# GPU를 이용한 Gabor Texture 특징점 기반의 금속 패드 변색 분류 알고리즘

# Discolored Metal Pad Image Classification Based on Gabor Texture Features Using GPU

최 학 남, 박 은 수, 김 준 철, 김 학 일*
(Xue-Nan Cui, Eun-soo Park, Jun-Chul Kim, and Hakil Kim)

**Abstract:** This paper presents a Gabor texture feature extraction method for classification of discolored Metal pad images using GPU(Graphics Processing Unit). The proposed algorithm extracts the texture information using Gabor filters and constructs a pattern map using the extracted information. Finally, the golden pad images are classified by utilizing the feature vectors which are extracted from the constructed pattern map. In order to evaluate the performance of the Gabor texture feature extraction algorithm based on GPU, a sequential processing and parallel processing using OpenMP in CPU of this algorithm were adopted. Also, the proposed algorithm was implemented by using Global memory and Shared memory in GPU. The experimental results were demonstrated that the method using Shared memory in GPU provides the best performance. For evaluating the effectiveness of extracted Gabor texture features, an experimental validation has been conducted on a database of 20 Metal pad images and the experiment has shown no mis-classification.

**Keywords:** discolor, texture, GPU, P2C transform, pattern map

## I. 서론

GPU (Graphics Processing Unit)는 1999년에 NVIDIA에서 처음 발표되었고 주요 목적은 그래픽 처리 작업을 수행하는 것이다. 최근 들어 GPU 장치의 비약적인 발전으로 인하여 그래픽 처리뿐 만아니라 일반 응용 프로그램에서도 GPU를 사용할 수 있게 되었다. 최근 출시되고 있는 GPU들은 병렬처리와 수학연산에 특화 되어 있어 트랜지스터가 흐름제어나 데이터 캐싱보다 데이터 처리에 집중되어 있다. 따라서 데이터 처리가 많은 응용분야에서는 기존의 CPU를 이용하여 처리 하는 것보다는 GPU를 이용하면 훨씬 좋은 성능을 가져올 수 있다. 아래의 그림 1에서도 알 수 있듯이 CPU와 GPU의 구조특성상 GPU가 훨씬 많은 ALU를 가지고 있음을 알 수 있다. 이러한 구조적 특성 때문에 GPU에서의 처리속도가 훨씬 빠르게 된다. 특히 영상처리와 같이 반복 연산이 많은 응용분야에서는 더욱 많은 성능향상을 가져올 수 있다.

Texture 정보는 영상처리의 많은 분야에서 유용하게 사용되는 정보이고 매우 중요한 정보이다. 많은 Texture 정보 추출 알고리즘들은 통계학적 기반의 방식을 취하거나, 웨이블릿과 같은 멀티채널 필터기반의 방식을 취한다[1]. Texture 특징점 추출 알고리즘의 기존 연구를 살펴보면 Co-occurrence matrices를 이용한 GLCM (Gray Level Co-occurrence Matrix) 방법[5], Spot, Line, Edge, Corner 등 영상 고유의 특성을 이용한 Local-binary pattern 방법[3], Gabor filter를 이용한 방법[2,4], 처리할 영상에 PCA (Principle Component Analysis)를 적용하여 사용할 필터를 구성하는 방법[1]등이 있다.

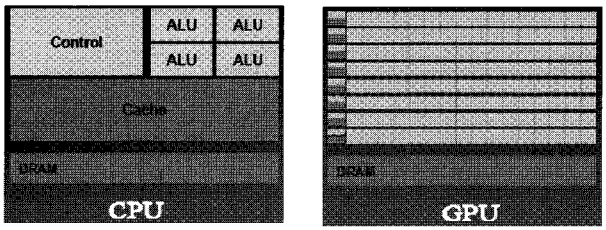



그림 1. CPU와 GPU의 구조 비교도[7].
Fig. 1. Comparison of CPU and GPU structures.

이러한 방법들은 장단점을 가지고 있어 응용분야에 따라 적용되고 있다. GLCM방법과 같은 경우는 표면의 거칠기 등을 판단하는 기준으로 사용되고, PCA를 이용한 방식은 성별구분 알고리즘에 사용되고 있다[6]. 특히 Gabor 필터를 이용한 방식은 조명의 영향을 적게 받고 회전에 강인한 특징점으로 알려져 있다[1,2]. 이처럼 Texture 특징점은 영상처리에 있어서 이미 아주 좋은 정보임을 증명 받고 있음에도 불구하고, 계산량이 많은 단점이 있어 실시간 처리가 필요한 응용분야에는 적용하기 어려운 실정이다.

본 논문에서는 이러한 처리시간 문제를 해결하기 위하여 Gabor filter 기반의 Texture 특징점 추출 알고리즘을 GPU를 이용하여 구현하였으며, 그 성능을 확인하기 위하여 CPU에 순차처리와 병렬처리 두 가지 방법으로 개발하여 비교하였다. 실험결과 GPU상에서 shared 메모리를 이용하였을 때 가장 좋은 성능을 나타내었고, 실시간 처리가 가능함을 확

* 책임저자(Corresponding Author)
논문접수: 2009. 5. 15., 채택확정: 2009. 6. 15.
최학남, 박은수, 김준철: 인하대학교 정보공학과
(xncui@vision.inha.ac.kr/espark@vision.inha.ac.kr/jckim@vision.inha.ac.kr)
김학일: 인하대학교 정보통신공학부 교수(hikim@inha.ac.kr)
※ 본 논문은 삼성전기 생산기술연구소에서 지원하여 연구하였음.

인하였다.

