# 실수 정의역 데이터 시각화와 그 응용 사례

이성호° 박태정 감형렬 김창헌

고려대학교

pocorall@korea.ac.kr taejung.park@gmail.com neary@korea.ac.kr chkim@korea.ac.kr

# Visualizer for real number domain data and its applications

Sung-Ho Lee°, Tae-Jung Park, Hyeong Ryeol Kam, and Chang-Hun Kim

Korea University

## 요 약

거리장, 볼륨 텍스처와 같은 3차원 실수 정의역을 갖는 데이터를 다룰 때, 이를 효과적으로 시각화하는 것은 중요한 주제이다. 본 논문에서는 간단한 인터페이스 구현체를 입력으로 받아 절단면 보기, 확대, 특정 위치의 값 조회 등의 작업을 효과적으로 수행하는 프레임워크를 제안한다. 인터페이스 구현체에는 다양한 알고리즘을 자유롭게 구현해 넣을 수 있으며, 여러 가지 종류의 데이터들을 조회하고 알고리즘 수행 결과를 쉽게 평가할 수 있다.

## Abstract

Effective visualizing is an important issue when one processing real number domain volume data such as distance fields, or volume texture. In this paper, we introduce a framework for inspecting, magnifying, cross-section viewing of real number domain volume data from an implementation of a simple interface. The interface can be freely implemented from any kind of existing algorithm, so that we can easily view the result and evaluate the algorithm.

**키워드:** 실수 정의역, 데이터 시각화

**Keywords:** real number domain, data visualization

## 제 1 절 서론

과학, 공학 시뮬레이션 및 연산, 물리기반 애니메이션 개발, 그래픽스 연구 등에서 거리장, 볼륨 텍스처, 프렉탈 데이터와 같은 3차원 실수 정의역을 갖는 데이터를 다룰 때, 생성된 데이터를 시각화하는 작업은 효과적인 결과물 도출을 위해 필수적이다. 그러나 3차원 데이터의 특성상 2차원 디스플레이에 일목요연하

*투고일: 2010-07-09 / 심사일: 2차 2010-09-11 / 게재확정일: 2010-09-13

게 표현하기는 어려우며, 절단면 보기, 확대, 특정 위치의 값 조회, 삼각화, 레이 캐스팅 등의 작업을 용이하게 하기 위해서는 추가적인 도구가 요구된다. 또한 멀티코어 환경에서의 성능 향상을 위해 멀티스레드로 렌더링하려는 경우에도, 효율적인 작업 스케줄링을 구현하는 것은 사소한 문제는 아니다. 임시방편의 간단한 뷰어를 작성해서 데이터를 확인하는 경우가 보통이지만, 앞서 말한 효율적인 브라우징과 렌더링 기능을 갖추는 데에는 많은 노력이 필요하다. 본 논문에서는 이러한 문제를 해결하기 위한 일반화된 가시화 프레임웍의 설계 원칙과 구조에 대해서 논한다.

본 논문의 프레임웍에서 입력으로 받아들이는 데이터의 가장 큰 특징은 실수 정의역을 가진 2차원 혹은 3차원 필드라는 점이다. 데이터의 정의역이 양의 정수인 경우, 일반적인 비트맵 이미지 뷰어로 값을 조회하는 것이 쉽게 가능하고, 좀 더 고급 분석 기능이 요구된다면 Mathlab, Mathmatica와 같은 상업용 계산 소프트웨어의 가시화 기능이나, Visualization toolkit과 같은 가시화 라이브러리를 이용하는 방법도 있다. 데이터의 정의역이 실수인데, 간단한 양함수(explicit function)로 이루어진 경우도 역시 기존의 가시화 소프트웨어를 이용해 렌더링하는 방법이 존재하지만, 입력 데이터가 거리장, 볼륨 텍스처, 프렉탈과 같은 절차적인(procedural) 방법에 의해 만들어지는 경우에는 기존 방법을 통해 자유롭게 시각화하는 것은 용이하지 않다.

