그래픽 하드웨어 가속을 이용한 실시간 색상 인식

(Real-time Color Recognition Based on Graphic Hardware Acceleration)

김 구 진 † 윤 지 영 †† 최 유 주 †††
(Ku-Jin Kim) (Jiyoung Yoon) (Yoo-Joo Choi)

요 약 본 논문에서는 야외 및 실내에서 촬영된 차량 영상에 대해 실시간으로 차량의 색상을 인식할 수 있는 GPU(Graphics Processing Unit) 기반의 알고리즘을 제시한다. 전처리 과정에서는 차량 색상의 표본 영상들로부터 특징벡터를 계산한 뒤, 이들을 색상 별로 조합하여 GPU에서 사용할 참조 텍스쳐(Reference texture)로 저장한다. 차량 영상이 입력되면, 특징벡터를 계산한 뒤 GPU로 전송하고, GPU에서는 참조 텍스쳐 내의 표본 특징벡터들과 비교하여 색상 별 유사도를 측정한 뒤 CPU로 전송하여 해당 색상명을 인식한다. 분류의 대상이 되는 색상은 가장 흔히 발견되는 차량 색상들 중에서 선택한 7가지 색상이며, 검정색, 은색, 흰색과 같은 3가지의 무채색과 빨강색, 노랑색, 파랑색, 녹색 과 같은 4가지의 유채색으로 구성된다. 차량 영상에 대한 특징벡터는 차량 영상에 대해 HSI(Hue-Saturation-Intensity) 색상 모델을 적용하여 색조-채도 조합과 색조-명도 조합으로 색상 히스토그램을 구성하고, 이 중의 채도 값에 가중치를 부여함으로써 구성한다. 본 논문에서 제시하는 알고리즘은 다양한 환경에서 촬영된 많은 수의 표본 특징벡터를 사용하고, 색상 별 특성을 뚜렷이 반영하는 특징벡터를 구성하였으며, 적합한 유사도 측정 함수(likelihood function)를 적용함으로써, 94.67%에 이르는 색상 인식 성공률을 보였다. 또한, GPU를 이용함으로써 대량의 표본 특징벡터의 집합과 입력 영상에 대한 특징벡터 간의 유사도 측정 및 색상 인식 과정을 병렬로 처리하였다. 실험에서는, 색상 별로 1,024장씩, 총 7,168장의 차량 표본 영상을 이용하여 GPU에서 사용하는 참조 텍스쳐를 구성하였다. 특징벡터의 구성에 소요되는 시간은 입력 영상의 크기에 따라 다르지만, 해상도 150×113의 입력 영상에 대해 측정한 결과 평균 0.509ms가 소요된다. 계산된 특징벡터를 이용하여 색상 인식의 수행시간을 계산한 결과 평균 2.316ms의 시간이 소요되었고, 이는 같은 알고리즘을 CPU 상에서 수행한 결과에 비해 5.47배 빠른 속도이다. 본 연구에서는 차량만을 대상으로 하여 색상 인식을 실험하였으나, 일반적인 피사체의 색상 인식에 대해서도 제시된 알고리즘을 확장하여 적용할 수 있다.

키워드 : 색상인식, 특징벡터, GPGPU

Abstract In this paper, we present a real-time algorithm for recognizing the vehicle color from the indoor and outdoor vehicle images based on GPU (Graphics Processing Unit) acceleration. In the preprocessing step, we construct feature vectors from the sample vehicle images with different colors. Then, we combine the feature vectors for each color and store them as a reference texture that would be used in the GPU. Given an input vehicle image, the CPU constructs its feature vector, and then the GPU compares it with the sample feature vectors in the reference texture. The similarities between the input feature vector and the sample feature vectors for each color are measured, and then the result

· 저자 김구진은 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 본 연구를 수행하였음(R04-2004-000-10099-0)

† 정 회 원 : 경북대학교 컴퓨터공학과 교수
kujinkim@yahoo.com
†† 학생회원 : 경북대학교 컴퓨터공학과
greenseed@empal.com
††† 종신회원 : 서울벤처정보대학원대학교 컴퓨터응용기술학과 교수
yjchoi@suv.ac.kr

논문접수 : 2007년 9월 6일
심사완료 : 2008년 1월 26일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제14권 제1호(2008.2)

is transferred to the CPU to recognize the vehicle color. The output colors are categorized into seven colors that include three achromatic colors: black, silver, and white and four chromatic colors: red, yellow, blue, and green. We construct feature vectors by using the histograms which consist of hue-saturation pairs and hue-intensity pairs. The weight factor is given to the saturation values. Our algorithm shows 94.67% of successful color recognition rate, by using a large number of sample images captured in various environments, by generating feature vectors that distinguish different colors, and by utilizing an appropriate likelihood function. We also accelerate the speed of color recognition by utilizing the parallel computation functionality in the GPU. In the experiments, we constructed a reference texture from 7,168 sample images, where 1,024 images were used for each color. The average time for generating a feature vector is 0.509ms for the 150×113 resolution image. After the feature vector is constructed, the execution time for GPU-based color recognition is 2.316ms in average, and this is 5.47 times faster than the case when the algorithm is executed in the CPU. Our experiments were limited to the vehicle images only, but our algorithm can be extended to the input images of the general objects.

Key words : Color recognition, feature vector, GPGPU

1. 서 론

색상은 영상이 가지는 하나의 속성으로서 영상의 전반에 걸친 색상 정보는 영상 분류 및 내용 기반 영상 검색 분야에서 활용된다[1-4]. 영상 내부 각 영역의 색상 인식 또는 영상에 포함된 물체의 색상 인식은 영상의 분할(segmentation), 물체 인식(object recognition), 물체 추적(object tracking) 등의 여러 분야에서 광범위하게 사용된다[5-7]. Human-computer interaction이나 ubiquitous computing에 있어서도 영상 기반의 물체 인식 문제는 활발히 연구되고 있으며, 색상 인식 문제는 이러한 연구에 있어 기본적인 도구로 사용된다. 본 논문에서는 영상에 포함된 물체의 색상을 인식하는 방법의 한 가지로서, 야외 및 실내에서 촬영된 차량 영상으로부터 차량의 색상을 인식하는 방법을 제안한다.

야외에서 촬영된 영상의 경우, 영상 내에 표현되는 피사체의 색상은 촬영 당시의 조도, 조명, 피사체의 표면 재질, 그리고 주변 환경의 영향 등에 의해 결정된다. 동일한 피사체의 경우라도 촬영 환경에 따라 영상에서는 서로 다른 색상으로 표현될 수 있다. 한 대의 단색 차량을 촬영할 경우에도 주변 조명과 차량 표면에 반사되는 물체의 색상 등에 따라 단색이 아닌 다양한 색상으로 표현될 수 있다. 실내에서 촬영한 영상의 경우, 고정된 환경에서 촬영된 영상은 야외에서 촬영된 영상에 비해 피사체의 색상이 안정적으로 표현될 수 있지만, 여러 다른 장소에서 촬영된 영상은 야외 영상과 마찬가지로 피사체의 색상이 다양하게 표현될 수 있다. 그림 1은 야외 및 실내의 다양한 환경에서 촬영 된 검정색 차량의 영상을 예로 보이며, 검정색이 촬영 환경에 따라 영상마다 서로 다른 색으로 다양하게 표현된 것을 볼 수 있다. 또한, 하나의 영상 내에서도 차량의 후드 영역을 살펴보면, 흰색, 회색, 검정색 등 다양한 변화를 보임을 알 수 있다. 주변 환경에 특정한 색상이 존재할 경우에는 검정색 차량의 표면에 해당 색상이 반사되어 보일 것이다.

