일반화 다각형을 위한 plane-sweep 알고리즘의 구현

안진영[0] 유견아
덕성여자대학교 전산학과
{jyahn[0], kyeonah}@namhae.duksung.ac.kr

Implementation of a plane-sweep algorithm for generalized polygons

Jin-Young Ahn[0] Kyeonah Yu
Dept. of computer science, Duksung Women's University

요 약

일반화 다각형(generalized polygons)이란 선분과 호로 둘러싸인 R^2 영역으로 정의되는 확장된 다각형 개념으로 로보틱스 등의 응용 분야에서 다루는 중요한 도형군이다. 로보틱스에 응용되는 컴퓨터 기하학 알고리즘의 대부분은 선분이나 다각형을 다루도록 개발되어 있어 로봇 작업환경의 다양한 물체들을 선분만으로 모델링해야만 알고리즘의 적용이 가능하다. 기존의 알고리즘들을 일반화 다각형을 다룰 수 있도록 확대한다면 보다 유연한 모델링을 가능하게 할 것이다. 본 논문에서는 컴퓨터 기하학분야의 대표적인 알고리즘인 plane-sweep 알고리즘을 일반화 다각형을 다룰 수 있도록 수정하고 구현한다. 이를 로보틱스의 응용분야중 하나인 고정쇠 문제(fixturing)에 적용한다.

1. 서 론[1)]

컴퓨터 기하학의 가장 큰 응용 분야중 하나는 로보틱스라고 할 수 있다. 충돌없는 이동경로(collision-free path)를 계획하거나 부품을 순서대로 조립하는 일 등 거의 모든 로보틱스 분야에서 기하학적 모델을 다룬다. 로봇이 작업하는 환경의 2차원의 기하학적 모델은 다각형뿐 아니라 곡선을 포함한다. 그러므로 컴퓨터 기하학 분야에서 개발된 선분과 다각형을 다루는 알고리즘들을 로봇 환경에 적용할 때에는 곡선을 무수히 많은 선분으로 근사 치환(approximate)하여 적용하는 것이 일반적이다. 이와 같은 방법을 사용할 때, 너무 많은 수의 선분으로 근사 치환하면 계산량이 많아지며 적은 수의 선분으로 근사 치환하면 존재하는 답을 놓칠 수 있는 단점이 있다[1]. 또한 아주 정교한 움직임이 중요한 문제의 경우에는 충분히 많은 수의 선분으로 근사 치환한다 하더라도 정확한 답을 찾아 내지 못하는 경우가 발생한다. 이러한 점에 비추어 볼 때, 시간 복잡도를 희생하지 않고도 곡선을 다룰 수 있도록 알고리즘을 확장하는 것은 중요한 의미가 있는 일이라고 할 수 있다.

본 논문에서는 첫 단계로 원의 호를 포함하는 도형군을 다룬다. 선분과 호를 포함하는 도형군은 로보틱스에서 다루는 중요한 도형군으로 일반화 다각형(generalized polygons)이라고 부른다 [2,3]. 논문 [2]에서는 일반화 다각형의 수학적 정의를 제시하고 몇가지 컴퓨터 기하학의 알고리즘들을 일반화 다각형을 위해 확장할 수 있음을 보였다. 본 논문에서는 이중 대표적인 알고리즘인 plane-sweep 알고리즘을 확장하는 방법을 소개하며 실제 로보틱스 문제에 적용하기 위해 구현하고 적용한 예를 보여준다. 이 때 발생하는 특수한 경우(degeneracies)에 대해서 논한다.

1) 이 연구는 한국과학재단의 목적기초 사업(과제번호 R04-2001-00048)의 연구비 지원에 의해 수행됨.

2. Plane-sweep 알고리즘의 수정

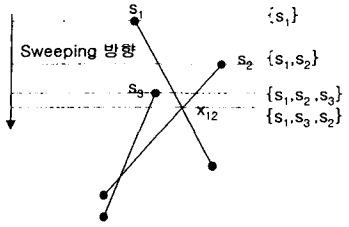
이 장에서는 기존의 plane-sweep 알고리즘에 대해 살펴보고 원의 호를 처리하기 위해 어떻게 수정되는지를 설명한다. 또한 수정된 알고리즘의 시간 복잡도에 대하여 살펴본다.

