Original Article

https://doi.org/10.12985/ksaa.2019.27.2.001
ISSN 1225-9705(print) ISSN 2466-1791(online)

최적 경로 계획을 위한 RRT*-Smart 알고리즘의 개선과 2, 3차원 환경에서의 적용

탁형태*, 박천건**, 이상철***

Improvement of RRT*-Smart Algorithm for Optimal Path Planning and Application of the Algorithm in 2 & 3-Dimension Environment

Hyeong-Tae Tak*, Cheon-Geon Park**, Sang-Chul Lee***

ABSTRACT

Optimal path planning refers to find the safe route to the destination at a low cost, is a major problem with regard to autonomous navigation. Sampling Based Planning(SBP) approaches, such as Rapidly-exploring Random Tree Star(RRT*), are the most influential algorithm in path planning due to their relatively small calculations and scalability to high-dimensional problems. RRT*-Smart introduced path optimization and biased sampling techniques into RRT* to increase convergent rate. This paper presents an improvement plan that has changed the biased sampling method to increase the initial convergent rate of the RRT*-Smart, which is specified as mRRT*-Smart. With comparison among RRT*, RRT*-Smart and mRRT*-Smart in 2 & 3-D environments, mRRT*-Smart showed similar or increased initial convergent rate than RRT* and RRT*-Smart.

Key Words : Optimal Path Planning(최적 경로 계획), Rapidly-exploring Random Trees star(RRT*), RRT*-Smart, Biased sampling(편향 샘플링), Initial convergent rate(초기 수렴 속도)

Ⅰ. 서 론

최적 경로 계획(Optimal path planning) 의 목적은 시작 지점과 목표 지점 사이에서 장애물과의 충돌이 없는 가장 짧은 경로를 찾는 것이다. 이는 운송, 탐색 및 구조 등의 분야에서 효율적인 유무인기의 운용을 위해 중요하다. 특히 각종 군사목적과 상업목적으로 각광받고 있는 무인 멀티콥터의 자율주행에 있어 경로 계획 문제의 해결은 필수적인 요소이다. Noreen(2016)에 따르면 대표적인 최적 경로 계획 알고리즘으로 Probability Roadmaps(PRM), Rapidly-exploring Random Tree(RRT)[1]와 같은 샘플링 기반의 알고리즘을 들 수 있다[2].

Rapidly-exploring Random Tree Star(RRT*)[3]는 RRT로부터 생성되는 경로가 최적 경로로의 수렴을 보장하지 않는 한계를 극복하기 위해 개발되었다. Nasir(2013)등에 따르면 삼각부등식

Received : 10. Jun. 2019. Revised : 18. Jun. 2019.
Accepted : 25. Jun. 2019
* 한국항공대학교 학부 항공우주 및 기계공학과
** 한국항공대학교 대학원 항공우주 및 기계공학과
*** 한국항공대학교 항공우주 및 기계공학부
연락저자 E-mail : slee@kau.ac.kr
연락저자 주소 : 경기도 고양시 덕양구 항공대학로 76

조건을 이용한 경로 최적화와 특정 영역 내에서 샘플링을 실시하는 편향 샘플링 기법을 적용하여 RRT*의 수렴 속도를 증가시킨 RRT*-Smart를 개발하였다[4]. 이희범(2014)은 RRT*-Smart의 성능을 높이기 위해 반복 수(Iteration)가 증가함에 따라 점차 편향 샘플링을 시행하는 비율이 늘어나도록 변경하였다[5].

RRT* 기반의 알고리즘은 반복 수 증가에 따라 점진적으로 최적 경로로 수렴한다. 온라인 경로 계획의 경우 동적인 환경 속에서 빠른 경로 생성을 필요로 한다. 또한 실제 알고리즘 적용 시 계산 능력의 제약으로 인해 높은 초기 수렴 속도가 요구된다. 따라서 상대적으로 적은 반복수로 낮은 비용의 경로를 찾는 것이 바람직 하다.

본 논문에서는 RRT*-Smart의 초기 수렴 속도 증가를 위해 편향 샘플링 방식을 개선한 mRRT*-Smart를 제안하고 2, 3차원에서의 적용에 대한 연구를 수행하였다. 또한 임의로 설정된 2-D, 3-D 환경에서 시뮬레이션을 진행하여 얻은 경로의 평균 비용을 기준으로 개선안과 RRT*, RRT*-Smart의 성능을 비교하였다.

