**2022 – 2 휴먼 인터페이스 미디어**

**과제 1 – 같은 모양 찾기**



Cross - Correlation Measuring

**20173709 소프트웨어학부**

**나원후**

**Ⅰ. 서론**

지난 수업 시간 동안 같은 객체를 찾는 방법에 대해서 배웠다. 가장 큰 분류로는 Convolution을 통한 방법과 Cross - correlation을 통한 방법이 있다. 본 과제 수행물에서는 이미지에서 패치를 직접 crop하여, 그 패치와 “같은 모양 찾기” 를 수행한다.

Shape

Description automatically generated

그림 1. “같은 모양 찾기”를 수행할 이미지

A picture containing text, clipart

Description automatically generated A picture containing clipart, pea

Description automatically generated A close-up of a hat

Description automatically generated with medium confidence  A close-up of a vegetable

Description automatically generated with medium confidence

(a) (b) (c) (d) (e)

그림 2. 이미지에 존재하는 객체

**Ⅱ. 이론적 배경**

2.1 Convolution

Convolution (합성곱) 은 이미지 프로세싱과 인공신경망 모델링에서 많이 쓰이는 개념이다. 합성곱은 하나의 함수와 또 다른 함수를 반전 이동한 값을 곱한 다음, 구간에 대해 적분하여 새로운 함수를 구하는 수학 연산자이다. 합성곱을 수행할 때 한 함수를 반전시킨다는 점에서 교환법칙이 가능해진다.

(1)

(2)

(2)

(1)의 𝜏에t - 𝜏 를 넣으면 (2)의 형태가 된다. (2)의 형태는 결국 (3)의 형태와 같다. 즉, 교환 법칙이 성립한다. Convolution은 반전된 함수와 원본 함수의 Cross-correlation이라 할 수 있다.

2.2 Cross-correlation

Cross-correlation (교차 상관)은 신호처리에서 주로 쓰이는 개념이다. 교차 상관은 두 series의 유사성의 척도이다.

(4)

이는 Convolution과는 반대로 하나의 함수를 반전하지 않고 두 함수 모두 그대로 사용한다. 이미지를 반전시키지 않기 때문에 정렬된 이미지 안에서 이미지 유사도를 찾는데 쓴다.

2.3 cv2

cv2 (open cv)는 imread를 통해 이미지를 읽어오며 3채널 numpy행렬로 읽어온다. 이때 각 채널은 RGB순이 아닌 BGR순으로 되어있다. 이미지가 numpy array 형식이기 때문에 image processing연산을 쉽게 구현할 수 있다.

**Ⅲ. 구현 내용**

3.1 Convolution과 Cross-correlation의 차이

이미지 연산에서 Convolution과 Cross-correlation의 차이는 Patch를 함수 f로 보았을 때, 패치를 좌우 반전을 하고 연산을 하는가, 아닌가에 있다. 그림 1을 보았을 때 같은 객체인 경우 모두 같은 모양으로 존재한다. 즉, 만약 Convolution을 적용하여 이미지 유사도를 측정한다면 그림 2 의 (a)의 경우 좌우 반전을 했을 때 더 많이 겹치는 그림 2의 (d) 와 유사도가 더 높게 나올 수 있다. 이렇게 이미지가 정렬되어 있는 상태에서 “같은 그림 찾기”는 Cross-correlation으로 수행되어야 한다.

3.2 Code outline

코드의 구성은 Cross-correlation을 계산하는 correlation.py, 이미지를 정규화하는 normalize.py, 결과를 종합하여 보여주는 show\_output.py, 그리고 모든 함수를 종합하여 사용하는 main.ipynyb로 이루어져 있다.

3.1 correlation.py

이 모듈 안에는 3가지 함수가 구현되어 있다. 각각의 함수가 수행하는 일은 다음과 같다.

- correlation (image, patch)

이미지를 순회하며 patch와 이미지의 Cross-correlation 값을 구한다. 리턴 값은patch와 이미지가 겹치는 영역의 픽셀 값을 element-wise 하게 모두 곱하고 sum한 값들의 2차원 배열이다. overflow를 막기 위해서 데이터 타입을 float64로 형변환하여 값을 저장한다.

- three\_channel\_correlation (image, patch)

이미지는 총 3개의 채널로 이루어져있다. 따라서 correlation값도 총 3가지가 나오며 three\_channel\_correlation은 총합을 반환한다. 본 과제물에서는 이 리턴값을 사용한다.

