# GPU 기반 곡률 추정 알고리즘

김선정

요약

본 논문에서는 3차원 물체의 외형 특징이 되는 곡률을 추정해내는 GPU (Graphics Processing Unit) 기반 알고리즘을 제안한다. 곡률은 3차원 물체의 간략화, 메쉬 분할 등과 같은 기하학적 모델링 분야에서 매우 유용한 수치이다. 그러나 그래픽스 기술의 발달로 대용량의 3차원 물체들이 응용 프로그램에서 사용되고 데이터 용량에 비례하여 곡률을 추정하는 계산 시간도 증가하게 되었다. 만약 실시간으로 곡률을 추정해야 하는 응용 프로그램의 경우, 3차원 물체의 기하학 정보를 모두 메인 메모리에 올려놓고 CPU에서 계산하게 되면 심각한 속도 저하를 가져올 수 있다. 본 논문은 곡률 추정을 위한 계산 시간을 단축시키기 위해, CUDA (Compute Unified Device Architecture)를 사용하여 꼭지점마다 수행되는 주요 계산 과정을 GPU에서 병렬로 수행하도록 구현하였다. 기존의 곡률 계산 방법보다 훨씬 빠른 수행 결과를 보인다.

# GPU Based Algorithm for Estimating Curvatures

Sun-Jeong Kim*

ABSTRACT

This paper proposes the GPU (Graphics Processing Unit) based algorithm for estimating curvatures which is one of properties of the shape of 3D objects. In geometric modeling, curvatures are very useful metrics for mesh simplification, mesh segmentation, and so on. Nowadays the advanced techniques for computer graphics let many applications use huge amounts of graphic data, so that the computing time for estimating curvatures of such a huge graphic object becomes to increase. If an application which requires real-time estimation of curvatures will perform the computation in the CPU with the main memory where are all geometry information of a 3D object, it must make the system to slow down. In this paper, to decrease the computing time for estimating curvatures, some crucial computing parts for each vertex are implemented in the GPU using CUDA (Compute Unified Device Architecture). The results show that the proposed algorithm is definitely faster than previous methods.

Key Words : Curvatures, Differential Geometry, MLS(Moving Least Squares), GPU(Graphics Processing Unit) Programming, CUDA(Compute Unified Device Architecture)

* 한림대학교 유비쿼터스 컴퓨팅학과 (✉ sunkim@hallym.ac.kr)
· 제1저자(First Author) · 교신저자(Correspondent Author) : 김선정
· 투고일자 : 2013년 1월 24일 · 심사수정일자 : 2013년 2월 15일 · 게재확정일자 : 2013년 2월 19일

## Ⅰ. 서 론

다양한 3차원 그래픽 모델들이 많은 응용 프로그램에 사용되면서, 3차원 물체들의 기하학적인 분석이 요구되기 시작했다. 예를 들어 특징 추출, 필터링, 인덱싱과 같은 알고리즘에서 3차원 모델의 기하학적 데이터와 함께 곡률과 같은 미분 기하 정보도 중요한 요소로 등장하게 되었다. 3차원 물체의 외형에서 법선 벡터와 곡률과 같은 미분 기하 정보는 순수 컴퓨터 그래픽스 외에도 신호 처리 분야에서 컨텍스트로 사용되고 있다.

3차원 모델이 이미지나 비디오와 같은 데이터 타입과 가장 큰 차이점은 불규칙하게 샘플링된 데이터라는 점이다. 3차원 표면 위에 꼭지점들은 불규칙적으로 분포되어 있고, 특별한 경우를 제외하고는 모든 꼭지점의 연결성(모서리로 연결된 이웃 꼭지점의 개수)도 동일하지 않다. 그러므로 불규칙적인 삼각형의 크기와 모양으로 구성된 메쉬에 대해서도 곡률 추정 알고리즘이 견고해야지만 유용할 것이다.

실시간으로 곡률을 추정해야 하는 응용 프로그램의 경우, 3차원 물체의 기하학 정보를 메인 메모리에 올려놓고 CPU에서 계산하는 방법에는 한계가 있기 때문에, GPU 메모리를 사용하는 GPU 프로그래밍의 이용이 점차 증가하고 있는 추세이다. 본 논문에서는 3차원 메쉬의 곡률을 빠르게 추정하는 GPU 알고리즘을 제안한다.