II. 본 론

2.1 알고리즘

2.1.1 RRT* 알고리즘

RRT*는 노드(Node) 집합 $x \in X$를 상태공간으로 정의할 때 시작 노드 x_{init}에서 목표 영역 $x_{goal} \in X_{goal}$로의 경로 생성을 목표로 한다. 알고리즘의 결과로 생성되는 $T=(V,E)$는 노드와 모서리의 집합으로, V는 장애물이 없는 공간에 생성되어 충돌 없이 T에 추가된 노드를 의미하고 E는 노드들을 잇는 모서리를 의미한다.

RRT* 알고리즘의 과정을 Figure 1에 나타내었고 Figure 2는 RRT*의 pseudo code를 나타낸 것이다. Figure 1. a)~b)에서 보이듯이, $RandomSample$로 생성된 노드 (x_{rand})가 V의 노드 중 $Nearest$ 함수로 찾아진 가장 가까운 노드($x_{nearest}$)와 충돌 없이 연결 가능하다면 V에 새로운 노드(x_{new})로 추가한다. 추가된 노드로부터 Figure 1. c)~d)와 같이 특정 반경 이내의 노드들을 X_{near}로 설정하고, 이 중 x_{init}로부터 x_{new}까지 경로의 비용(Cost)이 최소가 되도록 하는 노드(②)를 부모 노드로 재 연결한다. 그 후 Figure 1. e)~f)와 같이 X_{near}의 노드 중 x_{new}를 부모 노드로 할 때 최소 비용이 되는 노드(⑤)를 재 연결한다.

RRT*는 Figure 1. c)~f)의 재 연결 기법을 통해 Figure 1. a)~b)의 과정으로만 이루어진 RRT와 달리 비용정보를 기준으로 경로를 점근적으로 최적에 수렴하도록 한다는 데 의미가 있다.

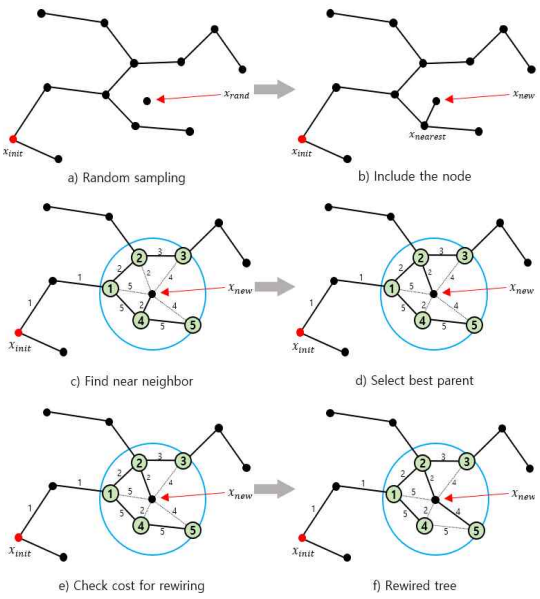


Fig 1. RRT, RRT* Algorithm

Algorithm 1: RRT* pseudo code

Input : x_{init}, X_{goal}
Output: Tree $T = (V, E)$

$V \leftarrow x_{init}$, $E \leftarrow \emptyset$;
for $i = 1, ..., N$ **do**
  $x_{rand} \leftarrow$ RandomSample(i);
  $x_{nearest} \leftarrow$ Nearest(T, x_{rand});
  $x_{new} \leftarrow$ Steer($x_{nearest}, x_{rand}$);
  if ObstacleFree($x_{nearest}, x_{new}$) **then**
    $V \leftarrow V \cup \{x_{new}\}$;
    $X_{near} \leftarrow$ Near(T, x_{new}, r);
    ChooseParent ($X_{near}, x_{nearest}, x_{new}, E$);
    Rewiring (X_{near}, x_{new}, E);
  end
end

Fig 2. RRT* pseudo code

- $RandomSample(i)$: 상태공간(X)에서 무작위로 노드(x_{rand})를 뽑는다.
- $Nearest(T,x_{rand})$: Euclidean norm을 이용하여 V에서 x_{rand}로부터 가장 가까운 노드를 반환한다.
- $Steer(x_{nearest}, x_{\neq w})$: $x_{nearest}$로부터 x_{new}로의 비용을 유도한다.
- $ObstacleFree(x_{nearest},x_{new})$: $x_{nearest}$와 x_{new}가 장애물과의 충돌 없이 이어지는지 확인하여 참 또는 거짓 값을 반환한다.
- $Near(T,x_{new},r)$: x_{new}로부터 특정 반경(r)이내에 존재하는 노드들을 반환한다.
- $ChooseParent(X_{near},x_{nearest},x_{new},E)$: X_{near}의 노드들 중 x_{new}와 연결 시 x_{new}로의 경로 비용이 최소가 되도록 하는 노드를 x_{new}의 부모 노드로 설정한다.
- $Rewiring(X_{near},x_{new},E)$: X_{near}의 노드들 중 x_{new}를 부모 노드로 설정 할 시 최소 비용이 되는 노드들의 부모 노드를 수정한다.