많은 프레임워크들이 해당 분야에 특화된 언어(DSL; Domain specific language)를 제시하는 반면, 우리는 자바(Java)나 그루비(Groovy)와 같은 범용 언어를 사용한다. 범용 언어를 사용함으로써, DSL을 배우는 데 드는 학습 시간을 절약할 수 있고, 효율적이고 안정적인 컴파일러와 런타임 가상머신을 이용할 수 있으며, 기존에 개발되어 있는 광범위한 라이브러리를 사용할 수 있게 된다. 본 프레임워크에 입력으로 주어지는 데이터를 서술하는 데에는 자바 가상 머신(JVM; Java vertual machine)을 이용하는 모든 종류의 언어(Java, Groovy, JRuby, Scala, Jython)를 자유롭게 사용할 수 있다.

우리의 프레임워크는 데이터를 시각화하는 데에 실수 도메인 이미지를 정의하는 매우 간단한 프로그래밍 인터페이스를 이용하고, 여타의 다른 코딩이 부가적으로 필요하지 않기 때문에, 데이터를 서술하는 매우 간결한 코드만 제공하면 프레임웍이 제공하는 여러가지 시각화 도구들을 활용할 수 있게 된다. 또한, 한 번 작성한 코드가 2차원과 3차원 정의역 데이터를 표현하는 데에 공통으로 사용될 수 있으므로 더 범용적이다. 3차원 데이터가 일반적으로 효과적으로 시각화하기 더 까다롭다는 점을 고려할 때, 2차원으로 시각화하여 데이터의 무결성을 확인한 후, 3차원 데이터 연산에 사용할 수 있으므로 디버깅 과정에서 유용하다.

과학 및 공학 시뮬레이션 수행시에는 거리장, 만델브로 집합, 노멀 맵 등의 스칼라 필드 뿐만 아니라, 컬러 텍스처, 텐서장 등의 벡터 필드 데이터를 조회하려는 경우도 흔하다. 우리의 프레임워크는 스칼라 필드와 벡터 필드를 모두 표현할 수 있으며, 두 경우 모두 공통의 사용자 인터페이스를 이용한다.

전형적으로 입력 데이터는 특정 파라미터에 대한 함수인 경우가 빈번하다. 예를 들어 쥴리아 집합(Julia set)은 파라미터 $\mathbf{c} = \mathbb{R}^2$가 변화함에 따라 데이터가 변화한다. 런타임에 입력 데이터의 파라미터를 변경시키고 그에 따른 변화를 실시간으로 관찰하는 것도 효과적인 디버깅을 위해 필수적인데, 본 논문의 프레임워크에서는 별도의 컴파일 없이 JVM에서 제공하는 인트로스펙션(introspection)을 사용해서 인터렉티브하게 파라미터 변경이 가능하도록 한다.

우리 프레임워크의 또다른 장점으로는 렌더링 지원을 들 수 있다. 멀티코어 활용을 위한 멀티스레드 렌더링 작업 스케줄링, 그리고 거리장을 위한 레이캐스팅[1] , 삼각형 메쉬 생성[2, 3] 이 제공된다. 이런 방법들은 데이터의 성격에 무관하게 공통으로 요구되는 기능이므로 프레임워크에서 제공하는 것이 합당하다.

지금까지 소개한 내용을 바탕으로, 본 논문에서 다루는 프레임워크의 장점을 요약하면 다음과 같다:

- 범용 언어 사용 (JVM기반)
- 간결한 표현
- 2,3차원 실수 정의역 겸용
- 스칼라, 벡터 공역 겸용
- 런타임 파라미터 변경 (인트로스펙션 사용)
- 렌더링 지원 (멀티스레드 스케줄링, 레이캐스팅, 삼각형 메쉬 생성)

## 제 2 절 관련 연구

Upson et al.[4] 과 Schroeder et al.[5] 의 연구는 데이터를 처리하는 모듈을 추상화하고, 모듈들의 파이프라인을 구성해서 시각 데이터 처리의 재사용성과 이용편의성을 높였다. 객체지향 프로그래밍 측면에서, 이 연구들에서 추상화시킨 모듈이 데이터 처리기인 반면, 우리가 제안하는 프레임워크에서는 데이터가 추상화된다. 우리 모델에서는 추상화된 데이터들의 데코레이터와 컴포지터[6] 패턴을 통해 데이터가 가공되기 때문에, 파이프라인을 통하는 데이터의 용량 문제로부터 자유로우며, 우리의 추상화된 데이터는 실수 정의역을 가지므로, 데이터의 해상도와 용량 사이의 트레이드 오프도 발생하지 않는다.