AFVI (Automated Final Vision Inspection) 시스템은 AOI (Automated Optical Inspection) 시스템보다 정교한 알고리즘을 필요로 하고, 정교한 알고리즘을 개발하기 위하여 texture 정보를 이용해야 한다. 따라서 본 논문에서는 Gabor 필터를 이용하여 texture 특징점을 추출하고, 추출된 특징점을 이용하여 AFVI 마지막 단계에서 사용하게 될 금속 패드의 변색 분류 알고리즘을 개발하고자 한다. 현재 생산현장에서는 변색 분류에 관한 알고리즘이 적용되지 않고 사람이 직접 검사하는 방식을 취하고 있는 실정이다.

본 논문은 서론, Gabor 필터를 이용한 Texture 특징점 추출 알고리즘, 금속패드 변색 분류 알고리즘, 실험결과 및 분석, 결론과 같이 다섯 부분으로 구성되었다. 서론에서는 GPU 하드웨어 동향과 Texture 추출 알고리즘에 대한 기존 연구에 대해 서술하였고, Gabor 필터를 이용한 Texture 특징점 추출 알고리즘과 금속 패드 변색 분류 알고리즘 부분에서는 제안한 알고리즘과 알고리즘 개발방법에 대해 서술하였으며, 실험결과 및 분석에서는 본 논문에서 개발한 알고리즘에 대한 결과와 그에 대한 성능평가 및 분석을 진행하였으며, 마지막으로 결론 및 향후 연구에 대해서 서술하였다.

## II. Gabor 필터를 이용한 픽셀 패턴 기반의 Texture 특징점 추출 알고리즘

### 1. Gabor 필터

Gabor 필터는 아래의 식 (1)과 같이 주어진다. 식 (1)에서 $y' = -x\sin\theta + y\cos\theta$, $x' = x\cos\theta + y\sin\theta$이고, $\lambda$는 코사인 함수의 파장을 나타내고, $\theta$는 Gabor 함수의 방향성을, $\psi$는 위상차를, $\sigma$는 가우시안 함수의 표준편차를 나타내며, $\gamma$는 Gabor 필터의 모양을 결정하는 요소이다.

$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\psi\right) \quad (1)$$

Gabor 필터적용은 기타 필터 적용과 마찬가지로 아래의 식 (2)와 같이 입력영상과의 Convolution을 통하여 이루어진다.

$$G_F(x,y;\lambda,\theta,\psi,\sigma,\gamma) = I(x,y) * g(x,y;\lambda,\theta,\psi,\sigma,\gamma) \quad (2)$$

본 논문에서는 다양한 Gabor 특징점을 추출하기 위하여 $\sigma = 2.0$, $\lambda = 0.55$, $\psi = \frac{\pi}{2}$, $\gamma = 1$을 취하고, $\theta = \{0, \frac{\pi}{8}, \frac{2\pi}{8}, \frac{3\pi}{8}, \frac{4\pi}{8}, \frac{5\pi}{8}, \frac{6\pi}{8}, \frac{7\pi}{8}\}$를 취하였다. 즉 기타 요소들은 변하지 않고 방향성만 틀리게 하여 적용하였다. 따라서 식 (2)를 $G_F(x,y;\theta(m))$와 같은 형식으로 아래에서 표현한다.

### 2. 특징점 추출 방법

특징점 추출방법은 다음과 같은 두 단계로 나눌 수 있다. 첫 번째 단계는 Gabor 필터를 적용한 영상을 이용하여 pattern map을 구하는 단계이고, 두 번째 단계는 pattern map을 이용하여 최종 Texture 특징벡터를 추출하는 단계이다.

식 (3)에서 $G_F(\theta(m))$은 $G_F(x,y;\theta(m))$를 나타내고, $m$은 $\theta$의 인덱스를 나타내며, 여덟 개의 필터를 사용했을 경우 $m = 8$이다.

$$G_F(\theta(k)) = \max\left(G_F(\theta(1)), G_F(\theta(2)), \cdots G_F(\theta(m))\right) \quad (3)$$

식 (4)에서 $PM$은 pattern map을 나타내며 $k$는 식 (3)에 의해 구하여진 $G_F$ 값들 중 maximum 값의 인덱스를 나타낸다. 따라서 $PM$의 값의 범위는 1~$m$사이의 값을 가지게 되며 필터의 개수에 종속된다.

$$PM(x,y) = k(1 \le k \le m; \text{정수}) \quad (4)$$

식 (5)는 특징점 window 범위 내에서의 특징점 추출 식이고, $F_H F_W$는 특징점 window 크기를 나타내며, $h_l(x,y)$는 식 (6)에서 정의되었다. 식 (7)은 최종 Gabor 특징벡터를 나타내고, 식 (5)에 의해 구해진 $f_l$로 구성된다.

$$f_l(i,j) = \sum_{x=i-\frac{F_H-1}{2}}^{i+\frac{F_H-1}{2}} \sum_{y=j-\frac{F_W-1}{2}}^{j+\frac{F_W-1}{2}} h_l(x,y), l = 1,\cdots m \quad (5)$$

$$h_l(x,y) = \begin{cases} 1 \; if \; PM(x,y) = l \\ 0 \; otherwise \end{cases} \quad (6)$$

$$F_G = (f_1, \cdots, f_m) \quad (7)$$

### 3. CPU와 GPU을 이용한 개발방법

본 논문에서는 CPU상에서의 순차처리, CPU상에서의 OpenMP를 이용한 병렬처리, GPU상에서의 global 메모리를 이용한 방법과 shared 메모리를 이용한 방법 등, 네 가지 방식으로 위에서 제시한 알고리즘을 개발하였다.