2.1.2 RRT*-Smart 알고리즘

RRT*-Smart의 알고리즘을 Figure 4에 나타내었다. RRT*-Smart는 처음 경로가 찾아지기 전까지 RRT*와 동일한 과정을 거쳐 노드들을 추가한다. 처음으로 경로가 찾아지면 전체 알고리즘이 실행된 횟수인 반복 수(i)를 n으로 저장한다. 경로가 찾아진 이후부터 Figure 3과 같이 삼각부등식 조건을 사용하여 경로 최적화($Path\,Optimization$)를 실시한다. 최적화된 경로가 기존 최적화 경로보다 낮은 비용을 가질 경우 해당 경로상의 노드들(x_1, x_2, x_3)을 비콘 노드로 설정한다. n이후의 반복 수부터 특정 반복 수(Biasing ratio; b)마다 편향 샘플링($RandomSample(i,x_{beacons})$)이 진행된다. 편향 샘플링은 저장 된 비콘 노드들 중 임의로 선택된 하나의 비콘 노드로부터 특정 반경(Biasing Radius) 이내의 원형 영역에서 랜덤으로 노드를 생성하여 수렴 속도를 증가시킨다.

RRT*-Smart의 성능을 결정하는 파라미터는 biasing radius와 biasing ratio이다. Biasing radius는 편향 샘플링이 이루어지는 원형 영역의 반경이며, Biasing ratio는 편향 샘플링이 이루어지는 비율을 결정하는 값이다.

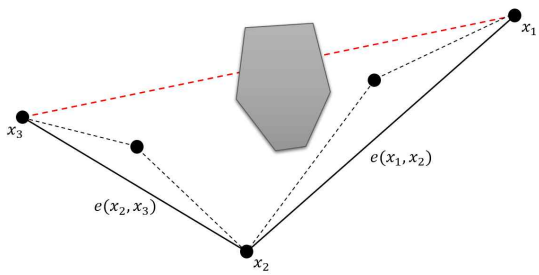


Fig 3. Path Optimization

Algorithm 2: mRRT*- Smart pseudo code

Input : x_{init}, X_{goal}
Output: Tree $T = (V, E)$

$V \leftarrow x_{init}$, $E \leftarrow \emptyset$;
for $i = 1, ..., N$ **do**
  if $i = n + b, n + 2b, ...$ **then**
    $x_{rand} \leftarrow$ RandomSample$(i, x_{beacons})$;
  else
    $x_{rand} \leftarrow$ RandomSample(i);
  end
  $x_{nearest} \leftarrow$ Nearest(T, x_{rand});
  $x_{new} \leftarrow$ Steer$(x_{nearest}, x_{rand})$;
  if ObstacleFree$(x_{nearest}, x_{new})$ **then**
    $V \leftarrow V \cup \{x_{new}\}$;
    $X_{near} \leftarrow$ Near(T, x_{new}, r);
    ChooseParent$(X_{near}, x_{nearest}, x_{new}, E)$;
    Rewiring(X_{near}, x_{new}, E);
    if InitialPathFound **then**
      $n \leftarrow i$;
    end
    $(T, optcost) \leftarrow$ PathOptimization(T, x_{init}, x_{goal});
    if $optcost_{new} < optcost_{old}$ **then**
      $x_{beacons} \leftarrow$ PathOptimization(T, x_{init}, x_{goal});
    end
  end
end

Fig 4. mRRT*-Smart pseudo code

- $RandomSample(i,x_{beacons})$: 비콘 노드($x_{beacons}$)에서 임의로 하나를 선정하고 해당 비콘 노드를 기준으로 편향 샘플링을 실시한다.
- $InialPathFound$: x_{init}로부터 x_{goal}로의 경로가 처음으로 찾아졌는지 확인하고 참 또는 거짓을 반환한다.
- $PathOptimization(T,x_{init},x_{goal})$: x_{init}와 x_{goal}사이의 경로를 최적화하고 최적화된 경로의 비용, 노드를 반환한다.

