1. 개요

영상처리(image processing)란 영상을 대상으로 하는 신호처리(signal processing)의 한 분야로서, 영상의 화질 향상, 소실된 정보의 복원, 데이터의 압축, 영상의 인식 등을 포함한다. 영상처리의 역사는 살펴보면, 20세기 중반까지는 아날로그(analog) 방식의 초보적이고 단순한 의미의 영상처리가 주를 이루었다. 그 이후, 아날로그 TV가 개발되고 아날로그 방송(Broadcasting)이 보편화 되면서 아날로그 영상처리 방법에 많은 발전이 있었다.

본격적인 영상처리 기술은 1960년대 미국에서 위성으로부터 전송받은 달 표면 사진의 화질을 복원시키는 방법에 대한 연구가 디지털 영상처리의 시초가 되었다. 특히, 컴퓨터 기술의 발전에 따라 디지털 신호처리 기술을 활용한 많은 영상처리 기술이 개발되었으며, 2000년대 들어와서 디지털 방송의 시작과 함께 영상처리 기술은 멀티미디어의 핵심기술로 자리매김하게 되었다.

현재 널리 알려진 영상처리의 분야는 영상의 기하학적 변형(geometric transform), 화질개선(enhancement), 복원(restoration), 데이터 압축(compression), 객체인식(object recognition) 등을 들 수 있다. 이들 영상처리 기술을 활용한 구체적인 영상처리 응용은 아래와 같다.

- 얼굴검색 및 인식
  - 영상 안에 존재하는 얼굴의 위치를 찾고, 그 얼굴에 해당하는 사람을 인식 하는 기술

- 내용기반 영상검색
  - 영상 안에 존재하는 사람, 사물, 색상정보 등을 인식하여 원하는 영상을 자동으로 찾아주는 기술

- 컬러영상의 화질개선
  - 전체적으로 어둡거나 밝은 영상을 눈에 보기 좋게 변환하는 화질 개선
  - 디지털카메라, HDTV

- 의료영상분야
  - 정교한 분석이 필요
  - 화질개선, 영상분석

- 문서처리
  - OCR(Optical Character Recognition)
  - 필기체인식

- 공장자동화
  - 공장에서 생산되는 제품의 결함을 검출
  - 실시간처리필수적

- 보안, 감시시스템
  - 침입자 감시
  - 군사적 용도

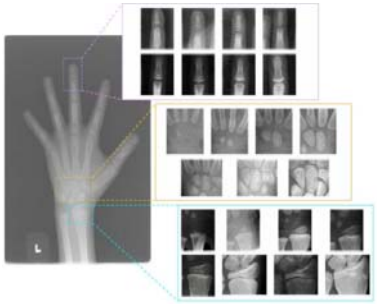
- 자동 동영상 요약
  - 스포츠 동영상의 하이라이트 생성
  - 뉴스, 교육용 비디오의 자동 편집 및 분류

- 증강현실
  - 현실세계의 영상에 컴퓨터가 생성한 그래픽스를 첨가하여 보여주는 가상 현실(virtual reality) 기술 의하나
  - 객체인식(object recognition)과 추적(tracking) 기법 등을 필요