2.1 Plane-sweep 알고리즘

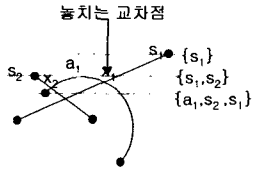
Plane-sweep 알고리즘은 선분의 집합이 주어졌을 때, 선분 사이의 교점(intersection)을 효과적으로 구하는 알고리즘이다[4]. 다각형은 선분의 집합으로 표현할 수 있으므로 두 다각형의 합집합이나 교집합을 구하는 데에도 응용될 수 있다. 이 알고리즘의 원리를 간단히 살펴보면 [5] 수평의 선(sweep-line), L이 선분을 구성하는 y값 중에 가장 큰 값으로부터 시작하여 아래 방향으로 내려가면서(sweeping) 다음과 같은 이벤트가 발생할 때 , L의 상태를 변화시킨다.

① 선분의 시작점
② 선분의 끝점
③ 두 선분 사이의 교점

선분의 시작점과 끝점은 y축 정렬된 방향을 기준으로 정한다. L의 상태는 동적 큐(dynamic queue), $\mathcal{Q}$로 표현되는데 이벤트 ①이 발생하면 해당하는 선분을 큐에 넣고 이벤트 ②가 발생하면 해당하는 선분을 큐에서 제거하며 이벤트 ③이 발생하면 $\mathcal{Q}$에서 해당하는 두 선분의 위치를 바꾼다. 큐 $\mathcal{Q}$은 x의 값으로 정렬된 순서를 유지하도록 한다. 이와 같이 하면 $\mathcal{Q}$의 상태가 변할 때마다 새로이 이웃하게 되는 선분들 사이의 교점을 구하면 된다. 그림 1(a)에 plane-sweep 알고리즘이 전개되는 예를 보여준다.



(a) Plane-sweep 알고리즘



(b) Plane-sweep 알고리즘이 호에 대해 실패하는 경우

그림 1. Plane-sweep 알고리즘

2.2 수정된 plane-sweep 알고리즘

일반화 다각형을 선분과 호로 둘러 쌓인 R^2 영역으로 정의된다. 그러므로 일반화 다각형을 다룰 수 있기 위해서는 선분끼리의 교점 뿐 아니라 호와 선분, 호와 호의 교점도 고려해야 한다. 기존의 plane-sweep 알고리즘을 그대로 적용할 수 없는 이유는 원의 호는 y축 혹은 x축을 따라 단순 증가 혹은 단순 감소하는 성질을 만족하지 못하므로 호의 끝점들을 x축이나 y축에 따라 정렬한다는 것이 의미가 없기 때문이다. 그림 1(b)는 호의 시작점보다 위에서 선분 s_1과 교차하는데 이를 발견하지 못하는 예를 보인다. 호 a_1이 등장했을 때에는 a_1의 오른쪽 이웃한 선분은 s_1이 아니라 s_2가 되기 때문에 a_1과 s_1의 교점을 구하는 일은 없게 된다.

이를 해결하기 위한 방법은 호를 단순 증가 및 감소하는 성분으로 나누어 처리하는 것이다. 즉, sweep-line L의 방향과 L과 수직 방향으로 호를 분할(decompose)하면 정렬이 필요한 모든 축에 대해 단순 증가 혹은 감소하므로 plane-sweep 알고리즘이 적용 가능해 진다.

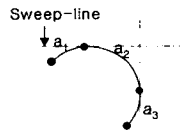


그림 2. 3 조각으로 분할된 호

그림 2는 그림 1에서 문제가 되었던 호를 위에서 설명한 방법대로 분할한 것이다. 이와 같이 하여 다시 plane-sweep 알고리즘을 적용하면 큐 $\mathcal{Q}$에 s_2가 삽입된 후, a_1과 a_2가 삽입되게 되어 위에서 지적한 문제점이 해결되게 된다.

2.3 수정된 알고리즘의 시간 복잡도

Plane-sweep 알고리즘의 시간 복잡도를 먼저 살펴보면 선분의 수가 n이고 교점의 수가 k라고 할 때, 각 선분마다 2개의 이벤트가 있으므로 총 (2n+k) 이벤트가 발생한다. 각 이벤트가 발생할 때마다 큐를 정렬해야 하는데 예로써 큐를 이진 트리로 유지하는 경우 선분이 새로이 삽입되거나 제거될 때마다 O(log n)의 시간 복잡도가 요구된다. 그러므로 전체적인 시간 복잡도는 O((n+k)log n)이다.

