차원 압축을 통한 최근접점 탐색 알고리즘의 속도 개선

강혜란[0] 남현우 위영철
아주대학교 정보통신 전문대학원
{jelly75, manner, ycwee}@ajou.ac.kr

Speed Improvement of Nearest Neighbor Search Algorithm using Dimension Compression

Hye-Lan Kang[0] Hyeon-Woo Nahm Young-Chul Wee
Graduate School Of Information and Communication, Ajou University

요 약

본 논문에서는 최근접점 탐색 알고리즘(Nearest Neighbor Searching)을 사용하여 고차원에서 질의점을 효과적으로 찾기 위한 방안을 제안한다. 최근접점 탐색에서 정확도와 실행속도는 반비례 관계를 가지며 기존에 제안된 최근접점 탐색 알고리즘의 경우, 차원이 증가할수록 탐색 시간이 기하급수적으로 증가하게 되어 고차원에서 질의점을 탐색할 경우 실행시간이 현저하게 길어진다. 최근접점 탐색을 실세계에서 적용할 경우 정확도도 중요하지만 실행 속도 또한 중요하다. 이 점을 감안하여 본 논문에서는 고차원 데이터를 저차원으로 압축하여 질의점을 탐색하고 압축 이전과 이후의 결과를 비교한 후, 이를 통해 정확성과 실행속도의 관계를 분석한다. 본 논문에서는 제안한 차원 압축을 이용할 경우, 정확성이 중요한 요소가 아닌 탐색에서 상당한 실행속도가 개선될 것으로 기대된다.

1. 서 론

최근접점을 찾는 문제는 그래픽스 분야뿐만 아니라 데이터 마이닝, 패턴인식, 데이터 압축, 멀티미디어, 데이터베이스, 통계학 등의 응용분야에서 다양하게 다루어지고 있다.

최근접점 탐색은 d 차원공간인 R^d공간에 주어진 데이터 집합 S에 대해 질의점이 주어지면 S의 데이터들을 전처리한 후 최근접점 탐색 알고리즘을 사용하여 주어진 질의점을 빠른 시간 내에 찾는 알고리즘이다.[1]

최근접점 알고리즘의 문제점중의 하나는 차원이 높아짐에 따라 시간복잡도가 증가하여 실행 시간이 증가한다는 것이다.

본 논문에서는 이와 같은 문제를 해결하기 위한 알고리즘을 제안하며, 또한 한 질의점의 최근접점을 압축하지 않는 차원과 압축한 차원에서 찾는 실험을 통해, 차원에 따른 실행시간과 거리의 오차를 비교하여 차원과 실행시간, 오차의 연관성을 분석한다.

본 논문에서 제안하는 알고리즘을 사용함으로써 효과적인 최근접점 알고리즘 개발에 기여를 할 수 있을 것으로 기대된다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 최근접점 탐색 알고리즘의 전체적인 개요와 기존의 알고리즘과 제안된 알고리즘에 대해 분석한다. 3장에서는 제시된 알고리즘을 실험하기 위한 개요와 실험결과를 제시한다. 5장에서는 실험결과를 통한 결론과 향후 연구과제를 제안하며 논문을 맺는다.

2. 최근접점 탐색 알고리즘의 개요

2.1 최근접점 탐색 알고리즘

최근접점을 찾는 알고리즘은 거리 공간 X에서 n개의 데이터 점들의 집합 S가 주어져 있을 때, 질의점 q에 대해서 가장 최근접한 점을 찾는 알고리즘이다.[2]

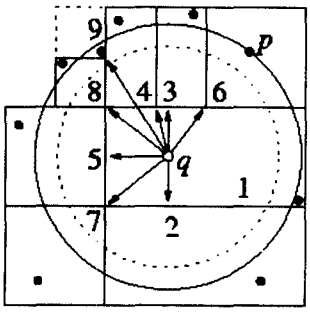
오른쪽 [그림 1]과 같이 실제 최근접점은 점선으로 표시된 9번 박스에 있는 점이나 거리 계산의 허용 오차를 고려하지 않게 되면 p 점이 매번 최근접점으로 선택되어지게 된다.[2]

실제 최근접점이 선택되기 위한 방법은 질의점으로부터 하향오차나 상향오차 즉 상대 오차를 허용하는 반경 내에 있는 점들을 찾아 거리 계산을 함으로써 적절한 최근접점을 탐색한다.[2]

최근접점을 찾는 문제는 데이터 마이닝, 패턴인식, 데이터 압축, 멀티미디어 데이터베이스, 통계학 등의 응용분야에서 다양하게 다루어지고 있다. 고차원에 대한 최근접점 탐색 문제는 물체의 특징을 다수의 벡터로 표현하여 처리하게 되며 이러한 점들 사이의 거리 계산은 일반적으로 Minkowski metric을 사용한다.[4]

최근접점을 찾는 방법의 하나는 차원이 높아짐에 따라 시간복잡도가 증가하여 실행 시간이 증가한다는 것이다.

