3차원 공간에서 에이전트의 탐색을 통한 장면 그래프 생성

신동협, 김인철
경기대학교 컴퓨터과학과
email:ksw9446@kyonggi.ac.kr, kic@kyonggi.ac.kr

Scene Graph Generation by Exploration of Agent in Three-Dimensional Space

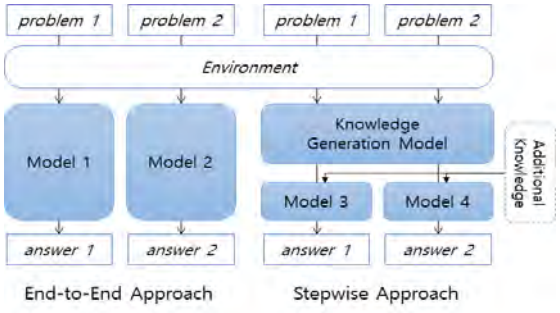
Donghyeop Shin, Incheol Kim
Department of Computer Science, Kyonggi University

요 약

장면 그래프는 영상 내 물체들의 정보를 나타내는 지식 그래프이다. 본 논문에서는 3차원 공간에서 에이전트의 탐색을 통해, 장면 그래프를 생성하는 모델을 제안한다. 3차원 공간에 대한 장면 그래프는 물체들의 위치, 종류, 속성뿐만 아니라 물체들 간의 관계 정보를 포함한다. 이에 따라 장면 그래프는 다양한 문제 해결에 기초 데이터로써 활용될 수 있다. 본 논문은 장면 그래프를 생성하기 위해 필요한 기능들을 정의하고, 기능에 따라 4가지 부분 네트워크들을 제안한다. 또한 각 부분 네트워크들의 학습 및 성능 평가를 위해, 3차원 실내 가상환경인 AI2-THOR에서 데이터들을 수집하였고, 다양한 실험을 통해 각 부분 네트워크들의 성능을 검증하였다.

1. 서론

딥러닝 기술의 발전에 따라, 영상에 관한 복합 지능 문제들에 관심이 높아지고 있다. 최근에는 VQA(Visual Question Answering)와 같은 단일 영상에 대한 복합 지능 문제와 달리, 3차원 공간에서 에이전트의 탐색을 요구하는 복합 지능 문제가 대두되고 있다. 대표적으로 IQA[1]에서는 에이전트가 주어진 질의에 따라 3차원 공간을 탐색하고, 질의에 대한 적절한 응답을 요구하는 문제를 제시하였다. 이러한 문제를 해결하기 위해선 주어진 질의를 이해하고, 이에 대한 응답을 생성하는 것도 중요하지만, 핵심 문제는 어떻게 3차원 공간을 이해하도록 학습된다는 장점이 있다. 하지만 문제 해결과정을 한 모델이 모두 처리하기 때문에, 모델의 구조 설계가 어렵고 모델이 적절히 학습되기까지 상당한 노력이 요구된다. 또한 학습된 모델은 동일한 환경이라도 새로운 문제에는 적용되기 어렵다. 본 논문에서는 이러한 단점을 해결하기 위해, 입력으로부터 문제의 답이 아닌, 3차원 공간에 대한 지식을 생성하는 모델을 제안한다. 이러한 지식은 신경망 내부에서 벡터 등으로 표현된 특징(feature)이 아니기 때문에, 새로운 외부 지식과 쉽게 결합 가능하다. 또한 3차원 공간의 정보들을 갖기 때문에, 다양한 문제 해결에 큰 도움을 줄 수 있다. 이러한 접근법은 (그림 1)의 단계별 접근방법(stepwise approach)에 해당한다.