Schroeder et al.[5] 은 범용 언어인 C++을 이용해 작성된 시각화 라이브러리를 소개한다. 다양한 종류의 기본 데이터 타입과 데이터 처리기를 제공하여 바로 사용할 수 있다. 이들의 구현에 기반이 되는 C++언어는 정적으로 타입을 처리하는 언어로서, 실행시간에 임의의 개체의 속성을 조회하고 갱신할 수 없다. 이에 반해 인트로스펙션을 제공하는 JVM기반인 우리의 프레임워크는 UI를 통해 재컴파일 없이 실시간으로 파라미터를 수정할 수 있고, 파라미터가 UI 프레임워크와 상호작용하기 위해 데이터 제공자는 파라미터를 선언하는 것 이외에 어떤 코딩도 추가로 요구되지 않는다. 덧붙여, Schroeder et al.[5] 의 연구는 라이브러리를 제공하는 방법으로서, 가시화를 담당하는 위젯을 구성하고 시작하는 부분의 프로그래밍이 필요한 반면, 우리는 프레임워크를 사용하는 접근방법을 취함으로서, 사용자는 입력 데이터를 정의하는 것 이외에 다른 코딩이 필요하지 않다.

## 제 3 절 입력

앞 절에서 밝혔듯이 우리 프레임워크에서는 데이터가 추상화된다. 추상화된 데이터는 아래에 표시된 간결한 형태의 자바 인터페이스로 정의되어 있다.

```
public interface ScalarImage {
  double getValue(double... x);
}
```

프레임워크는 데이터에 대해 위치 x의 값을 조회하는 인터페이스 이외에 어떤 제약도 가정하지 않으므로, 데이터 획득에 필요한 연산시간과 저장에 필요한 메모리 요구량은 데이터의 구현체에 따라 상이하게 결정된다.

인터페이스 ScalarImage 의 구현체를 작성하면 프레임워크가 렌더링과 사용자 인터페이스를 제공한다. 아래와 같이 줄리아 집합을 간결한 그루비 코드로 작성해서 시각화할 수 있다.

```
class JuliaImage implements ScalarImage {
  double cx=0.0
  double cy=0.7
  public double getValue(double... x) {
    double r = x[0], i = x[1], m = 0.0, n = 0.0
    int j = 0
    while((j < 256) && (r * r + i * i < 4.0)) {
      j++
      m = r * r - i * i
      i = 2.0 * r * i + cy
      r = m + cx
    }
    return j
  }
}
```

그림 1은 이렇게 생성된 줄리아 집합을 시각화한 것이다. 파라미터 cx와 cy는 사용자가 GUI를 통해

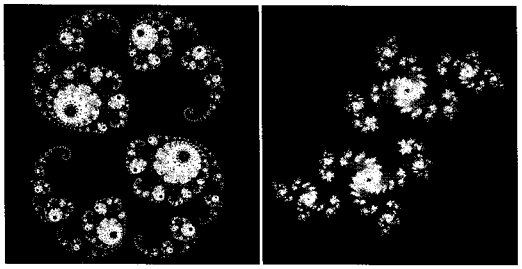

그림 1: 줄리아 프랙탈 이미지. $\mathbf{c} = [0.285, 0.01]$ (왼쪽), $\mathbf{c} = [0.7, 0]$ (오른쪽). 파라미터 $\mathbf{c}$는 프레임워크의 GUI를 통해 인터렉티브하게 수정할 수 있다.

실시간으로 변경하고 시각적인 피드백을 얻을 수 있다. 프레임워크는 double형이 제공하는 정밀도의 한계만큼 확대와 축소를 지원하며, 사용자가 지정한 곳의 값을 숫자로 조회할 수 있으며, 컬러 코딩 스킴을 교체할 수 있다.