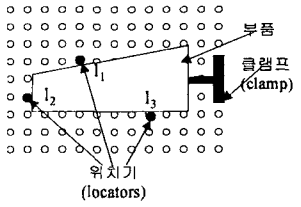
총 n개의 호와 선분의 교점을 구하는 경우, 2.2절에서 설명한 방식대로 호를 처리하면 하나의 호당 최대 4개의 호로 분할되며 하나의 호를 분할하는 데에는 상수 시간이 소요된다. 그러므로 최대 (8n+k) 이벤트에 대해 위와 동일하게 알고리즘을 적용하므로 전체 시간 복잡도는 변화가 없다.

3. 알고리즘의 구현 및 결과

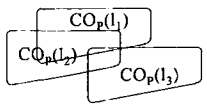
수정된 알고리즘은 Pentium III 플랫폼에서 Visual C++를 이용하여 구현하였다. 구현된 알고리즘은 공장에서 물건을 고정시킬 때 사용되는 모듈식 고정쇠에 적용하여 테스트하였다.

3.1 응용분야 - 모듈식 고정쇠(modular fixture)

고정쇠란 조립이나 선반가공 등의 제조공정 과정에서 부품을 고정시킬 때 사용되는 도구이다. 모듈식 고정쇠란 고정쇠의 재사용이 가능하도록 여러 모양의 구성요소를 조립하여 고정쇠를 생성하는 것으로 구성요소 중에는 다각형 뿐 아니라 실린더형의 위치기나 고정기 등이 다수 포함되어 있다.



(a) 3개의 위치기와 1개의 클램프로 고정된 부품



(b) 3개의 위치기에 대한 형상공간 표현

그림 3. 고정쇠 형상의 예[6]

그림 3 (a)은 고정쇠 요소들을 꽂을 수 있는 구멍이 격자 모양으로 배열된 모듈식 고정쇠의 구현 예를 보여준다. 클램프를 제거한 상태에서 부품을 삽입하고 클램프로 조여줌으로써 부품의 움직임을 허용하지 않는 form-closure 상태로 만든다. 부품의 평면 운동만을 고

려하면 작업공간에서 부품의 형상(configuration)은 (x,y,θ)으로 나타내어진다. 작업공간에 원과 다각형을 모두 포함하고 있기 때문에 형상공간(configuration space)은 그림 3(b)와 같이 호와 선분으로 이루어진 일반화 다각형으로 표현된다[6]. 부품의 방향이 θ일 때 클램프가 제거된 상태의 (x,y) 형상공간을 표시하며 원형 위치기둥은 형상공간에서 각각 일반화 다각형으로 표현되어 전체를 표현하기 위해서는 이들을 합집합하는 과정이 필요하다.

3.2 알고리즘의 구현

본 논문의 배경이 되는 [6]에서는 일반화 다각형의 꼭지점과 호를 일관성 있게 표현하기 위해 호를 유사 꼭지점(pseudo vertex)으로 간주하여 호의 시작점, 중점, 끝점 (v_i^-, v_i, v_i^+)의 3점으로 구성한 순서 집합을 택한다. 그림 4는 고정쇠 문제에서 다루게 되는 일반화 다각형의 유형인데 선분과 선분이 만나는 꼭지점의 경우에는 호의 시작과 끝이 같은 점인 특수한 경우로 취급되는 것을 볼 수 있다.