## Ⅱ. 관련 연구

3차원 표면의 곡률은 오래전부터 메쉬 간략화[1], 텍스쳐 합성 및 외형 표현[2], 비사실적 렌더링에서 윤곽선 추출[3] 및 라인 드로잉[4] 등과 같은 컴퓨터 그래픽스의 많은 분야에서 사용되어 왔다. 이러한 곡률 추정 방법은 3차원 메쉬가 미분 가능한 부드러운 표면을 표현하고 있다는 가정 하에 수행된다.

3차원 곡률 표현은 여러 가지가 있다. 그 중 normal curvature $\kappa_n$은 법선 벡터를 포함하는 평면으로 3차원 표면을 잘랐을 때 생기는 2차 곡선에 가장 잘 근사하는 원의 반지름의 역수를 뜻한다. 이런 normal curvature는 잘라내는 평면에 따라 다양하지만, 부드러운 곡면에서는 다음과 같은 수식을 만족한다.

$$\kappa_n = (s\ t)\begin{pmatrix} e & f \\ f & g \end{pmatrix}\begin{pmatrix} s \\ t \end{pmatrix} = (s\ t)\boldsymbol{II}\begin{pmatrix} s \\ t \end{pmatrix}$$

여기서 (*s*, *t*)는 탄젠트 평면에서의 직교 좌표계를 표현하는 단위 벡터이다. 행렬 II는 대칭행렬로 Weingarten 행렬 또는 2차 fundamental tensor이며, 다음과 같이 표현될 수 있다.

$$\kappa_n = (s'\ t')\begin{pmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{pmatrix}\begin{pmatrix} s' \\ t' \end{pmatrix} = \kappa_1 s'^2 + \kappa_2 t'^2$$

여기서 $\kappa_1$과 $\kappa_2$는 기본 곡률(principal curvatures)이고 (*s*', *t*')는 기본 곡률 방향(principal directions)으로 곡률이 최댓값과 최솟값을 가질 때 normal curvature의 (*s*, *t*)를 뜻한다. 이 기본 곡률은 컴퓨터 그래픽스에서 remeshing, smoothing, 메쉬 분할 등과 같은 여러 분야에서 사용되고 있다. 이러한 기본 곡률을 추정하는 방법은 매우 많으나 3종류로 나누어 볼 수 있다.

### 2.1 Patching Fitting 방법

이 방법은 로컬 영역에 있는 꼭지점 데이터를 이용하여 분석 가능한 표면(특히 다항식)으로 근사함으로써 3차원 표면의 곡률을 분석한다. 만약 부드러운 표면에 꼭지점들이 놓여 있다면 이 방법으로 정확한 곡률 계산이 가능하다[5]. 그러나 불안정한 degenerate한 곳엔 꼭지점들이 놓여 있으면, 표면 근사가 부정확하거나 불가능하므로 곡률 계산

도 힘들어 진다. 그러나 만약 점 데이터가 존재하고 꼭지점마다 fitting 방법을 이용하여 법선 벡터를 추정한다면, 이런 degenerate 경우를 피할 수 있고 정확도도 높일 수 있다[6].

### 2.2 Normal Curvature 기반 방법

우선 각 꼭지점에 연결되어 있는 모서리들의 방향으로 normal curvature를 추정하고, 그 normal curvature들을 사용하여 2차 fundamental tensor를 찾는다. 일반적으로 점 $p_i$에서 $p_j$ 방향으로 normal curvature를 계산할 때 다음과 같은 수식을 사용한다.