촬영 당시의 조도, 조명, 피사체의 재질, 배경 등이 미리 알려진다면, 주어진 입력 영상에 표현된 피사체의 실제 색상을 인식하기는 상대적으로 용이할 것이다. 그러나, 이러한 정보가 없는 상태에서 입력된 임의의 영상에 대해 촬영 당시의 조건을 추측하여 피사체의 실제 색상을 인식하기는 어려운 일이다.

본 논문에서는 촬영 당시의 조건이 알려지지 않았다는 가정 하에, 주어진 임의의 차량 영상에 대해 차량의 색상을 인식하는 알고리즘을 제시한다. 주어지는 입력 영상은 조명이 있는 실내 또는 야외 등 촬영 장소에 제약이 없으며, 영상 내의 차량은 육안으로 색상 구별이 가능한 상태라고 가정한다. 또한 영상 내의 중심부는 항상 차량 영역을 포함한다고 가정한다. 색상 인식 알고리즘의 기본 아이디어는 다음의 가정에서 출발한다.

각각의 차량 색상 별로 다양한 환경에서 촬영된 표본 색상들이 데이타베이스로 구축된다면, 주어진 입력 영상의 차량 색상과 가장 유사한 특성을 갖는 표본 색상을 발견함으로써 입력 영상 내의 차량 색상을 인식할 수

그림 1 다양한 환경에서 촬영된 검정색 차량의 영상

있을 것이다.

영상은 그 자체가 색상 정보로 구성되어 있으므로, 표본 색상을 구축하기 위해 영상 자체의 색상 정보를 사용할 수 있다. 그러나, 영상은 일반적으로 대용량의 크기와 대량의 정보를 가지므로 영상 자체를 이용하여 표본 색상을 구성하기는 어렵다. 따라서, 본 논문에서는 차량의 색상이 갖는 특성을 뚜렷이 반영하는 특징벡터를 구성하여 이들을 데이타베이스로 구축하고자 한다. 이때 특징벡터를 구성하기 위해 고려할 점은 다음의 세 가지로 요약된다.

1. 색상을 표현하는 표본 특징벡터는 각각의 색상 별 차이점을 반영해야 한다.
2. 인식의 성공률을 높이기 위해서는 색상 별 표본 특징벡터의 개수가 충분히 많아야 한다.
3. 특징벡터의 크기가 클수록, 그리고 표본 특징벡터의 개수가 많을수록 색상 인식에 소요되는 수행시간은 길어진다.

본 논문에서는 이러한 고려 사항을 기반으로 무채색과 유채색이 혼재하는 차량의 색상 별 특성을 분석하여 인식 성공률이 높으면서도 가능한 한 크기가 작은 특징벡터를 구성하였다. 또한, GPU(Graphics Processing Unit)를 사용하여 표본 색상과 입력 색상 간의 유사도 측정을 병렬로 처리함으로써 수행 시간의 효율성을 높였다.

차량의 색상은 차량 후드 영역의 색상과 같다고 가정할 수 있다. 만약 입력된 영상에서 차량의 윈드실드, 그릴, 번호판, 헤드라이트 램프 등의 영역을 모두 제외 시키고 후드 영역만을 발견하여 분할할 수 있다면, 차량의 색상 인식에 있어 불필요한 정보를 상당히 제거할 수 있다. 그러나, 차량이 촬영된 방향이나 차량과 카메라의 거리 등에 제약이 없는 임의의 차량 영상에서 차량의 후드 영역만을 분할하는 과정은 그 자체가 해결하기에 어려운 영상 분할 문제이다. 본 논문에서는 표본 영상과 입력 영상에서 차량이 영상의 중심에 있다는 가정 하에, 주어진 영상들에 대해 전체 면적의 25%에 해당하는 중심부의 영역만을 사용하여 특징벡터를 추출하였다. 그림 2에서는 사용된 차량 영상에서 특징벡터 추출에 사용된 영역을 사각형으로 그려 표시하였다. 이 영역은 차량이 아닌 배경에 포함되는 부분을 대체로 제거할 수 있지만, 차량 영역 내부에서도 후드 부분과 차량 부품, 유리창 등을 모두 포함할 수 있다. 본 논문에서 제시하는 색상 인식 알고리즘은 차량의 후드 부분 외에 여러 부품과 유리창 등의 색상이 혼재한 상태의 영상에 대해 차량의 색상을 인식한다.

분류의 대상이 되는 색상은 가장 흔히 발견되는 차량의 색상들 중에서 선택한 7가지 색상이며, 3가지의 무채색인 검정색, 은색, 흰색과 4가지의 유채색인 빨강색, 노랑색, 파랑색, 녹색을 포함한다. 차량 영상에 대해 HSI (Hue-Saturation-Intensity) 색상 모델을 적용하여, 색조-채도(Hue-Saturation) 조합과 색조-명도(Hue-Intensity) 조합의 히스토그램을 계산한 뒤, 무채색 및 유채색의 특성을 강조할 수 있는 기법을 적용하여 특징벡터를 구성한다. 전처리 과정에서는 차량의 색상 별 표본 영상들로부터 특징벡터를 계산하고, 이들을 순차적으로 배열에 저장한 뒤 그래픽 카드의 텍스쳐 메모리로 전송하여 GPU에서 사용할 참조 텍스쳐(reference texture)로 저장한다. 인식하고자 하는 차량 영상이 입력되면, 특징벡터를 계산하여 그래픽 카드의 텍스쳐 메모리로 전송하고, GPU에서는 참조 텍스쳐 내의 표본 특징벡터들과 비교하여 색상 별 유사도를 측정한 뒤 그 결과를 CPU로 전송하여, CPU에서 색상을 인식한다.

실험에서는, 색상 별로 1,024장씩, 총 7,168장의 차량 표본 영상을 이용하여 GPU에서 사용하는 참조 텍스쳐를 구성하였다. 본 논문에서 제시하는 알고리즘은 다양한 환경에서 촬영된 많은 수의 표본 특징벡터를 사용하고, 색상 별 특성을 뚜렷이 반영하는 특징벡터를 구성하였으며, 적합한 유사도 측정 함수(likelihood function)를 적용함으로써, 94.67%에 이르는 높은 인식 성공률을 보였다. 또한, GPU를 이용함으로써 대량의 표본 특징벡터의 집합과 입력 영상에 대한 특징벡터 간의 유사도 측정 및 색상 인식 과정을 병렬로 처리하였다. 표본 영상에 포함되지 않는 1,050장의 입력 영상에 대해 실험한 결과 평균 2.316ms의 속도로 실시간에 색상을 인식하였으며, 이는 같은 알고리즘을 CPU 상에서 수행한 결과에 비해 5.47배 빠른 속도이다. 본 연구에서는 차량만을 대상으로 하여 색상 인식을 실험하였으나, 일반적인 피사체의 색상 인식에 대해서도 확장하여 적용이 가능하다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구를 제시한다. 3절에서는 차량 영상의 색상이 갖는 특성을 분석하여 특징벡터를 구성하는 방법을 제시하고, 색

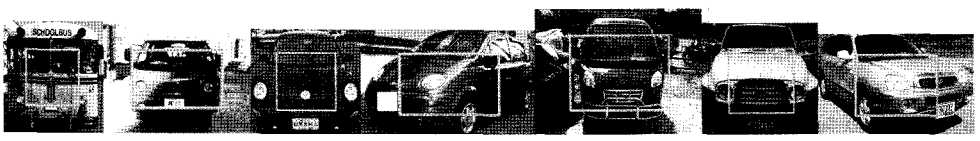

그림 2 색상 인식에 사용되는 영역

상 인식의 성공률을 높이기 위한 기법을 제시한다. 4절에서는 GPU 기반의 색상 인식 알고리즘을 제안하고, 5절에서 실험 결과를 보인다. 마지막으로 6절에서 결론을 내린다.