2.1.3 mRRT*-Smart 알고리즘

RRT* Smart에서 경로 최적화는 비콘 노드 사이의 장애물에 의해 중단된다. 따라서 경로 최적화가 완료되면 연속된 세 비콘 노드 (x_1, x_2, x_3)로 만들어지는 삼각형 내부에 최종적으로 수렴해야 할 지점이 존재하게 된다. 이에 착안하여 본 논문은 해당 지점으로 빠르게 수렴하도록 편향 샘플링 영역을 변경하였다.

개선안은 경로 최적화를 동일하게 실시한 뒤, 편향 샘플링이 이루어지는 반복 수마다 임의의 연속된 세 비콘 노드(x_1, x_2, x_3)를 선택한다. 그 후 Figure 3과 비콘 노드 간의 모서리 $e(x_1,x_2)$, $e(x_2,x_3)$을 포함하는 두 직선을 각각 x_1과 x_3을 기준으로 일정 각도씩 회전 시키며 맞은 편 모서리로의 교선을 생성한다. 생성된 교선이 장애물과의 충돌이 발생할 때까지 위 과정을 반복하고 충돌이 발생하면 충돌 직전의 마지막 교선들의 교점을 샘플링이 이루어지는 비콘 노드(x_{beacon})로 설정하고 특정 반경 이내의 영역에서 샘플링을 실시한다.

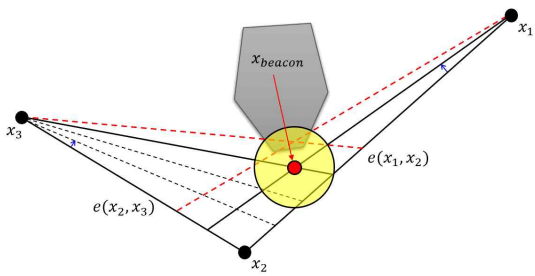


Fig 5. Modified beacon node

2.2 시뮬레이션

2.2.1 시뮬레이션 설정

RRT*, RRT*-Smart, mRRT*-Smart을 비교하였다. 2-D 환경은 하나의 사각형 장애물이 존재하는 단순한 환경과 Nasir(2013)가 사용하였던 환경[4] 중 임의로 세 가지 환경을 선택하여 총 4가지의 환경을 설정하였다. 3-D 환경은 일반적으로 접할 수 있는 계단 환경으로, 3-D 환경에서 편향 샘플링 영역은 구형으로 설정하였다. 경로의 비용은 Euclidean norm을 이용한 이동거리로 설정하였다. Table 1은 시뮬레이션을 진행한 각 환경의 크기, 최대 반복 수와 시뮬레이션 실행 횟수를 나타낸다.

알고리즘 구현은 C++를 이용하였고 AMD Ryzen 5 2600 3.4GHz, 16.00GB ram, VGA Radeon RX 580의 사양에서 시뮬레이션을 수행하였다.

Table 1. Simulation setting

환경	크기	최대 반복수	시뮬레이션 실행 횟수
2-D Case1	100×100	20,000	5
2-D Case2			
2-D Case3			
2-D Case4			
3-D Case	200×100×145	15,000	10

2.2.2 시뮬레이션 결과

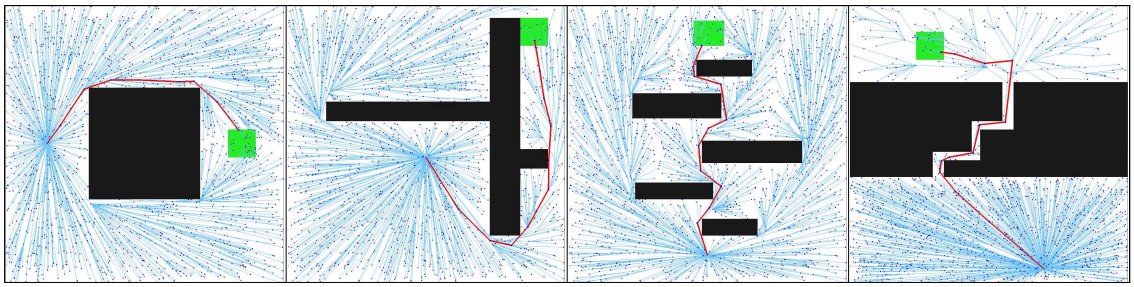
2.2.2.1 2차원 시뮬레이션 결과

2-D 환경에서 시뮬레이션 시 편향 샘플링 조건은 Table 2와 같은 값들로 설정하였다.

