Project1: Eigenface

2016024766 김서현

Index

프로젝트 설계

프로젝트 설계 - dataset

- dataset으로 쓰일 이미지들은 https://conradsanderson.id.au/lfwcrop/ 사이트에서 13278개를 구했다.
- 이미 grey cropped face image인 상태였기 때문에 이 부분에 대해서는 별다른 조치를 취하지 않았다.
- 13278개 중 5000개를 골라 dataset으로 정하고 32x32의 사이즈로 변환하였다.
- dataset 안에는 같은 사람의 다른 표정의 얼굴들이 포함되어 있다.
- 이렇게 만든 5000개의 이미지를 dataset으로 정하고 프로젝트를 시작했다.

프로젝트 설계 – SVD

- dataset의 행렬이 원점을 포함하는 vector space가 되도록 만들어야 한다.
- eigenface를 찾기 위해 SVD 혹은 PCA를 사용해야 한다.
  - Python의 라이브러리를 이용
- eigenface로 나온 것들이 orthonormal한 basis가 맞는지 확인해야 한다.
  - 아닐 경우 scaling을 통해 크기를 1로 만들어줘야 한다.
- basis의 계수들을 통해 face recognition을 할 수 있도록 구현해야 한다.

프로젝트 구현

프로젝트 구현 - SVD

- 우선 이미지들의 평균 벡터를 구하고, 평균벡터를 이미지 벡터에서 빼서 원점을 포함하는 Vector space가 되도록 만들었다.
- SVD 알고리즘을 이용하여 basis가 될 eigenface를 구하도록 하였다.
- SVD를 하기 위해서 scipy 라이브러리와 numpy 라이브러리를 사용하였다.
- basis로 사용되는 eigenface는 eigenvalue가 큰 순서로 100개를 뽑았다.
  - 처음에는 50개로 잡았으나 후에 테스트를 할 때 정확도가 낮다고 느껴 100개를 뽑는 것으로 수정하였다.
- basis들이 orthonormal한 지 확인하기 위해 basis끼리 내적도 해보고 크기도 구해보았다.

프로젝트 구현 – 얼굴 복원

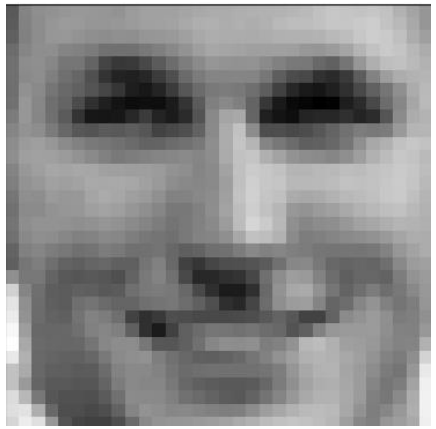
- SVD를 통해 구한 eigenface들과 이미지를 내적해서 각 이미지마다 계수들의 배열을 만들었다.
- 이렇게 구한 계수들의 배열과 eigenface들을 다시 내적하고 그 결과들을 합해서 원래 있던 이미지의 얼굴을 복원해냈다.
- 복원한 이미지를 출력하기 위해 matplotlib의 plt를 이용했다.

프로젝트 구현 - Recognition

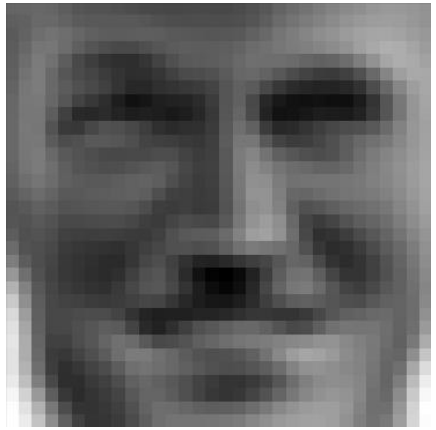
- 얼굴인식을 하기 위해서는 basis가 된 eigenface들의 계수 배열의 유사도를 검사해봐야 한다.
- 계수들의 유사도를 알기 위해 계수 배열 간의 distance를 구했다.
- Distance를 구하는 방법이 여러가지가 있었는데 그 중 유클리드 거리를 구하도록 했다.
- 유클리드 거리를 구하기 위해 scipy 라이브러리를 이용했다.