2. 관련 연구

현재까지 색상 인식과 관련된 연구는 주로 영상을 구성하는 각 픽셀이 어떤 색상 클래스에 속하는지 분류하는 방법에 대한 연구이다. Alvarez et al.[8]은 픽셀의 색상을 분류하는 방법으로 영상에 대하여 YUV 색상 공간에서 3차원의 색상 히스토그램을 구성한 뒤, 각 색상 클래스를 둘러싸는 3차원적인 타원체(ellipsoid)를 구성하여, 주어진 픽셀의 색상이 어떤 타원체에 속하는 지 판단함으로써 픽셀의 색상 클래스를 분류하는 방법을 제안하였다. Quinlan et al.[6]은 HSI 색상 히스토그램을 사용하여 색상의 클래스를 구성한 뒤 SVM machine 기법을 이용하여 픽셀의 색상을 분류하였다. Buluswar and Draper[9]는 영상 내 피사체의 색상을 인식하기 위한 방법으로 scene-context를 기반으로 하는 방법과 기계 학습(machine learning)을 이용하는 방법을 제안하였다. Vandenbroucke et al.[10]은 픽셀의 색상 정보를 이용하여 영상을 분할하는 기법을 제시하였다. 픽셀의 색상은 여러 색상 모델로 분석되고, 이들 중 픽셀이 속한 클래스(class)를 가장 잘 분류할 수 있는 다수의 색상 모델을 조합하여 개조된 색상 공간(adapted hybrid color space)을 구성한다.

GPU 기반의 알고리즘은 다양한 분야에서 적용되며, Owens et al.[11]의 survey 논문에 GPGPU(General Purpose GPU) 관련 내용이 정리되어 있다. GPU 프로그래밍을 작성하기 위한 기법들은 [12]와 [13,14] 등에 소개되어 있다.

3. 색상 특징벡터의 구성

본 연구에서 입력 및 표본으로 사용하는 차량영상은 야외 및 조명이 있는 실내에서 검정색, 은색, 흰색, 빨강색, 노랑색, 파랑색, 또는 녹색의 차량을 촬영한 것이다. 유채색에 속한 각 색상은 해당 색상에서 명암이 변화된 색상을 포함한다. 예를 들어, 진한 녹색과 연한 녹색의 차량들은 모두 녹색의 차량으로 분류된다. 입력된 차량 영상을 7가지 색상 중의 한 가지로 인식하는 것을 연구의 목표로 한다.

색상 모델에서 유채색과 무채색은 서로 다른 특성을 보인다. 유채색에 속하는 색상들의 경우, 촬영 환경이 변하더라도 색조값과 채도값은 색상 별로 거의 일정한 범위 안에 포함되지만, 명도값은 주변 조도의 변화에 따라 변동이 심하다. 반면, 무채색에 속한 색상들은 색조값과 채도값이 정의되지 않고, 명도(Intensity)에 의해서 색을 표현한다. 임의의 차량 영상이 주어질 때, 차량의 색상을 인식하기 위해서는 유채색과 무채색의 특성을 모두 반영하는 특징벡터가 필요하다. 3.1절에서는 특징벡터를 구성하기 위해 히스토그램을 이용하는 방법을 제안하고, 3.2절과 3.3절에서 색상 인식의 성공률을 높이기 위한 두 가지 기법을 제시한다. 3.4절에서는 제안된 기법들을 적용하여 특징벡터를 구성하는 알고리즘을 제시한다.

3.1 특징벡터의 구성

특징벡터의 구성 방법은 다음과 같다. RGB(Red-Green-Blue) 색상 모델로 구성된 입력 영상에 대해, RGB 값으로부터 색조와 채도 그리고 명도값을 분리하기 위해 HSI 색상 모델을 적용하여 특징벡터를 구성한다. HSI 색상 모델은 카메라와 피사체 표면의 방향, 광원의 방향, 광원의 명도, specular highlight 등에 대해 민감하지 않고 변화가 적다는 특성을 보인다[4]. 일반적으로 HSI 모델과 HSV(Hue-Saturation-Value) 모델은 색조, 채도, 명도 값을 분리하는 유사한 역할을 하지만, 결과로 생성하는 Hue와 Saturation의 값, 그리고 Intensity와 Value의 값에 어느 정도의 차이가 있다. 본 논문에서는 색상의 특성을 분석하는 데 유리한 색조, 채도, 명도 값을 구하기 위해 HSV 모델로부터 Hue와 Saturation의 값을, HSI 모델로부터 Intensity의 값을 계산하여 특징벡터의 구성에 사용하였다. 이후에 언급되는 HSI 모델은 두 색상 모델을 조합하여 구성한 색상 모델을 가리킨다.

수식 (1)은 RGB 색상 모델을 HSI 색상 모델로 변환하기 위해 사용하는 식이다. 여기에서 R, G, B는 각각 픽셀의 Red, Green, Blue 성분 값을 나타내고, MAX와 MIN은 R, G, B값 중에서 최대값과 최소값을 의미한다. 각 R, G, B값은 0과 255 사이의 정수값이라고 가정한다.

$$Hue = \begin{cases} Undefined, & if \quad MAX = MIN \\ 60 \times \frac{G-B}{MAX-MIN} + 0, & if \quad MAX = R \quad and \quad G \geq B \\ 60 \times \frac{G-B}{MAX-MIN} + 360, & if \quad MAX = R \quad and \quad G < B \\ 60 \times \frac{B-R}{MAX-MIN} + 120, & if \quad MAX = G \\ 60 \times \frac{R-G}{MAX-MIN} + 240, & if \quad MAX = B \end{cases}$$

$$Saturation = \begin{cases} 0, & if \quad MAX = 0 \\ \frac{MAX-MIN}{MAX}, & otherwise \end{cases}$$

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

HSI 색상 모델에서 유채색을 구분하기 위해서는 색조값이 핵심적인 역할을 하고, 무채색을 구분하기 위해서는 명도값이 핵심적인 역할을 한다. 여기에 채도값은 유채색과 무채색을 구분하는데 있어서 보조적인 역할을

수행할 수 있다. 무채색 영역은 이론적으로 Red, Green, Blue가 동일한 값이므로 색조값은 정의되지 않고 채도값은 0이 되지만, 실세계를 촬영한 영상에서는 시각적으로 무채색일지라도 실제로는 색조값과 채도값을 가질 수 있다. 이는 육안으로는 무채색에 해당하는 픽셀이라도 영상의 픽셀에서 Red, Green, Blue의 각 요소 값이 완전히 같은 경우가 드물기 때문이다.