(그림 1) 두 가지 문제 해결 방법

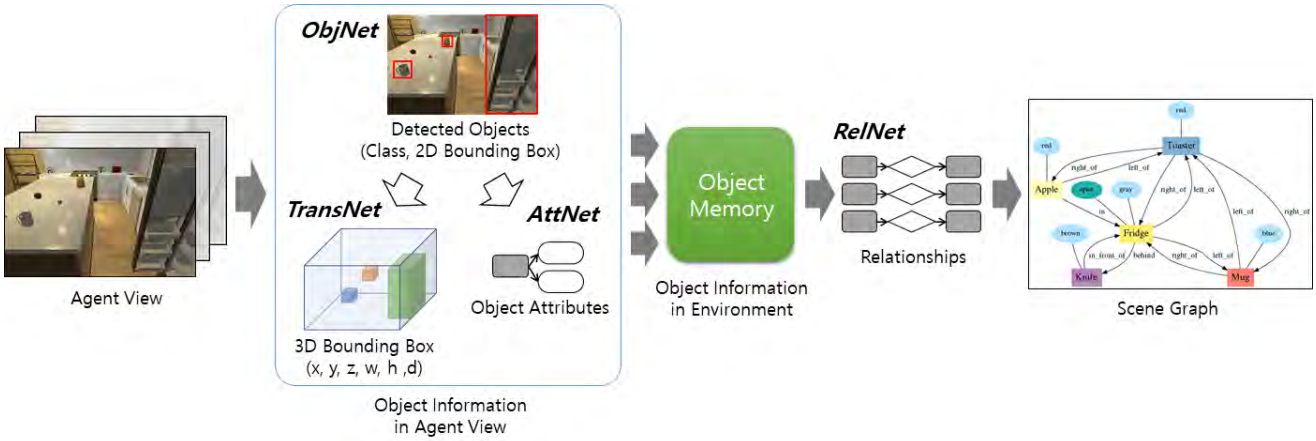
현재까지 대부분의 해결 방법은 (그림 1)의 왼쪽 부분과 같이, 입력으로부터 단 번에 답을 출력하는, 종단 간(end-to-end) 학습기반 모델을 사용하는 것이다. 이러한 해결 방법은 모델 내부에서 자동으로 3차원 공간을 이해

(그림 2) 3차원 환경에서의 장면 그래프 생성

본 논문에서는 (그림 2)와 같이, 3차원 공간을 표현하는 지식으로써 장면 그래프(scene graph)를 사용하였다. 장면 그래프는 최근 영상 이해를 위한 지식 표현 방법으로, 영상에 등장하는 물체들의 위치, 종류, 속성뿐만 아니라 물체들의 관계 정보까지 나타내는 지식 그래프이다. 최근에는 이러한 장면 그래프를 생성하려는 다양한 연구들이 진행되고 있다.

일반적인 장면 그래프 생성 과정은 물체 탐지(object detection) 과정과 관계 예측(relationship prediction) 과정

* 이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2018-0-00677, 촉각이 가능한 로봇 손으로 다양한 물체를 다루는 방법과 절차를 학습하는 로봇 손 조작 지능 개발)



(그림 3) 3차원 공간에서의 장면 그래프 생성 모델

으로 나뉜다. 물체 탐지 과정에서는 기존에 잘 알려진 Faster R-CNN 등의 물체 분류기를 사용한다. 관계 예측 과정에서는 탐지된 물체들의 다양한 특징들을 활용하여 관계를 예측한다. 대부분의 연구에서 물체의 시각 특징(visual feature)을 기반으로, 다른 특징들과 결합하여 예측에 사용하였다. [2]의 연구에서는 물체 종류 특징을 추가로 사용하였다. 이는 두 물체의 종류에 따라, 주로 발생하는 관계 정보를 이용하기 위함이다. 또한 [3]의 연구에서는 물체들의 공간 관계를 파악하기 위해, 각 물체의 위치 정보를 입력 특징으로 사용하였다.

하지만 장면 그래프 생성에 관한 기존 연구들은 한 영상만을 고려하기 때문에, 본 논문에서 다루는 3차원 공간에서 사용하기에 적절하지 않다. 이는 에이전트의 탐색으로 인해, 서로 다른 위치에서의 영상 정보를 입력받게 되고, 같은 물체가 다양한 영상에서 등장할 수 있기 때문이다. 이러한 중복 문제로 각 영상에서 생성된 장면 그래프는 단순히 더해질 수 없다. 또한 3차원 공간 좌표계를 고려하지 않았기 때문에 위치 기반의 중복 여부 판단도 불가하다. 본 연구는 이러한 문제를 해결하고 적절한 장면 그래프를 생성하기 위해, 필요한 기능들을 정의하고, 네 가지 부분 네트워크들로 이루어진 전체 모델을 설계하였다. 또한 학습 및 실험을 위해 AI2-THOR[4] 가상 환경을 이용하였다.