CPU상에서의 순차처리는 C++를 이용하여 개발하였고, 순차처리에 OpenMP를 적용하여 병렬처리를 하였다. OpenMP는 병렬처리 라이브러리로써 그림 2와 같이 제공되는 명령들을 순차처리에 적용하면 간단하게 병렬처리를 할 수 있다. 물론 개발 시 데이터 종속성, 병렬성 등을 고려해야 한다.

본 논문에서는 filtering, pattern map을 구하는 부분, 특징점 추출하는 부분을 그림 3에서와 같이 OpenMP를 적용하여 개발하였다. 그림 3의 각 단계는 II에서 서술한 알고리즘과 같이 데이터 종속성, 데이터 경쟁 등 병렬처리를 저해하는 요소들이 존재하지 않는다. 따라서 순차처리 프로그램을 기반으로 그림 2에서와 같이 간단한 OpenMP 함수들을 이용하여 구현이 가능하다.

GPU상에서의 개발은 NVIDIA에서 제공하는 CUDA를 이용하여 그림 4와 같이 개발하였다. 본 논문에서는 아래의 그림 5와 같이 블록과 그리드를 세팅하였다. 그림 5에서 bl, gr은 Gabor 필터 생성 시 사용하기 위한 세팅이고 kernel

```
for (int i=0; i<Height; i++){
    for (int j = 0; j<Width; j++){
        ...
    }
}
```

```
#pragma omp parallel for
    for (int i=0; i<Height; i++){
#pragma omp parallel for
        for (int j = 0; j<Width; j++){
        ...
        }
}
```

그림 2. 순차처리와 OpenMP를 이용한 병렬처리.

Fig. 2. Parallel processing using sequance and OpenMP.

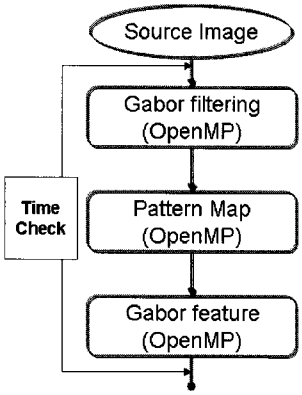



그림 3. CPU에서의 특징점 추출과정.
Fig. 3. Feature extraction with CPU.

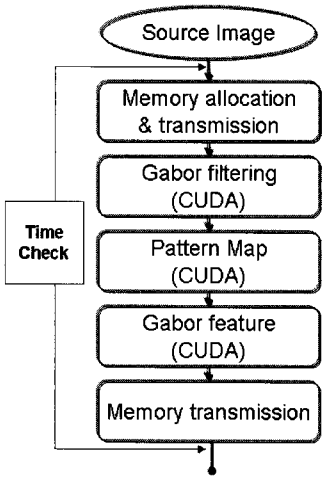



그림 4. GPU에서의 특징점 추출과정.
Fig. 4. Feature extraction with GPU.

```
dim3 bl ( kernel_size, kernel_size );
dim3 gr ( 1, 1, 1 );
dim3 block = dim3(16,16,1);
dim3 grid = dim3(width/block.x,height/block.y);
```

그림 5. 블록, 그리드 세팅.
Fig. 5. Block and grid setting.

size는 Gabor 필터 크기를 나타낸다. Block, grid는 filtering, pattern map 구하기, 특징점 추출 시 사용하기 위한 세팅이고, width, height는 영상의 크기를 나타낸다.

그림 6은 filtering에 대한 kernel함수 수행을 나타낸 그림이다. 그림에서 cuda_Global_Filter2D는 global 메모리를 이용한 kernel함수 실행을 나타내고, cuda_Shared_Filter2D는 shared 메모리를 이용한 kernel함수 실행을 나타낸다.

Global 메모리를 이용한 함수는 CPU상의 데이터를 Global 메모리에 올려서 그림 5에서 생성한 스레드에 의하여 계산된다. Shared 메모리를 이용한 함수는 Gabor kernel을 shared 메모리에 올려서 처리하여 성능향상을 가져왔다. shared 메모리는 on-chip 메모리이기 때문에 global 메모리를 사용하는 것 보다 훨씬 효과적이다. 실험을 통하여 성능을 확인할 수 있다.

```
cuda_Global_Filter2D<<< grid, block>>>
(d_src,width,height,kernel,kernel_size,kernel_size,d_dst);

cuda_Shared_Filter2D
<<< grid, block, sizeof ( float ) * kernel_size*kernel_size >>>
(d_src,width,height,kernel,kernel_size,kernel_size,d_dst);
```

그림 6. 커널함수 실행.
Fig. 6. Execution of the kernel functions.

## III. 금속 패드 변색 분류 알고리즘

본 논문에서는 금속 패드 변색 분류 알고리즘을 아래의 그림 7과 같이 구성하였다. 그림 7에서 preprocessing은 금속 패드 부분을 추출하는 부분과 추출된 ROI (Region of Interest)에 대하여 P2C (Polar to Cartesian)변환을 하는 부분으로 되어 있고, Gabor filtering, pattern map, feature extraction은 II부분에서 설명한 내용과 동일하며, classification은 변색 분류 알고리즘 부분으로써 변색과 거칠기를 분류한다.

1. 전처리(Preprocessing)

그림 8은 전처리 과정을 나타낸다. 전 처리 과정은 크게 두 단계가 있다. 입력 금속 패드 영상에 대하여 처리할 ROI영역을 추출하는 과정과 P2C변환을 하는 과정이 전처리 과정에 포함된다.