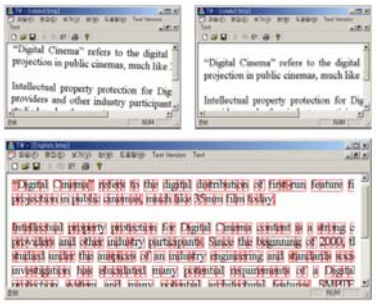
(a) 얼굴검색 및 인식

(b) 내용기반 영상검색

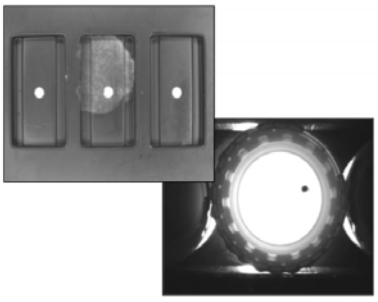
(c) 컬러영상의 화질개선



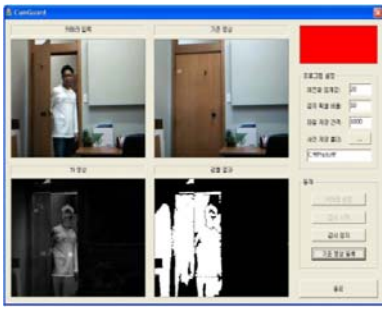
(d) 의료영상분야



(e) 문서처리



(f) 공장자동화



(g) 보안, 감시시스템

(h) 자동 동영상 요약

(i) 증강현실

그림 1.1 영상처리의 응용 사례

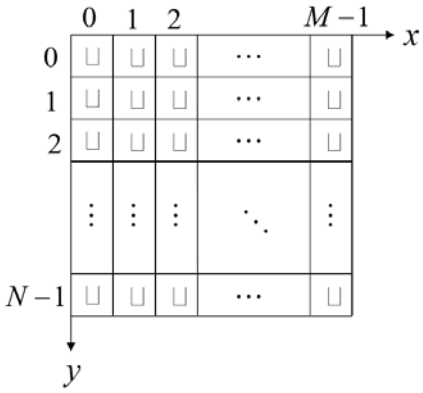
가. 영상신호

일반적으로 영상신호는 크게 흑백영상(gray image)과 컬러영상(color image)으로 구분할 수 있다. 그리고 영상의 기본단위를 화소(畵素) 또는 픽셀(pixel:picture element)이라고 한다. 예를 들어, HDTV에 사용되는 1024×720 영상의 경우, 737,280개의 픽셀로 영상이 구성되게 된다.

흑백영상의 경우 색상정보가 없이 오직 밝기 정보만으로 구성된 영상을 의미하며 보통 한 픽셀 당 8bits 들이 할당하게 된다. 이 경우 $2^8 = 256$개의 밝기 정보를 표현할 수 있게 된다. 또한 컬러영상의 경우, R, G, B의 3가지 색상정보를 가지고 있어서 다양한 색상을 표현 할 수 있다. 컬러영상에서 R, G, B 각각에 8bits을 할당하여 픽셀 당 24 bits을 사용할 경우 트루컬러(truecolor) 영상이라고 하며, $2^{24} = 16,777,216$ 가지의 색상 표현이 가능하게 된다.

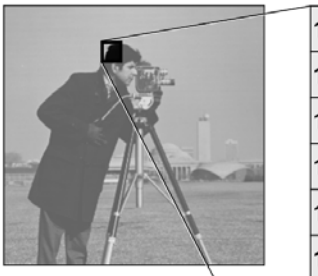
그림 2.1 흑백 그레이스케일(gray scale)영상의 밝기 표현 예

영상신호는 2차원 신호로서 수학적으로는 행렬 모델로 표현할 수 있다. 즉, 각 픽셀의 위치를 2차원 행렬의 행과 열의 위치로 표현할 수 있게 된다. 그림 2.2는 영상신호의 좌표계를 보여주고 있으며, 이를 행렬로 표현한 예를 보여주고 있다. 또한, 그림 2.3은 영상신호의 일부분의 픽셀 값 분포를 보여주고 있다. 그림 2.3은 그레이 스케일의 흑백영상이므로 픽셀 값의 분포가 0에서 255사이에 있는 것을 알 수 있다.



$$\mathbf{A}=\begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,M-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N-1,0} & a_{N-1,1} & \cdots & a_{N-1,M-1} \end{bmatrix}$$

그림 2.2 영상신호의 좌표계와 이를 행렬로 표현한 예



187	187	187	194	197	173	77	25	19	19
190	187	190	191	158	37	15	14	20	20
187	182	180	127	32	16	13	16	14	12
184	186	172	100	20	13	15	18	13	18
186	190	187	127	18	14	15	14	12	10
189	192	192	148	16	15	11	10	10	9
192	195	181	37	13	10	10	10	10	10
189	194	54	14	11	10	10	10	9	8
189	194	19	16	11	11	10	10	9	9
192	88	12	11	11	10	10	10	9	9