실행 결과 및 분석

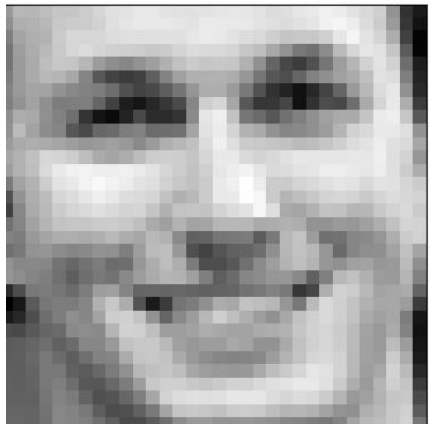
실행 결과 – 얼굴 복원



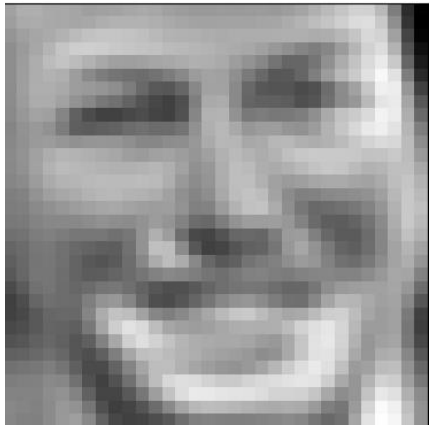
원본



복원

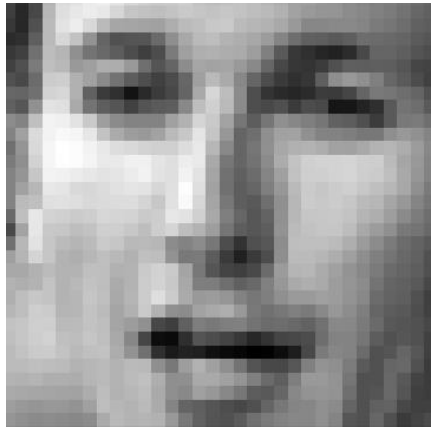


원본

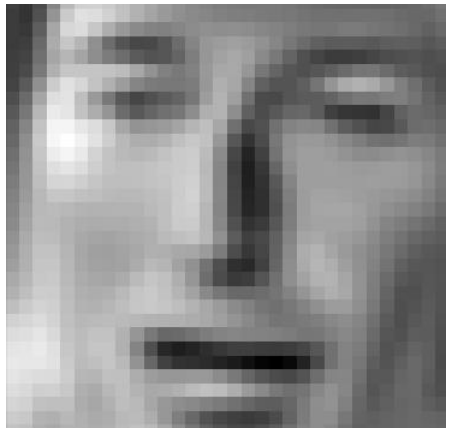


복원

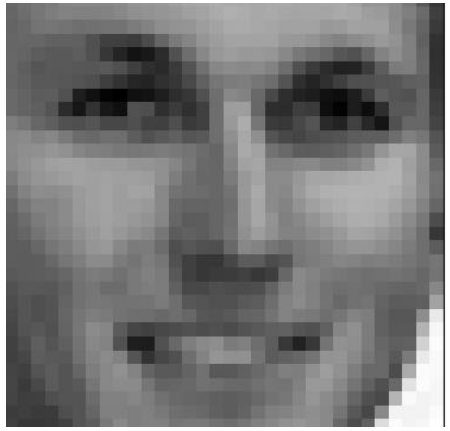
실행 결과 – 얼굴 복원



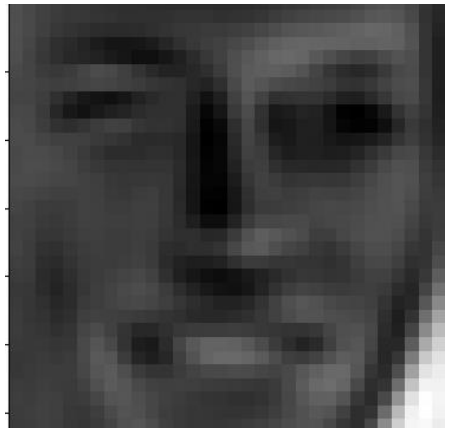
원본



복원



원본



복원

실행 결과 및 분석 – 얼굴 복원

- 위 슬라이드에서 보인 것처럼 원본 사진을 SVD 분할을 한 후 eigenface와 계수를 이용해 다시 복원했을 때 유사한 얼굴 이미지를 얻을 수 있었다.
- 하지만 완전히 동일하게 복원하지는 못하고, 색깔도 전반적으로 원본보다 더 진하게 복원되었다.
- eigenvalue가 더 큰 eigenface에 더 가중치를 줘서 복원을 해봤으나 얼굴 형태가 오히려 더 흐릿해지고, 색이 더 진해져서 그냥 계수와 eigenface의 곱으로만 복원했다.

실행 결과 및 분석 – basis 개수 (50 VS 100)


원본


100


50


원본


100


50

- eigenface의 개수를 100개로 했을 때와 50개로 했을 때의 결과이다.
- 50개로 했을 때에도 전반적인 윤곽은 원본을 따라가지만 100개로 한 결과와 비교했을 때 훨씬 흐릿하고 부정확하다.
- 이 결과를 통해 basis vector가 많을 수록 복원한 결과가 정확해지는 것을 알 수 있다.
- 하지만 basis vector가 더 적을수록 더 작은 차원에서 계산할 수 있는 것이기 때문에 필요에 맞게 계산속도와 결과의 정확도 사이에서 타협을 하면 될 것이라 생각한다.

실행 결과 및 분석 – recognition (1/2)

```
same
931.8568113826626
1330.218962087119
1597.5485306333674