$$\kappa_{ij} = \frac{2n_j \cdot (p_i - p_j)}{|p_i - p_j|^2}$$

만약 점 $p_i$에서 이웃 $p_j$로 향하는 모서리들이 균일하게 모든 방향으로 분포한다면, 기본 곡률은 $\sum_j \kappa_{ij}(p_I - p_j)(p_i - p_j)^T$의 eigenvalue들로 구할 수 있다[7]. 그러나 일반적인 3차원 모델은 모서리의 방향들이 균일하지 못하기 때문에, 다른 보완적인 방법으로 $\kappa_n$을 샘플링하고 least square를 이용하여 fitting 시키거나[8], 비슷한 fitting 방법을 사용하여 평균 곡률 $H = (\kappa_1 + \kappa_2)/2$과 가우시안 곡률 $K = \kappa_1\kappa_2$을 추정하는 방법[9]을 사용한다. 그러나 이 방법도 patch fitting 방법과 마찬가지로 불안정하다는 단점을 여전히 가지고 있다.

### 2.3 Tensor Averaging 방법

폴리곤 메쉬의 작은 영역에 대해 curvature tensor의 평균을 계산하는 방법[10]이다. 편평한 하나의 삼각형 위와 모서리 직선 위에서는 곡률이 0이지만 Voronoi 영역과 같은 곳에서 곡률의 평균은 0이 아니다. Curvature tensor의 평균을 구하는 방법은 잘 정의되어 있고 안정적이지만, 작은 삼각형들이 너무 밀집되어 있는 곳에서는 오류가 발생할 수 있을 수 있다.

## III. 본 론

기존 연구에서 Rusinkiewicz[11]은 CPU에서 곡률을 계산하였고, 최근 들어 Griffin et al.[12]은 Rusinkiewicz의 계산법을 GPU에서 구현하였다. 본 논문에서는 Griffin et al.에서 사용된 곡률의 병렬 계산 구조를 따라하되, 보다 정확한 곡률 계산을 위해 다른 곡률 계산법을 사용한다. Rusinkiewicz의 제안 방법은 각 삼각형 마다 곡률을 계산하기 위해 모서리로 연결된 꼭지점의 법선 벡터의 방향 차이에 대한 선형 제약조건을 만들고 이를 least square로 풀었다. 그리고 이렇게 계산된 삼각형의 곡률들을 가지고, 한 꼭지점에 연결된 삼각형들의 곡률을 평균 내어 꼭지점의 곡률로 사용하였다. 그러나 본 논문에서는 삼각형의 곡률들을 평균 내어 꼭지점을 곡률을 추정하는 tensor averaging 방법보다는, 직접 꼭지점의 곡률을 추정하는 patching fitting 방법과 normal curvature 기반 방법을 사용하여 정확한 곡률을 계산한다. Patching fitting 방법으로는 MLS(moving least square)를 이용한 local fitting 방법을 구현하고, normal curvature 기반 방법으로는 이산 곡률 추정 방법을 구현한다.

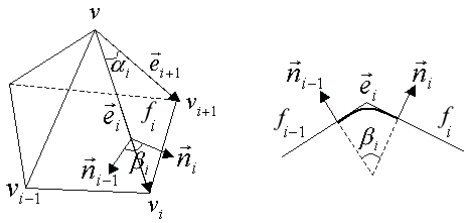
### 3.1 이산 곡률 추정 방법

이산 곡률 추정 방법은 원칙적으로 부드러운 곡면에서만 계산되는 곡률을 이산적인 메쉬에서 추정할 수 있도록 만든 방법으로 꼭지점의 가우시안 곡률 $K$와 평균 곡률 $H$를 다음과 같은 공식을 사용하여 계산할 수 있다.

$$K = \frac{1}{3\sum_{i=0}^{n-1} A_i}\left(2\pi - \sum_{i=0}^{n-1}\alpha_i\right)$$

$$H = \frac{1}{4}\sum_{i=0}^{n-1}\|\overrightarrow{e_i}\|\beta_i$$

여기서 $\alpha_i$는 삼각형의 내각, $\beta_i$는 이면각, $A_i$는 삼각형의 면적, $e_i$는 모서리를 뜻한다(<그림 1>).



