summary

1. 템플릿 매칭: `matchTemplate()`
2. 캐스케이드 분류기와 얼굴 검출: `CascadeClassifier()`, `detectMultiScale()`
3. HOG 앍고리즘과 보행자 검출: `HOGDescriptor()`, `setSVMDetector()`
4. QR 코드 검출: `QRCodeDetector()`, `detectAndDecode()`

1. 템플릿 매칭

1. 개념

- 찾고자하는 부분적인 영상
- 입력영상의 전체 영역에 대해 이동하며 비슷한 위치를 수치적으로 찾아냄 (유사도 / 비유사도 계산)
- 유사도가 가장 높은 곳은 밝게(255) 나타남
- `matchTemplate()`함수 사용

```
#회로도 칩셋 찾기
res = cv.matchTemplate(img, temp1, cv.TM_CCOEFF_NORMED)
res_norm = cv.matchTemplate(res, None, 0, 255, cv.NORM_MINMAX, cv.CV_8U)
```

2. 종류

이름	의미	유사(매칭o)	유사X
TM_SQDIFF	제곱차	0(완벽일치)	큰 양수
TM_CCORR	상관관계	큰양수	작은값
TM_CCOEFF	상관계수(평균밝기로 미리 보정후 상관관계)	큰양수	0에 가까운 양수나 음수
TM_CCOEFF_NORMED,TM_CCORR_NORMED	밝기차이 영향을 줄여주는 정규화 수식	1에 가까운 수	

- TM_SQDIFF: 템플릿이미지 - 원본이미지
  - R값: 작을수록 완벽히 일치된 것
  - R값을 최소화하려면? Maximize부분을 최대화하면 된다..

$$R(x,y) = \sum_{x',y'}\left(T(x',y') - I(x+x',y+y')\right)^2$$

Minimize

$$\left(T(x',y') - I(x+x',y+y')\right)^2$$

$$= \left(T^2(x',y') + I^2(x+x',y+y') - 2T(x',y')\cdot I(x+x',y+y')\right)$$

Maximize

  -

- TM_CCORR
    - 제곱차를 착안함
    - 단점: 원본 이미지 값을 그대로 곱해주해주므로 원본이미지 픽셀이 크면 무조건 R이커지는 문제 발생
    - 해결책: 정규화(nomalize)필요
- TM_CCOEFF_NORMED
    - 장점: 가장 좋은 결과 도출
    - 단점: 연산량이 많아져서 속도가 느림

2. 캐스케이드 분류기와 얼굴 검출

1. 특징

- 유사-하르 필터 집합으로부터 특징정보 추출해 얼굴여부 판단
- 부스팅기반으로 다양한 객체 검출 가능, 속도와 정확도 인정받음

2. 유사-하르필터

- 흑백 사각형이 서로 붙어 있는 형태로 구성된 필터
- 흰색영역: 모두 더함
- 검은색영역: 픽셀값 모두 뺌

3. 비올라-존스algorithm

- 24x24 크기로 정규화. 대략 18만개의 유사-하르필터 생성가능
- adaboost 알고리즘과 적분영상을 이용해 문제 해결함
- 캐스케이드 구조 사용
    - 1단계: 얼굴 검출에 가장 유용한 유사-하르필터 사용함, 얼굴이 아니라고 판단되면 유사-하르필터 계산을 수행하지 않음
    - 2단계: 1단계 통과하면 유사-하르필터 다섯개 사용해 검사함, 얼굴이 아니라고 판단되면 유사-하르필터 검사 안함
    - 3단계: 20개 사용 ...
- `CascadeClassifier(),`detectMultiScale()`함수 사용

```
classifier = cv.CascadeClassifier('haarcascade_frontalface_default.xml')
faces = classifier.detectMultiScale(img)
```

4. 남자애 2명 눈 검출하는 실습: xml 파일 중 검출을 위한 파일을 사용해 얼굴 찾은 후 눈 위치를 찾음

3. HOG 앍고리즘과 보행자 검출

1. HOG알고리즘

- 그래디언트 방향 히스토그램을 의미함
- 그래디언트 크기, 방향 성분 이용하여 서 있는 사람의 특징 벡터를 정의
- 서포트 벡터 머신 알고리즘으로 보행자 위치 검출
- 특징(64x128기준)

- 방향성분: 0~180도
- 크기분할(셀): 8x8
- 히스토그램 구하기: 20도단위로 나누어 총 9개의 빈 구함
- 블록: 인접한 4개 셀을 합친 것, 16x16
- 가로방향 블록: 7개 / 세로방향 블록: 15개 = 총 105개 블록 생성
- 히스토그램 실수 값 개수: 105x 36 = 3780

2. 데랄과 트릭스

- 수천장의 보행자영상, 보행자 아닌 영상에서 HOG 특징 벡터 추출
- 두 특징 벡터를 구분하기 위해(보행자/비보행자) 서포트 벡터 머신 알고리즘 사용
- `HOGDescriptor()`, `setSVMDetector()` 함수 사용

```
hog = cv.HOGDescriptor()
hog.setSVMDetector(cv.HOGDescriptor_getDefaultPeopleDetector())

detected, _ = hog.detectMultiScale(frame)
```

4. QR 코드 검출

1. 특징

- QR코드 내에서 3개의 정사각형 방향을 찾고 투시변환
- `QRCodeDetector()`, `detectAndDecode()`함수 사용

```
detector = cv.QRCodeDetector()
info, points, _ = detector.detectAndDecode(frame) #info: 저장된 문자열,
points: qr코드 영역
```