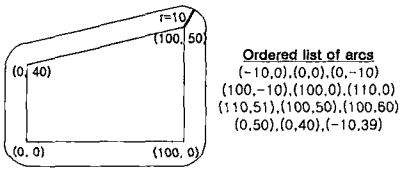
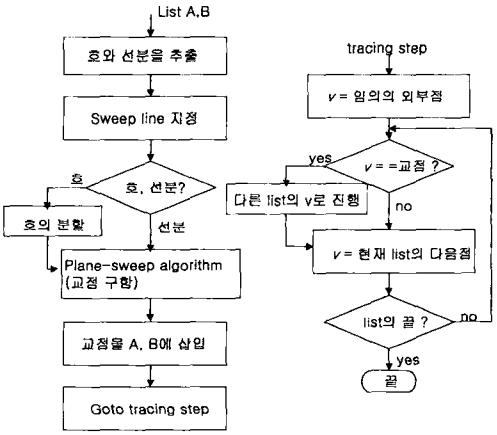


그림 4. 일반화 다각형의 표현 - 시계반대방향의 순서 집합

이와 같은 자료구조로 표현된 두 개의 일반화 다각형을 합집합하기 위한 프로그램의 흐름도는 다음과 같다. 두 개의 일반화 다각형이 A, B 리스트로 표현되었다고 하자.



3.3 실행 결과

그림 5는 임의의 호와 선분이 섞여 있는 집합이 주어졌을 때, plane-sweep 알고리즘을 적용한 결과 창들이다. 좌측 그림에서 주어진 호는 3등분되어(호위의 작은 선분들이 호가 나뉘는 곳을 나타냄) 실제로는 3개의 호와 3개의 선분 사이의 교점을 구한 결과이다. 가운데 그림은 그림 3의 예에서 필요했던 두 일반화 다각형들의 합집합을 plane-sweep 알고리즘을 이용하여 구한 결과를 보여준다. 도형 위에 흐린 굵은 선이 합집합을 나타내며 작은 원은 교점을 나타낸다. 우측 그림은 그림 3의 예를 오목(concave) 다각형에 확대 적용하여 합집합 결과에 구멍(hole)이 생기는 예이다.

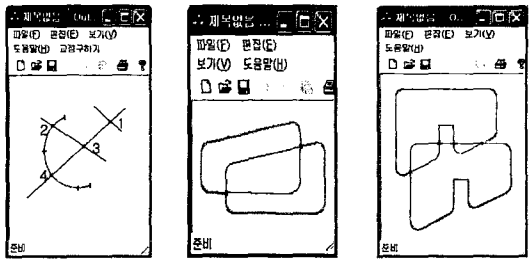


그림 5. Plane-sweep 알고리즘 적용 결과

4. 결론

본 논문에서는 기존의 plane-sweep 알고리즘을 원의 호를 포함하는 확장된 개념의 일반화 다각형에도 적용할 수 있도록 구현하였다. 일반화 다각형은 로보틱스 분야 등에서 보편적으로 볼 수 있는 도형군이므로 응용의 범위는 넓다고 할 수 있겠다.

본문 중에는 언급하지 않았지만 이와 같은 문제를 다룰 때에는 항상 특수한 경우(degeneracies)에 대한 고려가 필요하다. 즉, 선분이 다른 선분의 끝점과 만나는 경우, 두 선분이 겹치는 경우 등은 이미 기존의 알고리즘이 소개되면서 언급된 바 있으나 호와 선분, 혹은 호와 호의 접점이 교점으로 발생하는 경우 등이 새로운 특수한 경우로 간주되어 일반적인 plane-sweep 알고리즘의 흐름을 따르지 않는다. 그러므로 이 알고리즘을 경로를 찾는다거나 조립 계획을 하는 문제에 적용할 때에는 실제로 적용되는 문제의 특성에 따라 이러한 경우에 대한 명확한 답을 정의하여야 한다.

[참고문헌]

[1] D. Halperin, L. Kavraki, and J.-C. Latombe, "Robot algorithms" CRC Algorithms and Theory of Computation Handbook M. Atallah (editor), CRC Press, 1999.

[2] J.P. Laumond, "Obstacle growing in a nonpolygonal world", Inform. Proc. Letter, vol. 25(1), pp41-50, 1987.

[3] A. Rao and K. Goldberg, "Orienting generalized polygonal parts", IEEE international conf. on Robotics and Automation, pp2263-2268, 1992.

[4] M. de Berg et. al., Computational Geometry: Algorithms and Appplications 2nd edition, Springer, pp 19-43, 1998

[5] J. O'Rourke, Computational Geometry in C, Cambridge University Press, pp220-293, 1998

[6] K. Yu, "Planning of compliant motions for fixture loading", Tr. on Control, Automation, and System Engineering, vol 2(1), pp62-68, 2000.