거리장 데이터를 표현하기 위해서 프레임워크는 부가 기능을 제공한다. 먼저 아래와 같이 하이퍼스피어 거리장을 정의한 그루비 클래스 SphereImage를 렌더링하면 그림 2와 같이 일정한 간격으로 등고선(contour)을 표시할 수 있다.

```
class SphereImage implements ScalarImage {
  double radius = 0.3
  public double getValue(double... x) {
    double sum = 0
    x.each { sum += it * it }
    return Math.sqrt(sum) - radius
  }
}
```

위의 코드에서 알 수 있듯이, 인터페이스 ScalarImage의 정의역은 2차원에 한정되지 않고, 데이터 정의역의 차원에 중립적으로 구현할 수 있다. 몇차원 정의역 데이터로 렌더링할 것인지는 프레임워크가 렌더링 시에 결정하는데, 이런 특징은 디버깅에 특히 유용하다. 복잡한 이미지 구현체를 만드는 경우, 데이터를 열람하기 쉽고 연산이 빠른 2차원 렌더링으로 오류가 없음을 확인한 다음, 코드 수정 없이 시간이 오래 걸리는 3차원 렌더링을 바로 적용할 수 있기 때문

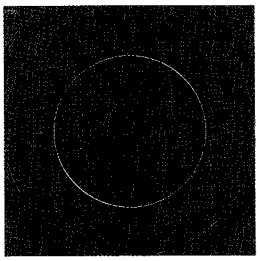

그림 2: 2차원 구형 거리장 이미지. 흰 선이 값이 0인 곳이고, 0.1의 배수마다 파란색 선이 그려져 있다.

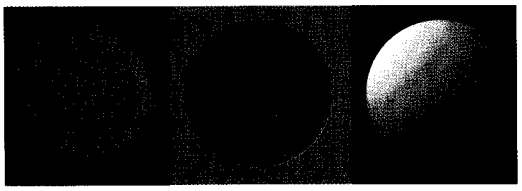

그림 3: 레이트레이싱된 3차원 구형 거리장 이미지. 맨 왼쪽부터 깊이 값, 노멀, 퐁 셰이딩.

이다.

3차원으로 렌더링한 결과는 그림 3에 제시된다. 3차원 렌더링에는 스피어 트레이싱 [1] 기법이 사용되었다. ScalarImage의 구현체로는 구와 같은 분석적 도형 뿐만 아니라, 복잡한 형태의 메쉬 데이터로부터 얻어낸 거리장을 표현할 수도 있다. 그림 4 에서는 Baerentzen과 Aanaes[7] 의 기법을 이용해서 삼각형 집합으로부터 거리장을 계산하여 렌더링한 예제를 보여준다. 또한, 프레임워크에서는 그림 5와 같이 듀얼 컨투어링[3] 방법을 이용해 삼각형 메쉬를 생성할 수 있다.

## 제 4 절 데코레이터

데이터의 특성과 용도에 따라 많은 거리장 계산 방법과 자료구조가 제안되어 왔으며 [8, 9, 7, 10] , 새로운 요구사항에 부응하는 새로운 거리장 표현법을 개발해

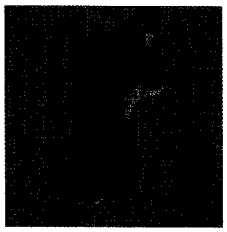

그림 4: 메쉬로부터 구해진 3차원 거리장을 레이 트레이싱으로 렌더링. 색상은 표면의 노멀을 의미한다.

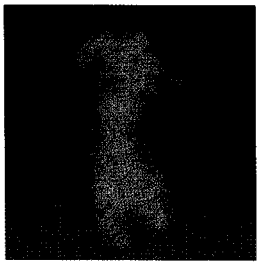

그림 5: 두얼 컨투어링[3] 기법을 이용해 생성된 삼각형 메쉬.

야 할 필요가 생길 수 있다. 거리장 표현법을 개발하는 데 있어서 거리장의 공간적 연속성과 거리장의 그라디언트의 크기가 1이 되는 성질은 중요하다. 이 성질을 쉽게 조회하기 위해 우선 스칼라 데이터의 그라디언트를 계산하는 클래스를 아래와 같이 구현한다.

```
class Gradient implements VectorImage {
  ScalarImage src
  double delta=0.0001
  public double[] getValues(double... x) {
    double[] result = x.clone()
    x.eachWithIndex { it, i ->
      double[] x2 = x.clone()
      x2[i] += delta
      result[i] = (src.getValue(x2) - src.getValue(x)) / delta
    }
    return result
  }
}
```

src는 그라디언트가 구해질 스칼라 데이터이고, delta는 그라디언트를 계산할 오프셋이다. 그라디언트 연산자는 벡터를 반환하므로 아래와 같은 벡터 필드용 인터페이스를 상속받아 구현했다.

```
public interface VectorImage {
  double[] getValues(double... x);
}
```