2. 3차원 공간에서의 장면 그래프 생성 모델

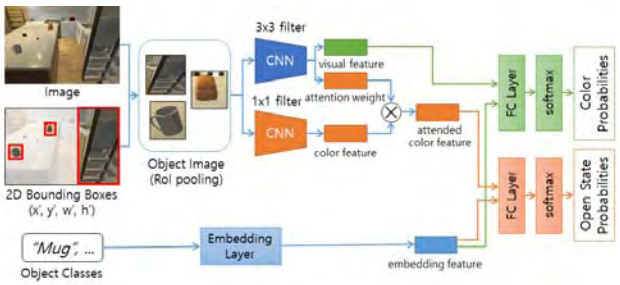
본 논문에서는 3차원 공간에서 장면 그래프를 생성하기 위한 효과적인 모델을 제안한다. 모델의 전체 구조도는 (그림 3)에서 표현되어 있다. 제안 모델은 물체 탐지 네트워크(Object Detection Network, ObjNet), 속성 예측 네트워크(Attribute Prediction Network, AttNet), 변환 네트워크(Transfer Network, TransNet), 관계 예측 네트워크(Relationship Prediction Network, RelNet)로 총 4가지 부분 네트워크들로 구성된다. ObjNet은 에이전트가 현재 바라보는 영상에서 물체들의 영역과 종류(class)를 찾는다. AttNet은 찾아낸 물체들의 여러 속성들을 예측한다. 또한 TransNet은 물체들의 좌표를 현재 에이전트의 관점이 아닌, 3차원 공간상의 절대 관점으로 변환한다. 에이전트는 위 세 네트워크를 이용하여 발견한 물체들을 물체 메모리(object memory)에 저장한다. 물체 메모리에서는 물체들의 3차원 좌표와 종류를 기준으로 중복된 물체들이 제거된다. RelNet은 물체 메모리에 저장된 물체들의 관계를 예측한다. 이후 예측된 결과들을 종합하여 장면 그래프가 생성된다.

2.1 물체 탐지 네트워크 (ObjNet)

물체 탐지 네트워크(ObjNet)은 (그림 2)와 같이 에이전트가 받아들이는 2차원 영상으로부터, 영상 내 물체들의 좌표와 종류를 알아낸다. 본 논문에서는 IQA[1]에서 YOLOv3[5] 물체 탐지기를 이용하여 학습시킨 네트워크를 차용하였다. 그 이유는 YOLOv3 물체 탐지기의 장점인 속도와 정확성에 있다. YOLOv3 물체 탐지기는 물체 영역 탐지와 물체 분류를 동시에 수행하는 한-단계(one-state) 모델이다. 이에 따라 빠른 처리 속도로 실시간 영상 처리가 가능하다. 또한 YOLOv3 물체 탐지기의 네트워크 구조상 작은 물체들을 탐지하는 데에 유리한 장점이 있다. 이는 작은 물체가 많이 등장하는 AI2-THOR 환경에서 매우 적합한 특징으로 볼 수 있다.

2.2 속성 예측 네트워크 (AttNet)

속성 탐지 네트워크(AttNet)은 (그림 4)과 같은 구조를 갖는다. 영상, 물체들의 좌표, 물체들의 종류를 입력받아 물체들의 속성들을 예측한다. 본 연구에서는 물체들의 속



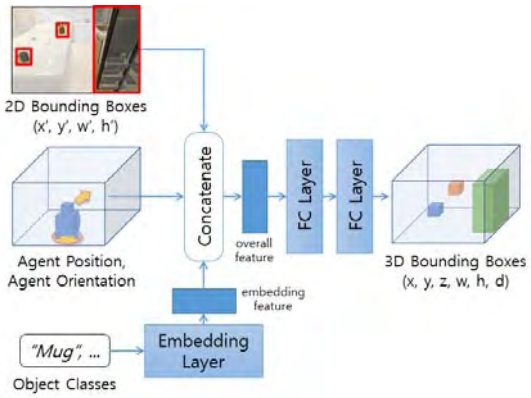
(그림 4) 속성 예측 네트워크(AttNet) 구조

성으로 색상(color)과 개폐 상태(open state)를 선택하였다. 색상은 해당 물체의 색상 종류를 나타내고, 개폐 상태는 해당 물체가 열려있는 상태인지, 닫혀있는 상태인지 혹은 열고 닫을 수 없는 물체인 지를 나타낸다. AttNet에서는 속성들을 예측하기 위해, 합성 곱 신경망(Convolutional Neural Network, CNN)을 이용하여 영상의 시각 특징(visual feature)을 추출하였다. 이는 영상이 특정 부분에서 나타내는 모양을 알아내기 위함이다. 한편 색상 예측의 경우, 영상 데이터 자체가 갖는 색상 값인 RGB 값이 중요하기 때문에, 추가적으로 1x1 크기의 필터를 갖는 CNN을 사용하였다. 이는 영상의 각 픽셀에 해당하는 3가지 색상 값만을 이용하여 픽셀에 대한 색상 벡터를 추출하기

위함이다. 시각 특징들은 물체 종류로부터 얻은 임베딩 특징(embedding feature)과 결합하여, 각 속성들을 예측하는 데에 사용하였다.