수식 (1)에 의하면 채도값은 RGB 각 요소의 값 중에서 MAX와 MIN값에 따라 결정된다. 무채색의 경우 RGB 각 요소의 값들이 거의 비슷하게 나타나므로 MAX-MIN은 항상 작은 값을 갖지만, 이를 MAX로 나누는 과정에서 채도값에 차이가 발생한다. 검정색은 RGB 각 요소들이 거의 0의 값에 가깝지만 흰색은 255에 가까우므로, 흰색은 채도값이 낮고, 채도값이 증가함에 따라 회색과 검정색이 된다. 따라서, 무채색에 속하는 색상을 구분하기 위해서는 명도값뿐만 아니라 채도값 또한 이용할 수 있다.

차량의 색상이 갖는 특성을 분석하고 특징벡터를 구성하기 위해, 히스토그램을 사용하였다. 히스토그램을 구성하기 위해 차량의 영상을 7가지 색상 별로 각각 5장씩 선정하여, 차량의 후드에서 반사가 거의 없고 차량의 원래 색상을 일관성 있게 나타내는 영역을 50×50의 해상도의 영상으로 추출하여 실험하였다. 3차원 배열의 각 인덱스를 Hue, Saturation, Intensity로 정의하고, 배열원소가 해당 HSI값을 갖는 픽셀의 개수가 되도록 3차원 히스토그램을 구성할 수도 있으나, 3차원의 경우 색상 별 특징을 분석하기 어렵고, 차후 색상의 특징벡터를 구성할 경우 특징벡터의 크기가 매우 커진다는 단점이 있다. 따라서, 본 논문에서는 3차원 히스토그램을 특정 평면으로 투영하여, 서로 다른 특성을 표현하는 두 종류의 2차원 히스토그램을 생성하여 사용하였다. 색조-채도(Hue-Saturation) 평면과 색조-명도(Hue-Intensity) 평면, 채도-명도(Saturation-Intensity) 평면으로 HSI의 각 요소값 별 픽셀의 개수를 투영한 2차원 히스토그램은 그림 3과 같다.

이론적으로는 무채색의 경우 색조값은 정의되지 않고 채도값은 항상 0이다. 그러나, 그림 3에서 제시하는 바와 같이 실제의 영상에서는 무채색에 대해서도 색조와 채도의 값이 존재한다는 것을 확인할 수 있다. 또한, 채도값의 변화에 따라 무채색에 속하는 색상들이 서로 구분되는 것을 확인할 수 있다. 색조-채도 및 색조-명도의 히스토그램에서 볼 수 있듯이 유채색의 경우, 서로 다른 색상이 차지하는 색조의 영역들이 뚜렷이 구분되지만, 무채색은 구분하기 어렵다. 그에 비해, 색조-명도 및 채도-명도 히스토그램에서는 무채색에 속한 색상들의 명도 영역이 뚜렷이 구분되지만, 유채색은 그렇지 않

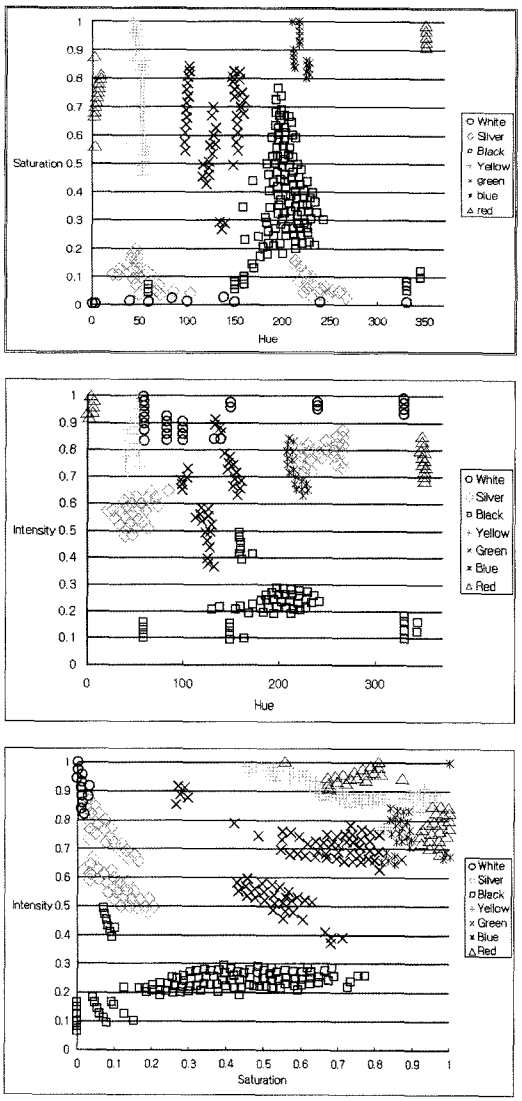

그림 3 색조-채도, 색조-명도, 채도-명도에 대한 2차원 히스토그램

다는 것을 확인할 수 있다. 채도-명도 히스토그램만을 살펴볼 경우에는 색상 별로 특징이 뚜렷하게 구분되지 않는다. 따라서, 본 논문에서는 HSI공간에서 구성된 색조-채도 히스토그램과 색조-명도 히스토그램에서 나타난 특징들을 이용하여 특징벡터를 구성한다.

특징벡터의 크기를 작게 유지하면서, 인식의 수행 속도 및 GPU의 저장 방식 등을 고려하여, 색조는 0~15 범위의 정수값으로, 채도와 명도는 각각 0~31 범위의 정수값으로 양자화한다. 이에 따라 색조-채도의 히스토그램을 16×32 크기의 행렬로, 색조-명도의 히스토그램을 16×32 크기의 행렬로 구성하고, 이들을 조합하여 최

종적으로 32×32 크기의 행렬을 만들어 특징벡터로 사용한다.

3.2 채도값에 대한 가중치 부여

영상에서 무채색과 유채색의 구분을 돕기 위해 채도값의 특징을 이용한다. 차량 영상에는 항상 무채색에 속하는 영역이 포함된다. 예를 들어 타이어나 라디에이터 그릴, 유리창, 또는 도로 등의 부분은 무채색인 경우가 많다. 따라서, 차량의 색상이 유채색인 경우라도, 차량의 영상 내에는 유채색과 무채색의 픽셀이 혼재하게 된다. 반면 무채색의 차량은 영상 내에 유채색의 색상을 가지는 경우가 상대적으로 적다. 따라서, 주어진 영상의 특징으로서 무채색 보다는 유채색을 강조 시키는 방법이 필요하다.

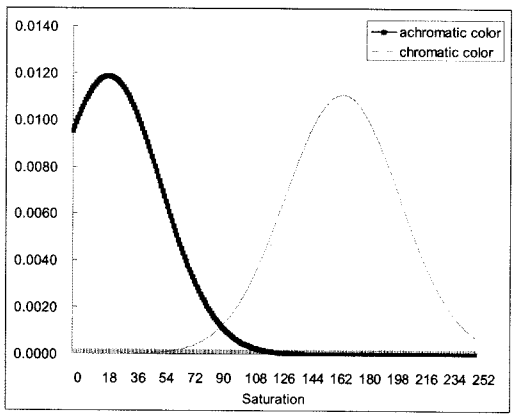

그림 4 무채색 및 유채색 차량의 채도값에 대한 확률 밀도 함수의 결과

그림 4에서는 무채색인 차량 영상 20장과 유채색인 차량 영상 20장에 대해 채도값의 평균 및 분산을 계산하여 확률 밀도 함수(probability density function)를 구한 결과를 보인다. 채도값의 범위를 0~255로 계산할 때, 무채색은 평균이 14.19, 표준편차가 48.41이며, 대부분의 채도값이 127 이하라는 특성을 갖는다. 유채색의 경우 채도값이 127 이하가 되는 경우가 있지만, 대체로 채도값이 작아질수록 시각적으로 유채색이라는 것을 판별하기 어려워진다. 따라서, 특징벡터의 요소값 중에서 채도값이 128이상인 것들은 가중치를 두어 유채색의 특징을 강조한다.

