실시간 OctoMap 업데이트를 위한 점구름 데이터 양자화

권용선[1], 윤성의[2]
[1]한국과학기술원 로봇공학제전공,[2]한국과학기술원 전산학과

Point-Cloud Data Quantization for OctoMap Update in Real-Time

Young-Sun Kwon[1], Sung-Eui Yoon[2]
[1]KAIST Robotics Program, [2]KAIST Computer Science
e-mail: tea984811@kaist.ac.kr, sungeui@gmail.com

요 약

많은 양의 점구름 데이터(point cloud)를 처리하기 위해 맵 자료구조를 사용할 때, 해상도가 높을수록 업데이트 시간이 기하급수적으로 증가하여 실시간성을 보장할 수 없다. 본 논문에서는 실시간 OctoMap 업데이트를 위한 점구름 데이터 양자화 기법을 제안한다. 이 기법은 3차원 화소를 이용하여 점구름 데이터를 양자화한 후, 기존 데이터를 대표하는 새로운 점구름 데이터를 산출한다. 이렇게 양자화 된 점구름 데이터를 이용하면 높은 정확도를 유지하며 실시간으로 맵 업데이트를 할 수 있다. Kinect 센서를 이용하여 실제 데이터를 처리해야하는 상황에서 실험을 진행하였으며, 결과는 제안된 양자화 기법을 사용하지 않았을 때에 비해 최대 45배 빠르게 데이터를 처리하고, 시각적 및 수치적으로 94% 이상의 높은 정확도를 보여준다. 제안된 양자화 기법을 통해 실시간을 보장함으로써 장애물이 움직이는 동적인 환경에 빠르게 반응할 수 있으며, 주어진 같은 시간 내에서 해상도를 높임으로써 모션플래닝이나 충돌탐지의 정확도를 높일 수 있다.

1. 서론

많은 로봇은 3차원 깊이 센서를 이용하여 3차원인 주변 지형에 대한 데이터를 얻고, 이를 처리하여 반응한다. 최근 키넥트(Kinect)나 액션(XTion)과 같이 저렴한 가격의 3차원 깊이 센서가 출시됨으로써 사용량이 증가되는 추세이다. 이러한 센서는 주변 지형 정보를 점집합인 점구름(point cloud) 데이터 형태로 제공하며, 짧은 시간마다 주기적으로 점구름 데이터를 제공한다. 점구름 데이터는 노이즈가 섞인, 많은 수의 3차원 점으로 이루어져 있기 때문에 이를 직접적으로 모션플래닝이나 충돌탐지 등의 응용분야에 적용하기 어렵다. 따라서 점구름 데이터를 효율적으로 처리하기 위해 격자 기반(Grid-based)의 맵 자료구조나 트리 기반(Tree-based)의 맵 자료구조를 사용한다. 하지만 처리해야 할 데이터의 양이 많고 자료구조가 가지는 해상도가 높을수록, 업데이트 시간이 기하급수적으로 증가하여 실시간성을 보장할 수 없다.

본 논문에서는 이를 해결하기 위한 방안으로 3차원 화소(Voxel) 기반의 양자화 기법을 제안한다. 이 기법은 센서로부터 입력된 점구름 데이터를 3차원 화소를 기반으로 양자화 하여, 기존 데이터를 대표하는 새로운 점구름 데이터를 산출한다. 양자화 된 데이터를 OctoMap 자료구조에 업데이트하며, 이를 통해 기존 대비 높은 해상도에서 많은 양의 데이터를 실시간으로 처리할 수 있다.

2. 본론

2.1 OctoMap

점구름 데이터는 많은 수의 3차원 점으로 구성되어 있기 때문에 이를 응용분야에 적합하도록 처리하기 위해 맵 자료구조를 사용한다. 더 나아가 로보틱스 분야에서는 데이터에 존재하는 센서 노이즈를 처리하기 위해 확률적 맵 자료구조를 사용한다. 지금까지 점유 격자 맵(Occupancy Grid Map)[1] 자료구조나 점유 팔진트리(Occupancy Octree)[2] 자료구조를 널리 사용하였으며, 최근에는 팔진트리를 기반으로 하는 OctoMap[3] 자료구조에 대한 연구가 선행되었다.

