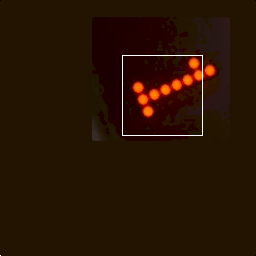
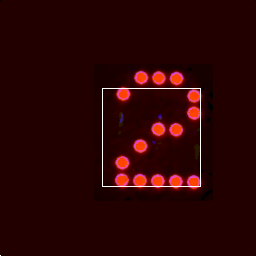
컴퓨터 비전 개론 과제#2

201911050 도담록

코드 설명

먼저 프로그램 안에서 구현한(수업 때 사용하지 않은) declear 함수를 이용하여, 밝기가 20 이하인 값들을 지워 노이즈를 약간 제거했다. 따라서, 밝기가 20 이하로 아주 어두운 경우는 측정이 불가능할 가능성이 있다. Declear 함수는 밝기가 일정 값 이하인 값들을 전부 0으로 만든다.

자연, 다른이(가) 표시된 사진

자동 생성된 설명그 후 median filter를 통해 noise를 제거하고, min-max normalization을 통해 밝은 값을 255에 가깝게 맞추고, declear 함수를 다시 이용하여 밝기가 235 이하인 값들을 제거해 엘리베이터의 숫자 값만 보이게 하였다. 일반적으로 불이 들어온 부분이 가장 밝은 부분이기 때문에, 불이 들어온 부분은 255에 가까운 값을 갖는다.

이때, 사라지지 않은 픽셀들을 “유효 픽셀”이라고 하겠다. 먼저 전체 이미지의 유효 픽셀 수를 센다. 그 뒤에, 정사각형의 사이즈를6, 8씩 늘려가며, 전체 유효 픽셀의 80% 이상을 포함하는 “가장 작은” 정사각형을 찾아내었다. 이때, 같은 사이즈의 정사각형 중 가장 많은 유효 픽셀을 포함하는 값을 포함하였다. 이렇게 될 때 생길 수 있는 문제로, 포함하는 유효 픽셀의 수가 같은 크기의 정사각형들이 있을 경우 선택의 문제가 있다. 좀 더 숫자들을 가운데에 몰아넣기 위해, 처음 나오는 위치에 적당한 값을 더해 주는 방식을 채택하였다.

1과 2-3을 구분할 때는 정사각형의 변의 사이즈를 6씩 늘려가며, 정사각형을 3\*3으로 분리해 정가운데 유효 픽셀 수의 비율을 통해 1과 2-3을 구분하였다. 1의 경우는 유효 값의 비율이 0.2를 넘었고, 2와 3은 그렇지 못하기 때문에 서로 구별할 수 있다. 하지만 숫자가 기울어져 있을 때는 완벽하게는 구분하지는 못하는 모습을 보였다. 크기가 달라지기 때문에 원을 사용하기 힘들어 정사각형 창을 사용했는데, 이렇게 사용하면 각도에 따라서 값이 달라질 수 있다. 하지만, 이 데이터에서는 2,3으로부터 1은 정확하게 분리해내는 모습을 보였다.

그림 1\_1 찾기

2와 3을 구분할 때는 사이즈를 8씩 늘려가며, 정사각형의 가로, 세로 3/8~5/8 부분의 유효 값의 비율을 통해 2와 3을 구분하였다. 3의 경우에는 가운데 부분이 양쪽으로 연결된 2와는 달리 가운데 부분이 오른쪽 한쪽으로만 이어져 있기 때문에 적은 값이 나오지만, 2-D의 경우에만 오류가 나왔다.

결과

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | | 2 | | 3 | |
| A | 1 | A | 2 | A | 3 |
| B | 1 | B | 2 | B | 3 |
| c | 1 | C | 2 | C | 3 |
| d | 1 | D | 3 | D | 3 |

2-D의 경우에만 제대로 값이 측정되지 않았다. 이유는 2-D의 경우에는 2의 오른쪽 위 변 불이 켜지지 않은 상태라, 2-3 구별이 불가능하기 때문이다.

토론

Size 혹은 각도 등이 정해져 있지 않은 환경이기 때문에, 1, 2, 3의 모양을 이용하여 Classification하는 것은 모든 각도, 사이즈의 1,2,3 모델을 필요로 하기 때문에 시간을 너무 갖는다. 따라서, 그림들이 갖는 특성들만으로 Classification을 해야 하는데, 2-D의 경우에는 우측이 끊겨 실제로도 2라고 보기 힘든 정도이기 때문에 Classification을 실패하였다.

코딩을 할 때 값들을 조금씩 바꾸어 가며 시도했는데, 이것이 기계학습의 과정과 비슷하다고 느꼈다. 기계학습의 경우도 PCA를 통해서 특징을 줄이는데, 여기에서도 1,2,3의 2가지 특성만을 이용하여 Classification을 하였다. 조금 더 supervised된 형태의 기계학습을 하는 느낌이었다.

위치와 크기를 특정한 후 그 중간에서 값들의 분산을 측정하여 2와 3을 classifying하려고 했으나, 위치가 정확한 중간을 잡지 못하는 경우가 있어 2와 3을 분리해내지 못했다. 2와 3을 정확하게 중심을 맞추어서 잡을 수 있다면 classification할 수 있을 것이다.

그 방법으로는 위에 사용한 분산을 이용해 분산을 최소화하는 값을 찾는 방법이 있겠지만, 이미 위치와 크기를 잡을 때 for 문을 2번 사용해 엄청나게 많은 iteration을 하기에 for문을 또다시 사용해서 분산을 최소화하는 위치를 잡을 수 없었다. 먼저 연결된 값을 찾기 위해서 dilation을 사용할 수도 없었다. 숫자들의 크기가 전부 다르기 때문에, dilation을 하면 작은 값들은 아예 뭉개져 버렸기 때문이다. 따라서 boundary를 측정하기 위해서 fourier transform을 사용해도, 점들 단위로 나올 수밖에 없고 이로써는 원하는 결과를 얻기 힘들었다.

만약 size가 정해져 있다면, u(1)d을 통해서 2를 판별하고, u(2)와 u(3)을 통해서 2와 3을 판별할 수 있을 것이다.