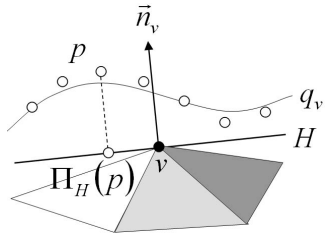

<그림 1> 이산 곡률 추정을 위한 내각 $\alpha_i$ 와 이면각 $\beta_i$
<Fig. 1> Angle $\alpha_i$ and dihedral angle $\beta_i$ for discrete curvature estimation

## 3.2 MLS를 이용한 미분 곡률 추정 방법

주곡률(principal curvatures)를 추정하기 위해 한 꼭지점 주위의 이웃점들을 이용하여 MLS를 local fitting 시킨 다음, MLS 다항식을 미분하여 주곡률의 방향과 최대/최소 곡률을 추정한다. 한 꼭지점과 k개의 이웃점을 이용하여, 탄젠트 평면을 도메인으로 하는 n차 다항식 평면을 찾아 fitting 시키는 방법으로 빠르게 수행하는 반면 좋은 품질을 나타내 포인트 렌더링과 같은 알고리즘에 많이 사용되었다. 그 수식은 다음과 같다.

$$\sum_{p\in P_v} \| q_v(\Pi_H(p)) - p \|^2 e^{-\frac{\|p-v\|^2}{h^2}}$$

여기서 h는 꼭지점 v로부터 MLS 근사가 영향을 주는 범위로 $h = \sqrt{\frac{\pi R^2}{n}}$ 이다. R은 꼭지점 v를 중심으로 $P_v$의 원소들을 포함하는 경계 구의 반지름이고, n은 $P_v$의 원소 개수이다. $\Pi_H$는 꼭지점 v를 원점으로 하는 참조 평면 H로의 직교 투영시키는 함수를 뜻한다(<그림 2>).




<그림 2> MLS 투영
<Fig. 2> MLS Projection

## 3.3 GPU 프로그래밍

GPU 기반 곡률 추정 알고리즘은 CUDA를 아용하여 구현되었다. <표 1>과 같은 7단계로 구성되어 있으며, 각 단계는 병렬로 처리되었다.

<표 1> 제안 알고리즘의 단계
<Table 1> The steps of proposed algorithm

| | |
|---|---|
| 단계 1 | 삼각형마다 법선 벡터를 계산<br>(면적도 함께 계산) |
| 단계 2 | 꼭지점마다 법선 벡터를 계산<br>(이웃하는 삼각형들의 법선 벡터의 평균) |
| 단계 3 | 꼭지점마다 이산 곡률 추정 |
| 단계 4 | 꼭지점마다 k개의 이웃점을 모음 |
| 단계 5 | 꼭지점마다 local fitting을 위한 로컬 좌표계 변환 수행 |
| 단계 6 | 꼭지점마다 MLS의 local fitting 수행 |
| 단계 7 | 꼭지점마다 주곡률과 주곡률 방향을 계산<br>(MLS의 basis 방향과 다항식 미분) |

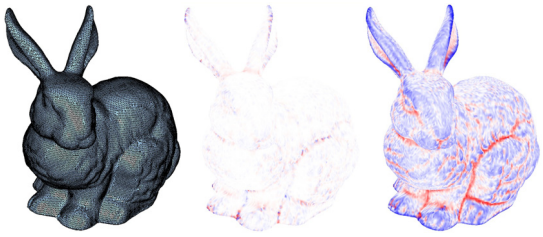
각 단계는 다른 단계들과 독립적이며 이전 단계

가 끝난 다음에 다음 단계가 수행된다. 꼭지점을 기본 단위로 계산하는 단계는 독립적이라 잠금 기능이나 동기화가 없어도 가능하나, 삼각형을 기본 단위로 계산하는 단계(예: 면적의 합이나 면 법선 벡터의 평균)에서는 데이터를 쓰는 도중 다른 쓰레드가 접근하지 못하게 하는 잠금 기능과 모든 쓰레드가 계산 완료하기를 기다리는 동기화 기능이 필요하다.

