바코드 영역 검출

김성영교수
금오공과대학교
컴퓨터공학과

개요

- 문제 설명
  - 바코드 영상으로부터 바코드 영역만을 분리하여 검출
  - 분리한 바코드 영역은 결과 영상에 사각형으로 위치 표시하고 검출 위치는 detect.dat 파일에 저장
  - 제시한 정확도 계산 프로그램을 사용하여 정확도 계산
  - 개별 진행

검출 프로그램 실행 방법

- 형식

  python detect.py --dataset *dataset*
  --detectset *result* --detect *detect.dat*

  - --dataset: 검출할 바코드 영상을 포함하는 **폴더**
  - --detectset: 검출 결과 영상을 포함하는 **폴더**
    - 결과 영상에서 검출된 바코드 영역은 색상 사각형으로 표시
  - --detect: 검출한 바코드 위치 저장 **파일**
    - 각 필드는 공백 문자로 구분하며 공백 문자의 개수는 무관
    - 반드시 utf-8 형식으로 저장해야 함
    - 형식: (fileID는 파일 확장자를 제외한 순수 파일명을 사용)

      fileID lefttopx lefttopy rightbottomx rightbottomy

정확도 계산 프로그램 실행 방법

- 형식

  python accuracy.py --reference *ref.dat*
  --detect *detect.dat*

  - --reference(-r): 바코드의 기준 위치를 포함하는 파일
  - --detect(-d): 검출 결과를 포함하는 파일
  - accuracy.dat 생성
    - 개별 영상 파일에 대한 정확도 계산 (F-Measure, IOU)
    - 제공하는 파일의 구조를 참조하여 생성
  - 공통 형식:

    fileID lefttopx lefttopy rightbottomx rightbottomy

정확도 계산 프로그램 실행 화면

```
>> python accuracy.py -r ref.dat -d detect.dat

total number of list:  73
total number of list:  73
Accuracy data in accuracy.dat
================================
average Precision:  0.7
average Recall:  0.87
average F1_Score:  0.75
average IOU_Score:  0.65
```

Dataset 폴더에서 특정 파일만 리스트

- Glob 모듈 사용
  - finds all the pathnames matching a specified pattern
  - results are returned in arbitrary order

```
import glob
dataset = "..\\images"
glob.glob(dataset + "\\*.jpg")
```

```
['..\\images\\ani1.jpg', '..\\images\\ani2.jpg', '..\\images\\ani3.jpg',
'..\\images\\circle.jpg', '..\\images\\coins.jpg', '..\\images\\document.jpg',
'..\\images\\document2.jpg', '..\\images\\gull_color.jpg', '..\\images\\hand.jpg',
'..\\images\\hand2.jpg', '..\\images\\nature.jpg',
'..\\images\\nature_grayscale.jpg', '..\\images\\pumpkin.jpg',
'..\\images\\pumpkin_dim.jpg', '..\\images\\receipt.jpg',
'..\\images\\rectangle.jpg']
```

String.rfind( )

- rfind( )
  - returns the last index where the substring str is found
  - or -1 if no such index exists

```
path = 'D:\\tt\\Dol-Guldur-001.png'
k = path.rfind("\\")
```

```
5
```

```
path = 'D:\\tt\\Dol-Guldur-001.png'
fname = path[path.rfind("\\")+1:]
```

```
Dol-Guldur-001.png
```

파일 이름 구분

- 파일 경로의 목록에서 파일 이름만 분리하여 구분

```
import glob

dataset = "..\\images"
for imagePath in glob.glob(dataset + "\\*.jpg"):
    # extract our unique image ID (i.e. the filename)
    fname = imagePath[imagePath.rfind("\\") + 1:]
    print(imagePath)
    print(fname)
```

```
path: ..\images\ani1.jpg
name: ani1.jpg
path: ..\images\ani2.jpg
name: ani2.jpg
path: ..\images\ani3.jpg
name: ani3.jpg
. . .
path: ..\images\rectangle.jpg
name: rectangle.jpg
```

결과에 대한 제출물 목록

- 결과 보고서
  - 문제 분석, 설계 및 구현 (알고리즘 포함), 실험 결과, 느낀 점 등
    - accuracy.py의 실행 결과 화면 포함
  - 정확도 분석 (원인 분석, 평균 정확도 제시 등)
    - 정확도 85%이상, 60~85%, 60% 미만의 영상 각 5개 이상 제시
    - 각 영상에 대한 중간 처리 결과 제시
  - 모든 영상을 처리하는데 소요 시간 제시
- 소스코드
- 바코드 검출 결과
  - 검출 결과 파일: accuracy.dat, detect.dat
  - 검출 결과 영상 (바코드는 사각형으로 표시하여 구분)

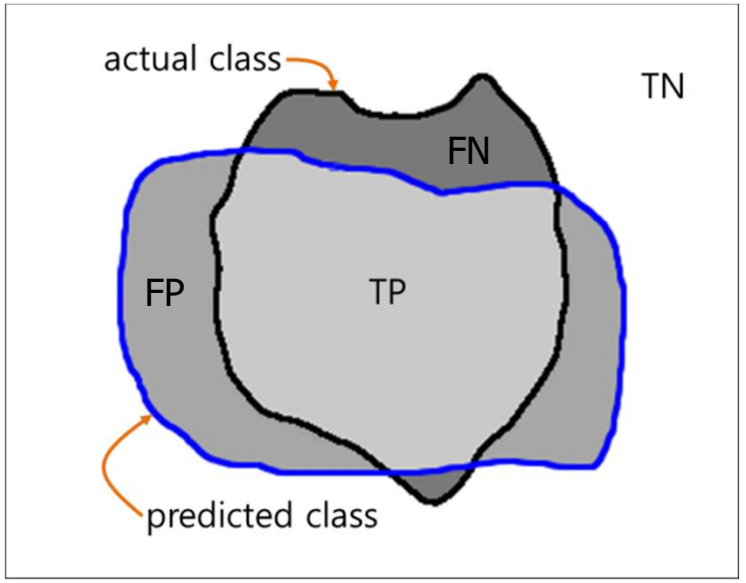
평가 기준

- 정확도: 15점
  - F Measure를 기준으로 사용 (IOU 참조)
  - ~85%: 15점, ~80: 13점, ~75%: 11점, ~70%: 8점, ~65%: 6점, ~60%: 4점, 60% 미만: 2점
- 방법의 타당성: 10점
- 프로그램 완성: 5점

참고: 혼동 행렬

- 혼동 행렬(confusion matrix): 오류 경향 분석

예측 / 실제	ω_1(YES)	ω_2(NO)
ω_1(YES)	n_{11} TP	n_{12} FN
ω_2(NO)	n_{21} FP	n_{22} TN



분류에서의 기준: 정확도accuracy

$$\mathrm{P} = \frac{n_{11} + n_{22}}{n_{11} + n_{12} + n_{21} + n_{22}}$$

검출에서의 기준: 참 긍정률true positive rate과 거짓 긍정률 false positive rate

$$\mathrm{TPR} = \frac{n_{11}}{n_{11} + n_{12}} \qquad \mathrm{FPR} = \frac{n_{21}}{n_{21} + n_{22}}$$

검색에서의 기준: 정밀도precision과 재현률recall

$$\mathrm{P} = \frac{n_{11}}{n_{11} + n_{21}} \qquad \mathrm{R} = \frac{n_{11}}{n_{11} + n_{12}}$$

$$F = 2 \times \frac{P \times R}{P + R}$$