3.3 색조값에 의한 무채색의 구분

영상에서 무채색과 유채색을 구별하기 위한 다른 방법으로 색조값의 특징을 이용한다. 무채색 영상에서 색조값의 범위가 다양하게 나타나면 유채색과 중복되는 특징벡터가 생길 수 있어 혼동을 가져올 수 있기 때문에 이를 피할 수 있는 방법이 필요하다.

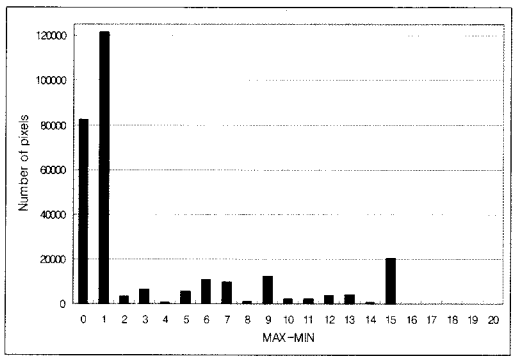

그림 5 무채색 차량 색상에 대한 RGB 각 요소의 MAX-MIN 값의 히스토그램

그림 5는 주변 환경의 영향을 거의 받지 않은 차량 영상에서 무채색에 해당하는 검정색과 은색 그리고 흰색의 차량 영상을 각각 20장씩 총 60장을 선별한 뒤, 차량 후드 영역을 50×50 픽셀의 영상으로 추출하고, 이들의 RGB 각 요소값 중 최대값(MAX)에서 최소값(MIN)을 뺀 결과에 대한 히스토그램을 구한 결과이다. 실험 결과 MAX-MIN 값의 범위는 최소 0으로부터 최대 15의 범위에 포함되었다. MAX-MIN의 계산 결과값이 작을수록 무채색에 근접하므로, 계산 결과의 최대값인 15이하의 픽셀들은 무채색이라 간주하고, 색조값을 *undefined*로 지정한다.

3.4 특징벡터 생성 알고리즘

이 절에서는 3.1절-3.3절에서 제안한 기법들을 이용하여 특징벡터를 구성하는 알고리즘 Feature_Vector_Generation을 제시한다. 알고리즘에서 사용된 RGB_to_HSI_Conversion($I[i][j]$)는 $I[i][j]$ 픽셀의 RGB 값을 해당 HSI 값으로 변환하며, 이때 hue의 범위는 최소 0, 최대 15, saturation과 intensity의 범위는 각각 최소 0, 최대 31의 정수가 되도록 정규화한 결과를 return한다. 실제 실험에서 *undefined* 값은 가능한 색조값의 범위 중에서 차량의 표본 영상들의 색조값으로 자주 사용되지 않는 값을 할당하였다. 특징벡터들은 다양한 크기의 표본 영상과 입력 영상에 대해 생성되므로, 알고리즘을 적용하여 생성된 특징벡터들은 영상의 픽셀 개수에 의해 정규화된다. 이해를 돕기 위해 알고리즘을 최적화하지 않은 상태로 기술하였다.

4. GPU 기반의 색상 인식 알고리즘

일반적으로 CPU를 이용하는 연산은 SISD(Single Instruction Single Data)의 기법으로 처리된다. 색상별로 구성된 대량의 특징벡터들과 입력 영상의 특징벡터를 비교하는 작업은 SISD보다는 SIMD(Single Inst-

알고리즘

```
Algorithm: Feature_Vector_Generation
Input:  I[h][w].RGB   /* 각 픽셀이 R, G, B값을 갖는 w×h 해상도의 입력 영상 */
        Saturation_weight  /* 채도값에 대한 가중치 */
        Saturation_threshold  /* 채도값에 가중치를 부여하기 위한 임계값 */
        Hue_threshold /* 색조값을 undefined 로 정의하기 위한 임계값 */
Output: v[32][32]          /* color feature vector of given image */
1. for i=0 to 15 do
     for j=0 to 31 do
       HS[i][j] = HI[i][j] = 0;     /* HS, HI 히스토그램의 초기화 */
2. for i=0 to h do
     for j = 0 to w do begin
       Max =Find_Max( I[i][j].R , I[i][j].G , I[i][j].B );
       Min = Find_Min( I[i][j].R , I[i][j].G , I[i][j].B );
       I'[i][j] ← RGB_to_HSI_Conversion ( I[i][j]); /*HSI 색상모델로 변환*/
       /* Max-Min 값이 Hue_threshold 이하인 경우, hue값으로 undefined 를 저장 *
       if Max-Min ≤ Hue_threshold then
          I'[i][j].hue = undefined ;
       /* Saturation 값에 대해서 가중치를 부여하며 HS 히스토그램을 구성 */
       if I'[i][j].saturation <= Saturation_threshold /* 무채색이라 추정할 경우 */
          HS [ I'[i][j].hue ][ I'[i][j].saturation ]++
       else                          /* 유채색이라 추정하여 가중치를 부여할 경우 */
          HS [ I'[i][j].hue ][ I'[i][j].saturation ]+= Saturation_weight ;
       /* HI 히스토그램 구성 */
       HI [ I'[i][j].hue ][ I'[i][j].intensity ]++;
     end
3. for i=0 to 15 do
     for j = 0 to 31 do begin
       v[i][j] = HS[i][j];
       v[i+16][j] = HI[i][j];
     end
```

ruction Multiple Data) 기법으로 처리할 때 뛰어난 효율성을 보일 수 있다. 본 논문에서는 GPU를 기반으로 SIMD 기법을 이용하여 효율적으로 비교 작업을 수행하는 알고리즘을 제시한다(그림 6).

GPU에서 사용하는 참조 텍스쳐는 전처리 과정에서 미리 생성된다. 차량 색상의 표본 영상들로부터 특징벡터를 계산한 뒤, 이들을 색상 별로 조합하여 참조 텍스쳐(Reference texture)를 구성한다. 참조 텍스쳐는 GPU 내의 기억장치에 최초로 한 번 저장된 후, 모든 입력 영상의 색상 인식에 대해 반복적으로 사용할 수 있으므로, 이 과정은 실행 시간에 영향을 주지 않는다.

특징벡터 간의 유사도를 측정하기 위해서는 3가지 함

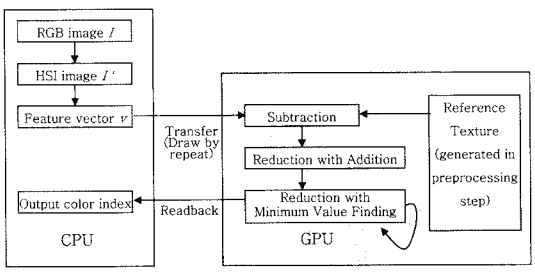

그림 6 GPU 기반의 색상 인식 알고리즘

수를 고려하였고, 그 중 가장 성공률이 높은 city-block distance 함수를 이용하여 GPU 기반 알고리즘을 구성

하였다. 고려된 3가지 함수 및 성공률과 계산 시간은 5절의 실험 결과에서 설명한다.

그림 6에서 제시된 알고리즘에서 GPU 상의 각 단계들은 다음과 같은 역할을 수행한다.