ROI 영역 추출과정은 우선 Otsu[8]방법을 이용하여 이진화를 하고, 모폴로지 연산을 진행하여 잡음을 제거한 후, 무게중심을 이용하여 ROI영역의 중심좌표를 계산하고, 반지름을 계산하여 ROI영역을 확정한다.

다음 중심좌표와 반지름 정보를 이용하여 P2C변환을 하

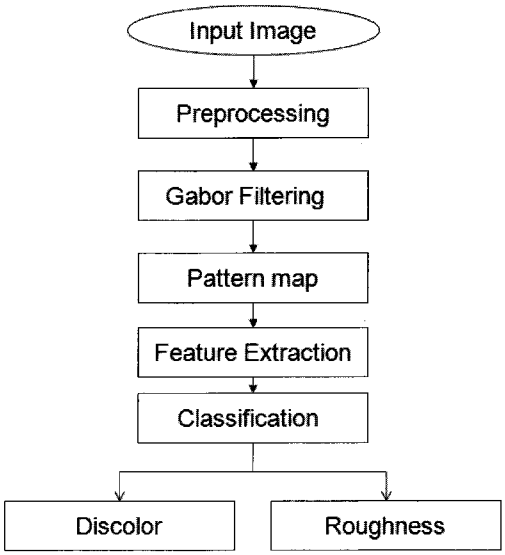



그림 7. 금속 패드 변색 분류 알고리즘.
Fig. 7. Gold pad discolor classification algorithm.


그림 8. 전처리 과정.
Fig. 8. Flowchart of preprocessing.

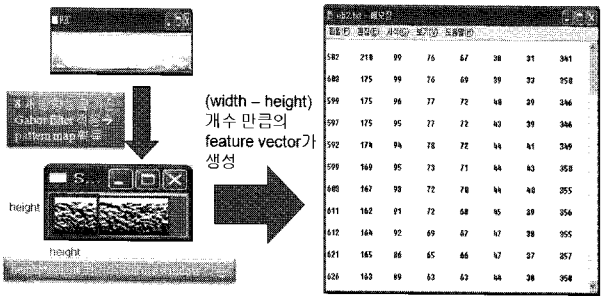



그림 9. Gabor 특징점 추출 과정.
Fig. 9. Process of Gabor feature extraction.

여 펼쳐진 영상을 얻는다. P2C 변환을 하면 반지름 × 회전각밀도 크기의 P2C영상을 얻는데 여기서 회전각밀도는 1도 단위로 하면 360이 되고, 2도 단위로 하면 180이 된다. P2C 변환을 하는 이유는 ROI영역의 정보가 손실되지 않기 위함이다. 만약 P2C변환을 하지 않고 사각형 형태의 ROI를 지정하면 금속 패드 영역의 일부 정보가 사라지게 된다.

2. Gabor 특징점 추출

그림 9는 전처리 과정에서 획득한 P2C 영상에 대하여 II에서 제안한 방법을 적용하는 과정을 나타내다. 그림 9에서 P2C영상에 적용할 Gabor 필터 파라미터는 각각 $\sigma=2.0$, $\lambda=0.55$, $\psi=\frac{\pi}{2}$, $\gamma=1$로 하고 방향 파라미터는 $\theta=\{0, \frac{\pi}{8}, \frac{2\pi}{8}, \frac{3\pi}{8}, \frac{4\pi}{8}, \frac{5\pi}{8}, \frac{6\pi}{8}, \frac{7\pi}{8}\}$와 같이 8방향을 적용하여 pattern map을 구하고, pattern map 영상에 height×height 크기의 feature window를 적용하여 width-height개수만큼의 feature vector를 획득한다. 여기서 height와 width는 각각 pattern map 영상의 크기를 나타낸다.

3. 변색, 거칠기 분류 알고리즘

변색 분류 알고리즘은 추출된 feature vector 에 대하여 평균과 표준편차 값을 구하고 임계값을 취하여 임계값 이상이면 변색 아니면 거칠기로 판단하였다. 즉 아래의 식 (8)~(11)과 같이 특징벡터에 대하여 처리한 후, 식 (12)~(15)를 이용하여 변색, 거칠기 영상을 분류한다. 식 (8)~(15)에서 S는 Score를 나타내고, R은 변색, 거칠기 분류결과를 나타내고, S와 R의 아래첨자는 각각 특징벡터에 표준편차와 평균연산을 적용하는 순서를 나타낸다.

$$S_{SM}=std(mean(F_G)) \tag{8}$$

$$S_{SS}=std(std(F_G)) \tag{9}$$

$$S_{MM}=mean(mean(F_G)) \tag{10}$$

$$S_{MS}=mean(std(F_G)) \tag{11}$$

$$R_{SM}=\begin{cases}Discolor & \text{if } S_{SM}<T_{SM}\\ Roughness & otherwise\end{cases} \tag{12}$$

$$R_{SS}=\begin{cases}Discolor & \text{if } S_{SS}<T_{SS}\\ Roughness & otherwise\end{cases} \tag{13}$$

$$R_{MM}=\begin{cases}Discolor & \text{if } S_{MM}<T_{MM}\\ Roughness & otherwise\end{cases} \tag{14}$$

$$R_{MS}=\begin{cases}Discolor & \text{if } S_{MS}<T_{MS}\\ Roughness & otherwise\end{cases} \tag{15}$$

## IV. 실험결과 및 분석

본 논문에서 제안한 알고리즘에 대한 성능을 평가하기 위하여 CPU와 GPU에서 각각 두 가지 버전으로 특징추출 알고리즘을 개발하였으며, 256×256. 512×512 일반 텍스쳐 영상을 이용하여 성능을 평가하였으며, 실제 생산 공정에서 획득한 12장의 변색 영상과 8장의 거칠기 영상에 대하여 분류알고리즘을 적용하였다. 다음의 표 1은 실험환경에 대한 표이다.

