신경망 기반의 텍스춰 분석을 이용한 효율적인 문자 추출
(Efficient Text Localization using MLP-based Texture Classification)

정 기 철 † 김 광 인 †† 한 정 현 †††
(Kee Chul Jung) (Kwang In Kim) (Jung Hyun Han)

요 약 본 논문은 MLP와 MultiCAMShift 알고리즘을 이용한 텍스춰 기반의 영상 내 문자 추출 방법을 제안한다. MLP를 이용한 텍스춰 분석기는 별도의 특징값 추출 단계 없이 다양한 환경의 입력 영상에 대해 효과적으로 문자 확률 영상을 생성하며, 문자 확률 영상 상에서 수행되는 MultiCAMShift 알고리즘은 국소 탐색만으로 효율적으로 문자 영역을 추출할 수 있다.

키워드 : 신경망, 텍스춰, 문자추출

Abstract *We present a new text localization method in images using a multi-layer perceptron (MLP) and a multiple continuously adaptive mean shift (MultiCAMShift) algorithm. An automatically constructed MLP-based texture classifier generates a text probability image for various types of images without an explicit feature extraction. The MultiCAMShift algorithm, which operates on the text probability image produced by an MLP, can place bounding boxes efficiently without analyzing the texture properties of an entire image.*

Key words : *MLP, Texture, Text Localization, MultiCAMShift*

1. 서 론

전통적으로 영상 데이타는 많은 시간과 노동력을 요구하는 수 작업에 의해 인덱싱되거나, 상대적으로 저 차원적인 색상, 모양, 질감 등의 정보를 이용하여 자동으로 인덱싱되고있다. 최근에는 영상에 내재된 문자를 추출하고 인식함으로써 인덱싱에 유용한 고급 정보를 얻을 수 있다는 장점으로 인하여, 영상 내의 문자 추출에 관한 연구가 멀티미디어 시스템, 전자 도서관, 비디오 인덱싱, 문서 구조 분석, 우편 영상 내의 주소 영역 추출, 자동차 번호판 추출 등 다양한 관련 분야에서 진행되고 있으나, 양질의 문서의 자동 문자 인식(OCR) 과는 달리 상당히 어려운 문제로 인식되고 있다[1-16].

많은 논문에서 문자 감지 (text detection)와 문자 추출 (text localization)을 비슷한 의미로 사용하고 있으나, 이들은 다음과 같이 조금 다른 의미를 지니고 있다. 엄밀한 의미에서 문자 감지는 주어진 입력 영상에 문자가 포함되어있는지 여부를 감지하는 것을 말하고, 문자 추출은 영상 내의 문자의 위치를 파악하는 것을 말한다. 많은 연구에서 입력 영상이 문자를 포함하고 있다고 가정하는데 이는 CD 상자나 책 표지 등을 스캐너로 입력받아 실험하는 경우에 일반적인 방법이다[5, 10]. 그러나 디지털 비디오와 같은 연속된 영상을 사용하는 경우에는, 일반적으로 문자를 포함한 프레임(frame) 보다 포함하지 않은 프레임이 더욱 많이 나타나는데, 이러한 환경에서 문자 감지 단계는 더욱 복잡한 단계인 문자 추출 단계를 불필요하게 수행하는 것을 줄임으로써 전체적인 시스템의 성능을 향상시킬 수 있다. 이를 위해 Gargi [16] 등은 연속된 영상에서 문자가 나타날 때 P-, B-프레임에서의 intracoded 블록 개수의 증가치를 보고 문자 감지를 수행하고, Wernicle와 Lienhart [15]는 균일한 시차에 따라 샘플링된 프레임에 대해서만 문자 추출을 수행한다. Zhong [4] 등은 문자 추출기를 문자 감지로도 사용하였는데 이는 문자 추출기가 실시간으로 수행될 수 있기 때문이다.

기존의 문자 추출 방법은 크게 연결 성분 방법 (connected component method : CCM)과 텍스춰 (tex

† 정 회 원 : PRIP lab Michigan State Universty
kjung@ece.skku.ac.kr
†† 비 회 원 : 한국과학기술원 인공지능연구실
kimki@ai.kaist.ac.kr
††† 종신회원 : 성균관대학교 전기전자컴퓨터공학부 교수
han@ece.skku.ac.kr
논문접수 : 2001년 5월 14일
심사완료 : 2001년 11월 7일

ture) 기반 방법으로 나눌 수 있다. 연결 성분 방법은 색상, 질감 등의 정보를 이용하여 입력 영상을 분할한 후, 각 연결 성분들을 특정 조건들을 이용하여 문자 영역과 비-문자 영역으로 나누는 방법이다[1, 5, 6]. Lienhart와 Studer [1]는 비슷한 색상, 크기의 연결 성분을 문자로 간주하여 추출하고 모션 정보를 이용하여 추출 결과를 향상시켰다. Jain과 Yu [5]는 입력 프레임을 각기 다른 색상으로 분할한 후, 각 분할된 조각들을 크기나 배열 모양 등의 조건들을 이용해서 걸러냄으로써 문자 추출을 수행한다. Kim [6] 등은 한글과 한자 등의 다-조각 문자 (multi-segment character)를 포함하는 입력 영상에서 비-문자 성분들을 걸러내기 위해 클러스터를 기반으로 한 템플릿을 사용하였다. CCM과 달리 텍스춰 기반 방법은 문자 영역의 텍스춰 성질을 이용하기 위해 Gabor filter, wavelet, spatial variance 등의 텍스춰 분석기를 사용하는 방법이다[3, 4, 7, 8, 11-15]. Li [3] 등은 wavelet을 이용한 특징값 추출 후 신경망을 이용하여 문자 영역을 추출하였다. Zhong [4] 등은 DCT 압축된 도메인에서 직접 텍스춰 특징을 이용한 방법을 사용하였다. Zhong [7] 등은 그레이 영상에서의 국소 영역 변화 (local spatial variation)을 이용하여 높은 차이를 보이는 영역을 문자 영역으로 간주하였으며, 또한 이 방법을 CCM과 결합하여 사용하였다. Jain과 Karu [18]는 서류 영상 내의 문자, 그래픽, 반음영 영역 (halftone region)을 구분하기 위해 학습 기반의 텍스춰 분석 기법을 사용하였다

본 논문에서는 텍스춰 기반의 새로운 문자 추출 방법을 제안한다. 문자 추출을 위한 일반적인 방법인 CCM은 구현이 쉬운 반면, 문자 크기와 문자 간의 거리 등에 많은 영향을 받으며, 정밀한 영상 분할 알고리즘을 필요로 하기 때문에 비디오 영상과 같은 잡음이 많은 저해상도 영상 또는 다양한 크기의 영상들에는 적합하지 않다. 문자 추출에 상당히 효과적이기는 하지만 텍스춰 기반의 방법 또한 (1) 텍스춰 분석기 생성의 어려움; (2) 텍스춰 분석 단계에서의 많은 계산량; (3) 텍스춰 분석 후의 후처리 과정에서 사용하는 파라미터 설정의 어려움과 그에 따른 전체적인 시스템의 성능저하와 같은 몇 가지 단점들이 있다.