1. Transfer: CPU에서 계산된 특징벡터 v_{input}은 repeat 매개변수를 사용하여 OpenGL에서 반복된 사각형으로 그려진다. 이 때, 참조 텍스처와 동일한 크기만큼 v_{input}을 반복하여 drawing을 함으로써, 참조 텍스처 내의 각 특징벡터는 v_{input}과 한 개씩 같은 텍스처 좌표로 대응된다.

2. Subtraction: 특징벡터 v_{input}과 GPU 내의 참조 텍스처에 포함된 표본 특징벡터 간의 차이를 같은 좌표값에 대응하는 요소 별로 계산하여 절대값을 취한다.

3. Reduction with addition: 특징벡터 단위로 Subtraction 단계의 결과값을 합산하여 새로운 텍스처 T'에 저장한다.

4. Reduction with minimum value finding: 위의 단계에서 생성된 텍스처 T'에서 인접한 2×2 행렬마다 최소값을 구하여 크기가 행의 크기와 열의 크기가 1/2로 감소된 새로운 텍스처 T''에 저장한다(reduce 단계). 핑퐁 방식[13,14]을 적용하여 T''에 저장된 값들에 대해 다시 reduce를 수행하여 텍스처 T'에 저장한다. 텍스처의 크기가 1×7×4로 감소될 때까지 핑퐁 방식으로 계속 reduction을 수행한다.

5. Readback: 위의 단계를 거쳐 최종적으로 계산된 결과를 CPU로 전송한다.

6. Output color index: CPU에 readback된 값 중 최소값을 발견하여 해당하는 색상명, 즉, 입력된 특징벡터와 가장 유사도가 높은 표본 특징벡터가 속한 색상명을 결과로 출력한다.

그림 7에서 참조 텍스처와 입력된 특징벡터에 의해 생성된 입력 텍스처의 예를 보인다. 참조 텍스처는 2차원 텍스처로 구성되며, 텍셀 한 개는 R, G, B, A의 값을 가질 수가 있다. 따라서, $H \times W$ 크기의 2차원 텍스처를 생성하면 실제로는 $H \times W \times 4$ 크기의 3차원 배열의 저장 공간이 생성된다. 본 연구에서는 3.4절에서 제시된 특징벡터 생성 알고리즘에 의해 색상 별로 16×16×4개의 표본 영상에 대해 32×32 행렬 형태의 특징벡터를 생성한 뒤, R, G, B, A의 저장 장소에 저장하여 참조 텍스처를 생성한다. 그림 6에서 $T_{color,i,j,k}$ 는 해당 색상의 표본 영상 1장에 대해 생성한 32×32 특징벡터이며 i,j는 텍스처 상의 행렬 시작 좌표를, k는 R, G, B, A 저장 장소 중 소속된 장소를 나타낸다. 입력 텍스처는 입력 영상에 대해 생성한 32×32 크기의 특징벡터 v_{input}을 참조 텍스처와 같은 크기의 텍스처에 반복하여 drawing 함으로써 생성한다.

그림 8에서는 생성된 참조 텍스처와 입력 텍스처가 Subtraction, Reduction with addition, Reduction with minimum value finding 단계를 거치는 과정을 보인다. Subtraction 단계에서는 참조 텍스처 내에 저장된 표본 특징벡터와 입력 텍스처 내의 입력 특징벡터 간의 차를 각 원소 별로 계산하여 그 절대값을 새로운 텍스처에 저장한다. Reduction with addition 단계에서는 참조 텍스처 내에서 특징벡터 단위로 원소값들을 모두 더함으로써 텍스처의 크기를 16×(16×7)×4로 감소시킨다. 이 텍스처에 대해 2×2 행렬의 원소값들 중 최소값을 선택하는 Reduction with minimum value finding 을 반복 수행하여 최종적으로 1×7×4 크기의 텍스처를 얻는다. 이 텍스처는 CPU로 readback되며, 이들 중에서 최소값

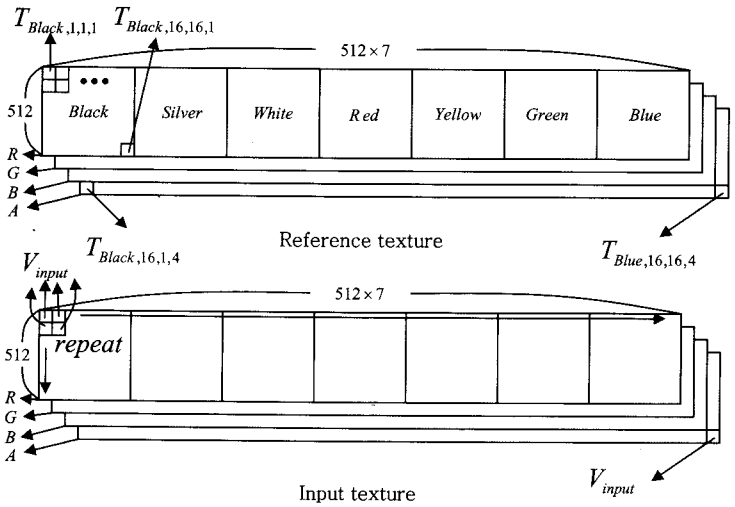

그림 7 참조 텍스처와 입력 텍스처

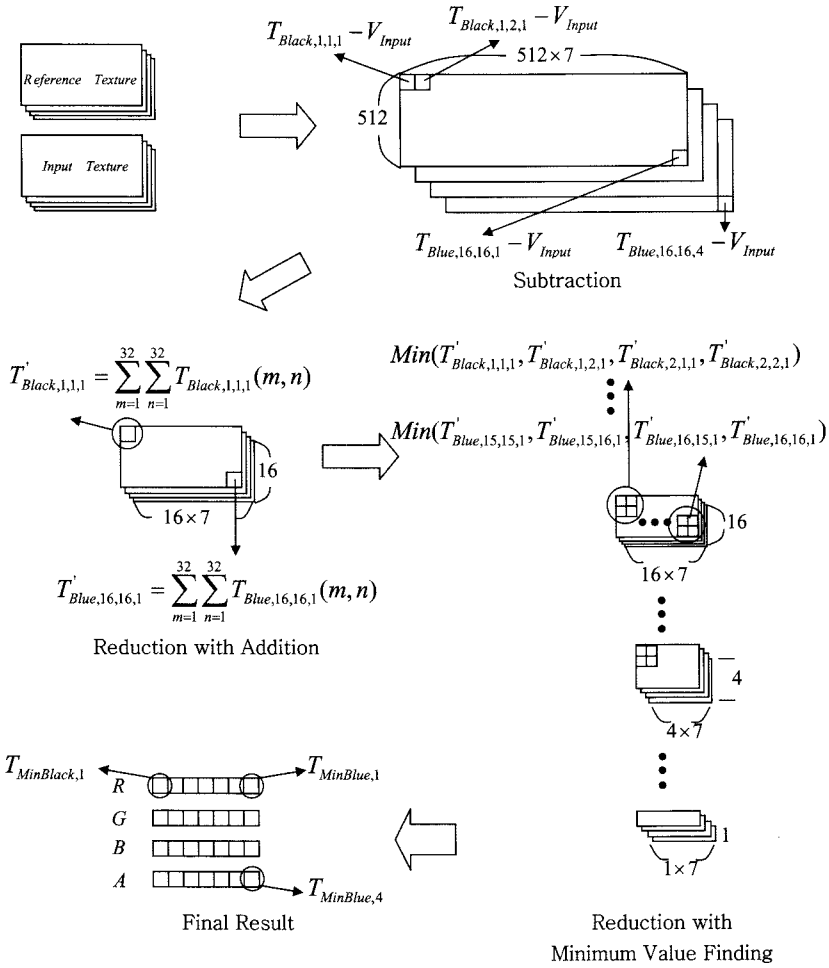

그림 8 참조 텍스쳐와 입력 텍스쳐에 대한 GPU 알고리즘 적용

이 있는 색상 영역을 차량의 색상으로 출력한다.