CPU와 GPU상에서의 특징점 추출 성능을 확인하기 위하여 논 본문에서는 다양한 실험을 진행하였다. 그림 10은 256×256 크기의 원본 텍스쳐 영상을 나타내고, 그림 11은 CPU상에서의 순차처리, 병렬처리 및 GPU상에서의 Global 메모리와 shared 메모리를 사용했을 때의 pattern map 결과를 0~255값으로 normalization 해서 출력한 결과이며, 그림 12는 추출한 특징점을 나타낸다.

병렬처리를 함에 있어서 가장 중요한 것은 순차처리와 같은 결과를 도출하는 동시에 성능향상을 가져와야 한다는 것이다. 그림 11, 12로부터 순차처리와 병렬처리에서 같은 결과를 얻을 수 있음을 확인하였다. 특징벡터의 마지막 열은 결과를 확인하기 위하여서 삽입한 count값이다. 그림 11, 12에서 좌상 영상은 순차처리 결과를 나타내고, 우상 영상은 OpenMP를 이용했을 때의 결과를 나타내며, 좌하 영상은 Global 메모리를 이용한 GPU처리 결과를 나타내며, 우하 영상은 Shared 메모리를 이용한 GPU 처리결과를 나타낸다.

표 1. 실험환경.
Table 1. Experimental environment.

| | |
|---|---|
| CPU | Intel® CoreTM2 CPU 6600 @ 2.4GHz, 2401 MHz |
| Memory | 2GB |
| Cash Memory | L1: 32KB, L2: 4MB |
| SIMD | SSE, SSE2, SSE3, SSSE3 |
| GPU | NVIDIA Gforce 8800GT |
| OS | Windows XP |
| Tool | Microsoft Visual Studio 2005 & MATLAB7.01 |

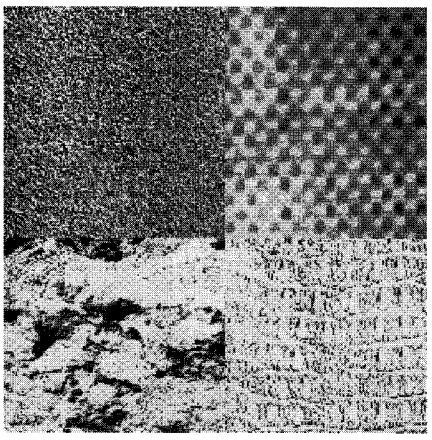


그림 10. 원본 텍스쳐 영상.
Fig. 10. Original texture image.

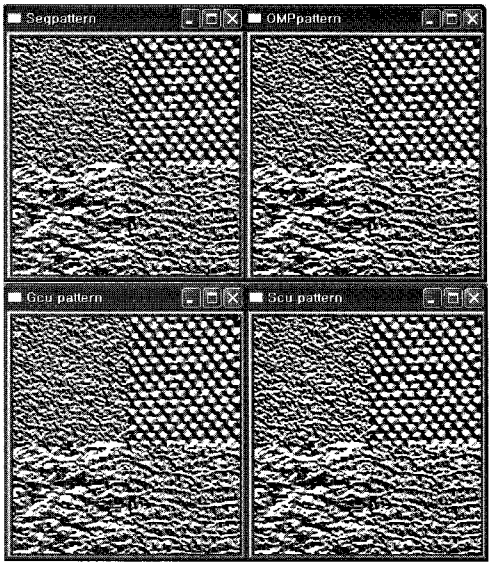



그림 11. 패턴 맵 결과 영상.

Fig. 11. Pattern map result images.

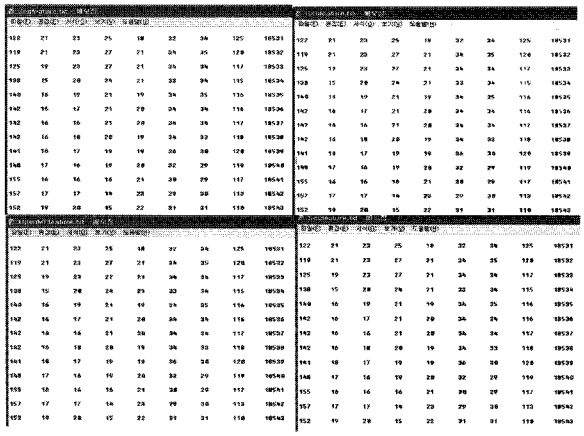


그림 12. 특징벡터 결과.

Fig. 12. Feature vector results.

그림 13은 256×256크기의 영상에서 특징점 window 크기를 15×15로 하고, Gabor 필터 크기를 각각 3×3, 5×5, 7×7, 9×9, 15×15로 했을 때의 결과이다. 실험결과 GPU상에서 shared 메모리를 이용하였을 때 가장 좋은 성능을 나타냄을 확인하였다. 여기서 GPU 프로세싱에서 시간체크는 영상 데이터를 CPU에 GPU로 전송하는 시간과 GPU에서 CPU 전송하는 시간을 포함하여 체크하였다. 이 부분은 GPU를 사용하기 위하여 반드시 필요한 Overhead 이다.

