**고급소프트웨어실습**

**중간 리포트**

**(CSE 4152)**

**Due: 2020년, 10/19일, 23:59분**

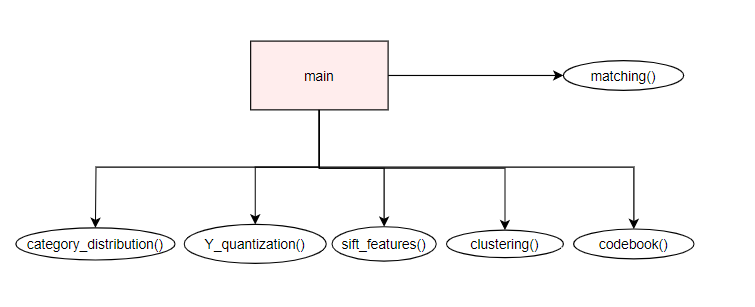
**학번: 20171672**

**반 번호: 2반**

**이름 : 이정원**

1. **이미지 검색 프로그램의 구현에 필요한 주요 함수들의 관계도 그래프를 자유로운 형식으로 그리고, 각 함수들의 역할에 대하여 설명하시오. (이미지는 모두 로컬 하드 드라이브에 저장되어 있는 것으로 가정. 함수의 개수는 주요 기능 위주로 5개 이상 10개 이하로 선정할 것)**

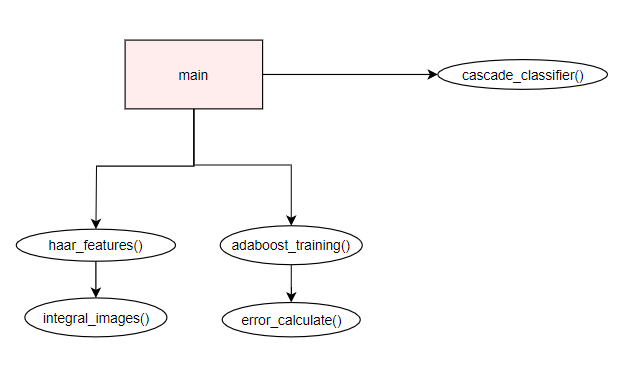
이미지 검색은 주어진 검색어 또는 이미지로부터 연관된 이미지를 찾는 것을 뜻한다. 이를 위해 Bag-of-words 모델과 SIFT를 통해 이미지들의 특징점을 추출하는 과정이 필요하다.



|  |  |
| --- | --- |
| **함수** | **설명** |
| main | 이미지 입력과 출력을 수행한다. 이미지들에서 필요한 정보들을 저장한다. main에서 이루어지는 과정은 크게 두 가지로 나눌 수 있다. 우선 하드 드라이브에 저장되어 있는 이미지들을 BoW를 이용하여 code book을 생성하고 난 뒤(1), 검색이 필요한 이미지(input 이미지)를 입력받아 앞서 생성한 code book을 참조하여 이미지 검색 과정을 실행한다.(2)  (1)  모든 이미지들에 대하여 카테고리 별로 분류하기 위해 category\_distribution() 호출. 이후 Y\_quantization() 호출한다. 카테고리 별 전체 이미지들의 SIFT feature 추출을 위해 sift\_features() 호출. codeword 추출을 위해 clustering() 함수 호출. codebook 생성을 위해 codebook() 함수 호출.  (2)  검색이 필요한 이미지(input 이미지)에 대해 Y\_quantization(), sift\_feautres() 함수 호출. 그리고 카테고리 매치를 위해 matching() 함수 호출. 어느 카테고리로 검색이 되었는지 결과를 출력한다. |
| category\_distribution() | 입력받은 이미지들을 카테고리 별로 분류한다. |
| Y\_quantization() | 이미지의 YUV 밝기 정보 값인 Y값을 8단계로 나눠 quantization한다. |
| clustering() | 추출된 features에 대해 클러스터링을 수행하여 클러스터 센터(center)들인 codeword를 찾아낸다. 이를 위해 k-means 알고리즘을 이용한다. |
| codebook() | 찾아낸 codeword들로부터 codebook을 생성한다. |
| sift\_features() | 입력받은 이미지 정보들을 이용해 SIFT features를 찾고 특징점을 추출해 특정 배열에 저장하는 역할을 하는 함수이다. |
| matching() | input 이미지에 대해 특징점들이 codebook에 어느 카테고리에 매칭이 되었는지 확인하는 함수이다. 매칭이 많이 된 카테고리를 선별해 반환한다. |

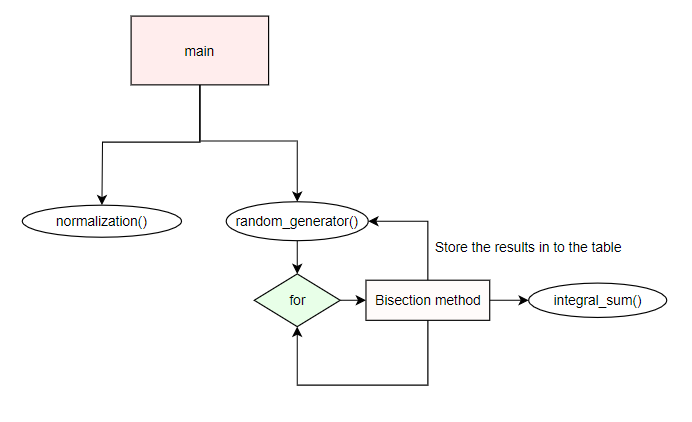
1. **얼굴 인식 프로그램의 구현에 필요한 주요 함수들의 관계도 그래프를 자유로운 형식으로 그리고, 각 함수들의 역할에 대하여 설명하시오. (이미지는 모두 로컬 하드 드라이브에 저장되어 있는 것으로 가정. 함수의 개수는 주요 기능 위주로 5개 이상 10개 이하로 선정할 것)**

