The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 16, No. 5, pp.1-6, Oct. 31, 2016. pISSN 2289-0238, eISSN 2289-0246

http://dx.doi.org/10.7236/JIIBC.2016.16.5.1
JIIBC 2016-5-1

CUDA를 이용한 고속 영상 회전 알고리즘에 관한 연구

A Study on High Speed Image Rotation Algorithm using CUDA

권희철*, 조형진**, 권희용***

Hee-Choul Kwon*, Hyung-Jin Cho, Hee-Yong Kwon***

요 약 영상 회전은 영상 처리나 영상 패턴 인식에서 중요한 전처리 방법 중 하나이다. 영상 회전은 회전 행렬의 곱으로 이루어 진다. 그러나 기존의 방법은 대량의 실수 연산과 삼각 함수 계산을 필요로 하므로 수행 시간이 오래 걸린다. 본 논문에서는 이 같은 두가지 주요 지체 연산과정을 제거한 새로운 고속 영상 회전 알고리즘을 제안한다. 제안된 알고리즘은 단지 2개의 전단 연산을 행하므로 매우 빠르다. 또한 최신 병렬 처리 기술인 CUDA를 적용한다. CUDA는 최근 널리 보급된 GPU를 이용한 대용량 병렬처리 계산 아키텍쳐이다. GPGPU는 그래픽 전용프로세서이므로 화소 단위의 병렬처리에 탁월한 성능을 보인다. 제안된 알고리즘은 기존의 회전 알고리즘과 다양한 크기의 영상에 대해 비교 실험한다. 실험 결과는 제안된 알고리즘이 기존의 방법보다 8배 이상의 매우 우수한 성능을 보인다.

Abstract Image rotation is one of main pre-processing step in image processing or image pattern recognition. It is implemented with rotation matrix multiplication. However it requires lots of floating point arithmetic operations and trigonometric function calculations, so it takes long execution time. We propose a new high speed image rotation algorithm without two major time-consuming operations. It use just 2 shear translation operations, so it is very fast. In addition, we apply a parallel computing technique with CUDA. CUDA is a massively parallel computing architecture using prevailed GPU recently. As GPU is a dedicated graphic processor, it is exellent for parallel processing of pixels. We compare the proposed algorithm with the conventional rotation one with various size images. Experimental results show that the proposed algorithm is superior to the conventional rotation ones.

Key Words : High Speed Rotation Operation, Image Processing, Image Rotation, Transform Matrix

Ⅰ. 서 론

영상 처리나 영상 패턴 인식에서 인식에 앞서 입력 영상을 응용 처리가 용이하게 변화시키는 전처리 단계는 필수적이다. 그림 1과 같이 전처리 단계에서는 입력된 원본 영상에서 회색 조 변환, 기울기 추출 및 영상 회전 변환 과정을 수행한다.[1] 이러한 전처리 단계 중 회전 변환은 가장 많은 시간이 소요된다. 더욱이 고전적인 방식에서 사용되는 행렬의 곱에 의한 회전 변환은 대량의 실수 연산과 삼각 함수 계산을 필요로 하므로 수행 시간이 긴 문제가 발생한다.[8] 따라서 보다 빠르게 영상을 회전 시킬수 있는 방법이 필요하다.

본 논문에서는 주어진 영상에서 대상 이미지 영역을 분리하고, 대상 이미지가 기울어진 크기만큼 고속으로 회전 보정하는 알고리즘을 제시한다. 이 방법은 대량의 실수 연산과 삼각 함수 계산을 제거하고, 이동 변환으로

*정회원, 안양대 컴퓨터공학과
**정회원, 안양대 컴퓨터공학과
***정회원, 안양대 컴퓨터공학과 (교신저자)
접수일자 : 2016년 9월 22일, 수정완료 : 2016년 10월 6일
게재확정일자 : 2016년 10월 7일

Received: 22 September, 2016 / Revised: 6 October, 2016 / Accepted: 7 October, 2016
***Corresponding Author: hykwon@anyang.ac.kr
Dept. of Computer Science & Engineering, Anyang University, Korea

대체하여 계산을 단순화 하여 연산 횟수를 줄여 고속 회전이 가능하게 한다. 또한 고속의 병렬처리를 수행할 수 있는 CUDA를 적용함으로서 속도 개선을 극대화 시키도록 하였다.

그림 1. 영상 전처리 흐름도
Fig. 1. Image preprocessing flow

본 논문에서는 삼각함수를 사용한 고전적인 방식을 Matrix방법으로 호칭하며, 이동변환에 의한 단순한 회전 방식을 Shearing방법으로 호칭한다. Shearing방법 이전에는 Matrix방법에서 속도 개선을 위하여 해당하는 기울기에 대한 삼각함수의 값을 미리 계산해서 Table에 보관하고 해당하는 값을 필요 시 사용하는 방법을 사용하였다.

제안된 알고리즘을 구현은 기존의 회전 알고리즘들과 다양한 크기의 영상에 대해 비교 실험 하였다. 실험 결과는 제안된 알고리즘의 속도가 훨씬 개선되어, 기존 회전 알고리즘에 비해 월등하게 우수함을 확인하였다.

II. 관련 연구

기존의 일반적인 회전 알고리즘은 사인(Sine)과 코사인(Cosine)에 의한 회전 변환 행렬을 이용하여 회전시킨다. 그림 2와 같이 2차원 회전 변환 행렬 공식을 주어진 픽셀의 x, y좌표 값에 적용하고, 그 결과 값을 영상의 새로운 좌표로 하여 이동시킨다[1].

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

그림 2. 점(x,y)의 점(x, 'y')로의 2차원 회전식
Fig. 2. Rotation of the point(x,y) onto point(x, 'y')

그림 3과 같이 임의의 점(x, y)를 기준으로 위 행렬의 곱을 계산하게 되면, P라는 점에서 θ만큼 회전된 P'라는 점(x', y')을 얻을 수 있다.

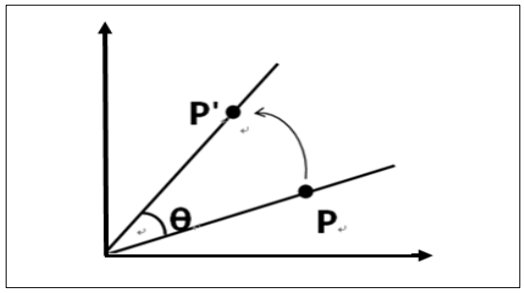

그림 3. 화소 P의 회전
Fig. 3. Rotation of a pixel P

행렬 공식을 이용할 경우 영상 픽셀을 한번 접근하여 회전된 위치로 이동시키는 것이 장점이지만, 회전 공식의 복잡성과 부동소수점 계산 및 삼각함수 계산 등의 단점 때문에 프로그램 상에서 많은 시간이 소요된다. 따라서 실제 응용에서는 기울기 값에 대한 Sin, Cos 값을 보관한 Table을 이용하기도 한다.[9]

본 논문에서는 DDA(Digital Differential Analyzer) 기법에 의한 고속 선 그리기 알고리즘을 영상 회전에 응용하고자 한다[2]. DDA 기법은 선분의 기울기를 y의 증가분을 x의 증가분으로, 혹은 x의 증가분을 y의 증가분으로 나누어 표현한 것이다. 다시 말해, x또는 y는 항상 1씩 증가하고, 그림 4와 같이 기울기 자체가 반대 차원의 증가분이 된다. 따라서 다음 어떤 교차점이 정해지면 다음 교차점은 이전 교차점의 기울기를 더한 값이 된다.

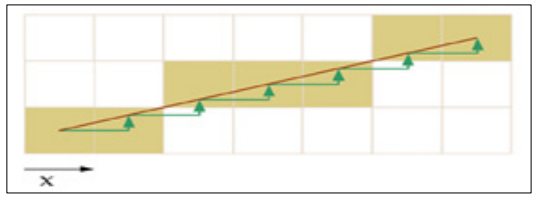

그림 4. DDA에 의한 선분 그리기
Fig. 4. Line drawing by DDA

DDA 알고리즘은 부동소수 덧셈의 정수화 과정에서 뒷자리가 잘려나가는 오류가 누적됨에 따라 실제 선분과 달라질 수 있다.[3] 이러한 단점에도 불구하고 연산횟수가 적고, 빠른 장점이 있어, DDA를 회전 행렬 곱에 적용하는 경우를 보이고 있다.[4]

기존 Matrix방법의 개선 방법으로 3회전 Shearing이 제안되었는데 2회전 Shearing방법에 비해 정밀한 대신 속도가 떨어진다. 3회전 Shearing방법은 2D회전 행렬의 분해(decomposition)에 기초하여 고속 래스터 회전 알고리즘을 세 개의 전단(Shear) 행렬의 곱으로 유도해 냈다. 래스터의 전단(Shearing)은 주사선(scan-line)을 기초로 이루어지며 특히 효율적이다. 유용한 전단 근사화(approximation)는 이웃한 픽셀들의 평균화 이고 이때 섞는 비율은 각각의 주사선에 대해 (상수로)일정하다. 우리 기술은 이들을 모두 합쳐 기존의 방법보다 빨리(aliasing이 없는)래스터들을 회전시킨다.[5]

본 연구에서 제안하는 시스템을 구현하기 위해 CUDA에 관한 지식이 필요하다. CUDA는 NVIDIA사에서 자신들의 그래픽 카드를 범용 적으로 활용할 수 있도록 제공하고 있는 기술이다. CUDA는 GPU의 개념이 나오고 나서 GPGPU등을 거쳐 완성이 되었고 지금도 계속 발전 중에 있다. CUDA (Compute Unified Device Architecture)는 NVIDIA사에서 기존의 GeForce 시리즈를 그래픽 전용처리를 위한 구조에서 유연성을 향상시켜 범용적인 프로그램을 처리할 수 있도록 변경하고 GPU를 이용하여 범용적인 프로그램을 개발할 수 있도록 '프로그램 모델', '프로그램 언어', '컴파일러', '라이브러리', '디버거', '프로파일러'를 제공하는 통합 개발 환경을 말한다.

CUDA를 이용한 프로그램은 높은 연산 처리 능력, 병렬 프로그램의 확장성, 저렴한 가격, 편리한 설치 등의 장점을 가지게 된다. CPU보다 월등한 연산 처리능력을 가지고 있고 또한 다수의 ALU로 인해 대규모 데이터를 멀티스레드로 실행하여 처리가 가능하다. 같은 처리능력의 시스템을 구축하는데 CPU를 증가시키는 것 보다 훨씬 저렴하게 구축할 수 있다.[6]

Ⅲ. CUDA 고속회전 알고리즘 설계

대상 이미지를 기계를 통해서 스캔하면 원본 영상은 기울어지게 되는데, 그림 5와 같이 기울기는 가로, 세로 모두 1/4, 1/2, 3/4 지점에서 탐침하여 탐침에 성공한 두 점을 가지고 구한다.

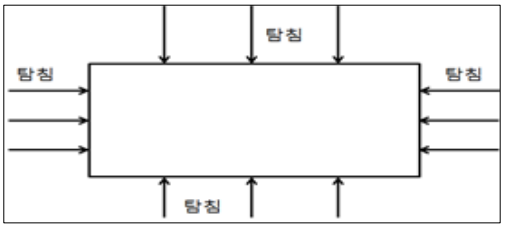

그림 5. 탐침 방법
Fig. 5. Probing method

그림 6과 같이 가로 증분과 세로 증분은 A', B', A, B 점의 좌표 값으로 구할 수 있다. 여기서 WX는 식(1)과 같이, HY는 식(2)와 같이 구할 수 있다.

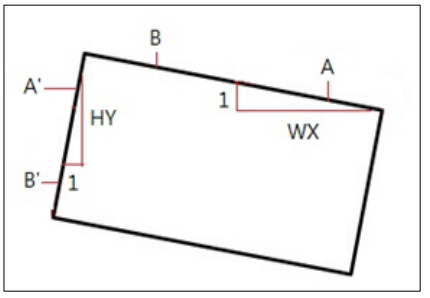

그림 6. 단위 픽셀 당 X축, Y축의 증분
Fig. 6. Horizontal and vertical ncrements per one pixel

$$WX = (A.x - B.x) / (A.y - B.y) \quad (1)$$

$$HY = (A'.x - B'.x) / (A'.y - B'.y) \quad (2)$$

위에서 구한 장축방향 기울기(가로 증분)를 WX, 단축방향 기울기(세로 증분)를 HY라고 정의한다. 그러면 WX, HY를 이용한 영상의 회전 변환 절차는 다음과 같다.

WX는 하나의 Y Pixel 당 변화한 X Pixel 개수를 의미하며, HY 는 하나의 X Pixel 당 변화한 Y Pixel 개수를 의미한다. 입력 영상에서 한 줄의 가로 Pixel을 처리하기 위해 동일한 줄에 대한 가로 증분 값을 구하고, 각각의 Pixel에 대한 세로 증분 값을 구하여 이동 변환하는 방법으로 결과 영상을 만들어낸다.

기울어진 영상은 그림 7과 같이 2가지 경우가 있다. ①과 ②는 회전시키는 방향이 다르므로 어느 방향으로 기울어 졌는지를 알아야 하는데 그 것은 WX와 HY를 통해

서 알 수 있다. ①의 경우에는 WX가 양수이고 HY가 음수이다. ②의 경우에는 WX가 음수이고 HY가 양수이다. 이 결과 값을 통해서 어느 방향으로 영상이 기울어 졌는지를 알 수 있다. 기울어진 방향에 따라 영상을 회전 변환하는 방향이 달라진다.

그림 7. 2가지 회전 영상
Fig. 7. Two cases of a slanted image

WX를 통해서 Pixel 데이터를 수정할 때 ①의 경우에는 그림 8과 같이 왼쪽에서 오른쪽으로 진행하면서 Pixel 데이터를 위 방향으로 이동하고, ②의 경우에는 ①과 반대로 왼쪽에서 오른쪽으로 진행하면서 Pixel 데이터를 아래 방향으로 이동한다. 예를 들어 WX = 2일 경우 X축으로 2만큼 이동할 때마다 Y축으로 1만큼 아래에 있는 Pixel 데이터를 가져온다. 여기서 주의할 점은 이 방법으로 아래에 있는 Pixel데이터를 가져오다 보면 영상의 범위를 넘어간 곳에서 데이터를 가져오게 되는데 그 경우에는 Pixel 데이터는 0 (여기서 0은 배경을 의미)으로 채워준다.

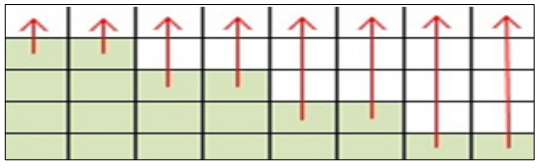

그림 8. X증분(WX)을 이용한 Pixel 이동
Fig. 8. Pixel translation with X increments(WX)

마찬가지로 HY를 통해서 Pixel 데이터를 수정할 때 ①의 진행 방향은 그림 9와 같이 아래에서 위로 진행하며 좌측 방향으로 수정하고, ②의 경우에는 ①과 반대로 아래에서 위로 진행하며 우측 방향으로 수정한다.

예를 들어 HY = 2일 경우 Y축으로 2만큼 이동할 때마다 X 축으로 1만큼 오른쪽에 있는 Pixel 데이터를 가져온다. 이때 주의할 점은 이 방법으로 오른쪽에 있는 Pixel 데이터를 가져오다 보면 영상의 범위를 넘어간 곳에서 데이터를 가져오게 되는데 그 경우에는 Pixel 데이터는 0 (여기서 0은 배경을 의미)으로 채워준다.[7]

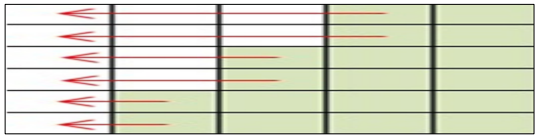

그림 9. Y증분(HY)을 이용한 Pixel 이동
Fig. 9. Pixel translation with Y increments(HY)

이상의 Shear연산을 이용한 회전 알고리즘을 CUDA로 구현한 결과는 그림 10과 같다.

```
// blockDim  : Block이 가지고 있는 쓰레드의 수
// blockIdx  : Block 번호
// threadIdx : Block안에 있는 쓰레드의 번호
int i = pThread.blockIdx.x;
int j = pThread.threadIdx.x;
int newlocation, newlocation3;
int jRi3, newx, newy;

newx = (int)(i - (pHeight - j) / pWX);
newy = (int)(j - i / pWX);
newlocation = (newy * pWidth + newx);
newlocation3 = 3 * newlocation;
jRi3 = (j * pWidth + i) * 3;

if (영상 범위내 인 경우)
{
 pOutData[newlocation3 + 0] = pInData[jRi3 + 0];
 pOutData[newlocation3 + 1] = pInData[jRi3 + 1];
 pOutData[newlocation3 + 2] = pInData[jRi3 + 2];
}
```

그림 10. CUDA-Shear 처리방법을 이용한 회전알고리즘
Fig. 10. Rotation algorithm using CUDA-Shear

WX와 HY를 이용하여 Pixel을 수정한 전·후 결과 영상은 Matrix Table처리 방법은 그림 11과 같으며, CUDA-Shear 처리 방법은 그림 12와 같다. 회전 보정을 하고 나면 똑바른 이미지(지폐) 그림 11, 그림 12 ②의 모양을 얻을 수 있다. 두 방법 모두 동일하게 결과가 나타나고 있다.

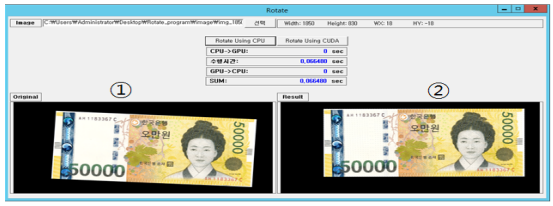

그림 11. Matrix-Table방법에 의한 회전 결과
Fig. 11. Rotation result using Matrix-Table

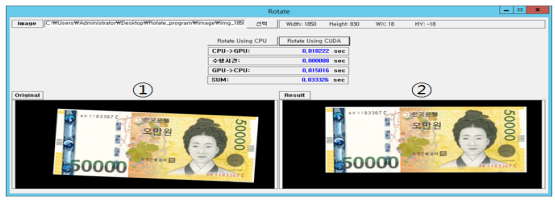

그림 12. CUDA와 Shear연산에 의한 회전 결과
Fig. 12. Rotation result using CUDA and Shear operation

Ⅳ. 실험 및 결과

제안된 알고리즘은 기존의 회전 알고리즘과 다양한 크기의 영상에 대해 비교 실험을 하고 성능을 비교하였다. 실험환경은 표 1과 같으며, 실험대상 이미지는 표 2와 같다.

표 1. 실험 환경
Table 1. Experimental environment

구분	사양
CPU	Intel 쿼드코어 Q9400 (2.66GHz)
RAM	DDR3 8GB(1333MHz)
Graphic Card	NVIDIA GeForce GTX560
OS	Microsoft Window서버 2012
개발언어	C#

표 2. 실험 대상 이미지들의 크기
Table 2. Size of experimental Images

이미지 사이즈	이미지 픽셀
300 * 170 (A)	51,000
950 * 540 (B)	513,000
1270 * 660 (C)	838,200
1850 * 820 (D)	1,517,000

기존의 회전 알고리즘의 경우 하나의 픽셀을 회전시키기 위한 연산의 종류 및 횟수가 비교 연산 1번, 곱셈 4번, 덧셈 4번을 하게 된다. 그리고 코사인과 사인의 값을 구하기 위한 연산은 부동소수점 계산을 하게 되기 때문에 실제 연산은 상대적으로 긴 시간을 필요로 한다. 따라서 m * n 영상의 경우 회전에 필요한 총 연산의 횟수는 1* m * n 번의 비교 연산, 4 * m * n 번의 부동소수점 곱셈과 덧셈, 그리고 각 곱셈마다 부가적인 SIN, COS 함수 계산이 필요하다. 이를 개선하기 위해 SIN, COS Table을 이용할 경우 총 연산 시간에서 SIN, COS 함수 계산 부분을 단축할 수 있다. 그러나 본 논문에서 제안하는 방식을 이용하여 회전 할 경우 m(height) + 2 * m * n번의 비교 연산과 2번의 덧셈 연산을 필요로 한다.

실험은 각각 10회씩을 수행하여 평균 값으로 계산하였으며, 위 4가지 방법과 CUDA를 적용한 실험한 결과 표 3과 같은 성능을 확인할 수 있었다. 제일 고전적인 Matrix 방법에 비해 본 논문에서 제안한 Shear방법을 CUDA에 적용한 방법이 8배 이상의 처리 속도 개선을 나타내고 있다. 표4에서는 CUDA방법의 수행시간 중 실제 연산시간 보다는 메모리 이동시간이 절대적으로 많이 소요되는 것을 알 수 있다. (20배 이상)

표 3. 회전 성능 비교
Table 3. Rotation performance comparison
(단위 mS)

구분	A	B	C	D
Matrix 방법	9.55	89.77	148.28	279.02
Matrix 방법(Table처리)	3.19	22.93	39.57	75.14
Shear 방식	2.51	26.44	42.65	73.25
Shear 방법(Table처리)	2.23	22.35	36.49	64.05
CUDA방법	1.38	10.51	17.00	32.71
성능비교(Mtrix : CUDA)	**6.9배**	**8.5배**	**8.7배**	**8.5배**

표 4. CUDA알고리즘의 세부 소요 시간
Table 4. Detailed time of CUDA algorithm
(단위 mS)

구분	A	B	C	D
총 소요 시간	**1.386**	**10.514**	**17.003**	**32.715**
CPU->GPU(메모리이동시간)	0.637	5.811	9.668	17.792
CUDA연산시간(GPU)	0.059	0.086	0.087	0.087
GPU->CPU(메모리이동시간)	0.690	4.617	7.248	14.836

Ⅴ. 결 론

본 논문에서는 최근 널리 보급되고 있는 GPGPU를 이용한 대용량 병렬처리 계산 아키텍처인 CUDA를 이용한 초고속 영상 회전 알고리즘을 제안 하였다. 기존의 회전 행렬을 이용한 방식의 단점인 삼각함수와 부동소숫점 계산은 이동 변환 Shear Operation으로 해결하였다.

제안된 알고리즘은 기울어진 지폐 영상에 대해 회전 변환을 수행할 때 시간 복잡도(Time Complexity)를 연산 횟수 및 연산 방식 측면에서 크게 낮추어 전체적인 속도 개선을 달성하였다. 부동 소수점 연산의 정수화에 따른 약간의 정밀도 하락이 있지만, 원본 영상과 회전된 영상의 데이터가 손실된 정도를 비교 확인한 결과 의미 있는 손실은 없는 것으로 확인하였다. 최종 속도 비교 결과는 평균 8배 이상의 향상을 보였다.

향후 연구과제로는 첫째, CUDA에서 GPU 계산속도에 비해 상대적으로 많은 시간이 소요되는 메모리 이동 시간에 대한 개선 연구가 필요하다. 둘째, 비록 의미 없는 손실일지라도 정밀도를 요구하는 영상의 처리를 위하여 손실된 데이터에 대한 보정 작업에 대한 연구가 필요하다.

References

[1] E. Angel, “Interactive Computer Graphics”, Pearson International Edition, 2009.

[2] M. S. Park and 1 Person, "Datawise Discriminant Analysis For Feature Extraction", Journal of Korean institute of intelligent systems, v.19 no.1, 2009.

[3] S. W. Shin, and 3 Person, "Image Enhancement with Rotating Kernel Transformation Filter Generated by Bresenham's Algorithm", The Korean Institute of Electrical Engineers, Volume 61, Issue 6, 2012.

[4] Leonid Morozand and 3 Person, “A Comparison of Standard One-Step DDA Circular Interpolators with a New Cheap Two-Step Algorithm”, Hindawi Publishing Corporation, Modelling and Simulation in Engineering, Volume, 2014.

[5] Alan W. Paeth, “A Fast Algorithm for General Raster Rotation”, Graphics Interface ‘86

[6] Jason Sanders and 1 Person, “CUDA BY EXAMPLE An Introduction to General- Purpose GPU Programing”, ADDISON- WESLEY, 2011.

[7] Hyungjin Cho and 3 Person, “High Speed Rotation Algorithm for the Banknote Classification”, Proceedings of the Autumn Conference Korean Mutimedia Society, VOL. 17 No. 2, 2014.

[8] Myung Jae Lim and 2 Person, “Rehabilitation System through Image Analysis Method”, The Institute of Internet, Broadcasting and Communication (IIBC), VOL. 10 NO. 6, 2010.

[9] Byung Jik Son and 3 Person, “Displacement Measurement of Cable Stayed Bridge using Digital Image Processing”, Journal of the Korea Academia-Industrial cooperation Society, VOL. 17 NO. 5, 2016.

저자 소개

권 희 철(정회원)

- 1982년 8월 : 숭실대 기계공학과(학사)
- 2013년 2월 : 안양대 대학원 컴퓨터공학 (공학석사)
- 2016년 3월 : 안양대 대학원 컴퓨터공학 (공학박사 수료)
- 1985년 4월 ~ 1992년 4월 : 동부제철(주)
- 1992년 4월 ~ 1998년 8월 : 동부CNI(주)
- 1999년 12월 ~ 현재 : ㈜넥스트소프트 대표이사
- E-Mail : heeckwon@empal.com

<주관심분야 : 영상처리, 토목계측모니터링, 웹공학>

조 형 진(정회원)

- 2014년 2월 : 안양대학교 컴퓨터공학과(학사)
- 2016년 2월 : 안양대 대학원 컴퓨터공학 (공학석사)
- 2016년 2월 ~ 현재 : (주) MVTech 기술연구소 연구원
- E-Mail : cho9038@naver.com

<주관심분야 : 패턴인식, 영상처리>

권 희 용(정회원)

- 1983년 2월 : 서울대학교 전자계산기공학과(학사)
- 1986년 2월 : 서울대학교 전자계산기공학과(공학석사)
- 1993년 2월 : 서울대학교 전자계산기공학과(공학박사)
- 1986년 ~ 1995년 : 한국통신 연구개발단 선임연구원
- 1995년 ~ 현재 : 안양대학교 컴퓨터공학과 교수
- E-Mail : hykwon@anyang.ac.kr

<주관심분야 : 패턴인식, 영상처리, 병렬처리응용>