경계선 탐지 (Edge Detection)

http://www.harrisgeospatial.com

Intro

이미지 처리에 있어서 경계선(Edge)이란 개념은 화소(Pixel)의 값이 급격하게 또는 불연속적으로 변화하는 부분을 뜻합니다. 즉 화소값이 확연하게 차이가 나는 영역을 식별하여 서로 구분 또는 분할해주는 경계선을 탐색하는 작업이라고 보면 됩니다. IDL에서는 이러한 **경계선 탐지 기능을 수행하는 관련 기능 함수들**이 여러 종류가 내장되어 있습니다. 예제를 통하여 주요 관련 함수들의 기능을 각각 소개해보고자 합니다.

예제 데이터 준비

예제로 사용할 이미지 데이터는 JPG 파일로부터 불러오고자 합니다. 이 파일은 아래 링크를 통하여 받으실 수 있습니다.

다운로드 링크

이 파일에는 원래 RGB 트루컬러(True Color) 이미지가 수록되어 있지만, READ_JPEG 명령으로 이 파일로부터 데이터를 읽어오는 과정에서 **그레이스케일(Grayscale)의 단색 이미지로 변환**합니다. 그래야만 이후의 각종 처리 과정에서 사용할 수 있기 때문입니다. 이 과정은 다음과 같습니다.

```
file = 'orlando_l3harris.jpg'
READ_JPEG, file, img, /GRAYSCALE
HELP, img
sz = SIZE(img, /DIM)
```

이제 원본 이미지 데이터를 다음과 같이 표출합니다. 그 모습은 다음 그림과 같습니다.

```
win0 = WINDOW(DIM=sz, /NO_TOOLBAR, $
  WINDOW_TITLE='Original')
im0 = IMAGE(img, MARGIN=0, /CURRENT)
```

앞서 언급했듯이 원본은 RGB 트루컬러 이미지이지만, 의도적으로 단색광 이미지로 읽어들였기 때문에 이와 같이 흑백 계열의 색상으로 표출됩니다.

* 예제 이미지는 IDL/ENVI 제작사의 모기업인 L3Harris사의 배너가 NBA 농구 Orlando Magic 팀의 홈경기장에 뜬 모습을 캡쳐한 것입니다.

ROBERTS, SOBEL, PREWITT 함수

IDL에서 지원되는 경계선 탐지용 함수들 중 ROBERTS, SOBEL, PREWITT 함수들부터 소개합니다. 가장 먼저 ROBERTS 함수부터 보면 사용 문법은 매우 간단합니다. 다음과 같이 대상 이미지 배열을 인수로 부여하기만 하면 됩니다.

```
img_edge = ROBERTS(img)
HELP, img_edge
PRINT, MIN(img_edge), MAX(img_edge)
win = WINDOW(DIM=sz, /NO_TOOLBAR, $
  WINDOW_TITLE='Result')
im = IMAGE(img_edge, MARGIN=0, $
  /CURRENT)
```

여기서는 ROBERTS 함수로 처리된 결과를 img_edge라는 배열로 얻고, 이 결과 배열에 관한 몇가지 기본 정보들을 출력한 다음, 그래픽 창을 띄워 표출하였습니다. 그 모습은 다음과 같습니다.

결과 이미지를 보면 경계선(Edge)에 해당되는 부분들만 높은 화소값을 띄도록 처리된 것을 알 수 있습니다. 참고로 ROBERTS 함수의 이름은 이 기법의 원래 명칭으로서 실제로 이 기법을 고안한 사람의 이름에서 따온 것입니다. 이후에 소개될 다른 경계선 탐지 함수들 역시 이러한 경우가 많습니다.

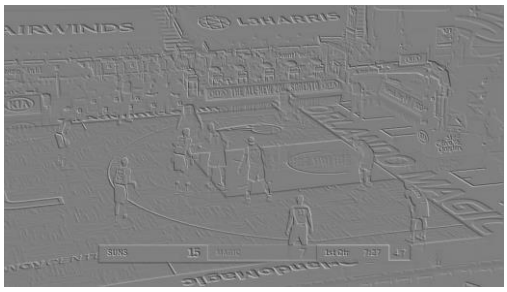
그리고 지면 관계상 각 기법에 관한 세세한 설명은 생략합니다. 그러한 내용은 IDL 도움말을 통해서 살펴보시기 바랍니다. 그리고 SOBEL 및 PREWITT 함수의 경우도 ROBERTS 함수와 비슷한 느낌의 결과를 산출합니다. 위의 예제 코드에서 ROBERTS 함수가 사용된 부분만 다음과 같이 바꿔주면 됩니다. 그 결과는 다음 그림과 같습니다.

```
img_edge = PREWITT(img)
```

SHIFT_DIFF, EMBOSS 함수

이번에는 SHIFT_DIFF 및 EMBOSS 함수를 살펴봅시다. 먼저 다음과 같이 SHIFT_DIFF 함수를 사용하여 처리된 결과를 얻으면 됩니다. 그 모습은 다음 그림과 같습니다.

```
img_edge = SHIFT_DIFF(img)
```