참조 텍스쳐의 구성은 7가지 색상의 참조 텍스쳐들 각각에 대해서 텍스쳐 크기를 가로와 세로의 비율이 같도록 한다. 그리고 유사도 측정 알고리즘을 적용한 다음 reduction 연산을 수행해 나간다. 이렇게 7가지 색상의 참조 텍스쳐 크기를 가로와 세로의 비율이 같도록 하는 이유는 reduction 연산을 적용 한 뒤 7가지 색상 각각에 대한 계산결과를 얻기 위해서다. 이렇게 얻어진 값을 메인 메모리로 readback 하여 7가지 색상의 결과 중에서 가장 근접한 값을 찾아 이에 해당하는 색상을 인식 색상으로 결정한다. 이러한 과정을 거치면 참조 텍스쳐에 따로 색상에 대한 인덱스를 할당하지 않아도 되기 때문에 연산 효율을 높일 수 있다. 단 리드백 된 7가지 색상의 결과 중에 최소값을 찾는 과정은 CPU에서 수행해야 하지만, 인덱스를 위해 별도의 저장공간을 활용하는 방법에 비하면 훨씬 빠른 속도로 색상을 인식 할 수 있으므로 효율적이다.

CPU와 GPU간의 데이타 전송 속도는 CPU와 메인 메모리간의 데이타 전송 속도보다 느리다. CPU와 메인 메모리는 메인보드의 버스를 통해 직접 연결이 되어있는 데 반해, GPU는 AGP 또는 PCI Express 방식의 확장기기이기 때문이다. 본 논문에서는 가능한 한 CPU와 GPU 간에 전송되는 데이타의 양을 감소시킴으로써 효율적으로 GPU 기반 알고리즘을 구성하였다.

5. 실험 결과

본 논문에서 제안한 알고리즘은 Intel Dualcore 2.13 Mhz의 CPU와 1G byte의 RAM 및 ATI 사의 Radeon X1950 (PCI Express, Memory:512M)가 탑재된 PC에서 실험되었다. GPU 프로그램의 작성은 Cg를 이용하였다.

실험에 사용한 차량 색상은 검정색, 은색, 회색, 빨강색, 노랑색, 녹색, 파랑색으로 분류되며, 이들은 다양한 차종에 대해 중국의 도로에서 촬영한 정면 영상 및 한국과 미국의 중고차 회사의 웹페이지를 통해 획득한 차량의 정면, 측면, 후면 등 촬영 각도 및 배경이 다양하게 구성된 영상들로 총 8,218장이다. 이 영상들은 7가지 색상 별로 각 1,174 장으로 구성되었다. GPU의 참조 텍스쳐에는 7가지의 색상 별로 각각 1,024장의 영상에 대해 32×32 크기의 표본 특징벡터를 생성하여 저장하였다. 각 색상 별로 32×32의 행렬로 구성된 1,024개의 특징벡터들을 512×512 크기의 텍스쳐 RGBA 평면에 각각 순차적으로 저장하였다. 텍스쳐의 RGBA 평면을 모두 사용함으로써, 각 색상마다 1,024개의 특징벡터가 참조 텍스쳐에 저장될 수 있다. 입력 영상으로는 표본 영상에 포함되지 않은 영상을 선택하여 색상 별로 각 150장씩, 총 1,050장을 사용하여 실험하였다.

입력 영상에 대해 특징벡터를 구성한 뒤 표본 특징벡터와의 유사도를 측정하기 위해서 3가지의 측정 방법을 적용하여 실험하였다. 임의의 벡터 간의 유사도를 측정하기 위한 함수는 다양하게 존재하지만, GPU의 기능을 충분히 활용하기 위해서는 두 벡터 내의 대응되는 원소들에 대해 병렬로 계산할 수 있는 함수가 적절하다. 본 연구에서는 특징벡터 간의 유사도 측정을 위해 Euclidean distance, City-block distance, Cosine-angle distance를 고려하였다. 두 개의 특징벡터 $U=(u_1,u_2,...,u_n)$와 $V=(v_1,v_2,...,v_n)$가 주어질 때, 다음의 수식들은 두 벡터 간의 City-block distance, Euclidean distance, Cosine-angle distance를 나타낸다.

$$D_{City-Block}(U,V)=\sum_{i=1}^{n}|u_i-v_i| \tag{2}$$

$$D_{Euclidean}(U,V)=\sqrt{\sum_{i=1}^{n}(u_i-v_i)^2} \tag{3}$$

$$D_{Cosine-Angle}(U,V)=\frac{\sum_{i=1}^{n}(u_i v_i)}{\sum_{i=1}^{n}u_i^{\,2}\sum_{i=1}^{n}v_i^{\,2}} \tag{4}$$

3.4절의 알고리즘에 의해 특징벡터를 생성한 뒤 색상 별로 150장의 입력 영상에 대해 유사도 측정 함수 City-block distance, Euclidean distance, cosine-angle distance를 적용한 결과, 표 1과 같은 인식 성공률을 얻었다. 실험 결과 City-block distance가 가장 인식 성공률이 높고 계산 시간이 짧은 것으로 측정 되었다.

유사도 측정 함수로 City-block distance 함수를 사용할 때, 인식 성공률은 특징벡터를 구성하는 기법에 따라 달라질 수 있다. 표 2에서는 색조-채도 평면과 색조-명도 평면의 조합에 의해 32×32 크기의 특징벡터를 구성할 때, 3.2절의 기법, 3.3절의 기법들을 적용한 경우와 적용하지 않은 경우에 대해 각각 인식 성공률을 비교하였다. 실험 결과 3.2절과 3.3절에서 제시한 두 가지 기법을 동시에 적용시킨 경우가 가장 효과적으로, 94.67%에 이르는 성공률을 얻었다. 실험에서 *Saturation_threshole* 값은 127, *Saturation_weight* 값은 2, *Hue_threshold* 값은 15를 사용하였다.

특징벡터를 구성하는 과정은 CPU에서 수행되며 이때 소요되는 시간은 입력 영상의 해상도와 영상에 포함되어 있는 색상의 종류와 양에 따라 다르다. 해상도가 150×113인 영상으로부터 특징벡터를 만드는 과정의 평균 수행 시간은 0.509ms가 소요되었다.

특징벡터가 구성된 후, 입력 영상의 특징벡터와 표본 특징벡터 간의 유사도를 측정하여 색상을 인식하는 모든 과정을 CPU에서 수행한 결과와 GPU 기반으로 수행한 결과를 표 3에서 비교하였다. 색상 인식 알고리즘의 평균 수행 시간은, CPU 상에서만 수행할 경우 평균 12.67ms가 소요되고, GPU 기반의 알고리즘을 이용한 경우 평균 2.316ms가 소요되므로, GPU 기반의 알고리즘이 CPU 기반의 알고리즘에 비해 5.47배 빠른 속도를 보였다.