간단히 설명하면, 이산 곡률 추정 방법에서는 쓰레드를 이용하여 삼각형의 속성(예: 법선벡터, 면적)을 병렬로 계산한 다음, 각 꼭지점의 가우시안 곡률과 평균 곡률을 역시 병렬로 계산한다. MLS 미분 곡률 방법에서는 이산 곡률 추정 방법과 동일하게 우선 삼각형의 속성을 병렬로 계산한 다음, 꼭지점의 속성(예: 법선 벡터, 탄젠트 평면의 좌표계, 이웃하는 점들의 집합)을 병렬로 계산한다. 그리고 각 꼭지점마다 MLS 다항식을 병렬로 계산한 다음, 다항식을 미분하여 기본 곡률을 계산한다.

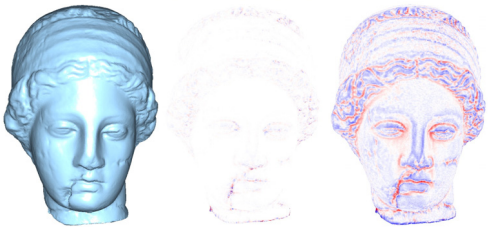
## Ⅳ. 실험 결과

<그림 3>은 이산 곡률 추정 방법에 대한 결과 그림이다. 기본 곡률의 평균인 평균 곡률은 모서리(edge)를 잘 표현하고, 기본 곡률의 곱인 가우시안 곡률 코너(corner)를 잘 표현한다. 이 두 곡률을 잘 이용하면, 3차원 모델의 외형 특징을 잘 추출해 낼 수 있다.

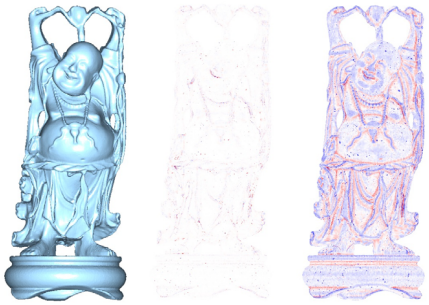


(a) Stanford bunny 모델

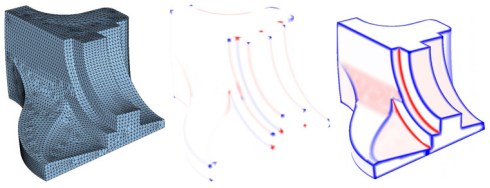
(b) Hand bone 모델



(c) Venus 모델



(d) Happy buddha 모델



(e) Fan disk 모델

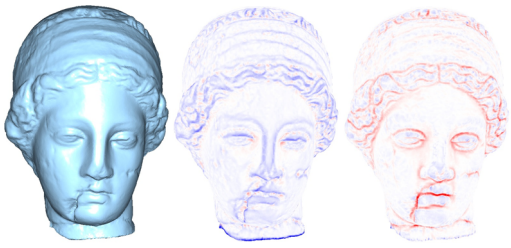
(f) Saddle 모델

그림 3. 이산 곡률 추정 결과 (좌: 원본 모델, 중앙: 가우시안 곡률, 우: 평균 곡률) (파란색: 양의 곡률값, 빨간색: 음의 곡률값)

Fig. 3. Results of estimating discrete curvatures (left: original model, middle: Gaussian curvature, right: mean curvature) (blue: positive curvatures, red: negative curvatures)

(a) Hand bone 모델

(b) Golf club 모델

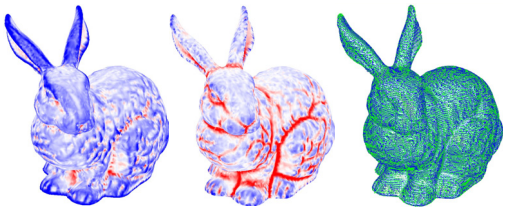


(c) Venus 모델

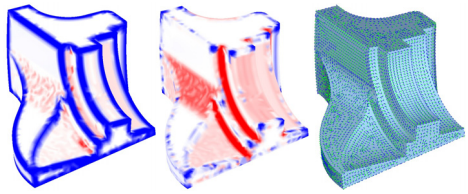
(d) Horse 모델

그림 4. MLS 미분 곡률 추정 결과 (좌: 원본 모델, 중앙: 최대 곡률, 우: 최소 곡률) (파란색: 양의 곡률값, 빨간색: 음의 곡률값)