문자나 배경 영상의 다양한 변형으로 인해 각각의 적용 분야 마다 텍스춰 분석기를 수 작업으로 생성하는 것은 쉽지 않다. 본 연구에서는 문자 영역과 비-문자 영역을 분류하는 텍스춰 분석기를 생성하기 위해 MLP를 사용함으로써 다양한 환경에 적응성을 지니는 분석기를 자동으로 생성한다.

두 번째 단점은 텍스춰 분석 단계의 많은 계산량으로 인한 전체적인 시스템의 수행 속도 저하이다 [3, 8, 13, 14, 18]. 특히 텍스춰 기반의 컨벌루션 (convolution) 필터링 기법은 문자 추출을 위해서 전체 입력 영상을 모두 스캔 (scan) 해야 한다. 특정 필터로 전 영상을 컨벌루션하는 것은 상당한 수행시간을 필요로 하는 작업인데, 본 연구는 이를 문자 확률 영상 (text probability image: TPI) 상의 다수의 시작 노드에서 continuously adaptive mean shift (CAMShift) 연산을 수행하며 문자 영역을 추출함으로써 극복한다.

텍스춰 분석 후 문자 영역을 설정하기 위해 일반적으로는 연결 성분 생성 또는 프로젝션 등의 후처리 과정을 수행하게 되는데 [3, 5, 8, 14, 15], 이에 사용되는 파라미터들을 구하는 일은 쉽지 않고, 파라미터 값의 설정이 전체적인 시스템의 성능에 영향을 줄 수 있다. 제안한 방법은 기존의 방법에 비해서 효율적으로 보다 정확한 문자 영역을 추출하며, 사용되는 파라미터들의 값에 큰 영향을 받지 않는다.

본 논문의 구성은 다음과 같다. 제 2장에서는 MLP와 MultiCAMShift 알고리즘을 이용한 문자 추출 방법에 대해 설명하고, 제 3장에서는 제안된 방법을 이용한 실험 결과와 분석을 보이고, 제 4장에서는 결론과 향후 연구 분야를 기술한다.

2. 문자 추출

본 절에서는 문자 영역 추출을 위한 텍스춰 분석 신경망과, 일반적인 mean shift 알고리즘, 그리고 Multi CAMShift 알고리즘을 기술한다.

2.1 텍스춰 분석 MLP

문자 영역과 비-문자 영역을 구분하는 텍스춰 분석기를 구성하는 어려움을 극복하기 위해 학습을 통한 텍스춰 분석기 생성에 관한 연구가 진행되고 있는데 [3, 8, 12, 18], 본 논문에서는 MLP를 사용하여 다양한 크기나 모양의 문자와 배경에 적응할 수 있는 텍스춰 분석기를 구성한다. MLP는 2개의 은닉층, 1개의 출력 노드로 구성되며 인접층의 노드들은 모두 연결되어있고, 입력층은 그림 1에 표시되어 있는 바와 같이, 입력 영상에서 M×M 크기의 입력창 내의 회색으로 표시된 화소들의 인텐서티 값을 사용한다. 그림 1은 신경망의 입력층이 생략되어 있고, 대신 입력 화소의 모양으로 이를 대신한다. 이러한 입력 구조는 텍스춰 분석 분야에서 성능과 속도 향상에 좋은 것으로 알려져 있다 [18]. MLP를 이용하여 입력 영상을 스캔 함으로써 생성된 TPI의 각 화소는 대응하는 입력 영상의 각 화소를 문자와 비-문자 클래스로 구분한다.

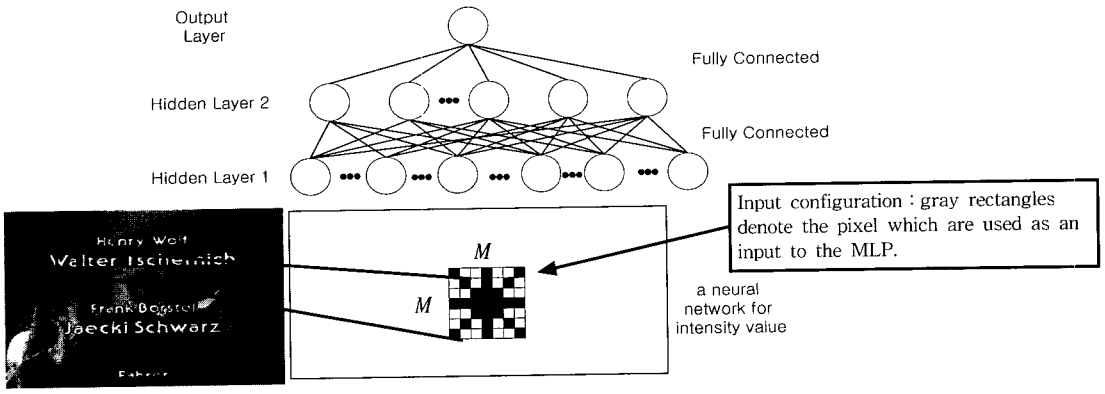


그림 1 텍스춰 분석기 MLP의 구조

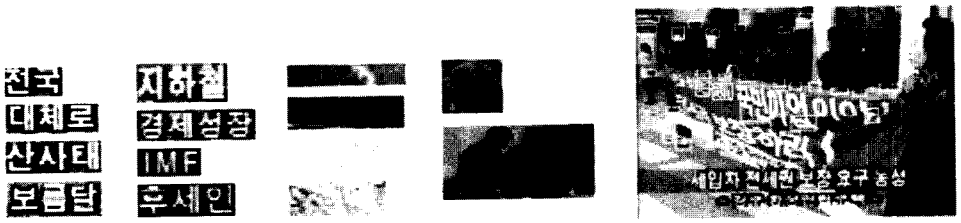


그림 2 신경망의 학습 데이타 : (a) 문자 데이타, (b) 비-문자 데이타, (c) 오인식 예

TPI는 [0..1]사이의 값을 지니며, 각 값은 입력 영상 내의 해당하는 화소의 문자 여부를 나타낸다.

신경망의 학습 단계에서 비-문자 클래스 학습을 위해 부트스트랩 (bootstrap) 방법을 사용한다[17]. 이는 초기에 정해진 비-문자 클래스 데이타를 이용해서 신경망을 학습 시킨 후에, 부분적으로 (partially) 학습된 신경망을 테스트 영상에 테스트하여 오인식 된 데이타로 신경망을 재 학습시키는 방법이다. 그림 2는 이의 예를 보이는 것으로써, (a)는 문자 클래스의 학습 데이타 예, (b)는 비-문자 클래스의 학습 데이타 예, (c)는 흰색 사각형으로 표시된 부트스트랩 과정에 오 인식된 예이다.

2.2 Mean Shift Iterations (MSI)

본 논문에서는 문자 추출을 위해 변형된 MSI를 사용한다. MSI는 특징값의 확률 분포 상의 경사면을 반복 수행에 의해 탐색하는 방법으로, 최고점 (peak, mode)을 통계적 측면에서 효과적으로 찾아주는 방법으로, 초기에 클러스터 분석과 패턴 분류 등에서 많이 사용되었으며 최근에서 영상 분할, 얼굴 추적과 같은 다양한 컴퓨터 비젼 관련 분야에서 사용되고 있다[19].