1309.7672601763068
different
2314.149154716732
1353.8237379170564
2838.7934936392294
1479.7036452795094
1472.0684727309497
```

가중치가 없을 때

```
same
24027.352900339236
42089.38732561614
43438.338852833665
28717.83919547242
different
94038.79458293122
29002.912482338084
125702.16380298919
35065.62981981428
29207.154057625503
```

가중치: Eigenvalue/2000

- recognition을 하기 위해 두 이미지의 basis 계수의 유클리드 거리를 구한 결과이다. 위의 same 칸은 같은 인물의 다른 사진을 비교한 것이고, 밑의 different에서는 서로 다른 인물의 사진을 비교한 것이다. (9개의 test case)
- 유클리드 거리가 클수록 유사도가 떨어지는 것이기 때문에 거리가 크면 다른 인물이고, 거리가 작으면 동일 인물이라고 판단하기 위해서 거리를 계산했다.

실행 결과 및 분석 – recognition (2/2)

- 평균을 따져보면 different의 거리가 더 크긴 하지만 same에 있는 거리가 different에 있는 거리보다 더 큰 경우도 있다.
- 그래서 계수 배열을 구할 때 eigenvalue/2000의 값을 곱해서 eigenvalue가 클수록 더 큰 가중치를 갖도록 만들어보았다. 하지만 오른쪽의 결과가 보여주듯 same과 different를 완전히 나눌 수 있게 값이 나오지는 않았다.
- 따라서 왼쪽 결과를 토대로 recognition을 하기 위해서 유클리드 거리가 1400 이상이면 다른 인물, 1400미만이면 같은 인물로 판별한다면 9개의 test case 중 7개는 알맞게 분류할 수 있다.
- 더 확실한 기준을 만들기 위해서는 훨씬 많은 테스트를 해봐야 하겠지만 지금으로서는 1400을 기준으로 잡고 판단하는게 가장 옳은 것으로 생각된다.