- convolution\_patch (image)

이미지의 좌우를 반전해 convolution을 계산 가능하도록 한다. Convolution을 계산하고 싶다면 패치를 변환한 후 three\_channel\_correlation 함수에 순회할 이미지와 함께 파라미터로 넣어준다.

3.2 normalize.py

더 좋은 결과를 얻기 위해서는 이미지 전처리, 정규화가 필수적이다. 이 모듈에서는 이미지 전처리에 관한 5가지 함수를 제공한다.

- no\_normalization (image)

이미지의 데이터 타입만 float64로 변경해주고 픽셀에 아무런 연산을 하지 않는다. 이미지의 기본 데이터 타입은 BYTE로, 형변환을 하지 않으면 데이터 손실이 일어날 수 있다.

- normalize\_subtraction (image)

이미지의 모든 픽셀에서 128을 빼준다. 한 픽셀의 값의 분포는 0 ~ 255 이므로 의도적으로 median값을 빼주어 더 좋은 결과를 기대할 수 있다. 이때 파라미터로 들어오는 image 는 numpy array로 – 연산을 하면 모든 픽셀에서 연산이 이루어진다.

- normalize\_pixel (image)

이미지의 모든 픽셀 값을 해당 이미지의 높이 \* 너비만큼 나눈 값을 반환한다.

- normalize\_reverse (image)

이미지의 모든 픽셀을 255에서 빼주어 반전된 이미지를 구한다. Cross-correlation을 구할 때 패치와 이미지가 겹치는 부분의 픽셀들을element-wise하게 곱하여 모두 sum하게 되는데, 이미지가 하얀 부분의 픽셀 값은 RGB모두 255로 항상 값이 높게 나오게 된다. 이를 방지하기 위해서 이미지 반전을 활용할 수 있다.

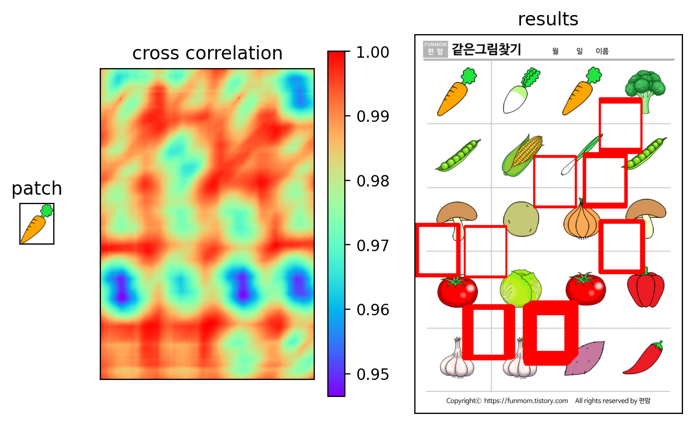
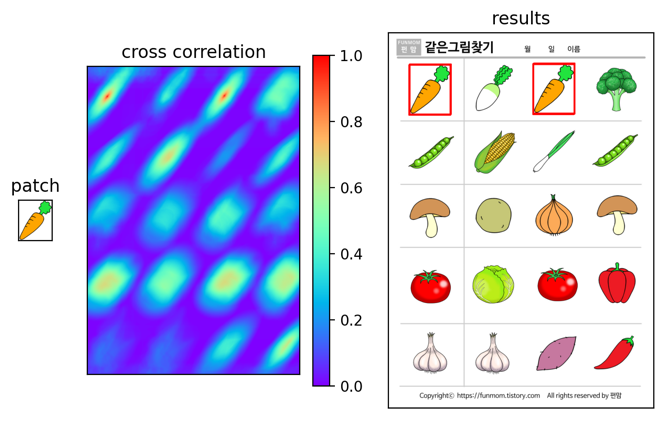
- normalize\_0to1 (image)

이미지의 픽셀 분포를 0에서 1사이의 값으로 정규화한다. 이미지가 음수 값을 가질 것을 대비해 모든 값에서 이미지의 최소값을 더한 뒤, 그 이미지의 최댓값으로 나눈다.