유클리디언 공간 X상의 데이타 집합을 S, K를 커널 (kernel) 함수라고 하고 소문자 s와 x는 각각의 인스턴스라고 할 때, x에서의 샘플 평균(sample mean)은 다음과 같다.

$$m(x)=\frac{\sum_{s\in S}K(\|s-x\|)s}{\sum_{s\in S}K(\|s-x\|)} \quad (1)$$

이 때 $m(x)-x$를 *mean shift*라고 하고, MSI는 현재의 mean을 새로운 mean $m(x)$로 이동하는 과정을 수렴까지 반복하는 것을 말한다. 위의 커널을 다음과 같은 λ-ball 모양의 평면 커널 (flat kernel)을 사용하면[19]

$$K(x)=\begin{cases}1 & if\ \|x\|\le\lambda\\ 0 & if\ \|x\|>\lambda\end{cases}, \quad (2)$$

평면 커널을 검색창 (search window)으로 구현할 수 있다. 다음은 mean shift 알고리즘에 대한 개괄적인 설명이다.

1. Set up a search window size λ, and initial location x of the search window.
   Repeat Steps 2 and 3 until convergence (mean location moves less than a threshold θ).
2. Based on the mean shift vector, derive a new mean location.
3. Center the search window at the new mean location computed in Step 2.

그림 3 Mean Shift Iterations

MSI는 직관적이며 간단하고 잡음에 상당히 강한 면을 보인다. 이러한 이유로 최근 얼굴 추출 및 인식과 같은 컴퓨터 비젼 분야에서도 많이 사용되고 있다. 그러나 MSI는 검색창의 크기가 고정되어 있기 때문에 크기가 변하는 물체의 검출이나 추적에는 적합하지 않다. 이러한 이유로 Bradski 와 Pisarevsky [20]는 얼굴 추적 분야에서 검색창의 크기를 동적으로 변경하면서 MSI를 수행하는 CAMShift 알고리즘을 사용하였고, Comaniciu와 Ramesh [21]는 얼굴 추적 (face tracking)에 사용되는 MSI 방법을 다수의 시작 노드에서 수행함으로써 얼굴 탐지 (face detection)에 사용하였다.

2.3 MultiCAMShift 알고리즘

본 논문에서는 TPI 상에 위치한 다수의 검색창에서 변형된 CAMShift 알고리즘을 반복 수행함으로써 영상 내의 문자를 추출한다. 시작 단계에서 TPI 상의 각 검색창이 문자 영역을 포함하고 있는지를 결정하며 (text detection), 연속된 일련의 단계에서 2차원 모멘트 (moment) 계산을 통해서 문자 영역의 크기와 위치를 구하고 인접한 화소를 병합하면서 문자 영역을 찾게 된다 (text localization). 매 번의 반복 수행과정에서 검색창의 크기를 문자 영역의 크기에 비례해서 수정하고, 겹치는 노드들을 제거하기 위해 노드 병합 과정을 수행한다. 수렴된 후 각 문자 영역들은 크기와 가로 세로 비율을 이용하여 필터링을 거친다. 본 논문에서는 이를 MultiCAMShift 알고리즘이라 하며, 이는 전체 입력 영상의 탐색 없이 필요한 부분만을 스캔 함으로써 전체적인 수행속도를 개선할 수 있다.

기존의 컴퓨터 비젼 관련 분야에서의 MSI의 응용에서는 주로 색상 정보를 이용하여 대상 물체의 확률 분포를 정의한다 [20, 21]. 그러나 문자 영역은 색상 공간상에서 다양한 분포를 나타내기 때문에, 이러한 색상 정보를 사용하는 것은 쉽지 않다. 따라서 본 논문에서는 MLP를 이용하여 만든 TPI 상에서 MultiCAMShift 알고리즘을 수행한다 (2.1절). 아래는 본 논문에서 사용한 MultiCAMShift 알고리즘이다.

문자 영역의 위치와 크기를 구하기 위해 연산이 간단하고 잡음에 효과적인 모멘트를 사용하였다. 이차원 $p+q$ 차 모멘트는 다음과 같이 기술할 수 있다.

$$Mpq(i)=\sum_x\sum_y x^p y^q TPI(x,y). \quad (3)$$

문자 영역의 중심 좌표 (sample mean location)는 K를 평면 커널을 사용할 때

$$mean_X(i)_t=\frac{M_{10}}{M_{00}},\ meanY(i)_t=\frac{M_{10}}{M_{00}}. \quad (4)$$

로 설정할 수 있고, 구해진 중심 좌표 값에 따라서, 입력 영상에서의 문자 영역을 CAMShift를 이용하여 탐색하는데, mean shift 값이 정해진 임계값 (ε_X, ε_X) 이하일 때까지 반복 수행한다.

■ Set up the initial locations ($mean_X(i)_0$, $mean_y(i)_0$) and sizes ($\lambda_x(i)_0$, $\lambda_y(i)_0$) of search windows Ws on the image depending on the applications.

■ **Do**

For each window $W(i)$

{

- Generate the TPI within $W(i)$ using MLP.
  for all pixel(x, y) in an image, where $\| x\text{-}mean_X(i)_t \| \le \lambda_X(i)_t \|$ $y\text{-}mean_Y$ $(i)_t \| \le \lambda_y(i)_t \|$
  if pixel(x,y) has been convolved by MLP in former iterations, re-use TPI(x,y), else perform convolution using MLP at pixel(x,y).
- Based on the mean shift vector, derive the new location and size of a text region.
- Move the $W(i)$ to the new location and change its size.

}

Merge overlapping nodes.

Increment the iteration number t.

While ($\| mean_X(i)_t - mean_X(i)_{t-1} \| > (_x$ **or** $\| mean_Y(i)_t - mean_Y(i)_{t-1} \| > (_y$)

Filtering localized text regions using size and aspect ratio of the region-bounding rectangles.

그림 4 MultiCAMShift 알고리즘

문자 영역의 폭 *(width)*과 높이 *(height)*는 문자 영역에 대응하는 사각형 영역의 가로, 세로 길이에 해당하는 것으로, 2차원 모멘트로 구성된 행렬의 eigen value를 구함으로써 다음과 같이 표현할 수 있다[23].

$$width(i)_t=\sqrt{2(a+c)+2\sqrt{b^2+(a-c)^2}},$$
$$height(i)_t=\sqrt{2(a+c)-2\sqrt{b^2+(a-c)^2}} \quad (5)$$

where $a=M_{20}/M_{00}-(M_{10}/M_{00})^2$, $b=2(M_{11}/M_{00}-M_{10}M_{01}/M_{00}^2)$, and $c=M_{02}/M_{00}-(M01/M_{00})^2$.