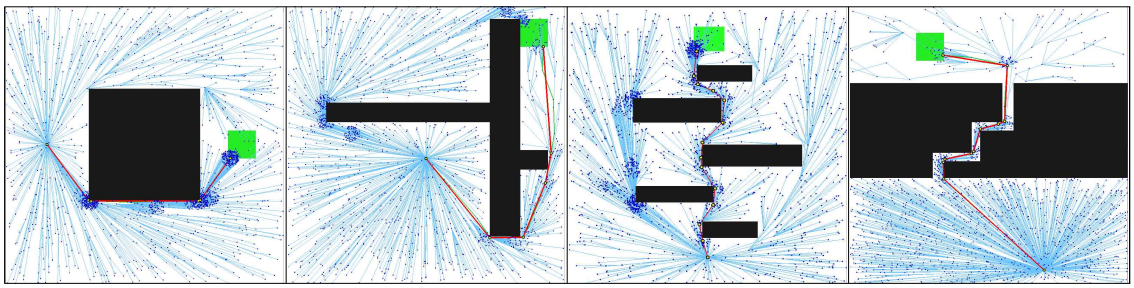
Table 2. Bias sampling setting

	Biasing ratio	Biasing radius
RRT*-Smart	1~5	1~9
mRRT*-Smart	1~5	0.1~9

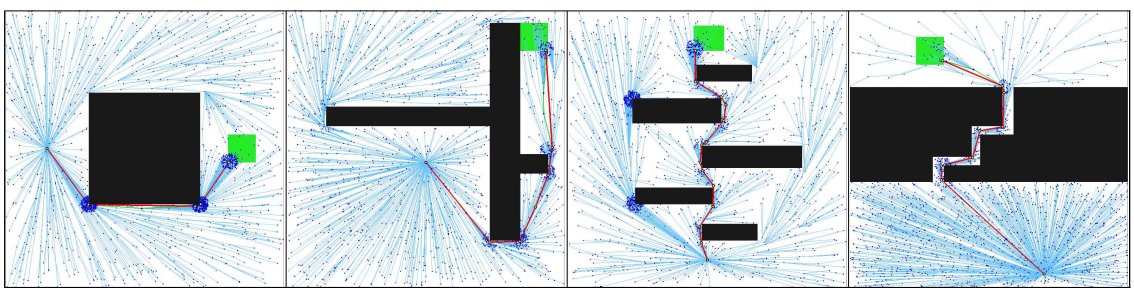
Figure 6는 Case 1, 2, 3에 대하여 반복 수 2,000일 때, Case 4에 대하여 반복 수가 3,500일 때 시뮬레이션 결과이다. 여기서 biasing radius 값의 경우 RRT*-Smart가 가장 좋은 결과를 나타내는 값인 3을 선택하여 mRRT*-Smart와 비교하였다. 또한 biasing ratio 값의 경우 RRT*-Smart와 mRRT*-Smart의 특성이 가장 잘 나타나는 값인 2를 선택하여 비교하였다. Figure 7은 각 2-D 환경별로 20,000번의 반복 수(i)에 대한 비용의 그래프를 나타낸다. Table 3은 임의로 설정한 초기 반복 수에서의 알고리즘별 평균 비용을 나타낸다. 2-D Case 4의 경우 좁은 길을 통과해야 하는 문제점으로 인해 2916번째의 반복 수에서 최초 경로가 찾아지므로 다른 환경보다 높은 반복 수를 기준으로 비교하였다.



(a) Case 1 (n=2000) (b) Case 2 (n=2000) (c) Case 3 (n=2000) (d) Case 4 (n=3500)

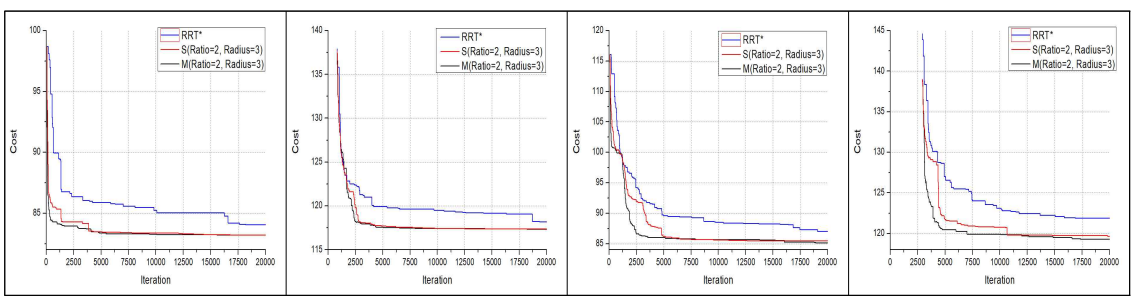


(e) Case 1 (n=2000) (f) Case 2 (n=2000) (g) Case 3 (n=2000) (h) Case 4 (n=3500)