우리의 프레임워크에서는 클래스 Gradient와 같이, 데이터를 감싸는 데코레이터 패턴, 혹은 여러 개의 데이터를 통합해서 감싸는 컴포지트 패턴[6] 을 이용함으로써 다양한 연산을 할 수 있다. 빌딩 블럭들은 단순하고 명확한 기능만을 수행하면 되고, 이들을 조합하여 추상 데이터 구현체의 트리를 구성함으로써 복잡한 요구사항을 만족하는 시각화를 이루어낸다. 빌딩 블럭의 계층 구조는 프레임워크가 모두 추적하여, 모든 블럭의 프로퍼티를 GUI로 조회하고 변경할 수 있는 인터페이스를 제공한다.

우리는 빌딩 블럭의 예로서 좌표 변환(확대, 이동, 회전, 거울상, 타일링 등), 값 변환(제곱, 절대값, 그라디언트, norm 등의 산술 연산), 거리장을 위한 CSG연산 등을 구현했는데, 이들은 범용적이며, 작성하기 쉽고, 프로퍼티를 조작하기 쉬우며, 효율적으로 동작한다.

## 제 5 절 세그먼트

데이터의 성격에 따라서 영역별로 데이터가 구분되는 경우가 있다. 이런 데이터를 효과적으로 표현하기 위해 아래처럼 세그먼트를 표시할 수 있는 인터페이스를 정의한다.

```
public interface SegmentedImage {
  Object getSegment(double... x);
}
```

입력 데이터가 SegmentedImage를 구현하고 있는 경우, 프레임워크는 렌더링에 사용하는 모든 쿼리 지점 x에 대해 getSegment()를 호출한다. 이 때 반환되는 개체들의 개체-동등성(object-equality)을 비교해서, 같은 세그먼트 개체를 반환하는 것으로 확인된 위

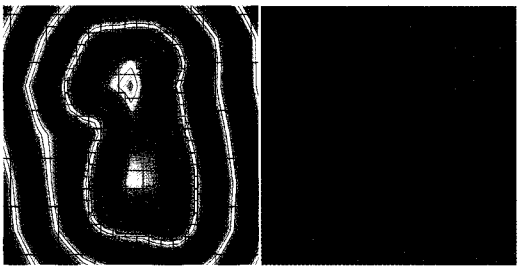

그림 6: Lefebvre 와 Hoppe가 제안한 자료구조[10]를 이용해 저장된 3차원 거리장의 단면(왼쪽)과 레이 캐스팅(오른쪽). 검정색이 세그먼트의 경계선을 나타낸다. 여기서는 트리의 노드가 각각의 세그먼트로 표시되었다.

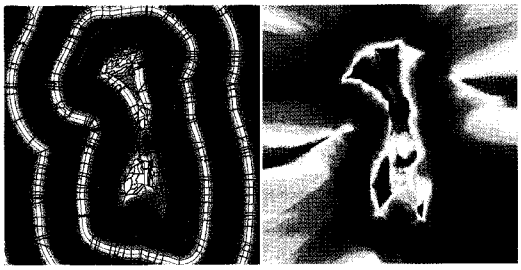

그림 7: Shevtsov[11] 의 기법을 이용해 계산한 거리장의 단면(왼쪽)과 거리장을 구하기 위해 탐색한 삼각형의 갯수(오른쪽). 왼쪽 그림에서 각 셀은 해당 위치에서 가장 근접한 삼각형이 소속된 Kd-Tree의 노드를 의미하고, 오른쪽 그림에서 셀은 도형의 내부 혹은 외부를 나타낸다.

치는 같은 세그먼트에 속하는 것으로 간주하고 렌더링한다.

같은 세그먼트인 부분의 의미는 데이터 구현체에 따라 다르게 주어진다. 예를 들어, 그림6과 그림 7는 같은 도형의 거리장을 계산한 것이지만, 자료구조의 차이에 따라 다른 방식으로 셀을 정의해서 시각화했다.