2.3 변환 네트워크 (TransNet)

변환 네트워크(TransNet)는 (그림 5)와 같은 구조를 가지며, ObjNet 결과 물체들의 좌표를 3차원 공간상의 좌표로 변환한다. 에이전트 위치를 기반으로 좌표를 변환하기



(그림 5) 변환 네트워크(TransNet) 구조

위해, 에이전트의 위치와 영상에서의 물체 좌표를 입력으로 주었다. 또한 물체의 실제 크기 정보를 이용하기 위해 물체의 종류를 추가로 입력하였다. 이는 물체의 종류에 따른 일반적인 크기 정보를 학습을 통해 배울 수 있기 때문이다. 물체의 크기 정보는 물체의 깊이 정보를 아는 데에 큰 도움을 준다. 예를 들어 영상에서 같은 영역을 차지하는 "Apple"과 "Fridge"가 있을 때, "Apple"은 "Fridge"보다 가까이 있음을 알 수 있다. 이러한 입력 특징들은 임베딩 층과 완전 연결 층을 거쳐, 3차원 공간의 좌표 정보인 <x, y, z, w, h, d>로 변환된다.

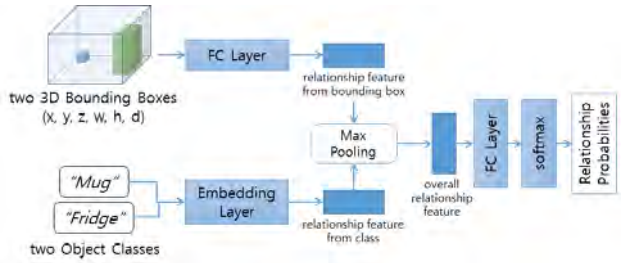
2.5 물체 메모리 (object memory)

ObjNet, AttNet을 거쳐 탐지된 물체와 그 정보들은 TransNet로 변환된 좌표를 기준으로 물체 메모리(object memory)에 저장된다. 이후 에이전트가 새로운 영상을 받아서 또 다른 물체들을 인식하면, 같은 방법으로 물체 메모리에 저장되고 NMS(Non-Maximum Suppression) 연산을 통해 중복된 물체를 제거한다. NMS 연산은 Yolo, SSD 등의 물체 탐지 모델에서 중복되는 물체를 제거하기 위해 사용되는 기술이다. 물체 메모리에서는 이와는 달리 3차원 좌표를 이용하여 NMS 연산을 수행한다. 먼저 종류가 같은 두 물체에 대해 겹치는 비율인 3차원 IoU(Interest of Union)를 구한다. 이는 두 물체의 교집합 공간 부피를 합집합 공간 부피로 나눈 값이다. 3차원 IoU가 일정 수준보다 높으면 물체 확률이 낮은 물체를 제외시킨다. 이러한 과정을 통해 물체 메모리에 중복 물체들을 제거한다.

물체 메모리는 IQA[1]의 공간 메모리(spatial memory)와 비슷해 보이지만, 상당한 차이점이 있다. 공간 메모리는 3차원 공간을 위에서 바라봤을 때 나타나는 2차원 좌표 공간을 나타낸다. 에이전트는 현재 위치 및 방향에 해당하는 공간 메모리 영역을 계산하고, 발견된 물체들을 해당 영역에 매핑 시킨다. 대신 중복 문제를 해결하기 위해, 물체의 객체 정보가 아닌, 물체의 확률 정보를 입력한다. 즉, 에이전트가 탐색을 실시하면서 현재 영역에 물체의 확률 정보를 갱신하는 것이다. 이에 따라 물체의 존재 여부를 명시적으로 저장할 수 없고, 물체의 크기나 속성 정보를 알 수 없다. 또한 위에서 봤을 때 겹치는 물체들의 경우도 고려할 수 없다.