그림 2.2 그레이 스케일 흑백영상에서 픽셀 값 분포의 예

나. 영상신호 입출력

일반적으로 잘 알려진 영상신호의 파일 포맷(file format)은 JPG, BMP, TIF 등이 있다. BMP와 TIF 형식은 run-length 압축방식을 사용한 무손실(lossless) 영상 압축 규격이다. 특히, TIF 형식은 팩스영상이나 도면영상 등의 이진 영상신호를 대상으로 한다. JPG는 JPEG에서 정한 국제표준 영상압축 규격으로 손실(lossy)과 무손실 모드를 포함한 사용자가 원하는 다양한 품질의 영상을 지원한다.

파이썬을 이용한 영상 입출력 방법은 scipi, pillow 등의 라이브러리에서 제공하는 함수

를 이용하면 되며, 대부분의 영상포맷 입출력을 지원한다. 현재 가장 많이 사용되는 영상처리 라이브러리는 OpenCV라고 알려진 라이브러리 이다. OpenCV에서는 영상처리에 필요한 다양한 관련 함수들을 제공하고 있다. 하지만 이 강의에서는 OpenCV 등에서 제공하는 이미 만들어진 ㄹ이브러를 이용하지 않고, low-level 방식으로 필요한 알고리즘을 직접 개발하는 것을 목표로 한다.

1) 파이썬을 이용한 영상 입출력

가) 파이썬에서 제공하는 영상 입출력 관련 라이브러리 불러오기

```
import numpy as np              #numpy 라이브러리
#scipy misc에서는 영상을 읽고 쓰는데 필요한 여러 함수를 제공한다.
from scipy import signal, misc
import matplotlib.pyplot as plt #그래픽 관련 라이브러리
from scipy import ndimage     #이미지 처리에 필요한 배열생성 지원
```

나) 이미지 읽기 함수

```
#Image Read
lena = misc.imread('lena_256.bmp')
row, col = lena.shape  #읽어들인 영상의 행과 열 크기를 알아낸다.
```

다) 이미지 화면에 출력

```
plt.imshow(lena)
plt.gray()
plt.show()
```

라) 이미지 파일로 출력

```
misc.imsave('lena_copy.bmp', lena)
```

< Example 1-1 > 영상신호 입출력

```
import numpy as np
from scipy import signal, misc
import matplotlib.pyplot as plt
from scipy import ndimage

lena = misc.imread('lena_256.bmp')    #Image Read
row, col = lena.shape                 #Get Image size
plt.imshow(lena)                      #화면에 영상 출력
plt.gray()                            #입력 영상이 컬러가 아니므로 흑백으로 지정
plt.show()                            #영상이 사라지지 않고 유지

misc.imsave('lena_copy.bmp', lena)    # 다른이름으로 파일 저장
```

2) 영상출력 시 자주 사용되는 그래픽 관련 함수들

가) subplot()함수

화면을 행과 열로 분리하여 여러 개의 그림 출력

Ex) plt.subplot(2,2,1) #2행 2열의 4개의 그림 중 첫 번째
    plt.subplot(2,2,3) #2행 2열의 4개의 그림 중 세 번째

나) 그림의 설명에 사용되는 함수

```
title()                #그림 제목
axis()                 #x-y 축 ON/OFF
xlabel(), ylabel()     #각각 x축과 y축의 축 설명함수
grid()                 #그림에 grid를 추가하는 함수
```

Ex) plt.title('Lena Image')
    plt.axis('off')

<실습 1-1>

1. Ex1-1 프로그램을 수정하여 "lena256.bmp", "lena_copy" 영상을 동시에 가로로 나열하여 출력하시오. 제목을 붙이고 axis의 ON/OFF를 선택하시오.
2. 인터넷 검색 등을 통해 6장의 사진을 찾아 이들 6장의 영상을 동시에 출력하시오.