그림 13에서 보면 256×256 영상에 9×9크기의 kernel을 적용했을 때 특징점 추출 처리시간은 순차처리에서 712.63 msec, OpenMP를 적용했을 때 429.07 msec, Global메모리를 이용했을 때 138.79 msec, shared 메모리를 이용했을 때 59.04msec이다. 이때 Global 메모리를 이용했을 때 순수 GPU 처리시간은 98.32msec이고, shared 메모리를 이용했을 때 순수 GPU 처리시간은 26.56msec로써 아래의 식 (8)에 의해 계산하면 각각 약 70msec 정도의 overhead가 필요함을 확인할 수 있다. 식 (16)에서 $O_H$는 overhead를 나타내고

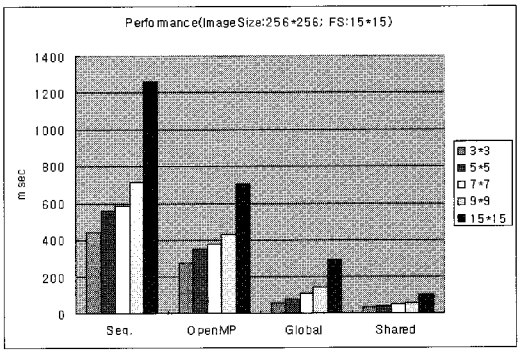



그림 13. Kernel window 크기변화에 따른 실험결과.

Fig. 13. Experimental results with changes in the kernel window sizes.

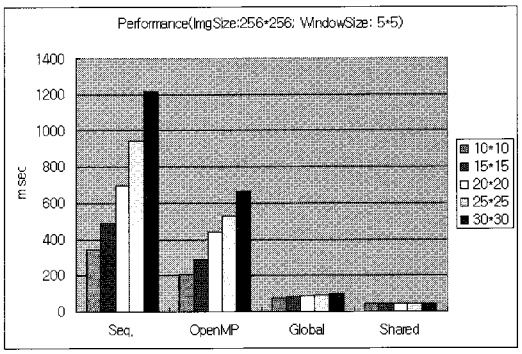
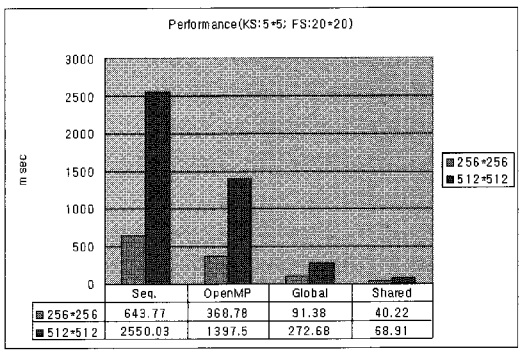



그림 14. 특징점 window 크기변화에 따른 실험결과.

Fig. 14. Experimental results with changes in the feature window sizes.




| | Seq. | OpenMP | Global | Shared |
|---|---|---|---|---|
| 256*256 | 643.77 | 368.78 | 91.38 | 40.22 |
| 512*512 | 2550.03 | 1397.5 | 272.68 | 58.91 |

그림 15. 영상 크기 변화에 따른 실험 결과.

Fig. 15. Experimental results with changes in the image sizes.

$T_{pt}$는 Total processing time을 나타내며, $G_{pt}$는 GPU processing time을 나타내다.

$$O_H = T_{pt} - G_{pt} \tag{16}$$

그림 14는 256×256크기의 영상에서 kernel window 크기를 5×5로 하고, 특징점 window 크기를 각각 10×10, 15×15, 20×20, 25×25, 30×30으로 했을 때의 결과이다. 그림에서 보는 바와 같이 특징점 window 크기가 커짐에 따라 처리시간이 많이 걸리는 것을 확인하였고, GPU상에서 shared 메모리를 이용하였을 때 기타 방식들 보다 가장 좋은 성능을

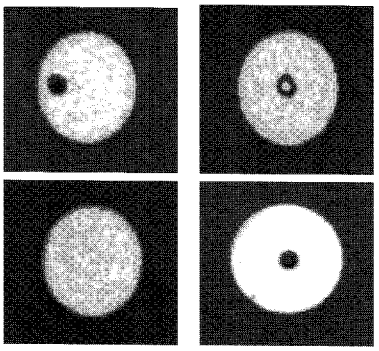


그림 16. 거칠기 영상 샘플.

Fig. 16. Roughness image samples.

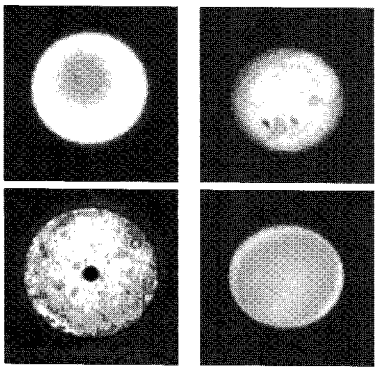


그림 17. 변색 영상 샘플.

Fig. 17. Discolored image samples.

나타냄을 확인할 수 있었다.

그림 15는 kernel window 크기와 특징점 window 크기를 고정하고 영상크기 변화에 따른 실험결과이다. 이 실험에서도 역시 shared 메모리를 이용한 방식이 가장 좋은 성능을 나타냄을 확인하였다.

그림 16과 그림 17은 본 논문에서 분류하고자 하는 거칠기 영상과 변색 영상 중에서 임의로 선택 된 샘플이다.