얼굴 인식 프로그램의 구현을 위해 Haar Cascade라는 머신러닝 기반의 오브젝트 검출 알고리즘을 이용한다. 직사각형 영역으로 구성되는 특징을 사용하기 때문에 직접 사용할 때보다 동작 속도가 빠르다. 찾으려는 얼굴이 포함된 이미지와 얼굴이 포함되지 않은 이미지를 사용하여 하르 특징 분류기를 학습시킨 후, 분류기를 이용해 얼굴을 검출한다.



|  |  |
| --- | --- |
| **함수** | **내용** |
| main | 이미지 입력과 출력을 수행한다. 이미지들에서 필요한 정보들을 저장한다. Haar features 검출을 위해 harr\_features() 함수를 호출하여 적분 이미지를 계산한다. 처리 성능 향상을 위해 adaboost\_training() 함수를 호출한다. 이후 input 이미지를 입력받아 cascade\_classifier()을 호출한다. 그리고 검출 결과를 출력한다. |
| haar\_features() | Haar features를 계산하는 함수이다. 모든 가능한 크기의 커널을 가지고 이미지 전체를 스캔하여 Haar features를 계산한다. 이는 이미지를 스캔하면서 위치를 이동시키는 인접한 직사각형들의 영역내에 있는 픽셀의 합의 차이를 이용한다. 사각 영역 내부의 픽셀들을 더하기 위해 integral\_images() 함수를 이용한다. 계산된 Haar features를 저장한다. |
| integral\_images() | Haar features를 계산하는 함수이다. 모든 가능한 크기의 커널을 가지고 이미지 전체를 스캔하여 Haar features를 계산한다. 이는 이미지를 스캔하면서 위치를 이동시키는 인접한 직사각형들의 영역내에 있는 픽셀의 합의 차이를 이용한다. 사각 영역 내부의 픽셀들을 더하기 위해 integral\_images() 함수를 이용한다. 계산된 Haar features를 저장한다. |
| adaboost\_training() | haar\_features() 함수를 통해 구한 Haar features들 중 최적의 특징을 선택하는 함수이다. 각 특징에 대해 얼굴이 포함된 이미지와 얼굴이 포함되지 않은 이미지를 분류하기 위한 최적의 임계값을 찾는다. 이를 위해 다음 과정을 반복한다.   * 처음에는 모든 Haar feature에 똑같은 가중치를 할당한다. * error\_calculate() 함수를 이용해 각 Haar feature의 에러률을 계산하고 에러률이 높다면 해당 Haar feature에 가중치를 증가시킨다. * 낮은 에러률을 보이는 Haar feature을 선택한다.   정확도가 충분히 높다면 반복문을 중지시킨다. |
| error\_calculate() | 각 Haar feature의 에러률을 계산한다. |
| cascade\_classfier() | input 이미지 정보를 읽어와 의미 있는 Haar features를 적용한 윈도우를 사용하여 얼굴을 검출한다. 얼굴이 있는 부분과 얼굴이 없는 부분을 나누어 계산 시간을 줄인다. 검출 결과를 반환한다. |

1. **주어진 확률 밀도 함수 p(x)를 통계적으로 따르는 stochastic process, X0, X1, X2, ... 를 생성해 주는 소프트웨어를 설계한 후, 구현에 필요한 주요 함수들의 관계도 그래프를 자유로운 양식으로 그리고, 각 함수들의 역할에 대하여 설명하시오.**



|  |  |
| --- | --- |
| **함수** | **내용** |
| main | 확률 밀도 함수 p(x)에 대한 x와 y값에 대한 정보를 입력받아 저장한다. 정규화를 위해 normalization() 함수를 호출한다. 난수 생성 개수를 입력받은 후, 본격적으로 난수를 생성하기 위해 함수 random\_generator() 함수를 호출한다. 이후 결과 출력 table을 출력한다. |
| normalization() | 입력받은 x값들은 [0, 1] 구간을 가지게, 그에 맞게 y값도 정규화시키는 과정을 수행하는 함수이다. 정규화된 정보는 새 table에 저장한다. |
| random\_generator() | 입력받은 난수 생성 개수만큼 과정을 반복한다. [0.0, 1.0] 구간의 크기를 가지는 난수를 생성한 후, Bisection Method를 사용해 함수의 근을 찾는다. 여기서 함수는 integral\_sum() 함수를 통해 구한 적분 함수이다. 근을 찾았거나 종료 조건에 만족해 반복문이 멈췄을 경우 근을 결과 출력 table에 저장한다. |
| integral\_sum() | 구분구적법을 이용해 적분 함수를 구한다. |