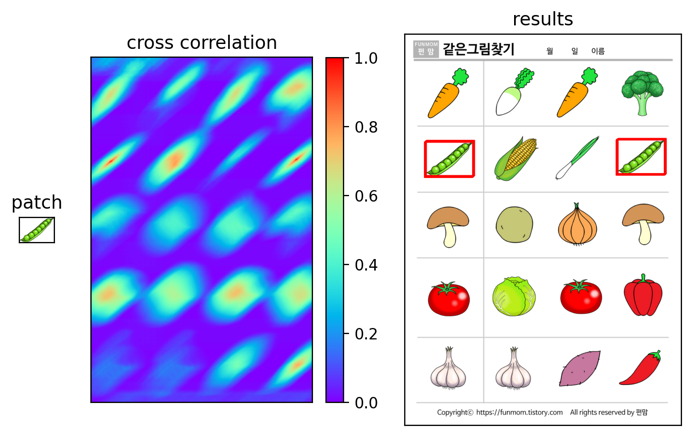
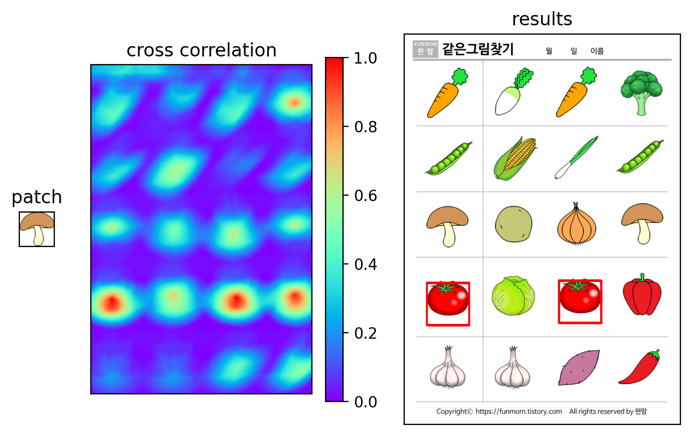
3.3 show\_output.py

해당 모듈에서는 patch, image, output, threshold값을 받아 결과를 출력한다. Output은 correlation값을 계산한 2차원 1채널 numpy array로, 이 값들 중 threshold보다 높은 값에 빨간 박스를 생성하여 “같은 모양 찾기” 를 수행한다.

**Ⅳ. 결과 및 평가**

Patch1 - Cross-correlation with no normalization (a) Patch1 - Cross-correlation with reverse N (b)

Patch2 - Cross-correlation with reverse N (c) Patch3 - Cross-correlation with reverse N (d)

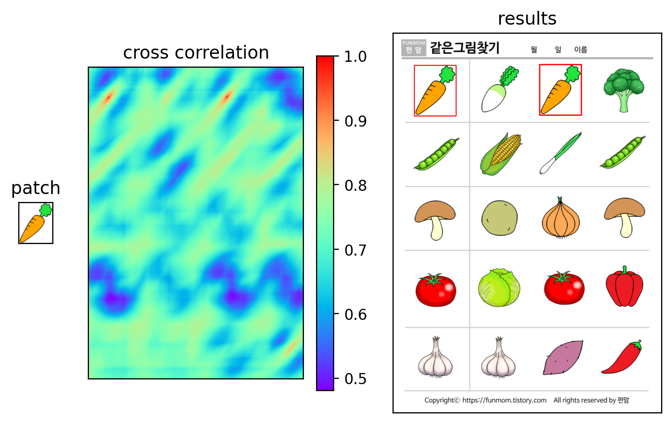
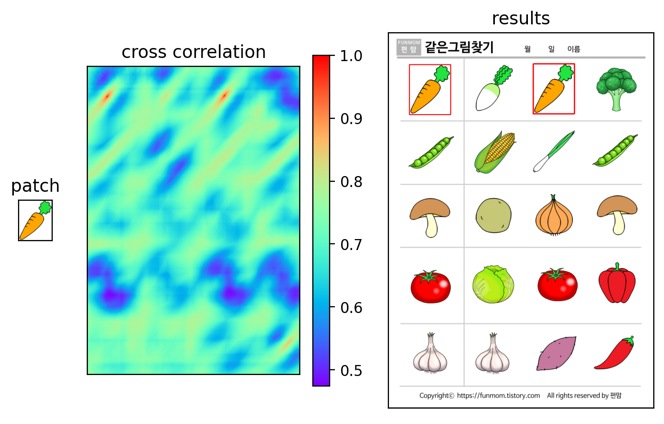
Chart