6. 결 론

본 논문에서는 야외 및 실내에서 촬영 된 차량 영상에 대해 실시간으로 차량의 색상을 인식할 수 있는 GPU 기반의 알고리즘을 제시하였다. 색상이 가진 특성을 분석하여, 채도값에 가중치를 두는 기법과 색조값에 의해 무채색을 구분하는 기법을 적용하여 특징벡터를 구성하는 방법을 제안하였으며, 대량의 표본 특징벡터들 중에서 입력된 특징벡터와 가장 유사성이 높은 것을 발견하기 위해 GPU를 이용하는 효율적인 알고리즘을 제시하였다.

표본 영상으로 7,168장을 사용하고, 표본 영상에 포함되지 않은 7가지 색상의 입력 영상 1,050장에 대해 제안된 알고리즘을 적용한 결과, 색상에 대한 인식 성공률은 평균적으로 94.67%의 높은 성능을 보였다. 한 장의 차량 영상에 대해 특징벡터를 구성하기 위해서는 150×113 해상도의 경우 평균 0.509ms가 소요되었다. 특징벡터가 구성된 후, GPU 기반으로 색상 인식 알고리즘을 수행하는 데 평균 2.316ms의 시간이 소요되었다. 이는 CPU를 이용하는 인식 알고리즘에 비해 약 5.47배 가량 빠른 속도로서, 효율성의 면에서 뛰어나다.

향후 연구 과제로는 본 논문에서 제안된 기법을 다양한 피사체에 대한 색상 인식 기법으로 확대하여 응용하는 과제를 고려하고 있다.

표 1 유사도 측정 함수에 따른 색상 인식 성공률

Likelihood function / Color	City-block	Euclidean	Cosine -angle
Black	93.3%	76.7%	72.0%
Blue	97.3%	87.3%	88.0%
Green	94.7%	92.0%	88.0%
Red	98.7%	98.7%	97.3%
Silver	88.7%	71.3%	73.3%
White	91.3%	76.0%	80.0%
Yellow	98.7%	98.0%	96.7%
Average	94.67%	85.71%	85.04%
Execution Time on CPU	12.67ms	760.42ms	1076.64ms

표 2 특징벡터의 구성에 따른 색상 인식 성공률(유사도 측정 함수: city-block distance함수)

	Either [a] or [b] is not used	[a] is used only	[b] is used only	Both [a]&[b] are used
Black	76.7%	72.0%	88.7%	93.3%
Blue	86.7%	90.0%	89.3%	97.3%
Green	90.7%	93.3%	92.7%	94.7%
Red	100%	100%	98.7%	98.7%
Silver	71.3%	69.3%	85.3%	88.7%
White	81.3%	80.0%	88.0%	91.3%
Yellow	98.7%	98.7%	98.0%	98.7%
Average	86.49%	86.18%	91.53%	94.67%

[a] : 유채색으로 추정되는 색상의 채도값에 가중치를 부여함 (3.2절 참조)
[b] : 무채색으로 추정되는 색상의 색조값에 'undefined' 를 할당함 (3.3절 참조)

참 고 문 헌

[1] Chapelle, O., Haffner, P., Vapnik, V. N., "Support vector machines for histogram-based image classification," *IEEE Transactions on Neural Networks*, Vol.10, No.5, pp. 1055-1064, 1999.

[2] Smith, J. R., Chang, S. -F., "Tools and techniques for color image retrieval," In: Sethi, I. K., Jain, R. C., eds., Storage & Retrieval for Image and Video Databases IV, vol. 2670 of IS&T/SPIE Proceedings. San Jose, CA, USA, pp. 426-437, 1996.

[3] Rui, Y., Huang, T. S., and Chang, S. -F., "Image retrieval: current techniques, promising directions and open issues," *Journal of Visual Communication and Image Representation*, Vol.10, No.4, pp. 39-62, April 1999.

[4] Jeong, S., Won, C. S., Gray, R. M., "Image retrieval using color histograms generated by Gauss mixture vector quantization," *Computer Vision and Image Understanding*, Vol.94, No.1-3, pp. 1077-3142, 2004.

[5] Park, J. B., Kak, A. C., "A New Color Representation for Non-White Illumination Conditions," *Technical Report, TR-ECE-05-06*, Purdue University, 2005.

[6] Quinlan, M. J., Chalup, S. K., Middleton, R. H., "Application of SVMs for colour classification and collision detection with AIBO robots," *Proceedings of Neural Information Processing Systems (NIPS)*, 2003.

[7] Browning, B., Veloso, M., "Real-Time, Adaptive Color-based Robot Vision," *Proceedings of IROS '05*, Edmonton, Canada, August 2005.

[8] Alvarez, R., Milan, E., Swain-Oropeza, R., Aceves-Lopez, A., "Color image classification through fitting of implicit surfaces," *Proceedings of IBERAMIA 2004*, LNAI 3315, pp. 677-686, 2004.

[9] Buluswar, S. D., Draper, B. A., "Color recognition in outdoor images," *Proceedings of Sixth International Conference on Computer Vision*, pp. 171-177, 1998.

[10] Vandenbroucke, N., Macaire, L., Postaire, J. -G., "Color image segmentation by pixel classification in an adapted hybrid color space: Application to soccer image analysis," *Computer Vision and Image Understanding*, Vol.90, pp. 190-216, 2003.

[11] Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Kruger, J., Lefohn, A. E., Purcell, T. J., "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, Vol.26, No.1, pp. 80-113, 2007.

[12] Fernando, R., Kilgard, M. J., *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics*, Addison-Wesley, 2003.

[13] Fernando, R., *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*, Addison-Wesley, 2004.

[14] Pharr, M., *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, Addison-Wesley, 2005.

표 3 GPU 기반 알고리즘과 CPU 기반 알고리즘의 계산 시간 비교

	색상 인식 알고리즘 수행 시간					
CPU	12.67ms					
GPU	Transfer	Subtract	Add	Reduce	Readback	Total
	0.233ms	0.049ms	0.109ms	0.101ms	1.824mw	2.316ms

김 구 진

1990년 이화여자대학교 전자계산학과 졸업(학사). 1992년 한국과학기술원 전자계산학과 졸업(공학석사). 1998년 포항공과대학교 컴퓨터공학과 졸업(공학박사). 1998년~2000년 Dept. of Computer Sciences, Purdue Univ., PostDoc. 2000년~2002년 아주대학교 정보통신전문대학원 BK21 조교수. 2002년~2003년 Dept. of Mathematics and Computer Science, University of Missouri-St. Louis, Visiting Assistant Professor. 2004년~현재 경북대학교 컴퓨터공학과 조교수 관심분야는 컴퓨터 그래픽스, 컴퓨터 비전, GPGPU, 곡면 및 기하모델링 등

윤 지 영

2006년 대구가톨릭대학교 컴퓨터공학과 졸업(학사). 2006년~현재 경북대학교 컴퓨터공학과 재학중(공학석사). 관심분야는 컴퓨터 그래픽스, 컴퓨터 애니메이션, GPGPU

최 유 주

1989년 이화여자대학교 전자계산학과(학사). 1991년 이화여자대학교 전자계산학과(석사). 2005년 이화여자대학교 컴퓨터학과(박사). 1991년~1993년 한국컴퓨터주식회사 기술연구소 주임연구원. 1994년~1999년 포스데이타주식회사 기술연구소 주임연구원. 2005년~현재 서울벤처정보대학원대학교 컴퓨터응용기술학과 전임강사. 관심분야는 컴퓨터 그래픽스, 가상현실, HCI, 컴퓨터비전, 의료영상처리 등