(i) Case 1 (n=2000) (j) Case 2 (n=2000) (k) Case 3 (n=2000) (l) Case 4 (n=3500)

Fig 6. A comparison of 2-D simulation results. RRT* is shown in (a)-(d), RRT*-Smart is shown in (e)-(f) and mRRT*-Smart is shown in (i)-(l)



(a) 2-D Case 1 (b) 2-D Case 2 (c) 2-D case 3 (d) 2-D case 4

Fig 7. Costs against iterations(S=RRT*-Smart, M=mRRT*-Smart)

Table 3. Path planning results at 1000, 1500, 2000 and 2500 iterations for 2-D case 1, 2, 3 & 3000, 3500, 4000, and 4500 iterations for 2-D case 4

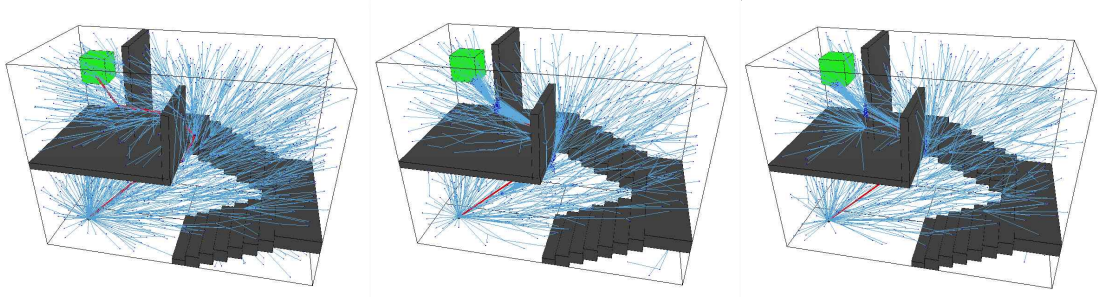
환경	알고리즘	Biasing Ratio	Biasing Radius	평균 비용 (n=1000)	평균 비용 (n=1500)	평균 비용 (n=2000)	평균 비용 (n=2500)	최적 비용
2-D Case1	RRT*	×		89.94	86.75	86.7	86.36	83.03
	RRT*-Smart	2	3	85.35	84.30	84.27	84.27	
	mRRT*-Smart			84.12	83.99	83.95	83.95	
2-D Case2	RRT*	×		135.8	123.9	122.5	122.4	117.2
	RRT*-Smart	2	3	130.2	123.8	121.6	119.8	
	mRRT*-Smart			129.5	124.3	120.9	118.2	
2-D Case3	RRT*	×		100.0	97.8	96.6	94.2	84.9
	RRT*-Smart	2	3	99.8	95.8	92.7	91.9	
	mRRT*-Smart			99.7	92.4	88.4	86.6	
2-D Case4	RRT*	×		143.9	132.5	130.1	128.7	118.8
	RRT*-Smart	2	3	136.4	129.5	128.9	123.2	
	mRRT*-Smart			135.9	124.3	121.9	120.8	

Figure 6. (e), (f), (g), (h)에서 RRT*-Smart는 장애물의 모서리가 아닌 지점에서도 집중적으로 샘플링이 이루어졌다. 반면에 mRRT*-Smart는 Figure 6. (i), (j), (k), (l)과 같이 장애물의 모서리 근처에서만 샘플링이 이루어진 것을 확인 할 수 있다. 이는 mRRT*-Smart가 적은 반복 수로도 장애물의 모서리를 빠르게 탐지함으로써 초기 수렴 속도가 RRT*-Smart보다 더 빠르게 수렴하도록 한다.

Figure 7와 Table 3으로부터 2-D 환경의 경우 mRRT*-Smart의 초기 수렴 속도가 RRT*, RRT*-Smart보다 비슷하거나 빠른 모습을 보이는 것을 알 수 있다.