## 제 6 절 결론과 향후 연구

본 논문에서는 2차원 혹은 3차원 거리장의 값을 쉽게 조회하는 프레임워크에 대해서 다루었다. 간결하게 정의된 인터페이스를 구현하면 프레임워크가 단면 및 등거리선 조회, 레이 트레이싱, 노멀 정보 표현 등을 담당한다. 인터페이스 구현체에는 다양한 알고리즘을 자유롭게 구현해 넣을 수 있으며, 단순하고 명확한 기능을 수행하는 데코레이터와 컴포지터를 빌딩 블럭으로 이용해서 복잡한 요구사항을 만족시키는 가시화를 수행하고 알고리즘 수행 결과를 쉽게 평가할 수 있다.

본 논문에서 제시한 프레임워크에서 멀티스레드를 이용한 렌더링을 지원하는 점을 확장하여, 앞으로 분산 환경에서 작동하는 가시화 시스템을 구성한다면 대규모 데이터의 가시화 과제를 위해 유용할 것이다.

## 감사의 글

본 연구는 지식경제부와 한국산업기술재단의 전략기술인력양성사업으로 수행된 연구결과이며, 이 연구에 참여한 연구자는 '2단계 BK21사업'의 지원비를 받았음.

## 참고 문헌

[1] J. C. Hart, "Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces," *The Visual Computer*, vol. 12, no. 10, pp. 527–545, 1996.

[2] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM SIGGRAPH*, vol. 21, no. 4, pp. 163–169, 1987.

[3] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of hermite data," *ACM SIGGRAPH*, pp. 339–346, 2002.

[4] C. Upson, T. Faulhaber, Jr., D. Kamins, D. H. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam, "The application visualization system: A computational environment for scientific visualization," *IEEE Comput. Graph. Appl.*, vol. 9, no. 4, pp. 30–42, 1989.

[5] W. J. Schroeder, K. M. Martin, and W. E. Lorensen, "The design and implementation of an object-oriented toolkit for 3d graphics and visualization," *IEEE Visualization*, pp. 93–100, 1996.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[7] J. A. Baerentzen and H. Aanaes, "Signed distance computation using the angle weighted pseudonormal," *IEEE TVCG*, vol. 11, no. 3, pp. 243–253, 2005.

[8] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, "Adaptively sampled distance fields: a general representation of shape for computer graphics," *ACM SIGGRAPH*, pp. 249–254, 2000.

[9] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel, "Multi-level partition of unity implicits," *ACM SIGGRAPH*, vol. 22, no. 3, pp. 463–470, 2003.

[10] S. Lefebvre and H. Hoppe, "Compressed random-access trees for spatially coherent data," in *Eurographics Symposium on Rendering*, 2007.

[11] M. Shevtsov, "Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes," *Computer Graphics Forum*, vol. 26, no. 3, pp. 395–404, 2007.

## 〈저자소개〉

**이성호**
- 2005년 고려대학교 컴퓨터학과 학사
- 2007년 고려대학교 영상정보처리협동과정 석사
- 2007년~현재 고려대학교 컴퓨터 전파통신공학과 박사과정
- 관심분야 : 지오메트리 합성, 시각화 프레임워크

**감형렬**
- 2008년 고려대학교 컴퓨터학과 학사
- 2010년 고려대학교 컴퓨터 전파통신공학과 석사
- 2010년~현재 고려대학교 영상정보처리협동과정 박사과정
- 관심분야 : 뉴로이미지, 지오메트리 응용

**박태정**
- 1997년 서울대학교 전기공학부 학사
- 1999년 서울대학교 전기공학부 석사
- 2006년 서울대학교 전기컴퓨터공학부 박사
- 2006년~현재 고려대학교 BK21 소프트웨어사업단 연구교수
- 관심분야 : 기하 정보 압축, TCAD, 3D 도형 모델링

**김창헌**
- 1979년 고려대학교 경제학과 학사
- 1988년 University of Tsukuba 전자정보 박사
- 1995년 3월~현재 고려대학교 컴퓨터학과 교수
- 2005년 11월~현재 고려대학교 컴퓨터과학기술대학원 원장
- 2005년 1월 ~ 2006년 3월 한국정보과학회 이사
- 2008년 3월 ~ 2010년 2월 한국컴퓨터그래픽스학회 회장
- 관심분야 : 컴퓨터그래픽스, 물리기반 시뮬레이션, Mesh Processing