 꼭짓점 픽셀값의 y값의 평균 계산
6. 5의 데이터를 기반으로 입력받은 픽셀값의 행 위치 파악
7. 해당 행의 위치에서 입력받은 픽셀값의 열 위치 파악
8. 해당 행/열 위치에서 가장 가까운 꼭짓점 3점을 통해 일차독립인 2개의 픽셀벡터 및 2개의 좌표벡터 계산
9. 8에서 구한 픽셀벡터의 일차결합으로 목표픽셀 계산
10. 9에서 얻은 일차결합의 계수를 활용하여 목표픽셀의 좌푯값 계산

Table 2. 픽셀-좌표 대응 알고리즘

이를 통해 꼭지점을 찾은 과정과 결과는 Fig. 11과 같다.



(a) 분석대상 이미지 (b) Laplacian Edge Detection 적용



(C) 조건문과 반복문으로 계산한 꼭지점 (d) 분석대상 이미지 위에 꼭지점을 표기

Fig. 11 꼭지점을 찾는 과정

C. Dataset

이후 개발한 메타학습 모델 검증을 위한 데이터셋 제작을 하였다. 임의의 두 좌표 사이에 블록을 운반하는 작업에 대한 데이터를 수집하였다. 카메라를 통해 촬영한 로봇의 이미지 및 로봇팔의 configuration을 수집하였다. 또한, 같은 작업을 사람손으로 반복하여 사람의 이미지 및 사람이 활용한 end effector의 토크값을 수집하였다. 로봇 data의 경우 총 1000개, 사람 데이터의 경우총 400개의 데이터를 수집하였다. 이를 설계한 메타학습 모델에 적용하여 학습여부를 확인하였다.

Result and Discussion

Dremel을 부착한 end effector의 경우 매우 높은 안정도로 자유로운 이동이 가능하며, milling역시 문제없이 진행됨을 확인할 수 있었다. Calibration의 경우 역시 데이터수집 과정에서 높은 정확도로 적용이 됨을 확인할 수 있었다.

반면, 메타학습 모델의 경우 loss가 감소하는 양상이 뚜렷하게 드러나지 않아 여러 방면에서 추가

개선이 필요하다. 우선, 메타학습 모델을 메타학습 전용으로 개발된 라이브러리를 활용하여 재구성할 계획이다. 또한, 학습을 위해 들어간 데이터의 calibration 에러문제 및 사람이 포함되는 등 데이터 품질의 문제가 존재하였다. 이 점을 개선하여 양질의 데이터셋을 수집할 계획이다.