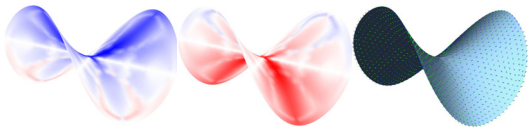
Fig. 4. Results of estimating MLS differential curvatures (left: origianl model, middle: maximum curvature, right: minimum curvature) (blue: positive curvatures, red: negative curvatures)



(a) Stanford bunny 모델



(b) Fan disk 모델



(c) Saddle 모델

그림 5. MLS 미분 곡률 추정 결과 (좌: 최대 곡률, 중앙: 최소 곡률, 우: 주곡률 방향) (파란색: 양의 곡률값, 빨간색: 음의 곡률값, 초록색/파란색 화살표: 최대/최소 곡률의 방향)

Fig. 5. Results of estimating MLS differential curvatures (left: maximum curvature, middle: minimum curvature, right: directions of principal curvatures) (blue: positive curvatures, red: negative curvatures, blue/green arrows: directions of maximum/minimum curvatures)

<그림 4>와 <그림 5>는 MLS를 이용한 미분 곡률 추정 방법에 대한 결과이다. Local fitting된 MLS 다항식을 미분하여 얻은 기본 곡률의 결과를 나타낸다. 기본 곡률은 주어진 꼭지점의 최대 곡률과 최소 곡률로 가우시안 곡률과 평균 곡률과 마찬가지로 3차원 모델의 외형 특징을 잘 유추해낼 수 있고, <그림 5>에서 보는 바와 같이 기본 곡률 방향을 이용하여 parameterization이나 texture mapping을

수행할 수 있다.

표 2. CPU vs. GPU 수행시간 비교
Table 2. Comparison with processing time in CPU vs. GPU

| 모델명 | 꼭지점 / 삼각형 (개) | CPU 계산 시간 (ms) | GPU 계산 시간 (ms) | GPU 메모리 사용량 (KB) |
|---|---|---|---|---|
| Happy buddha | 543,652 / 1,087,716 | 803 | 75.7 | 72,213 |
| Dragon | 437,645 / 871,414 | 631 | 59.3 | 54,614 |
| Hand bone | 327,323 / 654,666 | 449 | 34.9 | 35,801 |
| Golf Club | 209,779 / 419,554 | 330 | 42.0 | 24,583 |
| Venus | 134,345 / 268,686 | 215 | 28.2 | 15,218 |
| Teeth | 116,604 / 233,204 | 185 | 24.5 | 13,208 |
| Rabbit | 67,039 / 134,074 | 101 | 14.1 | 7,332 |
| Horse | 48,485 / 96,966 | 68 | 9.2 | 6,249 |
| Stanford bunny | 34,834 / 69,451 | 51 | 7.4 | 3,804 |
| Fan disk | 6,475 / 12,946 | 8 | 1.3 | 657 |

<표 2>는 곡률 추정을 위해 CPU와 GPU에서 수행된 시간을 비교한 것으로, 예측했듯이 모델 데이터의 양이 크면 클수록 GPU에서 계산 시간이 CPU에서의 계산 시간의 약 10% 밖에 걸리지 않는다는 것을 알 수 있다. GPU 프로그램은 NVIDIA GTX 480 그래픽 카드에서 CUDA 4.0으로 구현되었으며, GTX 480의 compute capability는 버전 2.0이며 15개 multiprocessor를 가지고 있다. CPU 프로그램은 Intel(R) Core(TM) i7 프로세서에서 12GB RAM 환경에서 구현되었다. CPU와 GPU에서의 곡률을 계산하는 수행 시간의 비교 수치는 10회 이상 곡률 계산의 수행 시간을 측정하여 평균을 구한 값이다. 두 프로그램의 계산 결과는 거의 비슷하나, 복잡한 모델의 경우 소수 뒷자리 부분에서 차이를 보인다. 이는 GPU는 GFLOP (giga floating point per operation)기능으로 보다 정확한 소수점 계산이 수행 가능하기 때문으로 유추된다.

