패치 히스토그램 추출 기반 패치 매칭

강현준

Histogram matching from histogram from patches

HyonjunKang

요 약

패치에서 추출한 히스토그램의 비교를 통하여 유사한 패치를 찾아 매칭한다.

Abstract

In the report, we implemented matching histogram based on similar histograms extruded from patches we selected manually.

Key words

SIFT, histogram, gradient feature vector, euclidean distance, opencv, C++

Ⅰ. 서 론

컴퓨터비전 분야에서 이미지 매칭은 전통적인 컴퓨터 비전의 한 문제로 연구되고 있었다. 여기서 우리는 CAU CV Class의 과제로 자체적으로(manually) 지정한 패치(patch)에서 추출한 다양한 히스토그램을 추출하고 거리(difference)를 비교하여 패치들을 매칭한다. 이를 통해서 본 보고서에서는 이미지 매칭에 대한 개념과 이미지 매칭의 단계 중 가장 첫 번째인 히스토그램에 관해 알아본다.

여기서 사용하는 히스토그램은 3가지로, Gray scale에서의 히스토그램, RGB scale에서의 히스토그램, Gradient distribution에서의 히스토그램을 활용한다. 본 보고서에서는 Opencv, Visual Studio C++에서 추출한 히스토그램을 활용하여 실험을 진행할 것이다

Ⅱ.실험 방법

1) 히스토그램

히스토그램은 표로 되어 있는 도수 분포를 정보 그림으로 나타낸 것이다[1]. 이미지에서 히스토그램은 보통 이미지의 픽셀값들의 도수 분포를 가지고 나타낸다. 본 보고서는 여기서 픽셀값들을 Gray scale, RGB scale, Gradient distribution에서 추출하여 도수 분포를 작성한다.

a) Gray scale

Gray scale에서 우리는 기존 3channel(RGB) 이미지를 Gray scale로 변경한다. 여기서 사용한 방법은 opencv의 자체 내장 함수인 cvtColor을 사용한다. COLOR_BGR2GRAY 변수의 공식은 다음과 같다.

$$\text{RGB[A] to Gray:} \quad Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

그림 1. BGR2GRAY[2]

이 뒤 도수분포는 변환한 Gray scale 값들을 사용한다.

b) RGB scale

RGB scale에서 픽셀값은 3channel 이미지의 각 channel의 픽셀값들을 활용한다. 여기서 더 변환을

거치지는 않고 직접 사용한다. 여기서 우리는 세 channel의 각각의 difference를 비교한다.

c) Gradient distribution(SIFT)

Gradient distribution에서 우리는 SIFT(Scale-Invarient Feature Transform)의 기본 원리를 사용한다. SIFT란 이미지의 크기와 회전에 불변한 특징을 추출하는 알고리즘이다. 여기서 특징이란, 영상에서 중요한 정보를 말하는 것으로 영상에서 중요한 정보를 포함하고 있는 지점을 특징점이라고 한다. 여기서 중요한 정보란, 색의 분포, 물체의 모양 등 여러 가지가 될 수 있다.

SIFT에서는 특징점의 특징을 구하기 위해 우선 대상 좌표 주위의 16*16 구역을 설정한다. 이를 패치라고 한다. 다시 패치 안을 4*4의 영역으로 구분한다. 이후 각 영역의 gradient histogram을 구한다.[3]

Gradient histogram은 원의 구간을 나누어 Gradient의 각도를 각 구간의 비율에 맞게 Gradient의 magnitude를 나누어 도수분포표를 만든다. 여기서 도수분포표를 히스토그램으로 만든 것이 다음과 같다. 이를 일렬로 concatenate 하면 SIFT의 Feature가 된다.

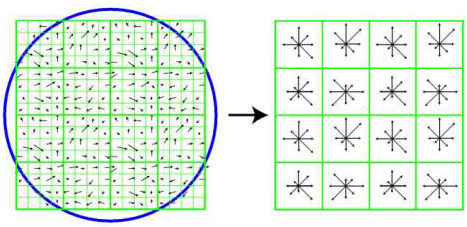


그림 2. SIFT Feature[3]

d) Gradient distribution(HOG)

우리는 두 번째 Gradient Distribution로서 HOG의 원리를 가져온다. HOG의 gradient histogram은 SIFT의 것과 비슷하다. 하지만 패치 크기가 8, 1개의 영역에서 0~180의 각도를 가지고 histogram을 만든다.[4] 또한 뒤의 히스토그램 거리를 측정 할 때 histogram의 bin을 한 칸씩 미뤄 20도씩 돌리는 효과를 얻는다.