본 논문에서는 이와 같은 문제를 해결하기 위한 알고리즘을 제안하며, 또한 한 질의점의 최근접접을 압축하지 않는 차원과 압축한 차원에서 찾는 실험을 통해, 차원에 따른 실행시간과 거리의 오차를 비교하여 차원과 실행시간의 연관성을 분석한다.



[그림 1] 최근접점 탐색 알고리즘의 개요

2.2 기존의 알고리즘

기존의 최근접점 알고리즘에서 한 개의 질의점에 대해서 가장 가까운 거리에 있는 세 개의 최근접점 데이터들을 탐색하게

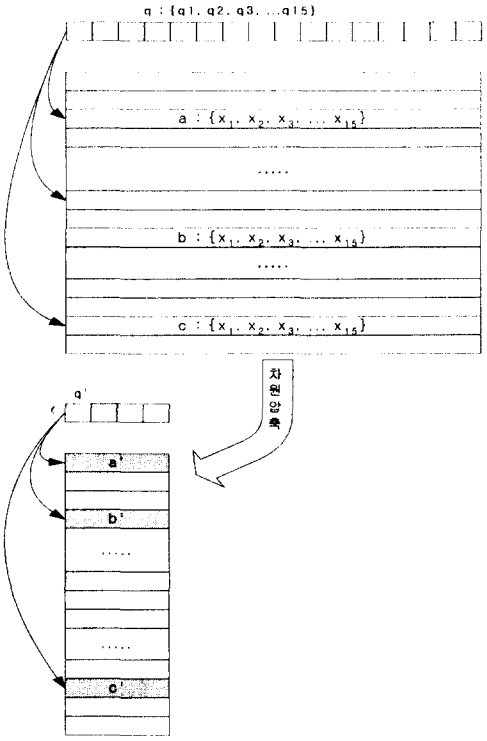
되며 [그림 2]에서 압축하기 이전과 같은 결과가 나타내게 된다.

2.3 제안한 알고리즘

제안된 알고리즘에서는 먼저 기존의 알고리즘과 같이 한 개의 질의점에 대해서 가장 가까운 거리에 있는 세 개의 데이터들을 탐색하게 될 때 찾은 세 개의 데이터 값들을 a, b, c 라고 한다면 d차원의 데이터 값들을 $\{ x_1, x_2, x_3, x_4 \ldots\ldots, x_d \}$ 으로 표현하고, 각 벡터값들을 $y_1 = x_1 + x_2 + x_3 + x_4 \ldots$ 와 같이 순서대로 네 개씩 묶어서 더한 값을 구한다. 네 개씩 묶은 합계값을 구하게 되면 $\{ y_1, y_2, y_3, \ldots\ldots, y_{\frac{d}{4}} \}$과 같은 $d/4$차원의 벡터값들이 생성된다.

$d/4$로 압축된 데이터들에서 압축한 질의점 q'으로 최근접점 데이터값 세 개를 탐색하고, 이때 찾은 세 개의 데이터 값들을 a', b', c'이라고 한다면 a', b', c'는 질의점에 대한 새로운 최근접점들이 된다.

제안된 알고리즘을 [그림 2]과 같이 표현할 수 있다.