문자 영역의 폭과 높이를 계산한 후에, 각 노드를 새로운 위치로 옮기고, 검색창의 크기를 다음의 식처럼 변경한다.

$$mean_X(i)_{t+1}=mean_X(i)_t,\ mean_Y(i)_{t+1}=mean_Y(i)_t \quad (6)$$
$$\lambda_x(i)_{t+1}=width(i)_t+Inc_x,\ \lambda_y(i)_{t+1}=height(i)_t+Inc_y \quad (7)$$

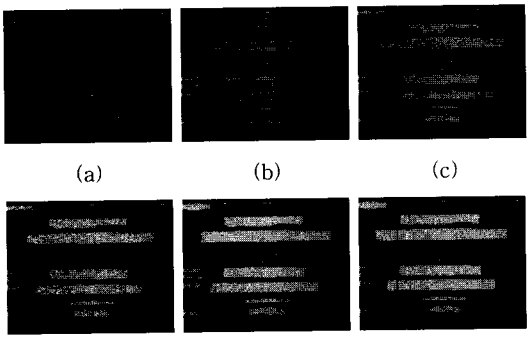
반복 수행 과정에서 불필요한 중복 수행을 하지 않기 위해서 노드들의 겹쳐진 비율이 일정이상이면 두 노드를 병합한다. 이와 같은 병합과정은 더 이상의 병합할 노드가 없을 때까지 수행한다. D_α와 D_β를 각각 두 개의 문자 영역 와 에 의해 점유된 영역이라고 할 때, 두 노드의 겹쳐진 정도는 아래의 식과 같다.

$$\Lambda(a, b) = \max(\text{size}(D_a \cap D_\beta)/\text{size}(D_a \cap D_\beta)/size(D_\beta)), \quad (8)$$

이때 size(λ)는 영역 λ내의 픽셀의 개수이다.

이와 같이 정의하고 경계치 T_O를 0.9로 설정하였을 때, 두 노드 a와 β는 다음의 조건에 의해 병합을 결정하게 된다.

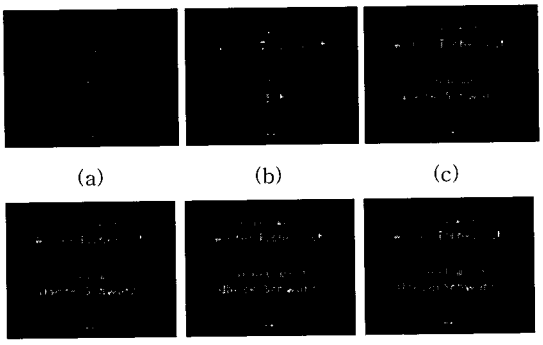
$$\begin{cases} \text{Single text}, & \text{if } T_0 \leq A(\alpha, \beta) \\ \text{Multiple texts}, & \text{otherwise} \end{cases} \quad (9)$$



(a) (b) (c)

(d) (e) (f)

그림 5 그림 1의 실험 영상에 대한 검색창의 변화 : (a) 부터 (f) 순서



(a) (b) (c)

(d) (e) (f)

그림 6 검색창의 변화에 따라 검출된 문자 영역 : (a) 부터 (f) 순서

그림 5는 그림 1의 실험 영상에 대한 Multi CAMShift 알고리즘 수행 중 검색창의 위치와 크기의 변화를 보인다. 그림 5(a)는 MultiCAMShift 알고리즘의 시작 단계의 검색창의 위치와 모양을 나타내며, 그림 5(b)-(f)에서 각 검색창의 위치와 크기가 변하면서 하나의 입력 영상에서의 문자열이 검출되는 과정을 볼 수 있다. 그림 5(f)에서 검은색 부분은 텍스춰 분석이 수행되지 않은 영역들이다. 이러한 MultiCAMShift 방법으로 기존의 텍스춰 기반의 방법들에서 문제점으로 지적되는 입력 영상 전역 탐색에 관한 문제점을 어느 정도 해결할 수 있다. 그림 5에서 회색으로 표시된 부분은 각 반복 수행 중의 *누적된* 그림으로 그림 5(f)는 6번 반복 수행동안 처리된 모든 화소를 나타낸다. 그림 6은 각 반복 수행중의 각 단계에서의 검출된 문자열의 모습이다. 따라서 그림 5와 6은 의미하는 바의 차이로 인하여, 모양이나 크기, 위치가 상이할 수 있다.

3. 실험 및 결과

3.1 데이타베이스

캡춰된 비디오 영상, MoCA 프로젝트의 실험 영상 [1], 스캔 영상, 웹 영상 등 다양한 실험 데이타를 이용하여 제안된 방법을 테스트하였다(표 1). 영상 텍스트(scene text)에 대한 제안한 방법의 유용성을 살펴보기 위해 영상 텍스트를 포함한 비디오 영상 또한 실험 데이타로 사용하였다. MLP의 학습을 위하여 비디오 영상에서 50개의 프레임 (57000개의 학습 패턴)을 사용하고, 나머지 프레임은 테스트에 사용하였다. 오류 검출의 편의를 위하여, 학습 영상과 테스트 영상 내의 모든 문자 영역을 사각형으로 표시하여 각 좌표를 수작업으로 구하였다.

많은 연구 결과에서 문자 추출의 결과를 비교하기 위해, OCR 단계, 문자(character) 단계, 화소(pixel) 단계 등의 여러 단계에서 정확도를 측정하고 있는데[15, 22], OCR 단계에서의 성능 비교는 문자 인식기의 성능에 영향을 받을 수 있기 때문에, 본 연구에서는 화소 단계와

표 1 영상 데이타베이스

Types	Video Frames	MoCA samples	Scanned Images	Web. Images	Scene Texts
Number of images	500	30	50	50	50
Image Sizes (pixels)	320×240	from 355×288 to 384×288	500×700	from 80×40 to 410×328	320×240
Text Sizes (pixels)	from 7 to 47	from 5 to 42	from 6 to 62	from 8 to 50	from 5 to 42

문자 단계의 추출률을 precision과 recall rate를 이용하여 측정하였다.

$$\text{precision}(\%)=\frac{\#\ of\ correctly\ detected\ text\ pixels\ or\ characters}{\#\ of\ text\ pixels\ or\ charaters}\times 100 \quad (10)$$

$$recall(\%)=\frac{\#\ of\ correctly\ detected\ text\ pixels\ or\ characters}{\#\ of\ text\ pixels\ or\ charaters}\times 100 \quad (11)$$

3.2 텍스춰 분석기 MLP 성능 분석

텍스춰 분석 신경망의 성능을 측정하기 위해 화소 단계에서의 precision과 recall rate를 측정하였다. 이는 MultiCAMShift 알고리즘을 수행하지 않고 MLP를 입력 영상의 모든 화소에 적용함을 의미한다. 따라서 표 2의 측정값은 문자 영역을 포함하는 사각형을 구하지 않고 MLP의 출력 결과를 ground truth 데이타와 직접 비교한 결과이다. 이전에 기술한 바와 같이 ground truth 데이타는 각 문자 영역을 포함하는 사각형을 수작업으로 표시하였다. 따라서 이 ground truth 데이타에서의 문자 화소는 실제 문자 화소 뿐만 아니라 문자 화소 주위의 화소까지도 포함하게 된다. 이러한 이유로 recall rate가 precision rate보다 낮은 값을 보이게 된다. MLP의 출력 노드의 값이 경계치 (threshold value) 보다 크면 문자 영역으로 간주하는데, 적절한 경계치를 설정하기 위해 경계치를 [0.3-0.7] 사이에서 달리하면서 실험하였다. 이 실험에서 표 2와 같이 [0.5,0.6] 구간에서는 비슷한 추출률을 보였으며, 본 실험에서는 경계치를 0.5로 사용하였다.