2) 히스토그램 거리

히스토그램간의 거리를 정의하기 위해 크게 두 가지, 유클리디안 거리와 해밍 거리 측정에 대해 알아보았다. 이중 SIFT와 같은 크기값이 있는 feature에 대해서는 유클리디안 거리를 사용하는 것이 주된 방법이다 (해밍 거리는 이진 feature들에 주로 사용된다). 우리는 이 실험에서 NORM_L2 즉 L2 거리를 활용한다.

cv2.NORM_L1: $\sum_i abs(a_i - b_i)$

cv2.NORM_L2: $\sum_i (a_i - b_i)^2$

cv2.NORM_L2SQR: $\sqrt{\sum_i (a_i - b_i)^2}$

그림 3. 유클리드 거리

3) 히스토그램 매칭

히스토그램 지정(histogram matching)은 히스토그램 둘의 확률분포를 유사하게 맞추는 방식으로, 본 보고서와는 큰 관련이 없는 기법이다. 하지만 이름만 같을 뿐 본 보고서에서는 두 이미지의 패치들에서 추출한 feature들의 히스토그램을 바탕으로 히스토그램의 거리를 측정하여 최소 거리의 패치를 매칭하는 것을 의미한다.

4) 실험 과정

- 기본 실험

1. 화면 안에 표시하기 위해 축소된 이미지를 표시한다.
2. 화면상에 표시된 이미지에 4개의 패치의 위치를 설정한다 이때 위치는 사전에 설정한다.
3. GRAY, RGB, SIFT, HOG 4가지 히스토그램으로 패치들을 분석한다.

- 랜덤 실험

1. 화면 안에 표시하기 위해 축소된 이미지를 표시한다.
2. 화면상에 표시된 이미지에 4개의 패치의 위치를 설정한다 이때 위치는 클릭으로 설정하여 매 이미지마다 다르게 설정한다, 이때 패치들은 (특징점이 잘 들어나게, 랜덤, 객체 밖)으로 설정한다.
3. GRAY, RGB, SIFT, HOG 4가지 히스토그램으로 패치들을 분석한다.

III. 결 론

1) 결과

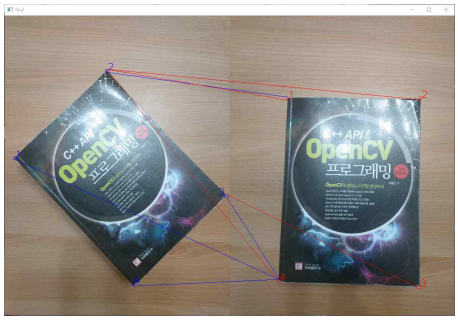
본 보고서에서의 실험은 결과를 내기에 어려운 실험이다. 객관적인 결과를 얻기 위해서는 다양한 이미지에서 다양한 패치들을 분석하여 이를 바탕으로 나뉘하는 것이 정상이다. 이 후 과정은 특별할 것이 없다 .