그림 18~21은 20장의 실험영상에 대하여 식 (8)~(11)을

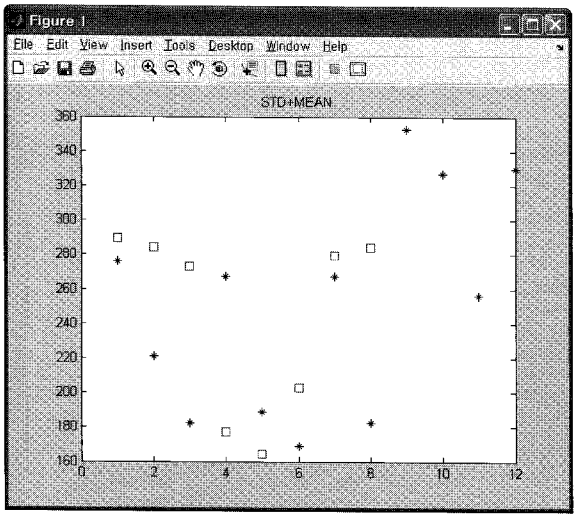



그림 18. 20장의 영상에 대한 $S_{SM}$ 결과.

Fig. 18. Plot of the 20 image $S_{SM}$ results.

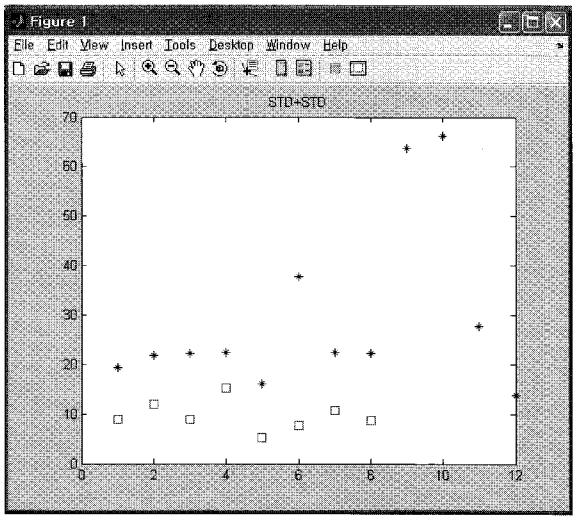



그림 19. 20장의 영상에 대한 $S_{SS}$ 결과.

Fig. 19. Plot of the 20 image $S_{SS}$ results.

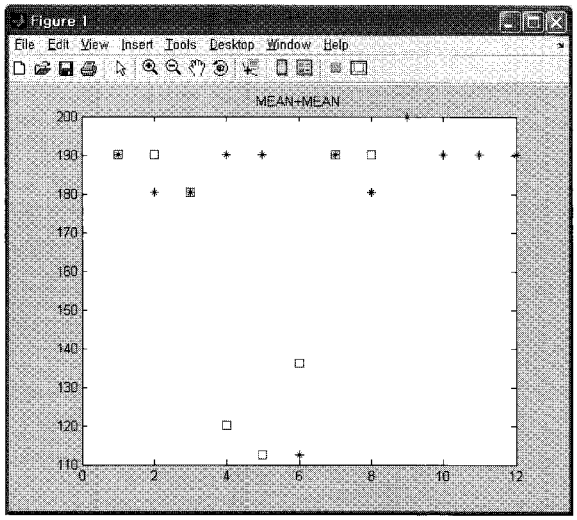



그림 20. 20장의 영상에 대한 $S_{MM}$ 결과.

Fig. 20. Plot of the 20 image $S_{MM}$ results.

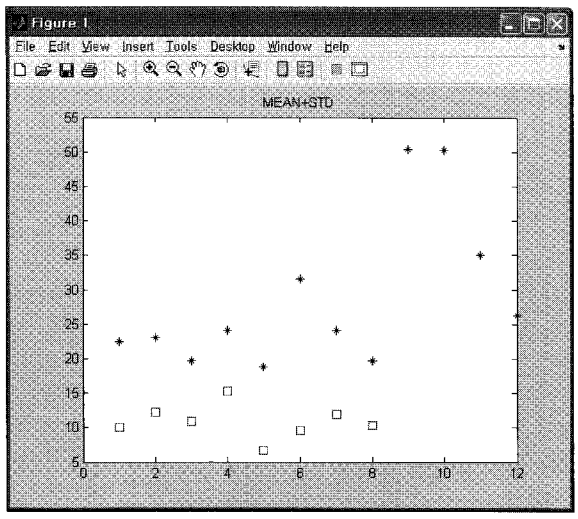



그림 21. 20장의 영상에 대한 $S_{MS}$ 결과.

Fig. 21. Plot of the 20 image $S_{MS}$ results.

표 2. 변색, 거칠기 분류 결과.
Table 2. Discolor and roughness classification results.

| | 과검율 | 미검율 | 전체 에러율 |
|---|---|---|---|
| $T_{SS}$ = 12 | 8.3% | 0% | 5% |
| $T_{MS}$ = 17 | 0% | 0% | 0% |

각각 적용했을 때의 결과를 MATLAB을 이용하여 plot한 결과이다. 여기서 별표(*)는 변색 영상에 대한 결과를 나타내고, 사각형(□)은 거칠기 영상에 대한 결과를 나타낸다. 그림에서 알 수 있다시피 식 (9)과 식 (11)이 비교적 좋은 성능을 내고 있다.

표 2는 식 (9)와 식 (11)을 이용했을 때의 결과를 분석한 표이다. 표에서 과검율, 미검율, 전체 에러율은 식 (17)~(19)를 이용하여 계산한다.

$$\text{과검율} = \frac{\text{과검수}}{\text{거칠기 영상 개수}} \tag{17}$$

$$\text{미검율} = \frac{\text{미검수}}{\text{변색 영상 개수}} \tag{18}$$

$$\text{전체 에러율} = \frac{\text{과검수} + \text{미검수}}{\text{전체 영상 개수}} \tag{19}$$

검사 시스템에서 미검이 발생하면 직접 사용자에게 불량품이 납품되기 때문에 미검은 발생하면 안된다. 따라서 본 논문에서는 미검이 발생하지 않는 조건에서 T값을 설정하였다. 표의 결과로부터 $T_{SS}$를 사용하였을 때 미검이 발생하지 않는 조건에서 8.3%의 과검과 5%의 에러율을 보였으며, $T_{MS}$를 사용하면 0% 에러율을 보임을 확인하였다.