## Ⅴ. 결론 및 향후 연구

본 논문에서는 GPU 기반 곡률 추정 알고리즘을 제안하였다. 가우시안 곡률과 평균 곡률을 구하기 위해서는 이산 곡률 추정 방법과 주곡률 방향을 구하기 위해서는 MLS 미분 곡률 방법을 사용하였다. 특히 빠른 곡률 추정을 위해 CUDA를 이용하여 주요 계산 부분을 GPU에서 병렬로 처리함으로써 속도 개선을 가져올 수 있었다.

향후 연구로는 GPU에서 계산된 곡률의 정확도를 좀 더 높이기 위해 계산 순서의 최적화에 대해 연구하고, 이렇게 계산된 곡률을 메쉬 분할이나 비사실적 렌더링 분야에 응용할 것이다.

## 참고 문헌

[1] P. Heckbert and M. Garland, "Optimal Triangulation and Quadric-Based Surface Simplification", Journal of Computational Geometry: Theory and Applications, Vol. 14, No. 1-3, pp. 49-65, 1999.

[2] G. Gorla, V. Interrante, and G. Sapiro, "Texture Synthesis for 3D Shape Representation", *IEEE Transactions on Visualization and Computer Graphics*, Vol. 9, No. 4, pp. 512-524, 2003.

[3] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive Contours for Conveying Shape", *ACM Transactions on Graphics*, Vol. 22, No. 3, pp. 848-855, 2003.

[4] T. Judd, F. Durand, and E. Adelson, "Apparent Ridges for Line Drawing", *ACM Transactions on Graphics*, Vol. 26,

No. 3, pp. 19:1-19:7, 2007.

[5] F. Cazals and M. Pouget, "Estimating Differential Quantities Using Polynomial Fiiting of Osculating Jets", *In Proc. of Symposium on Geometry Processing 2003*, pp. 177-187, 2003.

[6] J. Goldfeather and V. Interrante, "A Novel Cubic-Order Algorithm for Approximating Principal Direction Vectors", *ACM Transactions on Graphics*, Vol. 23, No. 1, pp. 45-63, 2004.

[7] D. L. Page, A. Koschan, Y. Sun, J. Paik, and A. Abidi, "Robust Crease Detection and Curvature Estimation of Piecewise Smooth Surface from Triangle Mesh Approximations Using Normal Voting", *In Proc. of Conference on Computer Vision and Pattern Recognition*, pp. 162-167, 2001.

[8] E. Hameiri and I. Shimshoni, "Estimating the Principal Curvatures and the Darboux Frame from Real 3D Range Data", *In Proc. of International Symposium on 3D Data Processing Visualization and Transmission(3DPVT)*, pp. 258, 2002,

[9] M. Meyer, M. Desbrun, P. Schröder and A. H. Barr, "Discrete Differential-Geometry Operators for Triangulated 2-Manifolds", *Visualization and Mathematics III*, Springer-Verlag, pp. 35-57, 2003.

[10] P. Alliez, D. Cohen-Steiner, O, Delillers, B. Lévy, and M. Desbrun, "Aniosotropic Polygonal Remeshing", ACM Transactions on Graphics, Vol. 22, No. 3, pp. 485-493, 2003.

[11] S. Rusinkiewicz. "Estimating Curvatures and Their Derivatives on Triangle Meshes", *In Proc. of 3D Data Processing, Visualization, & Transmission 2004*, pp. 486-493, 2004.

[12] W. Griffin, Y. Wang, D. Berrios, and M. Olano. "GPU Curvature Estimation on Deformable Meshes", *In Proc. of Interactive 3D Graphics and Games 2011*, pp. 159-166, 2011.

## 감사의 글

이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2011-0015072).

## 저자소개

**김선정(Sun-Jeong Kim)**

1996년 고려대학교 컴퓨터학과 학사
1996년 고려대학교 컴퓨터학과 석사
2003년 고려대학교 컴퓨터학과 박사

2005년~현재 한림대학교 유비쿼터스 컴퓨팅학과 부교수
※ 관심분야 : 컴퓨터 그래픽스, 3차원 게임, 가상현실