Description automatically generated Chart

Description automatically generated

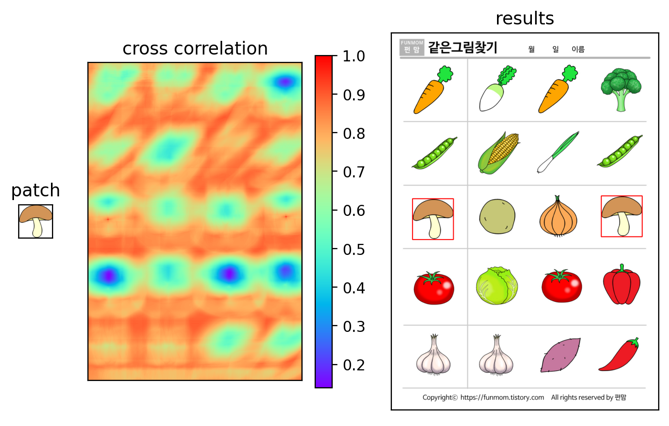
Patch1 - Cross-correlation with pixel N (e) Patch1 - Cross-correlation with subtract N (f)

그림3. Results – 1

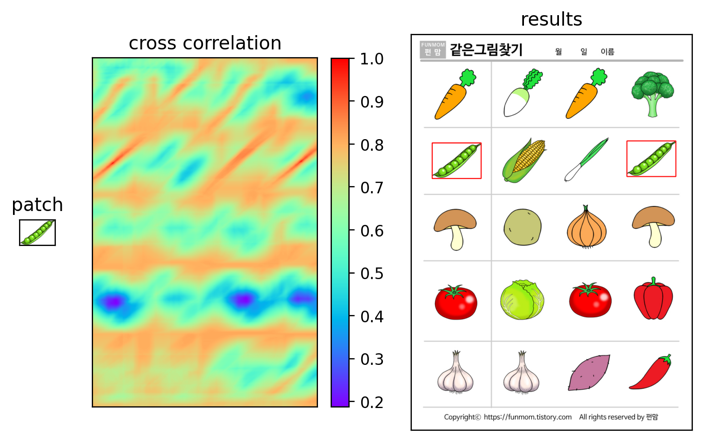
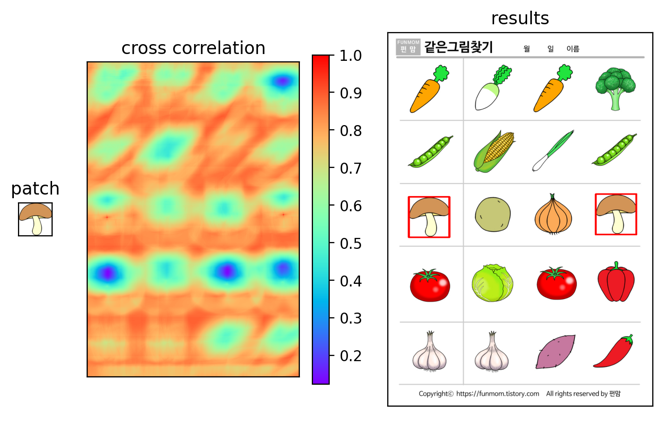
 

Patch2 - Cross-correlation with reverse, subtract N (a) Patch3 - Cross-correlation with pixel, subtract N (b)

Chart

Description automatically generated with low confidence 

Patch1 - Convolution with subtract N (c) Patch3 - Convolution with subtract N (d)

Patch2 – Cross-correlation with subtract N (e) Patch3 – Cross-correlation with subtract N (f)

그림4. Results - 2

4.1 테스트 용어 및 목표

그림 3, 4에서 명시된 **Patch 1, 2, 3은 그림 2의 (a), (b), (c)**에 대응한다. 또한 그림 3, 4에서 명시된 **N이란 Normalization으로, 정규화를 의미**한다. 각각의 정규화는 모두 normalize.py에 있는 함수로 구현되었다. 정규화의 대상은 **도메인 이미지와 패치 이미지**이며, 결과를 표시할 때에는 원본 이미지를 출력하도록 하였다.

이미지 객체 검출에서Cross-correlation 값을 활용하는데, 본 과제에서는 찾고자 하는 객체가 갖는 Cross-correlation 값을 **Objective value (목표 값)**로 칭한다. 원하는 객체 (patch) 가 아니지만 objective value보다 높은 Cross-correlation값을 갖는 영역을 **Upper error domain (상위 오류 영역)** 라 정의하겠다. 이 과제물의 목표는 Upper error domain을 없애고 Objective value가 가장 높은 Cross-correlation value가 되도록 하는 것이다.

4.2 정규화 처리를 하지 않은 경우

이미지에 아무런 전처리를 하지 않고 Cross-correlation을 구했을 때 결과는 “같은 그림 찾기”를 수행하기에는 부적절했다. Cross-correlation연산 수행 후 가장 높은 값은 흰색 부분이 많은 곳에 분포하기 때문이다. 따라서 패치의 객체모양대로 흰 픽셀이 존재하는 곳만 검출되었다. 그림3 (a) 를 보면 그림2 (a)의 객체 모양대로 흰 배경 (255)값이 있는 것을 확인할 수 있다.