## V. 결론

본 논문에서는 GPU를 이용한 고속 픽셀 패턴 기반의 texture 특징점 추출과 금속 패드 변색 분류 알고리즘을 제안하였으며, Gabor texture 특징 벡터 추출 성능을 평가하기 위하여 CPU상에서는 순차처리와 OpenMP를 이용한 병렬처리를 하였고, GPU상에서는 Global 메모리를 이용한 방법과 shared 메모리를 이용한 방법으로 개발하였다. 실험결과 CPU상에서의 순차처리를 기준으로 했을 때 OpenMP를 이용한 방법은 1.5배정도, GPU상에서의 Global 메모리를 이용했을 때 5~12배, shared 메모리를 이용했을 때는 10~40배의 성능향상을 보임을 확인하였다. 이러한 실험결과로 부터 shared 메모리를 이용한 GPU상에서의 처리속도가 가장 좋은 성능을 나타내는 것을 확인하였다.

본 논문에서 제안한 알고리즘에 의해 추출된 특징점의 성능을 검증하기 위하여 간단한 분류기를 설계하여 실험한 결과 20장의 변색, 거칠기 영상을 에러 없이 정확하게 분류함을 확인하였다. 하지만 본 논문에서 사용한 변색, 거칠기 영상의 수가 적어서 향후 본 알고리즘의 성능을 신뢰성 있게 검증하기 위하여 많은 DB를 구축하여 테스트해볼 필요가 있으며, 기타 여러 가지 분류기를 설계하여 그 성능을 비교하는 추가적인 연구가 필요하다.

## 참고문헌


[1] X. Y. Zeng, Y. W. Chen, Z. Nakao, and H. Lu, "Texture representation based on pattern map," *Signal Processing*, vol. 84, pp. 589-599, 2003.

[2] J. Melendez, M. A. Garcia, and D. Puig, "Efficient distance-based per-pixel texture classification with Gabor wavelet filters," *Pattern Anal Applic*, 11 pp. 365-372, 2007.

[3] H. Zhou, R. S. Wang, and C. Wang, "A novel extended local-binary-pattern operator for texture analysis," information Sciences 178 pp. 4314-5325, 2008.

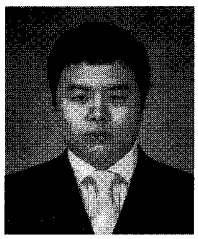
[4] S. E. Grigorescu, N. Petkov, and P. Kruizinga, "Comparison of texture features based on Gabor filters," *IEEE Transactions on Image Processing*, vol. 11, no. 10, Oct. 2002.

[5] C. E. Honeycutt and R. Plotnick, "Image analysis techniques and gray-level co-occurrence matrices for calculating bioturbation indices and characterizing biogenic sedimentary structures," Computer and Geosciences 34 pp. 1461-1472, 2008.

[6] H. Lu, Y. Huang, Y. Chen, and D. yang, "Automatic gender recognition based on pixel-pattern-based texture featre," *Journal of Real-time Image Processing* 3, pp. 109-116, 2008.

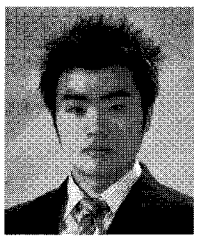
[7] NVIDIA_CUDA_Programming_Guide_2.0, 2008.

[8] H. F. Ng, "Automatic thresholding for defect detection," paterm recognition letters 27 pp. 1644-1649, 2006.

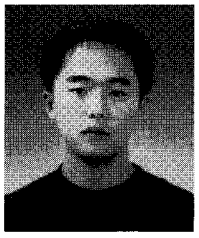

**최 학 남**

2004년 연변대학 응용수학과 졸업. 2007년 상명대학교 컴퓨터과학과 졸업. 2007년~현재 인하대학교 정보공학과 박사과정. 관심분야는 영상처리, 머신비전, 패턴인식, 병렬 영상처리.

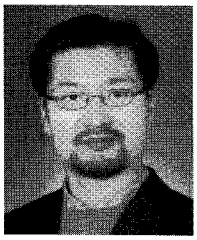

**박 은 수**

2007년 인하대학교 정보통신공학부 학사 졸업. 2007년~현재 인하대학교 정보공학과 석사과정. 관심분야는 컴퓨터 비전, 병렬 영상처리.


**김 준 철**

2008년 인하대학교 정보통신공학부 학사 졸업. 2008년~현재 인하대학교 정보공학과 석사과정. 관심분야는 컴퓨터 비전, 로봇비전, 병렬 영상처리.



**김 학 일**

1983년 서울대학교 제어계측공학과 학사 졸업. 1985년 (미) 퍼듀대학교 전기컴퓨터공학과 석사 졸업. 1990년 (미) 퍼듀대학교 전기 컴퓨터공학과 박사 졸업. 1990년 9월~현재 인하대학교 공과대학 교수. 2001년 2월~현재 한국생체인식포럼 시험평가분과 위원장. 2002년 1월~현재 한국정보보호학회 생체인증연구회 위원장. 2003년 3월~현재 ISO/IEC JTC1/SC37 (바이오인식), WG5(성능평가) 전문위원. 2005년 4월~현재 ITU-T SG17 Q.8 (Telebiometircs) Rapporteur. 관심분야는 바이오인식, 컴퓨터비젼, 패턴인식 및 병렬 영상처리.