1. 기본실험

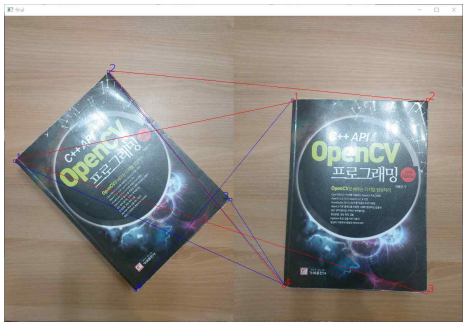
GRAY

RGB

SIFT



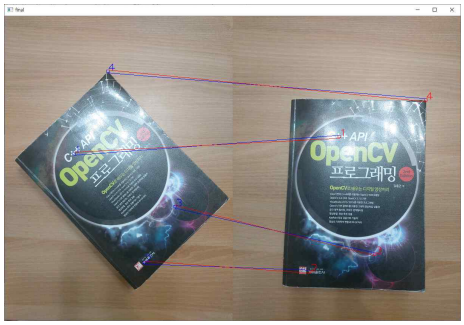
HOG



2. 랜덤실험

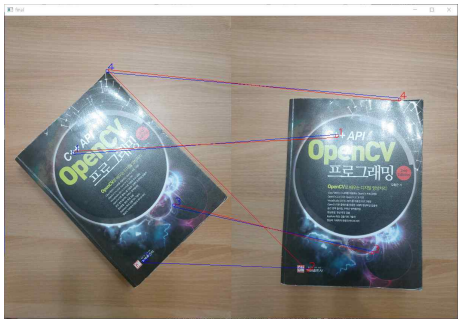
a) 특징점

GRAY

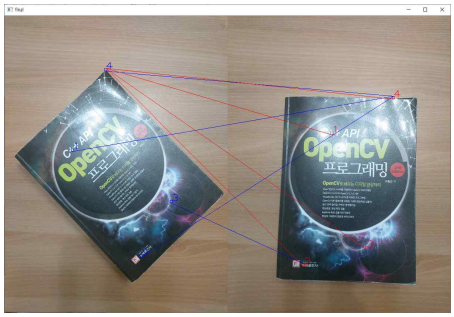


RGB

SIFT



HOG

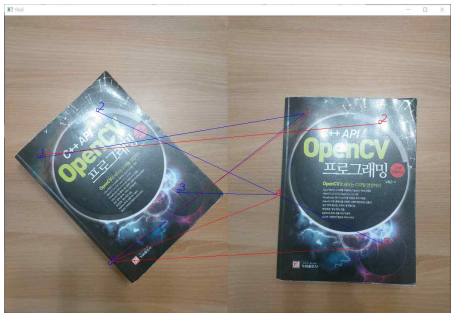


b) 랜덤

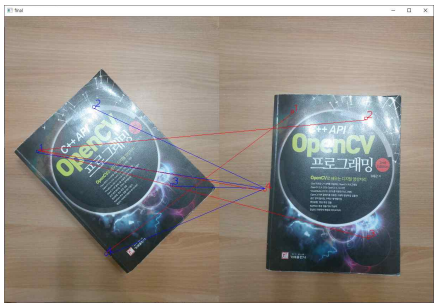
GRAY

RGB

SIFT



HOG



2) 결론

Color histogram은 거의 대부분에서 1~2개의 오차가 있거나 완벽히 맞춘 반면, Gradient histogram은 저조한 성능을 보여주었다. 즉 오히려 Gradient histogram 매칭보다 단순 Color histogram 매칭이 훨씬 좋은 성능을 보여줬다. 이는 Gradient histogram의 사용 방법에 있어서 큰 차이를 보인 것으로 추정된다. Gradient 기반 Feature extractor들은 이미지를 block으로 나누어놓고 block별 feature들을 가져와 유효한 feature들을 정의, 그 뒤에 알맞은 feature들끼리 매칭한다. 하지만 우리가 정의한 패치에 따르면 이것이 유효한 feature인지 알 수가 없다. 따라서 평균적인 성능을 보이는 것으로 추정되는 색 히스토그램이 더 정확도가 높이 나오는 것으로 추정된다.

Ⅳ. 부 록

GitHub 주소:

https://github.com/fender2758/CAU_CV_projects

(* 소스코드와 사용 이미지는 Github에 업로드)

참 고 문 헌

[1] 히스토그램 . (n.d.). https://ko.wikipedia.org/wiki/%ED%9E%88%EC%8A%A4%ED%86%A0%EA%B7%B8%EB%9E%A8.

[2] color_convert_rgb_gray . (n.d.). https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_gray.

[3] Nguyen-Khang Pham, Annie Morin, & Patrick Gros. (n.d.). Boosting of factorial correspondence analysis for image retrieval (pp. 3). n.p.: IEEE.

[4] Histogram_of_oriented_gradients . (n.d.). https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients.