CUDA 기반의 병렬 CAMshift 알고리즘을 이용한 객체추적 시스템

조지훈*, 이상구**

요약

본 논문에서는 PTZ(Pan-Tilt-Zoom) 카메라의 영상 객체 추적을 위한 강인한 실시간 시스템을 구현한다. 영상 객체 추적을 위하여 탐지된 이동 객체의 HSV 컬러 영상 분포에 기반을 둔 CAMshift 추적 알고리즘을 사용한다. CAMshift 알고리즘은 부동소수점 데이터의 연산량이 많기 때문에 CPU에 따라 끊김현상, 데이터 병목현상이 일어날 수 있다. 따라서 본 시스템은 이러한 단점을 극복하기 위해 CUDA 기반의 GPU(Graphic Processing Unit)를 사용하여 CAMshift 알고리즘을 병렬화 하였다. 그 결과 기존의 단일 CPU에서 실행시킨 프로그램보다 데이터 처리속도가 약 12~14배 가량의 속도 향상이 측정되었다. 본 시스템은 보다 큰 범위에서 연속적으로 움직이는 객체 추적을 위한 효율적이고 빠른 영상 감시 시스템에 활용될 수 있다.

An Object Tracking System using CUDA Based Parallel CAMshift Algorithm

Ji-Hoon Jo*, Sang-Gu Lee**

ABSTRACT

In this paper, we implement a real-time robust image object tracking system for PTZ(Pan-Tilt-Zoom) cameras. For image object tracking, we use the CAMshift algorithm based on the HSV color image distibution of detected moving objects. As CAMshift algorithm requires much floating-poing computations, image flickering and data bottleneck are sometimes occurred according to the CPUs. In order to overcome the drawbacks, in the proposed system, we parallelize CAMshift algorithm by using GPU(Graphic Processing Unit) in the CUDA enviorment. The experimental result shows that the speedup factor has 12~14 times faster as compared as CPU method. This system can be applied to an effective and faster image surveillance system for continuous object tracking in a wider area.

Key Words : CUDA, Parallel processing, GPGPU, Object Tracking, CAMshift

* 한남대학교 컴퓨터공학과 (✉invers83@naver.com)
** 한남대학교 컴퓨터공학과
· 제1저자(First Author) : 조지훈 · 교신저자(Correspondent Author) : 이상구
· 접수일(2013년 2월 18일), 수정일(1차 : 2013년 3월 13일), 게재확정일(2013년 3월 26일)

Ⅰ. 서 론

2000년대에 들어서면서 스마트 카메라, 그 중에서도 CCTV의 분야는 대학, 연구그룹, 비디오 감시 시스템을 개발하는 기업 등에서 관심을 갖고 연구를 진행해 왔다. CCTV의 경우 현대 사회에 이르러 강력범죄 및 보안감시에 아주 큰 요소로 활용되고 있으며, 이는 CCTV가 일반 카메라와 같이 단지 영상을 획득, 저장 만이 아닌 시스템 내에서 스스로 영상을 분석하고 패턴을 인식하는 일까지 가능해졌기 때문이다.[1]

최근 카메라 및 디바이스의 발전으로 고성능, HD급의 화질을 갖춘 카메라들이 출시되고 있고 그 중에서도 PTZ 카메라는 360도 회전 및 줌 인-아웃(in-out)이 가능하기 때문에 기존의 고정형 CCTV가 갖는 단점인 단방향 감시을 보완하여 카메라 주변의 모든 영역에 대해서 모니터링이 가능하다는 장점이 있다. 시각 장치를 통해 들어온 방대한 정보를 분류, 이해, 판단하기 위해서는 엄청난 양의 컴퓨팅 파워를 처리할 수 있어야 하며, 실시간 영상의 경우 340X240 영상의 경우라 하더라도 끊김이 없는 영상을 제공하기 위해서는 최소 30 프레임 이상을 제공하여야 한다. 하지만, 이러한 계산량은 단일 CPU가 감당하기에는 너무나도 부족하다.

기존의 하드웨어가 발전됨에 따라 고성능 프로세서를 탑재한 CPU들이 나오고 있지만 실시간으로 영상을 처리하는 데에는 한계가 있을 수 밖에 없다. 특히나 사람들의 왕래가 잦은 백화점이나 고속도로, 번화가의 경우 판별해야 할 객체의 수가 많아지기 때문에 CPU만으로 계산하기에는 한계가 있다[2]. 이러한 단점을 보완하기 위한 방법으로 병렬처리 기법이 대안으로 손꼽히고 있다.

2007년 NVIDIA에서는 자사의 그래픽 카드를 GPGPU로 활용할 수 있도록 CUDA(Compute Unified Device Architecture)를 발표하였다. CUDA는 NVIDIA 그래픽 카드를 장착하면 누구든지 쉽게 활용할 수 있다는 장점이 있으며, 이러한 장점을 바탕으로 다양한 연구 분야에서 활용되고 있다.

2012년 현재, CUDA의 경우 4.2버젼의 툴킷을 무료로 제공하고 있으며, NVIDIA Geforce를 사용하는 프로그래머라면 OS에 상관없이 누구든지 쉽고 빠르게 CUDA를 사용하여 병렬프로그래밍을 작성할 수 있다.

CUDA의 경우 기본적으로 C기반의 언어로 작성되어 있으나 4.2버전에서는 범용성을 높이기 위해 C/C++, OPENCL, DirectCompute, Fortran, MATLAB 등 여러 언어를 지원하고 있으며, 디버그 툴킷인 CUDA Visual profiler를 통해서 병렬 프로그래밍의 흐름을 보다 직관적으로 사용자에게 제공하고 있다.[3]

본 논문에서는 CUDA 기반의 GPU를 사용하여 CAMshift 알고리즘을 병렬화 하고 PTZ 카메라의 프로토콜과 RS-485 통신을 이용하여 카메라를 제어하는 방법으로 화면상의 객체의 위치를 화면 가운데에 위치하게 하는 실시간 추적 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 CAMshift의 개요, 3장에서는 GPU를 통한 병렬화 과정, 4장에서는 실험 결과 및 분석에 대해 알아보며, 5장에서는 결론을 다룬다.

Ⅱ. 객체 추적 방법

일반적으로 객체 추적 방법의 경우 입력되는 영상의 정보를 분석하여 이상 행위 유무를 감지하는 기술로써 배경 분리, 객체 식별, 객체 추적 단계로 나뉘게 된다.

배경 분리의 경우 카메라에서 입력되는 영상 프

레임에서 ROI(관심영역)과 그 외의 영역을 분리, 구분하여 객체를 탐지한다. 주로 현재 프레임과 이전 프레임간의 컬러 픽셀의 변화를 계산하여 분리하는 방법이 많이 쓰이고 있다. 최근에는 배경 영역에 대한 특성 정보를 가우시안 모델을 통해 정교하게 모델링 한 후 배경영역과 ROI를 구분하는 방법들이 많이 사용되고 있다. 객체 식별은 ROI내의 탐지된 객체가 탐지하고자 하는 객체인지 아닌지 판단 여부를 구분하는 과정이며, 객체 추적은 연속되는 프레임에서 식별된 객체의 이동 경로를 찾는 과정으로 칼만 필터, particle filter, Mean-shift 등의 다양한 알고리즘을 사용하여 추적하거나 혹은 알고리즘을 조합하여 추적하는 방법을 사용하고 있다.

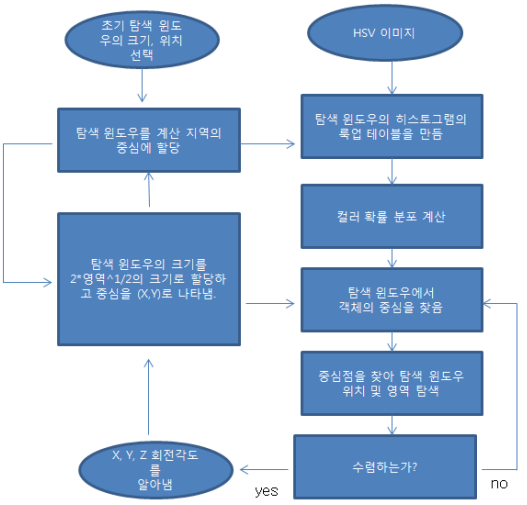


그림 1. CAMshift 알고리즘 순서도
Fig. 1 CAMshift algorithm flow chart

2.1 CAMshift 알고리즘

기존의 객체를 추적하는데 많이 쓰인 알고리즘 중 하나인 Mean-shift 알고리즘의 경우 추적 객체의 회전각도, 윈도우의 크기 등에 민감하다는 단점이 존재하였다.

본 논문에서는 객체를 추적하는 방법에 있어서 색 정보를 바탕으로 추적하는 알고리즘인 CAM shift 알고리즘을 사용하였으며 보다 정확한 객체 검출을 위해 HSV 모델중 Hue 값만을 사용하여 객체를 연속적으로 추적할 수 있게 하였다. CAMshift 알고리즘은 기본적으로 Mean-shift 알고리즘을 사용하면서도 객체의 크기 및 각도를 같이 계산한다. 본 논문에서 사용하는 PTZ 카메라로 추적하기 위해서는 객체의 크기에 영향받지 않는 CAMshift 알고리즘이 적합한 알고리즘이라고 할 수 있다.

CAMshift 알고리즘은 다음과 같은 단계로 나뉘어 수행된다.[4][5][6]

앞의 CAMshift 트래킹 알고리즘을 프로시저 형태로 표현하면 다음과 같다.

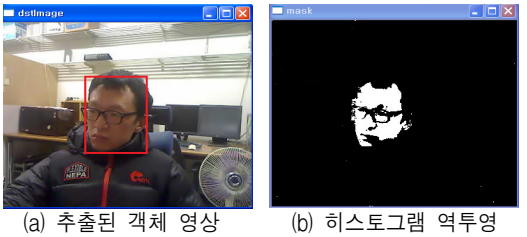
```
CAMshift_tracking( );
kernel = kernel_initialize( );
for  image_frame = 1: n
features = object_features(image_frame);
 repeat
  [M00 M01 M10]  = moments(kernel, features);
  Cw = centroid(kernel);
  Cg = [M01/M00,M10/M00];
  window_shift(kernel, Cg - Cw);
  kernel_size(kernel, width(M00),height(M00));
 until  |Cg - Cw| <= e;
end
```

여기서 M_{00}는 0차 모멘트 이고, M_{01}과 M_{10}은 각각 1차 모멘트(X)와 1차 모멘트(Y)로써 Cw를 통해서 무게 중심을 구한후 ROI의 위치를 이동, 크기 등을 재조절하여 중심 위치를 결정한다.

CAMshift 알고리즘은 객체의 컬러 정보, 즉, 컬러 확률 분포를 이용해 위치뿐만 아니라 회전 각도, 크기까지 빠르게 계산해서 추적할 수 있다.

객체의 추출은 기본적으로 배경영상과 입력영상의 차를 구함으로써 이루어진다. 차 영상을 통하여 객체가 추출이 되면 median 필터링 및 morphology를 통하여 프레임 내의 Salt and pepper 노이즈를 제거, 클러스터링을 통하여 객체를 라벨링한 후 추출한다. 추출된 객체는 다음 프레임에서의 객체간 유사도 검사를 통해 추적 객체로 선정한다. 객체간 유사도 검사는 최소 편차 값을 통해 구하도록 한다. 이때 검출된 객체의 크기가 추적 객체의 크기와 다르다면 크기 조정을 수행 후, 유사도 검사를 수행한다.

<그림 2> 는 CAMshift를 사용하여 실제 객체를 추적하는 것을 보여준다. (a)는 원 영상을, (b)는 색상(Hue), 채도(Saturation)의 히스토그램 유사도 역투영(Backprojection)을 나타낸다.



(a) 추출된 객체 영상 (b) 히스토그램 역투영

그림 2. CAMshift 추적
Fig. 2 CAMshift tracking

CAMshift 알고리즘의 경우 기존 Meanshift 알고리즘이 갖는 단점인 크기, 각도에 취약하다는 단점을 보완하여 추적하고자 하는 객체의 크기가 변하거나 각도가 달라진다 하더라도 연속적인 추적이 가능하다.

III. CAMshift 알고리즘의 병렬화

본 논문에서는 2장에서 설명한 CAMshift 알고리즘의 단점을 보완하기 위해 GPU를 응용한 CUDA 환경에서 병렬화 처리를 수행한다.

CAMshift의 병렬화 과정은 다음과 같다.

① CAMshift 알고리즘을 병렬화할때에 첫 번째 작업은 RGB-HSV 연산을 병렬화 및 모폴로지 연산과 같은 이미지 마스크를 병렬화 하는 것이다. HSV 변환의 경우 CAMshift 알고리즘이 객체를 쉽게 인식, 추적하기 위해서 컬러를 변환하는 것으로 각각의 픽셀은 항상 동일한 연산을 하기 때문에 병렬화 하는데 아주 좋은 연산이라 할 수 있다.

RGB-HSV의 변환은 각 픽셀의 주위 픽셀에 영향을 받지 않는 독립성이 존재하기 때문에 각 픽셀값을 CUDA 쓰레드에 전송, 다음과 같은 식을 계산 후, 결과 값을 리턴한다.

$$V = \frac{1}{3}(R+G+B)$$

$$S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)]$$

$$H = \cos^{-1}\left[\frac{\frac{1}{2}[(R-G)+(R-B)}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right]$$

이미지 마스크 경우, RGB-HSV 변환과는 달리 주위 픽셀의 값을 참조해야 하기 때문에 공유 메모리를 사용, 블록 복사를 통해서 이미지 마스크를 처리한다.[7]

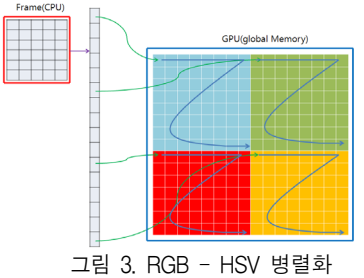


그림 3. RGB - HSV 병렬화
Fig. 3 RGB - HSV conversion parallelization

② ROI 내의 픽셀들에 대해서 Hue값만을 추출한후, ROI 내의 픽셀들의 Hue값들중에서 사용자가 지정한 Threshold값을 가지고 온다.(본 프로그램의 경우 0.8~1사이의 값만 가지고 옴)

③ ROI의 무게중심을 구하기 위해 M_{00}, M_{01}, M_{10}를 계산한다. M_{00}, M_{01}, M_{10}의 경우 ROI내의 픽셀들의 총 합이기 때문에 CPU가 아닌 CUDA에서 병렬로 처리하는 것이 속도적인 측면에서 큰 이득을 얻을 수 있다.

<그림 4>와 같이 ROI 내에서의 가중치를 구하는 연산은 총 3번에 걸쳐 계산된다. 먼저 M_{00}의 경우 ROI 내의 모든 픽셀의 값, M_{01}는 Y에 대한 가중치를 구하며, 마지막으로 M_{10}의 경우 X에 대한 가중치를 구하게 된다. 이와 같은 픽셀내의 모든 합을 구하기 위해서는 각 픽셀의 값을 블록 내의 쓰레드로 분배, 한 블록내의 쓰레드 값을 모두 더하여 CPU로 재전송, 블록들의 값을 다시 더하여 하나의 값으로 반환하며, 입력한 값의 개수와 상관없이 항상 하나의 출력데이터만을 생성한다. 하나의 입력 배열을 취하는 일반적인 절차와 그보다 더 작은 배열을 결과로 산출하는 계산을 Reduction이라 하며, <그림 4> 과 <그림 5>는 M_{01}의 Y에 대한 가중치를 구하는 그림과 병렬화를 위해 Reduction을 나타낸 것이다.

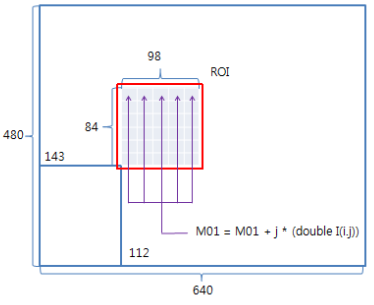


그림 4. M_{01} 가중치 계산

Fig. 4 M_{01} moments

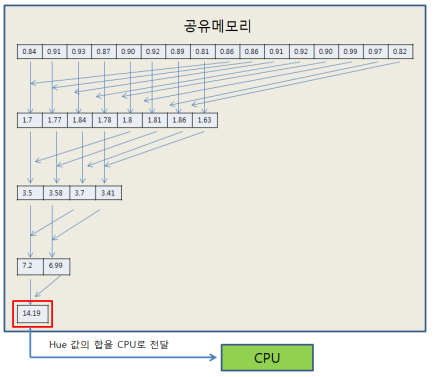


그림 5. Reduction 계산

Fig. 5 Reduction calulations

④ 가중치 값을 구한 후, CPU에 값을 전달하여 ROI의 무게 중심을 구한 후, 수렴 여부를 판단한다.

Ⅳ. 실험 결과 및 분석

4.1 제안시스템 개요

제안하는 CUDA 환경에서의 병렬 CAMshift 알고리즘을 이용한 움직이는 객체 추적 시스템을 실험하기 위한 환경은 <표 1>와 같다.

표 1. PC 개발환경

Table 1. PC developmet environment

	개발환경
H/W	CPU : Intel(R)Core(TM)2 2.0GHZ Memory : 2GB RAM, Graphic card : GTX Geforce 560
OS	MS Windows XP Professinal SP3
실험도구	MS Visual Studio 2008 CUDA Tool kit 4.2 CUDA Visual Profiler v1.1
실험카메라	SAMSUNG techwin SPD-1000

본 논문에서의 제안된 시스템의 개요도는 <그림 6>와 같다.

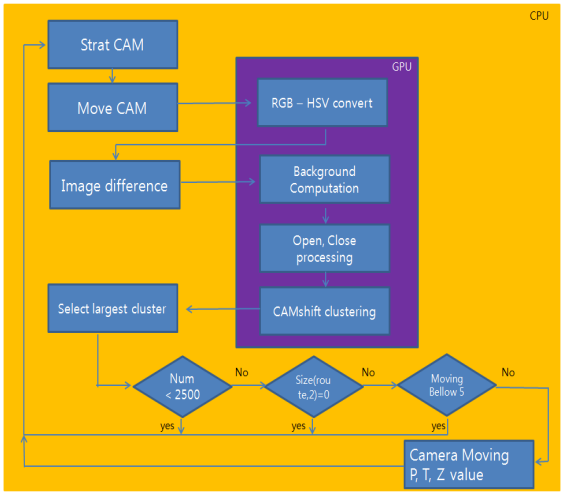


그림 6. 전체 시스템의 개요도
Fig. 6 System flow chart

<그림 6>와 같이 먼저 카메라가 객체를 인식하게 되면 배경을 제거 직전 프레임과의 비교, 데이터를 GPU에게 넘겨준다. GPU는 배경제거와 잡음을 없에는 모폴로지연산, CAMshift 연산을 수행한 후, 클러스터 데이터를 CPU에게 다시 돌려준다. 클러스터 추출 과정에서 클러스터의 크기는 2500을 반드시 넘어야 하며, 객체가 추출이 되면 객체의 움직임에 따라서 미리 지정된 Lookup Table을 통해 PTZ 값을 RS-485통신을 통해 전달한 후, PTZ 카메라는 수신된 PTZ값을 통해서 움직이게 된다.

본 논문의 실험에서는 삼성테크윈사 SPD - 1000 PTZ 돔 카메라를 사용하였다. 해상도는 640X480 동영상을 전송 받는다. 제안 알고리즘의 성능 입증을 위해서 실험대상은 여러 실험 객체 대상을 바꾸어 실험하였다. <그림 7>은 PTZ 카메라 상에서의 객체 추적 실험 영상이다.

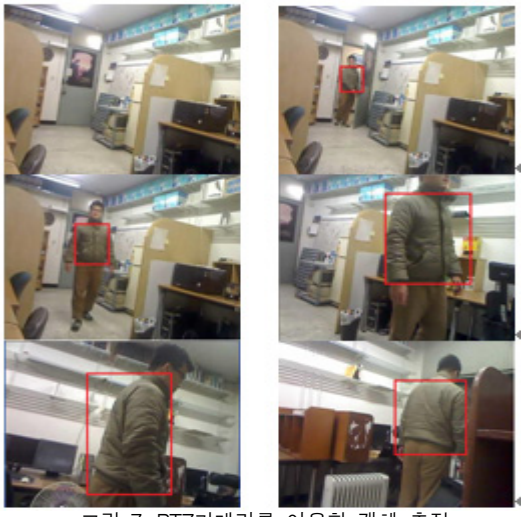
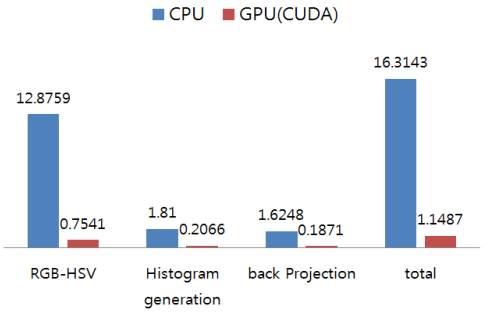


그림 7. PTZ카메라를 이용한 객체 추적
Fig. 7 Object tracking using PTZ camera

4.2 실험 결과

다음의 <표 2>는 CAMshift 알고리즘을 CPU (Intel(R)Core(TM)2 2.0GH)에서 실행했을때와 GPU(Nvidia Geforce 560)을 통해 실행했을 때의 성능 시간을 측정한 것이다. 이미지의 해상도는 640X480 크기이며, 초당 30프레임씩 처리하도록 설계하였으며, 각각의 프레임에서의 실행시간을 측정한 것이다[8].

표 2. CPU와 GPU의 수행시간 측정 비교(ms)
Table 2. Comparison of execution time in CPU and GPU



실험결과, 기존의 CAMshift에서 가장 많은 연산을 필요로 했던 RGB2HSV 연산의 경우 GPU로 병렬화 시킨 결과 가장 큰 속도 향상을 얻었다. RGB-HSV의 경우, 순차적으로 실행하는 CPU와는 달리 각 픽셀을 GPU상의 쓰레드들이 대응하여 연산하기 때문에 실행결과 약 17배 가량 성능 향상되었다. Histogram generation과 backprojection 연산의 경우 각각 9배 정도의 속도 향상이 측정되었고, CAMshift 알고리즘을 GPU로 수행 했을시에 CPU보다 약 14배 가량의 속도 향상이 측정되었다.

V. 결 론

본 논문에서는 CUDA 기반의 CAMShift 알고리즘을 이용하여 감시 영역 내의 객체 움직임을 판별하여 실시간으로 동작하는 영상 감시 시스템을 설계하였다. CAMshift 추적 알고리즘은 안정적인 성능을 자랑하지만, 각 프레임별로 객체를 인식, 추적하는데에 연산이 오래 걸린다는 단점을 보완하기 위해 CUDA를 기반으로 CAMshift 알고리즘을 병렬화 한 결과, 약 12~14배 정도의 속도 향상을 확인할 수 있었다. 향후 연구로는 배경 설정 등의 개선을 통한 주변 환경 및 조명에 강인한 커널함수의 선택과 이에 따른 효율적인 추적 기법 및 CUDA에서 제공하는 여러 가지 메모리들의 특성을 파악, 성능 최적화에 대해 연구할 필요가 있다.

참고문헌

[1] M. S. Kim, J. W. Han, "*Technical Trends of Smart Cameras*". Electronics and Telecommunications Trends, Vol..26, No.6, pp.139-153, 2011.

[2] Ebrahim Emami, Mahmood Fathy “Object Tracking Using Improved CAMShift Algorithm Combined with Motion Segmentation”, Machine Vision and Image Processing (MVIP), 2011 7th Iranian, pp 1-4, 2011

[3] https://developer.nvidia.com/category/zone/cuda-zone

[4] Gary R, Bradski, *"Computer Vision Face Tracking For use in a perceptual User Interface"*, Microcomputer Reseach Lab, 2002

[5] J. G. Allen, R.YD Xu, J. S. Jin *"Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces,"* Pan-Sydney Area Workshop on Visual Information Processing, pp.3-7, 2003.

[6] http://blog.naver.com/PostView.nhn?blogId=msnayana&logNo=80109766471&redirect=Dlog&widgetTypeCall=true

[7] Ji-Hoon Jo, Sang-Gu Lee, *"Sobel mask Operations Using Shared Memory in CUDA Environment*". Journal of The Korea Knowledge Information Technology Society, Vol.7, No.6, pp.117-123, 2012.

[8] Ji-Hoon Jo, Sang-Gu Lee, *“CUDA Based CAMshift Algorithm for Object Tracking Systems",* Recent Advances in Knowledge Engineering and Systems Science, pp.219-224, Feb. 2013.

감사의 글

본 논문은 2012년도 한남대학교 교비연구비 지원에 의해 수행되었음

저자소개

조지훈 (Ji Hoon Jo)

2011년 8월 : 한남대학교 컴퓨터공학과 졸업

2011년~현재 한남대학교 컴퓨터공학과 석사과정
※ 관심분야 : 객체추적, 영상처리, 영상 인식

이상구 (Sang Gu Lee)

1983년~현재 한남대학교 컴퓨터공학과 교수
※ 관심분야 : 영상처리, 임베디드 시스템, 패턴인식