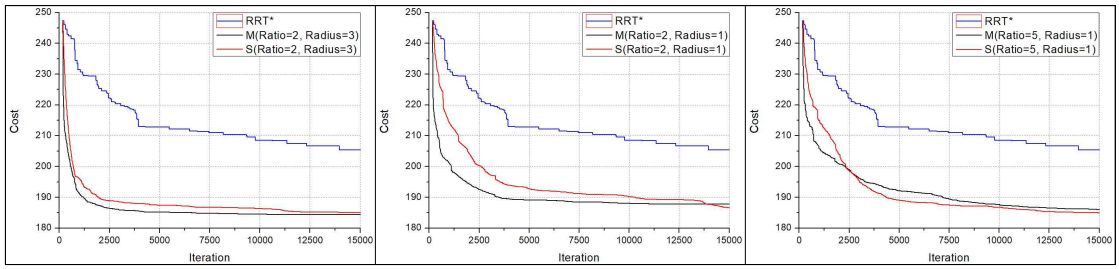
2.2.2.2 3차원 시뮬레이션 결과

Figure 8은 biasing ratio는 2, biasing radius는 3으로 설정하고 3-D Case에 대하여 반복 수가 2000일 때 시뮬레이션 결과이다. Figure 9는 3가지 조합의 biasing ratio, biasing radius의 경우에서 15000번의 반복 수에 대한 비용의 그래프를 나타낸다. Table 4는 임의로 설정한 초기 반복 수에서의 알고리즘별 평균 비용을 나타낸다. Figure 10은 반복 수가 2000일 때 3-D Case에서 생성된 노드들을 알고리즘별로 biasing ratio는 2, biasing radius는 1과 3인 경우를 나타낸 것이다.



(a) RRT* (b) RRT*-Smart (c) mRRT*-Smart

Fig 8. A comparison of 3-D simulation results at n=2000



(a) 3D-case(Ratio=2, Radius=3) (b) 3D-case(Ratio=2, Radius=1) (c) 3D-case(Ratio=5, Radius=1)

Fig 9. Costs against iterations(S=RRT*-Smart, M=mRRT*-Smart)

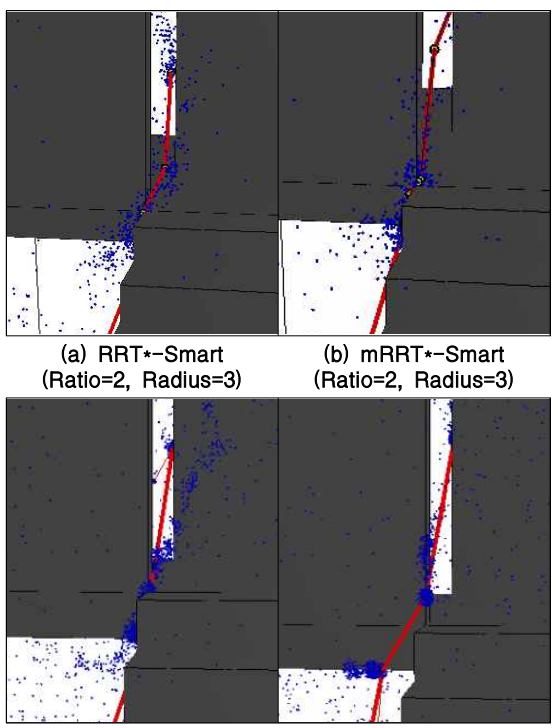
Table 4. Path planning results at 1000, 1500, 2000 and 2500 iterations for 3-D Case

환경	알고리즘	Biasing Ratio	Biasing Radius	평균 비용 (n=1000)	평균 비용 (n=1500)	평균 비용 (n=2000)	평균 비용 (n=2500)	최적 비용
3-D Case	RRT*	×		231.5	229.4	225.4	222.3	184.0
	RRT*-Smart	2	3	196.1	192.5	189.7	188.8	
	mRRT*-Smart			191.2	188.5	187.4	186.5	

Table 4의 평균 비용으로부터 mRRT*-Smart, RRT*-Smart, RRT*순으로 낮은 비용의 경로가 찾아진 것을 볼 수 있다. 또한 Figure 9를 보면, 전체 반복수에서 mRRT*-Smart의 초기 수렴속도가 RRT*와 RRT*-Smart보다 빠른 것을 확인 할 수 있다. Figure 10. (a), (c)에서 2-D환경과 마찬가지로 RRT*-Smart는 장애물의 모서리가 아닌 지점에서도 집중적으로 샘플링이 이루어졌다. 반면에 mRRT*-Smart는 Figure 10. (b), (d)와 같이 장애물의 모서리 근처에서만 샘플링이 이루어진 것을 확인 할 수 있다.

Figure 9. (b)와 같이 biasing ratio와 biasing radius를 모두 작게 설정할 경우 mRRT*-Smart로부터 얻어진 경로의 비용이 최적 경로 비용에 근접하지 않은 값(187.83)으로 수렴이 이루어졌다. 그 결과 13811회의 반복 수 이후부터 RRT*-Smart이 더 빠르게 수렴하는 것으로 나타났다. Figure 9. (c)와 같이 biasing ratio를 높게, biasing radius를 작게 설정할 경우 최종 수렴 값(186.00)은 최적 경로에 더 근접하였다. 그러나 초기 수렴 속도가 느려져 2356회 이후로 RRT*-Smart의 성능이 더 좋게 나타나는 것을 확인 할 수 있었다.