표 2 경계치에 따른 추출률

Threshold value	Precision	Recall
0.3	75.3	63.2
0.4	87.2	79.2
0.5	97.2	82.0
0.6	95.3	81.7
0.7	85.4	79.2

영상 내의 문자 추출에 있어서, 영상 내의 문자의 크기와 MLP의 입력창의 크기는 밀접한 관련을 가진다. 입력창의 크기가 작으면 상대적으로 처리 시간이 단축되는 반면, 부정확한 결과를 낼 수 있다. 따라서 사용 환경에 맞게 적당한 크기의 입력창을 설정하는 것이 중요하다. 2개의 은닉층을 가지고 각 은닉층마다 30개의 은닉 노드들을 가지며, 인텐서티 영상을 입력으로 사용하는 MLP로 다양한 크기의 입력창에 대해 실험하였다.

표 3은 입력창의 크기에 따른 추출률을 나타내며, 표 4는 문자 크기 별 추출률을 나타낸다. 문자 크기는 수작업으로 표시된 정보를 이용하여 각 문자열의 가로, 세로 길이 중 작은 길이를 의미한다. 상대적으로 낮은 추출률을 보이는 작은 문자들은 인덱싱을 위해서는 상대적으로 적은 중요도를 지니며, 이들은 또한 사용된 데이타베이스에서 상대적으로 적은 분포를 보인다.

표 3 입력창의 크기에 따른 추출률

Window Size	Precision	Recall
3×3	75.2	72.2
5×5	77.2	74.8
7×7	76.5	74.8
9×9	81.2	75.2
11×11	89.0	75.0
13×13	97.2	82.0
15×15	94.3	81.3
19×19	90.8	81.9

표 4 문자 크기에 따른 추출률

Text sizes (pixels)	Video frames		Scanned images		Web images	
	Precision	Recall	Precision	Recall	Precision	Recall
~ 9	81.3	62.2	82.8	68.2	81.1	62.4
10 ~ 19	93.4	85.2	94.3	84.3	98.5	89.2
20 ~ 29	97.4	84.2	97.3	81.3	98.2	83.4
30 ~ 39	96.1	81.6	96.3	82.4	97.5	84.2
40 ~	89.3	77.1	92.2	72.1	89.7	78.4

3.3 MultiCAMShift 에서의 파라미터

MultiCAMShift 알고리즘에는 검색창의 초기 위치와 개수, 종료 조건을 위한 경계치, 반복 수행 중의 검색창의 크기 증가량 등을 설정해야 한다. 서론에서 언급한 바와 같이 MultiCAMShift 알고리즘은 이들 파라미터들에 큰 영향을 받지 않는 것을 실험 결과에서 확인할 수 있었다.

초기 검색창의 개수와 위치는 각각의 적용 대상에 따라 다른데, 이들은 각 검색창 사이의 문자열을 놓치지는 않을 정도로 조밀해야하며, 적절한 수행시간을 보장하기 위해서는 가능한 드물게 배치되어야 한다. 실험 데이타에서 320×240 크기의 한글 자막을 가진 비디오 영상을 위해서는 다음과 같은 배치를 가진다.

$(mean_X(i\times 10+j)_0,\ mean_Y(i\times 10+j)_0)=\{(12+24\times(i),$

$50+100\times(j)$), *for* $0\le i<10$ *and* $0\le j<3$).

또한 355×288에서 384×288 크기를 가진 MoCA 데이타는 다음과 같은 배치를 가진다.

($mean_X(i\times 17+j)_0$, $mean_Y(i\times 17+j)_0$)= {($10+16\times(i)$, $25+75\times(j)$), *for* $0\le i<17$ *and* $0\le j<5$}.

이와 같은 배치는 문자열이 수평으로 배치되어 있으며 입력 영상의 1/3 (Korean character) 또는 1/5 (Roman character) 폭을 지닌다는 가정 하에 설정된 것이다. MoCA데이타가 조금 더 조밀한 배치를 지니는데, 이는 실험에 사용한 MoCA 영상 내의 독일어 문자의 크기가 한글 문자보다 상대적으로 작기 때문이다. 웹 영상, 스캔 영상, 영상 텍스트 영상은 각 영상의 크기를 (IW, IH)라고 할 때, IW/10×IH/10 개의 검색창을 균일한 분포로 위치 시켰다.

종료 조건으로는 두 경계치를 $\varepsilon_x=2$과 $\varepsilon_y=1$로 설정하였다. Mean shift의 크기가 가로, 세로 각각 ε_x, ε_y 보다 크거나 같으면 계속 반복 수행한다. 이 경계치는 세밀하게 조정하지 않았으며, 경계치의 값이 추출 결과에는 큰 영향을 미치지 않았다.

각 반복 수행마다 검색창의 크기 증가량을 $Inc_x=20$, $Inc_y=6$로 정하였으며, 수행 시간을 고려했을 때 부적당한 조기 수렴을 피하는 한에서 크게 설정하였다.

그림 7의 평균 mean shift 반복 횟수 그래프에서 좌측의 피크는 비 문자 영역에 해당하며, 대부분의 노드가 4-5번 반복 후에 수렴함을 알 수 있다. 이는 mean shift 알고리즘을 사용함에 따른 부가적인 부하가 극히 적음을 의미한다.

또한 MultiCAMShift 방법을 사용함으로써 얻게 되는 또 다른 장점은 수행시간이 영상 내의 문자영역의 크기에 비례한다는 점이다. 문자 영역이 적을 수로 수행시간이 작은데, 이러한 이유로 본 방법에서는 별도의 문자 감지 단계가 필요하지 않다는 장점이 있다.

3.4 실험 결과

본 절에서는 다양한 종류의 영상에 대한 실험 예를 보인다. 아래의 그림 8은 MoCA 프로젝트에서 사용한 데이타를 이용한 실험 결과 영상이다. 각 그림에서 상단의 (a)는 입력 영상열, (b)는 문자 영역 추출 결과 영상들이다. 그림 8은 정지된 배경 상에서 위로 움직이는 문자열에 대한 테스트 예이고, 그림 9와 그림 10은 움직이는 배경 상에서 위로 움직이는 문자열의 예이다. 모두 문자 영역의 크기, 너비, 높이, 가로 세로 길이 비 등의 정보를 이용하여 필터링을 수행하지 않은 결과이다.

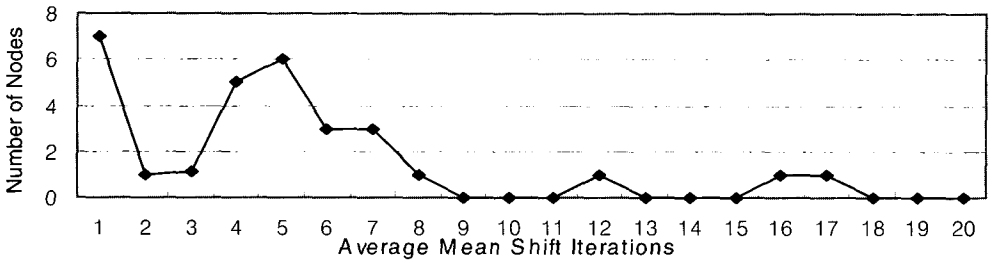


그림 7 프레임 당 평균 mean shift 반복 횟수

(a)

(b)

그림 8 고정 배경의 비디오 열에 대한 문자 추출 예 : 입력 영상 열;(b) 추출 결과 열

(a)

(b)

그림 9 배경이 변하는 비디오 열에 대한 문자 추출 예 : 입력 영상 열;(b) 추출 결과 열

(a)

(b)

그림 10 배경이 변하는 비디오 열에 대한 문자 추출 예 : (a) 입력 영상 열;(b) 추출 결과 열

그림 11은 비디오 영상에 대한 문자 추출 예를, 그리고 그림 12는 웹 영상에 대한 추출 예를 보인다. 몇몇 잘못 추출된 결과들을 볼 수 있으나 (그림 11(a),(c), 그림 12(a),(b),(c)), 이들은 OCR 과정에서 쉽게 걸러질 수 있다. 몇몇 문자는 일부만 잘려서 추출되기도 하였다 (그림 11(b)의 왼쪽 위). 충분히 큰 중요한 문자열은 대부분 성공적으로 추출된 반면 크기가 작고 배경과 밝기가 유사한 문자열들은 추출되지 않았다(그림 12(c)의 오른쪽 아래). 그림 13은 스캐너로 입력 받은 영상에 대한 테스트 결과이다. 각각의 입력 영상에 대해 추출된 결과 영상과 중간 단계에서의 MLP의 출력인 TPI 영상도 함께 보였다. 텍스춰의 질감이 문자 영역과 유사하거나 고주파 영역에서는 잘못된 추출 결과를 보인다 (그림 13(a), (c), (e)). TPI 영상내의 흰 사각형 영역은 최종적으로 검색된 부분을 나타내는데, 문자 영역을 검출하기 위해서 전체 영상을 탐색할 필요 없이 제한된 부분만을 탐색함으로써, 전체 시스템의 수행 시간을 상당히 줄일 수 있었다.

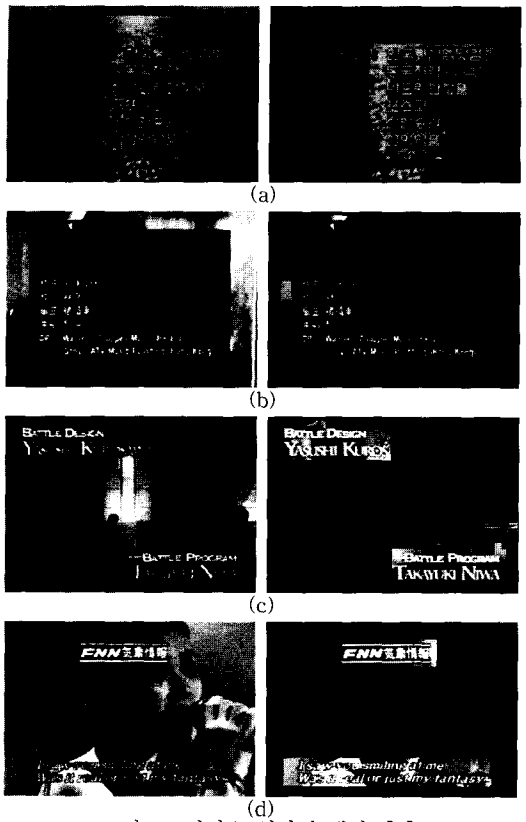


그림 11 비디오 영상에 대한 추출
예 : 입력 영상(왼쪽)과 추출 영상(오른쪽)

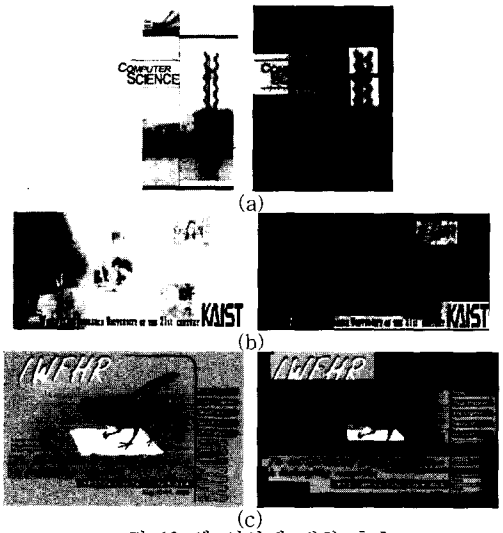


그림 12 웹 영상에 대한 추출
예 : 입력 영상(왼쪽)과 추출 영상(오른쪽)

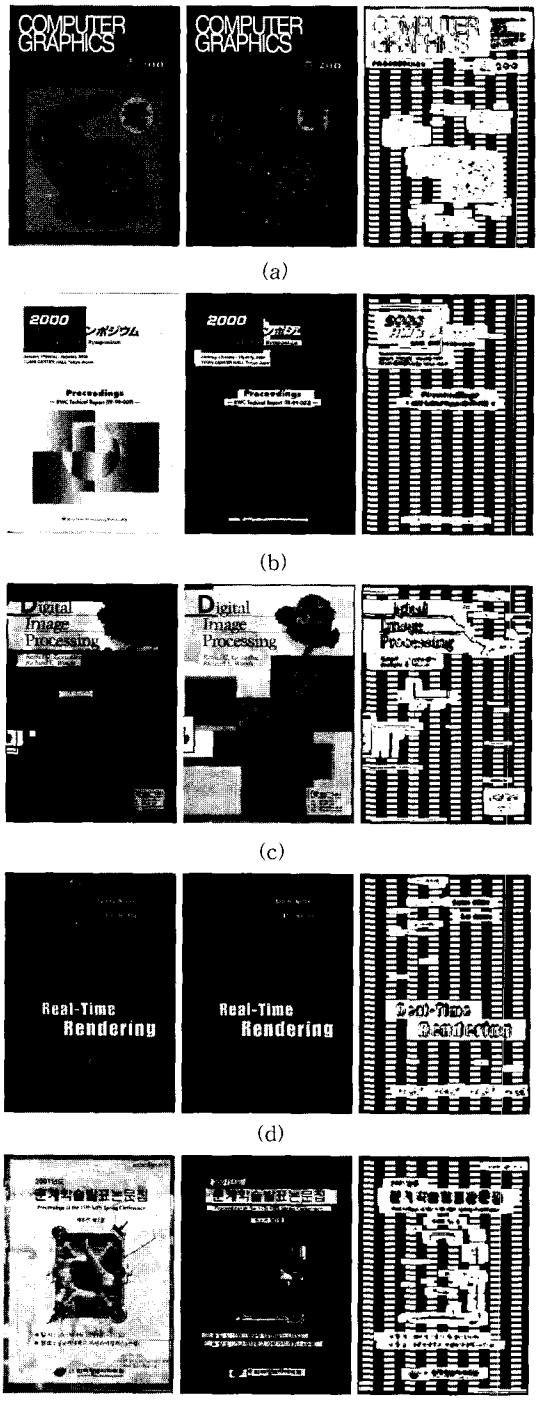


그림 13 스캔 영상에 대한
예 : 입력 영상(왼쪽), 추출 영상(가운데), TPI(오른쪽)

3.5 결과 분석

표 5는 MultiCAMShift 알고리즘을 사용하여 문자 영역을 추출한 후, 화소 단위와 문자 단위로 precision과 recall rates를 수정된 CCM 방법 [6]과 전체 영역을 탐색하는 MLP 방법 [8]과 비교한 결과이다. 표 5에 나타난 것처럼, 수행시간과 성능면에서 기존의 전체 영역 탐색 방법과 연결 성분 방법에 비해 장점을 보인다. 제한한 방법은 CCM과 비슷한 수행시간에 상대적으로 높은 precision과 recall rate를 보이고, 전체영역 탐색 방법에 비해 수행 속도가 4배 정도 향상되었으며 추출률 또한 향상된 것을 알 수 있다.