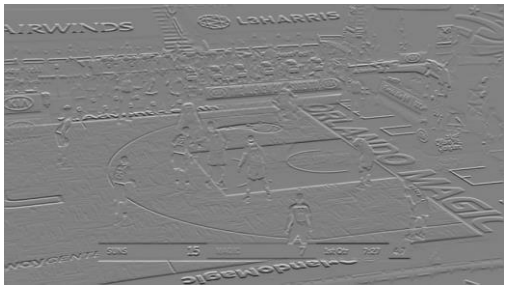


이 모습을 보면 약간 양각 또는 음각 처리된 듯한 느낌이 들 수도 있습니다. 그리고 EMBOSS 함수 역시 비슷한 효과를 나타냅니다. 다음과 같이 EMBOSS 함수를 사용하여 처리 결과를 얻어보면, 그 모습은 다음 그림과 같습니다.

```
img_edge = EMBOSS(img)
```

SHIFT_DIFF 함수에서는 Shift Difference 기법 그리고 EMBOSS 함수에서는 Embossing 기법이 적용됩니다. 두 기법 모두 특정 방향에 광원(Light Source)이 있다고 가정할 때 그 빛에 따라 명암이 구분되는 정도를 결과 이미지의 화소값으로 반영합니다. 광원의 위치는 조절이 가능한데, SHIFT_DIFF 함수에서는 DIRECTION 키워드로 그리고 EMBOSS 함수에서는 AZIMUTH 키워드로 조정할 수 있습니다. 만약 다음과 같이 EMBOSS 함수에서 AZIMUTH 키워드의 값을 90으로 설정할 경우에는 그 결과는 다음 그림과 같습니다.

```
img_edge = EMBOSS(img, AZIMUTH=90)
```



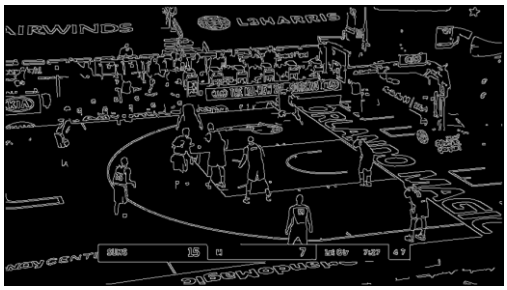
CANNY 함수

CANNY 함수에서는 Canny 알고리즘이 사용되는데, 경계선 탐지 분야에서 꽤 널리 사용되고 있고 효과도 좋은 기법으로서 1986년 John F. Canny에 의하여 고안되었습니다. 이 기법에서는 내부적으로 잡음(Noise) 제거, 경사도(Gradient) 계산, 경계선 방향 판별, Hysteresis 등의 과정을 거치게 됩니다. 함수의 사용 문법은 다음과 같이 매우 간단합니다.

```
img_edge = CANNY(img)
```

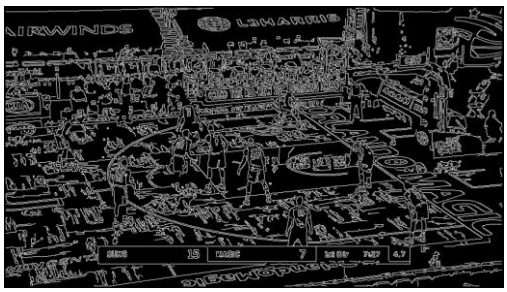
다만 CANNY 함수의 결과는 0과 1 값으로만 구성된 바이트형 배열로 산출되기 때문에, IMAGE 함수로 표출할 때 다음과 같이 MAX_VALUE 키워드를 함께 사용하는 것이 필요함을 유의해야 합니다. 결과가 표출된 모습은 다음과 같습니다.

```
im = IMAGE(img_edge, MAX_VALUE=1, $
   MARGIN=0, /CURRENT)
```



실제로 Canny 기법은 판별 과정에서 수평 및 수직 방향들을 모두 고려한다는 특성이 있으며, 결과 이미지에서 이를 느낄 수 있습니다. 그리고 CANNY 함수를 사용할 때에는 HIGH, LOW 키워드들을 추가적으로 사용할 수도 있습니다. 이것은 유효한 화소값 범위를 지정해주는 역할을 하는데, HIGH와 LOW의 디폴트 값은 각각 0.8과 0.4입니다. 만약 다음과 같이 0.6과 0.2로 설정해주면 그 결과는 다음 그림과 같습니다.

```
img_edge = CANNY(img, $
  HIGH=0.6, LOW=0.2)
```



즉 화소값 유효 범위를 조금 낮추면 더 자잘한 형체들까지 결과에 반영되는 효과를 얻게 됩니다. 이 키워드들의 값을 높이면 그 반대의 효과를 얻게 될 것입니다.

IDL에서는 앞서 소개한 함수들 외에도 EDGE_DOG, LAPLACIAN 등과 같은 경계선 탐지 기능 함수들도 지원되므로 함께 참조해보시면 좋을 것 같습니다.