4.3 이미지 반전

Patch를 찾을 도메인 이미지가 전체적으로 흰 부분이 많으므로 이미지를 반전 시켰을 때 정확도를 더 높일 것이라 기대할 수 있다. 그림 3 (b), (c) 에서는 실제로 객체를 잘 찾는 모습이나, 그림 3 (d) 에서는 다른 객체를 표시하고 있다. 이는 이미지를 반전 시켜도 그 이미지에서 밝은 , 혹은 픽셀 값이 높은 지역에 Cross-correlation값이 높게 분포하기 때문이다. 따라서 단순히 이미지를 반전 시키는 것만으로는 근본적인 해결책이 될 수 없다.

4.4 픽셀 정규화

픽셀 정규화는 이미지의 모든 값을 해당 이미지의 높이 \* 너비 값으로 나누는 연산이다. 픽셀 정규화는 객체 검출의 성능을 올려주진 않지만, Upper error domain과 Objective value간 차이를 줄여줄 수 있다. 그림 3 (e) 에서 threshold보다 높은 값이 그림 3 (a) 보다 적은 영역에서 드러나지만 여전히 Error domain에 객체를 검출하고 있다.

4.5 픽셀 평균값 정규화

모든 픽셀의 값은 0 ~ 255 사이의 값을 가지며 그 평균값은 약 128이다. Cross-correlation은 모든 픽셀을 element-wise하게 곱하고 더하는 연산이기 때문에 이미지가 유사하지 않을 때 **음수**를 더하여 패널티를 만들어주는 것이 이상적이다. 가장 간단한 방법은 모든 이미지 픽셀에 픽셀 평균값 (128)을 빼주어 **음수 영역**을 만들어주는 것이다. 그림 3의 (f), 그림 4의 (a), (b) 에서 보이듯이, 픽셀 평균값 정규화를 하면 모두 Cross-correlation값이 general하게 나오며, Upper error domain이 없이 성공적으로 객체 검출을 할 수 있다.

4.6 Convolution 적용하기

많은 경우에 이미지 검출에서 일반적으로 convolution을 사용한다. 본 과제에서는 patch를 좌우 반전하여 convolution값을 계산하였다. 하지만 도메인 이미지에서는 모든 객체들이 정렬되어 있으며 아무런 회전, 크기 변화, 반전 등의 transform이 없으므로 Cross-correlation을 사용하는 것이 일반적으로 더 높은 정확도를 보여준다. 그림 4의 (c) 에서 픽셀 평균값 정규화를 적용하였음에도 이미지의 흰 픽셀이 존재하는 곳만 찾아내거나, 패치의 일부분만 겹치는 곳을 검출하고 있다. 하지만 그림 4 (d) 와 같이 패치가 좌우 대칭인 특별한 경우 성공적으로 객체를 검출한다.

**Ⅴ. 논의**

본 과제에서는 Cross-correlation (Convolution의 경우 반전된 패치와 도메인의 Cross-correlation) 의 값이 저장된 output에서 일정 threshold 값 이상에만 bounding box를 표시했다. 이는 Output 별로 구체적으로 값을 설정해야 한다는 점에서 일반적이지 않다. Threshold값을 설정하지 않고 상위 10개의 위치를 찾는 방법처럼 특정 값에 의존하지 않도록 하는 것이 향후 개선 방안이라 할 수 있다.

**Ⅵ. 결론**

6.1 유의미한 정규화

그림 3 (f), 그림 4 (a), (b) 가 나타내듯, 픽셀 평균값 정규화를 제외하고 모두 다른 정규화를 했음에도 불구하고, 모두 동일한 output이 나왔다. 이는 양수 범위 내에서 이미지를 scaling하는 것 보다, 이미지 전체 내에서 픽셀의median 값을 0으로 만들어 음수와 양수 값을 만드는 것이 유의미한 Cross-correlation값을 찾는데 더 중요한 요인임을 알 수 있다.

6.2 가장 성공적인 객체 검출 방법

가장 성공적인 “같은 모양 찾기” 방법은 이미지를 **픽셀 평균값 정규화를 한 후, Cross-correlation을 구하는 것이다.** 그림 3 (f), 그림 4 (e), (f) 이 가장 이상적인 결과이다. 검출할 patch 부분만 값만 0.9 이상으로 값이 높고 나머지는 그 이하로 나오기 때문에 general한 접근 방법이라 할 수 있다.