제안한 방법을 다른 문자 추출 방법과 비교하는 것은, 현재 객관적인 공용 데이타가 없이 많은 연구자가 각자 자신의 데이타로 실험을 하기 때문에 상당히 힘들다. 이에 본 실험에서는 MoCA 사이트에서 다운로드한 데이타를 이용해서 실험하고 실험 결과를 이와 비교하였다. Lienhart [1]는 문자 영역 지정 후 이를 배경과 분리하는 작업을 수행하였다. 따라서 본 실험에서 또한 검출된 문자 영역 내에서 문자 추출을 수행하는데, 현 단계에서는 간단한 double thresholding을 기법을 사용하였다. 그림 14는 비교 그림으로써 Lienhart [1]의 방법에 비해 잘못된 추출이 더 적음을 알 수 있다.

그림 15는 제안한 방법을 영상 텍스트에 적용한 예이다. 영상 텍스트는 다양한 문자열의 배치 형태와 조명 등의 환경으로 인해 캡션 텍스트에 비해 훨씬 어려운 문제로 간주되며 상대적으로 많이 연구되지 않았다. MultiCAMShift를 영상 텍스트에 적용하기위해 다음의 식처럼 모멘트를 이용하여 문자열의 기울기를 추정하였다.

$$\tan 2\theta = \frac{b}{a-c} \tag{12}$$

이 때 θ는 수평축과 문자열의 장축의 법선 벡터가 이루는 각을 의미하고, a, b, c 는 식 5에 나타나있다.

이러한 방법은 Li [3] 등의 논문에서도 텍스춰 분석 후에 문자열의 기울기를 추정하기위해서 사용되었었다.

표 5 Precision과 recall 비교

		MLP + MultiCAMShift	MLP + Histogram analysis	Connected component
시간 (sec.)		0.7	2.7	0.6
화소 단위 (%)	Precision	93.1	87.2	78.8
	Recall	96.4	89.3	84.5
문자 단위 (%)	Precision	94.2	92.4	73.8
	Recall	98.5	94.7	82.3

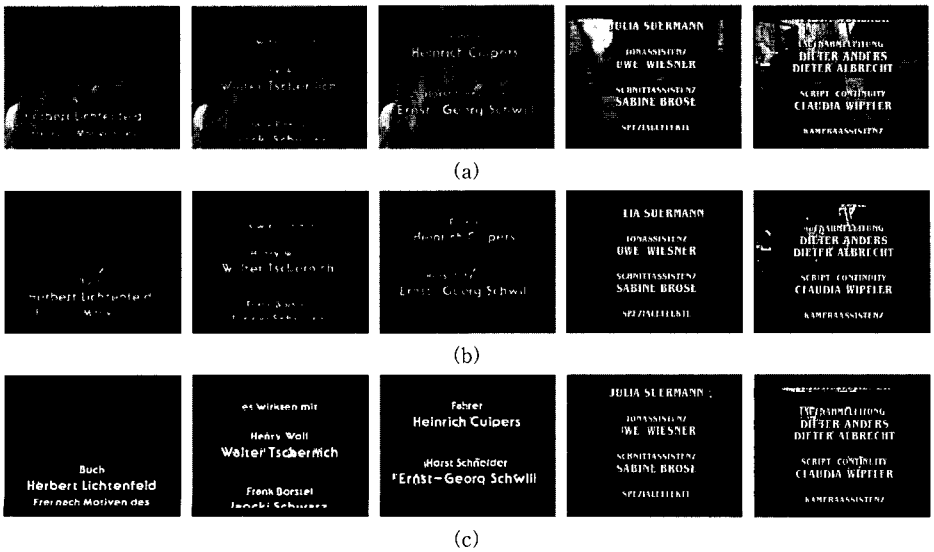


그림 14 문자 추출 예 : (a) 입력 영상, (b) Lienhart's 결과 예 [1], (c) 제안한 방법의 결과 예

그러나 본 논문에서는 텍스춰 분석과 동시에 문자열의 기울기 추정을 수행한다. 이에 따라 종료 조건으로 다음을 추가하였다.

$T_\theta \leq \theta(i)_t - \theta(i)_{t-1}$, T_θ는 경계치로써 0.5°로 설정하였다.

또한 수평 또는 수직으로 배열된 문자열이 기울기 추정에서의 오류로 인해 잘못 추출되는 경우를 막기위해서 기울기가 0°와 90°의 상하 편차가 2.5°보다 작은 경우에는 각각 0°와 90°로 수정하였다. 그림 15에서 좌측의 입력 영상에 대해서 기울기 추정을 고려하지 않은 경우(가운데)와 기울기를 고려한 경우(오른쪽)의 결과를 보인다. 기울기를 고려한 문자 추출 결과가 보다 적은 오류를 보이며 추출된 결과 또한 문자 인식에 유리한 모양을 나타낸다. 기울기를 고려할 때 대부분의 영상 문자열들이 정상적으로 추출되는 반면, 밀집된 캡션 텍스트들의 경우는 그림 16과 같이 잘못된 결과를 보이는 예가 있었다. 이는 상이한 문자열이 기울어진 하나의 문

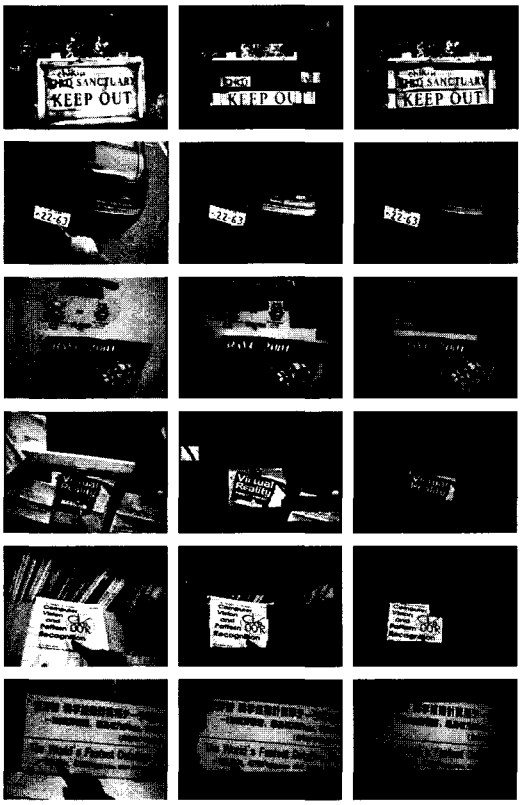


그림 15 영상 문자열에 대한 추출 예 : 입력영상(왼쪽), 기울기 추정 않음(가운데), 기울기 추정(오른쪽)

그림 16 수평 배치된 캡션 텍스트에 미치는 기울기 추정의 영향

자열로 추출되는 경우인데, 이처럼 기울기 추정을 하였을 때 기울어진 문자열의 추출에는 도움이 되지만, 밀집된 수평문자열의 추출에서는 오류를 보이기 때문에, 적용 분야에 따라 문자열의 배치에 관한 사전 정보가 있을 경우는 이를 이용하는 것이 바람직하다.

4. 결 론

본 논문에서는 MultiCAMShift 알고리즘을 이용한 텍스춰 기반의 문자 추출 방법을 제안하고 그 성능을 보였다. MLP를 이용하여 다양한 환경의 영상에 대한 텍스춰 분석기를 별도의 특징값 추출 없이 자동으로 구현하였으며, MultiCAMShift 알고리즘을 문자 검출에 사용함으로써, 영상 전체 영역을 탐색하지 않고도 빠른 시간에 더욱 정확한 문자 영역을 구할 수 있었다.향후의 연구 과제로 압축된 동영상에서의 수행 속도 개선을 위한 문자 추적, 더욱 자연스러운 모양의 문자열의 추출, 문자 인식 시스템과의 연계, 흑백 영상이 아닌 색상값을 이용한 방법에 관한 연구도 필요하다.

참 고 문 헌

[1] Rainer Lienhart and Frank Stuber, "Automatic Text Recognition In Digital Videos," *SPIE-The International Society for Optical Engineering*, pp. 180-188, 1996.

[2] Hae-Kwang Kim, "Efficient Automatic Text Location Method and Content-Based Indexing and Structuring of Video Database," *Journal of visual communication and image representation*, Vol. 7, No. 4, December, pp. 336-344, 1996.

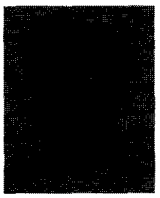
[3] Huiping Li, David Doerman, and Omid Kia, "Automatic Text Detection and Tracking in Digital Video," *IEEE Transactions on Image Processing*, Vol. 9, No. 1, pp.147-156, 2000.

[4] Yu Zhong, Hongjiang Zhang, and Anil K. Jain, "Automatic Caption Localization in Compressed Video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 4, 2000.

[5] Anil. K. Jain, and Bin Yu, "Automatic Text Location in Images and Video Frames," *Pattern Recognition*, Vol. 31, No. 12, pp.2055-2076, 1998.

[6] E.Y. Kim, K.Jung, K.Y.Jeong, and H.J.Kim, "Automatic Text Region Extraction Using Cluster-based Templates," *International Conference on Advances in Pattern Recognition and Digital Techniques*, pp. 418-421, 2000.
[7] Yu Zhong, Kalle Karu, and Anil K. Jain, "Locating Text In Complex Color Images," *Pattern Recognition*, Vol. 28, No. 10, pp. 1523-1535, 1995.
[8] K. Y. Jeong, K. Jung, E. Y. Kim, and H. J. Kim, "Neural Network-based Text Location for News Video Indexing," *Proceedings of International Conference of Image Processing*, 1999.
[9] Yassin M. Y. Hasan and Lina J. Karam, "Morphological Text Extraction from Images," *IEEE Transactions on Image Processing*, Vol. 9, No. 11, pp. 1978-1983, 2000.
[10] S. Messelodi and C. M. Modena, "Automatic Identifacation and Skew Estimation of Text Lines in Real Scene Images," *Pattern Recognition*, Vol. 32, pp. 791-810, 1999.
[11] Victor Wu, Raghavan Manmatha, and Edward M. Riseman, "TextFinder: An Automatic System to Detect and Recognize Text in Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21. No. 11, pp. 1224-1229, 1999.
[12] C. Strouthopoulos and N.Papamarkos, "Text Identification For Document Image Analysis Using a Neural Network." *Image and Vision Computing*, Vol. 16, pp. 879-896, 1998.
[13] Keechul Jung, "Neural Network-based Text Location using Color Texture Discrimination," *PhD. Thesis, Artificial Intelligence Laboratory, Kyungpook National University, Korea*, December 1999.
[14] Huiping Li and David Doermann, "A Video Text Detect System based on Automated Training," *International Conference on Pattern Recognition*, pp.223-226, 2000.
[15] Axel Wernicle and Rainer Lienhart, "On the Segmentation of Text in Videos," *IEEE International Conference on Multimedia and Expo*, Vol. 3, pp. 1511-1514, 2000.
[16] Ullas Gargi, Sameer Antani, and Rangachar Kasturi, "Indexing Text Events in Digital Video Database," *International Conference on Pattern Recognition*, pp. 1481-1483, 1998.
[17] K. K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 39-51, 1998.
[18] Anil K. Jain and Kalle Karu, "Learning Texture Discrimination Masks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 2, pp. 195-205, 1996.
[19] Yizong Cheng, "Mean Shift, Mode Seeking, and Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 8, August, pp.790-799, 1995.
[20] Gary R. Bradski and Vadim Pisarevsky, "Intel's Computer Vision Library: Application in Calibration, Stereo, Segmentation, Tracking, Gesture, Face and Object Recognition," *Proceedings of IEEE Conference of Computer Vision and Pattern Recognition*, Vol. 2, pp. 796-797, 2000.
[21] Dorin Comaniciu and Visvanathan Ramesh, "Robust Detection and Tracking of Human Faces with an Active Camera," *The 3rd IEEE International Workshop on Visual Surveillance*, pp.11-18, 2000.
[22] Sameer Antani, Ullas Gargi, David Crandall, Tarak Gandhi, and Rangachar Kasturi, "Extraction of Text in Video," *Technocal Report*, CSE-99-016, August 30, 1999.
[23] B.K.P. Horn, *Robot Vision*. MIT Press, 1986.

정 기 철
1994년 경북대학교 컴퓨터공학과 공학사. 1996년 경북대학교 컴퓨터공학과 공학석사. 1999년 Intelligent User Interfaces group at DFKI(The German Research Center for Artificial Intelligence, GmbH), Germany, 방문연구원. 2000년 Machine Understanding Division, ElectroTechnical Laboratory in Japan, 방문연구원, 경북대학교 컴퓨터공학과 공학박사. 2001년 PRIP(Pattern Recognition and Image Processing) lab., Michigan State University, 박사후 연구원.



김 광 인
1996년 동서대학교 컴퓨터공학과 공학사. 1998년 경북대학교 컴퓨터공학과 공학석사. 2000년 경북대학교 컴퓨터공학과 공학박사. 2000년 ~ 현재 한국과학기술원 박사후 연구원

한 정 현
1988년 서울대학교 공과대학 컴퓨터공학과 학사. 1991년 미국 씬씨내티대학교 전산학과 석사(Computer Science, University of Cincinnati). 1996년 미국 USC 전산학과 박사(Computer Science, University of Southern California). 1996년 ~ 1997년 미국 상무성 표준기술연구원(National Institute of Standards and Technology). 1997년 ~ 현재 성균관대학교 전기전자 및 컴퓨터공학부 조교수. 관심분야는 Solid Modeling, Computer Graphics, Computer Vision.