2.4 관계 예측 네트워크 (RelNet)

관계 예측 네트워크(RelNet)은 (그림 6)와 같은 구조를 갖는다. RelNet은 물체 메모리에 저장된 물체들의 공간 관계를 예측한다. 이를 위해 물체의 3차원 공간 좌표, 물



(그림 6) 관계 예측 네트워크(RelNet) 구조

체 종류를 입력받고, 두 물체의 입력 특징을 이용하여 물체간의 관계를 예측한다. 네트워크 구조는 각 특징으로부터 두 관계 특징을 추출한 뒤, 두 특징을 결합하여 관계 분포를 예측한다. 각 관계 특징을 추출하기 위해, 물체의 종류의 경우 임베딩 층을 이용하였고, 물체 좌표의 경우 완전 연결 층을 이용하였다. 이후 두 관계 특징을 최대 풀링(max pooling) 연산으로 결합하였다. 이 결합 방법은 3장의 성능 실험을 통해 결정하였다. 이후 결합된 관계 특징을 완전 연결 층에 입력하여 관계 분포를 예측하였다.

3. 구현 및 실험

3.1 데이터 집합

제안한 네트워크들을 AI2-THOR[4]에서 학습 및 평가하기 위해, 데이터 수집 프로그램을 개발하여 데이터 집합을 직접 구축하였다. AI2-THOR[4]에는 여러 실내 공간이 정의되어있다. 그 중 FloorPlan2~FloorPlan11 환경에 대한 데이터들을 수집하였고, FloorPlan2~FloorPlan9는 모델 학습에, FloorPlan10~FloorPlan11은 모델 평가에 사용하였다. 수집한 영상은 총 24,463장이며, 25개의 물체 종류, 6개의 색상 종류, 3개의 개폐 상태 종류, 6개의 관계 종류를 선정하여 영상에서 해당하는 데이터들을 수집했다. 이 중 개폐 상태 종류는 "opened", "closed", "unable"이며 "unable"은 "Apple"과 같이 여닫을 수 없는 속성을 의미한다. 그리고 관계 종류는 "in", "on", "in_front_of", "behind", "left_of", "right_of"이다. 학습에 사용된 영상은 20,463장이며, 평가에 사용된 영상은 4,000장이다.

3.2 구현

본 연구에서 제안한 모델의 부분 네트워크들은 서로 독립적으로 학습된다. 물체 탐지 네트워크(ObjNet)는 기존 IQA[1]에서 학습된 모델을 차용하였고, 나머지 네트워크들은 수집한 데이터를 바탕으로 학습하였다. 속성 예측 네트워크(AttNet)와 관계 예측 네트워크(RelNet)의 경우, 분류(classification) 문제를 해결하기 위해 크로스엔트로피(cross-entropy) 손실 함수를 사용하였고, 변환 네트워크(TransNet)는 회귀(regression)를 위해 평균 제곱 오차(mean square error, mse) 손실 함수를 사용하였다. AttNet, TransNet, RelNet의 학습률(learning rate)은 실험

을 통해 0.1, 0.001, 0.01로 설정하였고, 최적화 함수(optimizer)는 모두 확률적 경사하강법(stochastic gradient descent)을 사용하였다. 또한 제안한 네트워크들을 구현하기 위해 Ubuntu 16.04 LTS 환경에서 Python 딥러닝 라이브러리인 PyTorch를 사용하였다.

3.3 실험

본 논문에서는 학습한 부분 네트워크들의 개별 성능 실험을 수행 하였다. AttNet과 RelNet은 분류 정확도를 판정하기 위해 재현율(recall)을 사용하였고, TransNet은 회귀 정확도를 보이기 위해 정답과의 오차인 평균 제곱 오차를 사용하였다.

먼저 <표 1>은 AttNet의 입력 특징별 성능 실험 결과를 나타낸다. AttNet에서는 주어진 입력 특징들이 성능에 얼마나 영향을 미치는 지 실험하였다. <표 1>의 결과를 통해 물체들의 시각 특징뿐만 아니라 종류 특징도 성능에 좋은 영향을 미치는 것을 볼 수 있다. 이는 물체의 종류에 따라 색상과 개폐 상태를 예측할 수 있기 때문이다. 예를 들어, “Apple”은 대부분 “red”이고, “Spoon”은 대부분 “gray”라는 특징이 있다. 또 두 물체는 종류로부터 “unable”인 개폐 상태임을 바로 알 수 있다. 하지만 각 종류 마다 고유한 속성을 갖지 않는 경우가 많기 때문에, 종류만을 이용하여 결과를 예측하는 것은 어렵다. <표 1>에서 물체 종류만이 입력된 경우의 낮은 성능이 이를 뒷받침한다.

<표 2> 입력 특징 별 AttNet 정확도 성능

Input data	Output Data	Recall	
		each	average
image	color	61.40%	66.13%
	open state	70.85%	
class	color	45.88%	65.28%
	open state	84.68%	
image + class	**color**	**63.15%**	**76.43%**
	open state	**89.70%**	

<표 2>는 TransNet의 입력 특징별 성능 실험 결과를 나타낸다. 입력 특징은 물체의 상대 좌표(bounding box, bbox), 종류뿐만 아니라 환경으로부터 얻을 수 있는 에이전트와 물체와의 거리가 있다. 물체의 종류와 거리 값은 서로 대체될 수 있는 특징을 갖는다. 그 이유는 물체의 종류는 임베딩 층을 통해 물체의 크기를 예측하고, 이로부터 물체와의 거리를 유추할 수 있기 때문이다. <표 2>의 결과를 확인해보면 직접적으로 물체와의 거리를 입력한 결과가 성능이 좋지만, 물체의 종류를 입력했을 때도 물체의 상대 좌표만을 입력한 경우보다 성능이 향상된 것을 확인할 수 있다. 또한 크기 예측의 경우, 물체 종류를 입력했을 때, 더 높은 성능을 보였다.

<표 3> 입력 특징 별 TransNet 정확도 성능

Input Data	Output Data	MSE	
		each	total
bbox	position	1.0072	1.1809
	size	0.1735	
bbox + class	position	0.8406	0.8931
	size	**0.0525**	
bbox + distance	**position**	**0.2608**	**0.4182**
	size	0.1574	

<표 3>은 RelNet의 입력 특징 및 입력 특징의 결합 방법에 대한 성능 실험 결과를 나타낸다. 입력 특징은 두 물체의 3차원 좌표 정보와 두 물체의 종류이다. 결과적으로 두 입력 특징을 모두 사용한 것이 성능이 높았다. 이는 물체의 종류에 따라 관계를 예측하는 것이 어느 정도 가능하기 때문이다. 예를 들어, “Apple”과 “Fridge”가 주어진다면 “in”이라는 관계를 예측할 수 있을 것이다. 또한 결합 방법에 대한 실험 결과, 최대 풀링 연산을 적용한 것이 가장 높은 성능을 보였다. 이는 두 관계 특징 중, 확실한 특징을 통해 결과를 예측하는 것이 더 효율적이라는 것을 나타낸다.

<표 4> 입력 특징 및 결합 방법 별 TransNet 정확도 성능

Input Data	Fusion Method	Recall
bbox	-	78.66%
class	-	57.02%
bbox + class	concatenate	82.41%
	element-wise product	62.82%
	plus	83.23%
	max pooling	**84.63%**

4. 결론

3차원 공간에 대한 다양한 문제를 해결하기 위해, 기존 연구들은 입력으로부터 바로 최종 출력을 구하는 접근 방법 사용하였다. 이러한 방법은 중간 산물인 공간에 대한 지식을 확인할 수 없고, 다른 문제에 재사용할 수 없다는 단점이 있다. 본 논문은 이러한 단점을 해결하고자, 3차원 공간에 대한 지식을 직접적으로 생성하는 방법을 제안하였다. 생성된 지식은 사전 지식을 통해 개선될 수 있을뿐더러, 3차원 공간에 대한 다양한 문제 해결에 활용될 수 있는 장점이 있다. 본 논문은 공간에 대한 지식인 장면 그래프를 생성하기 위해, 4가지 부분 네트워크들을 제안하였다. 또한 AI2-THOR 가상 환경에서 데이터를 수집하고, 수집한 데이터로부터 각 부분 네트워크를 학습 및 검증하였다. 향후 연구에서는 3차원 공간으로부터 생성된 장면 그래프를 다양한 문제에 활용하는 방법이 연구되어야 할 것이다.

참고문헌

[1] D. Goron, A. Kembhavi, and M. Rastegari, et. al., "IQA: Visual Question Answering in Interactive Environments," Proc. of CVPR-18, pp.4089-4098, 2018.

[2] C. Lu, R. Krishna, and M. Bernstein, et. al., "Visual Relationship Detection with Language Priors," Proc. of ECCV-16, pp.852-869, 2016.

[3] B. Dai, Y. Zhang, and D. Lin, "Detecting Visual Relationships with Deep Relational Networks," Proc. of CVPR-17, pp.3298-3308. 2017.

[4] E. Kolve, R. Mottaghi, and D. Gordon, et. al., "AI2-THOR: An Interactive 3d Environment for Visual AI," arXiv preprint arXiv:1712.05474, 2017.

[5] J. Redmon, and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767, 2018.