OctoMap 자료구조는 팔진트리 기반의 자료구조로, 하나의 3차원 공간을 동일하게 8개의 하위 공간으로 나누고 이를 트리 형태로 구성하여 데이터를 저장한다. 저장하는 데이터는 3차원 깊이 센서를 통해 얻을 수 있는 주변 지형 정보로, 센서 노이즈가 존재하기 때문에 이를 고려하여 처리하는 것이 중요하다. 따라서 OctoMap은 센서 노이즈를 확률로 정의하고 있다. 또한, 역 센서 모델[2]과 Log-Odd 함수를 이용하여 노드의 점유 확률을 업데이트하는 식을 아래와 같이 정의한다.

여기에서 $L(n|z_{1:t})$는 1초부터 t초까지 센서 데이터 $z_{1:t}$에 의해 업데이트 된 리프 노드(leaf node) n이 가지는 점유 확률의 Log-Odd 값을 나타낸다. 그리고 $L(n|z_t)$는 t초에 측정된 센서 데이터 z_t를 리

프 노트 n에 업데이트하는 센서의 확률 모델로 식 (2) 와 같이 나타낼 수 있다.

$$L(n|z_{1:t}) = L(n|z_{1:t-1}) + L(n|z_t) \quad (1)$$

$$L(n|z_t) = \begin{cases} l_{occ} & \text{공간이 점유되어 있을 경우} \\ l_{free} & \text{공간이 비어있을 경우} \end{cases} \quad (2)$$

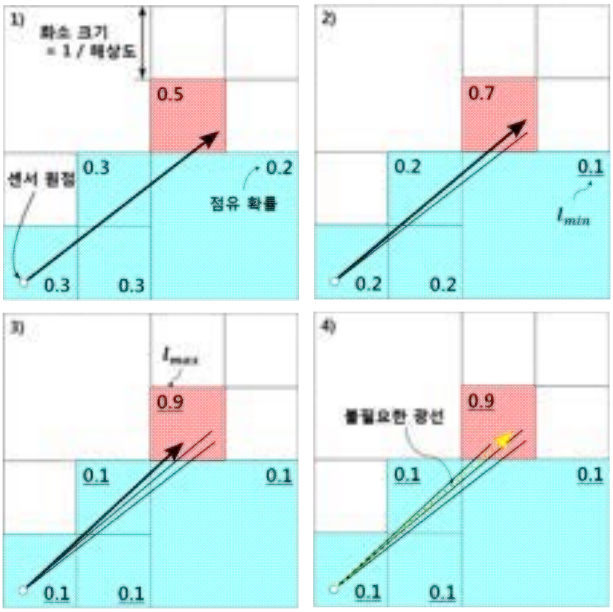
식 (1)의 역 센서 모델을 사용하는 경우, 많은 수의 센서 데이터로 인해 과신뢰(over-confidence) 문제가 발생할 수 있다. OctoMap에서는 이 문제를 해결하기 위해, 공간이 비어있다고 간주하는 확률($l_{\min}$)과 점유되어있다고 간주하는 확률($l_{\max}$)을 정의하며, 이 값을 넘지 않도록 하는 자름 방법(clamping policy)[4]을 사용한다. 이는 아래 (3)와 같이 표현된다.

$$L(n|z_{1:t}) \quad (3)$$
$$= \max(\min(L(n|z_{1:t-1}) + L(n|z_t)), l_{\max}), l_{\min})$$

2.2 불필요한 광선

공간에 하나의 점 데이터가 존재한다는 것은 센서 원점과 점 데이터 사이의 공간에 장애물이 없어 공간이 비어있다는 것으로 생각할 수 있다. 따라서 센서의 원점에서 발사되어 점 데이터를 향하는 광선을 정의 할 수 있으며, 이때 점 데이터는 광선의 끝점과 같다. OctoMap에서는 업데이트 시 모든 점 데이터에 대해 광선을 정의하고, Bresenham 광선 추적법 알고리즘[5]을 이용하여 광선이 통과하는 노드와 도달하는 노드를 계산한다. 이후 통과하는 노드는 공간이 비어있을 확률(free probability, l_{free})로 업데이트를 하고, 도달하는 노드는 공간이 점유되어있을 확률(occupied probability, l_{occ})로 업데이트를 한다.