[그림 2] 제안된 알고리즘

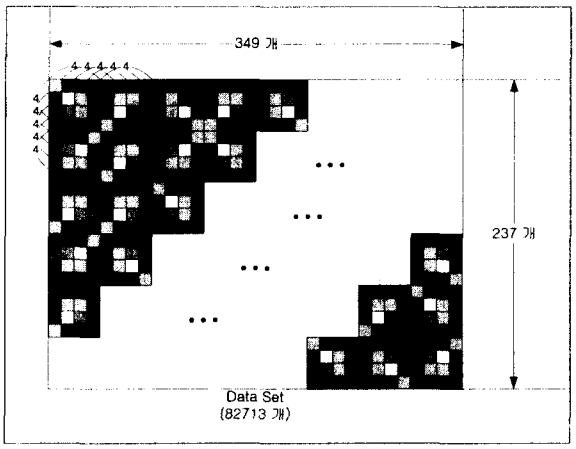
3. 실험결과 및 분석

3.1 실험 개요

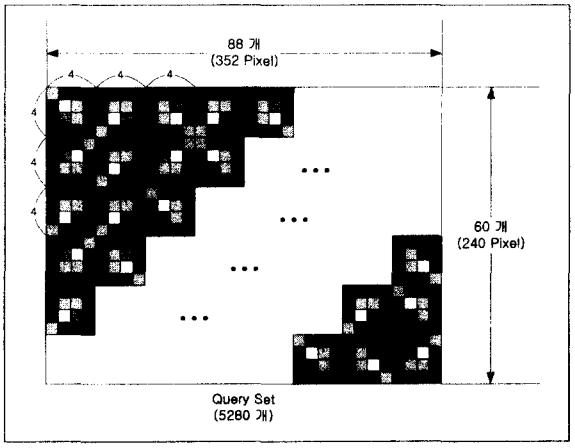
실험하기 위한 데이터집합과 질의점집합은 데이터의 연관성을 위해 동영상에서 캡쳐한 연속적인 352*240 픽셀 크기의 그레이 값이 들어 있는 이미지블럭 각각 2장씩, 5개의 데이터 집합을 발생시켜 실험하였다. 이미지블럭의 각 픽셀에는 그레이 색상값이 저장되어 있다.

[그림 3]에서는 4*4 픽셀값을 16차원의 데이터값으로 변환하여 한 픽셀단위로 움직이면서 82713개의 16차원 데이터집합을 생성하였다.

[그림 4]에서는 이미지블럭을 4*4단위로 나누어서 5280개의 16차원 질의점집합을 생성하였다.



[그림 3] 이미지블럭으로부터 데이터집합 생성



[그림 4] 이미지블럭으로부터 질의점집합 생성

3.2 분석 결과

[표 1]과 [표 2]에서의 실험결과를 분석해보면 질의점의 탐색에 있어서 정확도가 1/2로 감소되는 반면에 소요시간도 반 정도로 감소된다. 질의점의 탐색 시간 속도는 현저하게 빠른 속도를 제공함을 볼 수 있다.

또한 전처리과정의 소요시간도 1/2 정도로 감소된다.

	비압축의 전처리 과정(sec)	압축의 전처리 과정(sec)	비 압축의 질의 시간 (sec/질의)	압축의 질의시간 (sec/질의)
그림 1	17.99	5.33	0.0640486	0.00109641
그림 2	25.02	6.68	0.143914	0.00137051
그림 3	13.63	6.54	0.264428	0.00120808
그림 4	30.2	7.49	0.188481	0.00126899
그림 5	19.87	6.77	0.0522318	0.00123854

[표 1] 비 압축과 압축의 거리분석 결과

	비 압축 평균거리	압축 평균거리	비 압축과 압축의 동일한 데이터 갯수(%)
그림 1	10.32282	21.69428	7.57576
그림 2	5.618106	11.91123	1.90026
그림 3	5.804865	12.57472	8.66162
그림 4	5.004053	11.84894	3.24495
그림 5	2.829793	7.27479	4.09091

[표 2] 비 압축과 압축의 처리 속도 결과

5. 결론

본 논문은 정확도를 희생하여 속도를 높이는 방법을 제안하였다. 이 원리는 고차원을 저차원을 낮춤으로써 이루어진다. 이를 위해 본 논문에서는 차원을 낮추는 근거를 제시하기 위해 한 질의점의 최근접접을 압축하지 않는 차원과 압축한 차원에서 찾는 실험을 통해, 차원에 따른 실행시간과 거리의 오차를 비교하여 차원과 실행시간, 오차의 연관성을 분석하였다.

실험결과에서 정확도의 오차를 감수하더라도 질의점 탐색시간의 속도를 현저하게 높이기 위한 분야에서는 효율적인 탐색 알고리즘이라고 판단되며, 차원의 압축에 있어 다양한 방식으로의 접근이 계속 연구될 것이다.

6. 참고문헌

[1] S. Arya, D. Mount, N. Netanyahu, R. Silverman, A. Wu, An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions, Jounal of the ACM, 1998

[2] S. Arya, D. Mount, Approximate Nearest Neighbor queries in Fixed Dimensions, In Proc. 4th ACM-SIAM Sympos. Discrete Algorithms, 1993

[3] M. Moore, J. Wilhelms, Collision Detection and response for computer animation, ACM Computer Graphics, 1988

[4] F.P.Preparata, M.I.Shamos, Computational Geometry : An Introduction, Springer-Verlag, NY, 1985