3-D 환경에서 Biasing radius를 작게 설정할 경우 mRRT*-Smart의 문제점은 Figure 10. (c), (d)를 통해 확인 할 수 있다. mRRT*-Smart의 경우 연속된 세 노드로 만들어지는 삼각형 내부의



(a) RRT*-Smart (Ratio=2, Radius=3) (b) mRRT*-Smart (Ratio=2, Radius=3)

(c) RRT*-Smart (Ratio=2, Radius=1) (d) mRRT*-Smart (Ratio=2, Radius=1)

Fig 10. Nodes at 2000 iterations

장애물 근처에 편향 샘플 영역을 설정한다. 2-D 환경의 경우 최적 경로가 삼각형 내부 장애물의 모서리를 지나가지만 3-D 환경의 경우 최적 경

로가 삼각형 내부 장애물의 모서리를 지나가지 않는다. mRRT*-Smart는 장애물 근처로 빠르게 다가가므로 높은 초기 수렴 속도를 가질 수 있지만 다가간 장애물의 모서리가 최적 경로로부터 멀리 위치한다면 biasing radius가 작을 경우 편향 샘플링이 경로 개선에 기여 할 여지가 적어져 수렴 속도가 느려지게 된다.

Ⅲ. 결 론

본 논문에서는 장애물 정보를 활용하여 RRT*-Smart의 편향 샘플링 단계에서 샘플링 영역을 변경한 개선안인 mRRT*-Smart를 제안하였다. 2, 3차원의 환경에서 기존 알고리즘인 RRT*, RRT*-Smart와 개선안인 mRRT*-Smart를 이용하여 시뮬레이션을 수행하였다. 시뮬레이션 결과, 2차원 환경에서 기존 알고리즘인 RRT*, RRT*-Smart보다 mRRT*-Smart의 초기 수렴 속도가 빠른 것을 확인하였다. 3차원의 환경은 Biasing radius가 충분히 큰 경우에 mRRT*-Smart의 초기 수렴 속도가 빠른 것을 확인하였다. 높은 biasing ratio, 낮은 biasing radius를 사용 할 경우에는 적은 반복 수에서도 RRT*-Smart의 성능이 더 좋아지는 경향을 보였다. 제시된 알고리즘을 통해 생성된 경로는 I. Skrjanc(2010)가 사용한 Bezier curves[6]나 B. Lau가 사용한 spline[7] 등을 이용하여 실제 각종 이동 로봇(Mobile robot)에 적용할 수 있을 것으로 기대된다.

후기

이 논문은 2018년도 한국항공대학교 교비지원 연구비에 의하여 지원된 연구의 결과임

Reference

[1] S. M. Lavalle, “Rapidly-exploring random trees: A new tool for path planning”, 1998.

[2] Noreen, I., Khan, A., and Habib, Z. , “Optimal path planning using RRT* based approaches: a survey and future directions”, Int. J. Adv. Comput. Sci. Appl, 7(11), The Science and Information (SAI) Organization , 2016, pp. 97~107

[3] Karaman, S., and Frazzoli, E,. “ Sampling-based algorithms for optimal motion planning”, The international journal of robotics research, 30(7), Sage Publications, 2011, pp. 846~894.

[4] Nasir, J., Islam, F., Malik, U., Ayaz, Y., Hasan, O., Khan, M., and Muhammad, M. S,. “RRT*-SMART: A rapid convergence implementation of RRT”, International Journal of Advanced Robotic Systems, 10(7), Sage Publications, 2013, pp. 299.

[5] H, Lee, H, Kim, “2D Optimal Path Plannin using RRT*smart algorithm”, KSAS 2014 Fall Conference, The Korean Society for Aeronautical and Space Sciences, Jeju, 2014, pp. 1,237-1,240.

[6] I. Skrjanc, G. Klancar, “Optimal cooperative collision avoidance between multiple robots based on Bernstein-Bezier curves”, Robot. Auton. syst. Elsevier, 58(1), 2010, pp. 1-9.

[7] B. Lau, C. Sprunk, W. Burgard, “Kinodynamic Motion Planning for Mobile Robots Using Splines”, Proceedings of the EEE/RSJ International Conference on Intelligent Robots and Systems, St Louis, MO, USA, 2009, pp. 2427-2433.