그림 1은 총 4개의 점 데이터로 이루어진 점구름 데이터를 OctoMap에 업데이트하는 과정을 나타낸 그림으로, 각 사각형은 OctoMap의 리프 노트를 나타낸다. 파란색 노드는 l_{free}로 업데이트되는 노드이며 빨간색 노드는 l_{occ}로 업데이트되는 노드이다. 그리고 각 노드 안에 표현되어 있는 숫자는 점유 확률 $L(n|z_{1:t})$을 나타내고 있다. 업데이트 시 노드의 점유 확률이 $l_{\max}$에 도달하면 해당 노드는 점유되어 있다고 정의하며, 노드의 점유 확률이 $l_{\min}$에 도달하면 해당 노드는 비어있다고 정의한다. 그림 1에서 점유 확률이 $l_{\max}$ 나 $l_{\min}$에 도달한 노드를 표시하기 위해 점유확률에 밑줄을 그어 나타내고 있다.



[그림 1] 점구름 데이터의 업데이트 과정. 1)부터 4)번까지 새로운 광선을 추가하고, 이때 업데이트 되는 점유 확률을 나타냄. 이 예제에 사용된 Eq. (2) 및 Eq. (3)에 필요한 상수 값은 아래와 같음. $l_{occ} = 0.2$, $l_{free} = -0.1$, $l_{\max} = 0.9$, $l_{\min} = 0.1$

이때, 자름 방법을 사용할 경우, 그림 1의 4)와 같이 노드의 점유 확률에 영향을 주지 않는 광선(노란색 점선)이 존재할 수 있으며, 본 논문에서는 이러한 광선을 불필요한 광선이라고 정의한다.

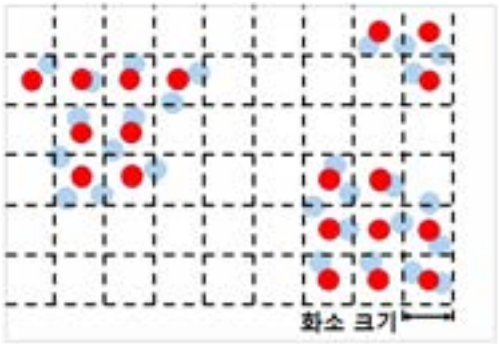
주어진 점구름 데이터를 OctoMap에 업데이트 할 때, 모든 데이터에 대해 광선을 정의하고 그 광선이 지나는 노드를 계산해야 하므로 불필요한 광선을 제거하기 위해서는 불필요한 점 데이터를 제거하는 것이 중요하다. 따라서 본 논문에서는 3차원 화소를 기반으로 점구름 데이터를 양자화 하여, 기존 데이터를 대표하는 새로운 점구름 데이터를 산출하는 3차원 화소 기반의 양자화 기법을 제안한다.

2.3 3차원 화소 기반의 양자화

불필요한 광선이 될 가능성이 있는 점 데이터 집합은 공간적으로 인접한 위치에 존재한다. 따라서 불필요한 광선을 제거하고, 이를 대표하는 광선을 계산하기 위해서 점 데이터를 공간적으로 분류해야 한다. 또한 제거된 데이터를 대표하는 데이터는 기존 데이터가 업데이트하는 노드와 공간적으로 인접한 노드를 업데이트해야한다.

이를 반영하여 본 논문에서 제안하는 기법에서는 공간적으로 다른 데이터를 묶을 수 있는 3차원 화소를 이용한다. 또한, OctoMap에서 사용하는 광선 추

적법 알고리즘도 3차원 화소를 사용하기 때문에 이를 양자화 단계에 적용하면, 대표하는 점 데이터를 업데이트에 이용할 경우에 기존 점 데이터들이 업데이트하는 노드들과 인접한 노드들을 업데이트 할 수 있다. 본 논문에서 제안하는 기법은 이를 토대로 점 데이터가 존재하는 3차원 화소의 중심점을 대표 데이터로 정의하고, 센서의 원점으로부터 새로운 대표 데이터로 향하는 광선을 대표 광선으로 정의한다.



[그림 2] 3차원 화소기반의 양자화

그림 2와 알고리즘 1은 본 논문에서 제안하는 3차원 화소 기반의 양자화 기법에 대한 개념도 및 의사코드를 나타낸 것이다. 먼저 센서 좌표계에 존재하는 점구름 데이터(그림 2의 파란색 점)의 범위를 계산(1라인)한 후, 이 범위를 미리 정의되어 있는 OctoMap의 키 공간으로 변환(2라인)한다. 여기에서 키 공간은 빠른 광선 추적 알고리즘 처리 및 빠른 노드 접근을 위해 정의된 공간이다. 이 공간으로 변환된 점구름 데이터를 3차원 격자로 나누어 3차원 화소로 이루어진 양자화 공간을 생성(3-4라인)한다. 이 공간은 제거할 데이터의 존재 여부를 저장할 수 있다. 이를 위해 모든 점 데이터에 대해 각각의 점 데이터를 양자화 공간으로 변환(6-7라인)한 후, 이 공간에서 점 데이터의 존재 여부를 저장(8라인)한다. 마지막으로 모든 양자화 공간에 대해 점 데이터가 존재하는지 여부를 확인(11라인)하고, 만약 점 데이터가 존재하였다면 해당하는 양자화 공간의 3차원 화소 중점을 센서 좌표계로 변환하여 저장(12-13라인)한다. 이 변환된 점을 대표 데이터로 정의하며, 이를 통해 제거된 점구름 데이터를 대표하는 새로운 점구름 데이터(그림 2의 빨간색 점)를 산출한다.

새로운 점구름 데이터는 기존 점구름 데이터를 대표하는 값이므로, 더 큰 값의 점유하고 있을 확률

[알고리즘 1] 3차원 화소 기반의 양자화

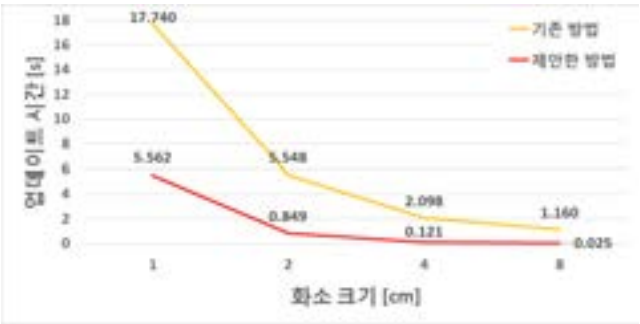
```
1  min, max ← MinMax(pointcloud)
2  minK, maxK ← coordToKey(min, max)
3  sizeK ← maxK - minK
4  voxel[sizeK.X * sizeK.Y * sizeK.Z] ← false
5  for i ← 1 to N // the number of points
6      pntK ← coordToKey(pointcloud[i])
7      idx ← Indexing(pntK.X, pntK.Y, pntK.Z)
8      voxel[idx] ← true
9  end
10  for i ← 1 to M // the number of voxels
11     if voxel[M] == true
12         cntK ← center(voxel[M])
13         newpoint ← push(keyToCoord(cntK))
14     end
15 end
```

($^{old}l_{occ} < {}^{new}l_{occ}$)과 더 작은 값의 비어있을 확률 ($^{old}l_{free} > {}^{new}l_{free}$)을 사용한다.

3. 결론

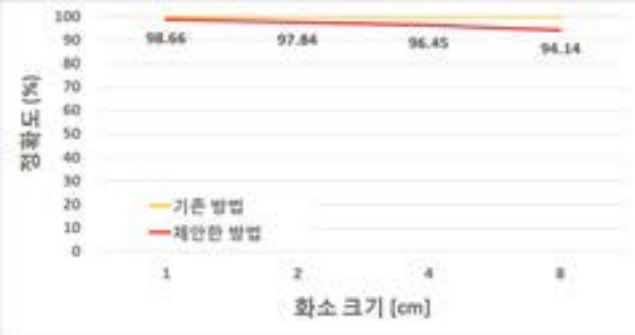
3.1 결과

3차원 점구름 데이터를 얻기 위해 상용 3차원 깊이 센서인 Kinect를 이용하였다. Kinect 센서로부터 매 프레임(frame)마다 30만개의 점구름 데이터를 획득할 수 있으며, 이를 OctoMap의 해상도에 따라 업데이트하는 실험을 수행하였다.



[그림 3] 화소 크기에 대한 업데이트 시간

그림 3의 그래프를 보면 해상도가 커짐에 따라 업데이트 시간이 기하급수적으로 증가하여 실시간이 보장되지 않는 것을 확인할 수 있다. 본 논문에서 제안한 방법은 기존 방법에 비해 최대 45배 빠르다. 또한 화소 크기가 8cm 경우에 업데이트 시간이



[그림 4] 화소 크기에 대한 정확도

25ms로, 센서의 입력 주기(30Hz=33ms) 내에 실시간으로 동작한다는 것을 보여준다.

제안한 양자화 기법을 통해 데이터의 수를 감소하였으므로 이에 따라 업데이트 시간이 줄어들게 된다. 더 나아가 그림 4에서 볼 수 있듯이 정확도는 모두 94%이상으로 유지하며 높은 신뢰도를 보여준다. 그림 4의 그래프에서 정확도는 기존 방법으로 업데이트 결과를 기준으로 삼아 같은 상태를 가지는 노드의 비율로 정의하였다. 이와 같은 결과는 3차원 화소를 이용하여 인접한 공간에 존재하는 데이터를 묶을 수 있었기 때문에 가능하였다. 또한, OctoMap에서 사용하는 키 공간을 이용하여 대표 데이터를 산출하였기 때문에, 기존에 업데이트를 해야 할 노드와 인접한 노드에 업데이트하였기에 가능하였다. 이 결과는 시각적으로도 비슷하다는 것을 그림 5에서 확인할 수 있다.

3.2 결론

본 논문에서는 불필요한 데이터를 제거하고 대표 데이터를 산출하여 실시간 성능을 보장할 수 있는 3차원 화소기반의 양자화 기법을 제안하였다. 본 논문에서 제안한 기법을 사용하지 않는 경우에 비해 정확도가 6% 정도 낮아지지만 최대 45배 빠른 처리 시간을 보장할 수 있었다. 앞으로 이 연구에서 더 나아가 양자화 된 데이터의 수에 따라, 노드에 업데이트하는 중요도를 변화시켜 높은 정확도를 유지시키는 방법에 대해 연구할 예정이다.

- 본 연구는 미래창조과학부 및 정보통신기술연구진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음. [R0126-15-1108]

참고문헌

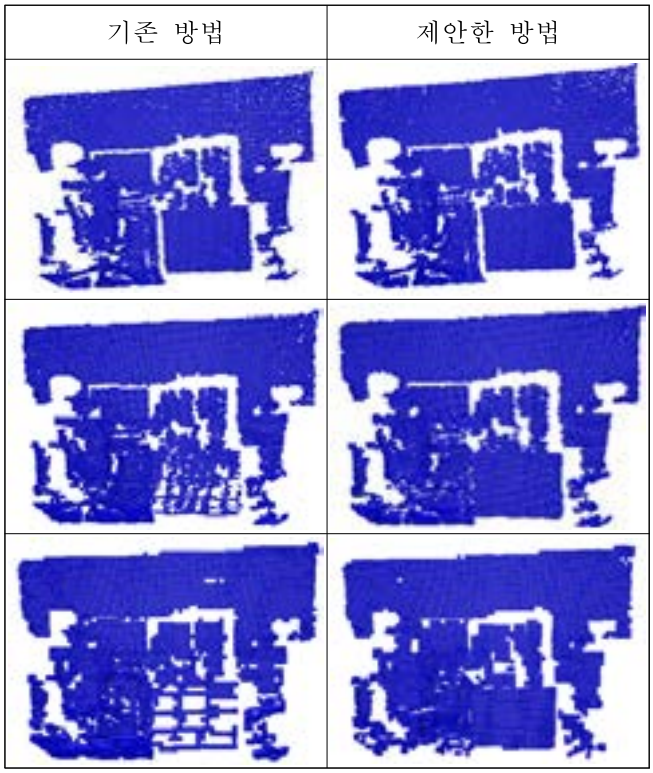
[1] H. Moravec and A. Elfes, “High resolution maps from wide angle sonar”, in Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), pp, 116-121, 1985.

[2] P. Payer, P. Hebert, D. Laurendeau, and C. Gosselin, “Probabilistic octree modeling of a 3-d dynamic environment”, in Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 1997.

[3] Kai M. Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard, “OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems”, in Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2010.

[4] M. Yguel, O. Aycard, and C. Laugier, “Update policy of dense maps: Efficient algorithms and sparse representation”, in Field and Service Robotics, Results of the Int. Conf. FSR, pp 23-33, 2007.

[5] J. Amanatides and A. Woo, “A fast voxel traversal algorithm for ray tracing”, in Proceedings of Eurographics, 1987.



[그림 5] 화소 크기에 대한 시각적 결과
위에서부터 차례대로 1cm / 2cm / 4cm