|  |  |  |  |
| --- | --- | --- | --- |
| [공개SW프로젝트] 최종보고서 | | | |
|  |  | | |
| **교과목명** | 공개SW프로젝트 | **담당교수** | 손윤식 |
| **프로젝트명** | openCV SURF를 이용한 명화 인식 프로그램 | **팀명** | 고래고래 |
| **진행기간** | 20.03.16 ~ 20.06.26 | | |
| **주제설명, 선정이유 및 목표** | 여러 오픈소스를 찾아보다가 텐서플로우를 기반으로 한 이미지 인식에 관한 프로젝트를 발견하게 되었다. 이미지 인식에 관한 프로젝트들을 더 찾아보다가 이미지 속 문자를 식별할 수 있는 OCR 기술과 이미지를 학습하여 이미지를 구분해내는 CNN 기술에 팀원 모두 관심을 가지게 되었다. 문자 식별보다는 대상의 특징 자체를 추출할 수 있는 나선형 신경망인 CNN 기술을 이용하기로 했고 거기에 적합한 프로젝트를 생각해보았다. 실생활에서 어디서 본 적 있는 그림이지만 이름이 바로 떠오르지않아 괴로웠던 경험이 한번씩은 있을 것이다. 또는 이름은 알지만 작가의 이름을 모르거나 그림에 대한 정보를 알고 싶을 때도 있다. 이런 궁금증이 생겼을 때 핸드폰을 통하여 바로 궁금증을 해결해 줄 수 있는 CNN 기반 이미지 인식 프로그램을 개발하기로 하였다. 어디서든 쉽게 쓸 수 있고 사용자들의 사용률을 높이기 위해 안드로이드 스튜디오를 사용한 안드로이드 어플로 구현하는 것을 목표로 하였다.  하지만 CNN 기법은 많은 양의 이미지 데이터를 학습시켜야하는 프로그램인데 반해 데이터셋이 한정되어 있고 명화의 사진이라던지 다양한 각도의 이미지 등 많은 데이터를 수집하기에는 현실적으로 불가능하다고 생각하였고 교수님과의 팀별 미팅에서의 피드백으로 남은 시간에서 프로그램 구현에 더 집중할 수 있도록 안드로이드 스튜디오를 사용한 안드로이드 어플 구현에서 일반 PC버전 프로그램으로 구현 방향을 바꾸었고, 데이터셋에 한계가 있기 때문에 CNN 기법 대신 현실적인 방법으로 사용할 수 있는 이미지 인식 알고리즘을 찾아보다 C++ 언어로 구현이 된 OpenCV 라이브러리를 이용하여 이미지나 사진을 인식할 수 있는 오픈소스 코드를 찾아서 이것을 이용하여 명화 인식 프로그램을 만들기로 하였다.  또한 SIFT(Scale-Invariant Feature Transform)이라는 이미지의 특징점을 추출하는 알고리즘을 이용하였다.  해당 프로그램은 명화의 사진이나 이미지를 인식해야 하는 프로그램이기 때문에 이미지의 크기나 찍는 각도, 회전에 상관없이 명화를 인식해야 한다. SIFT는 크기와 회전에 불변한 특징을 추출하여 매칠을 시키는 것이기 때문에 한 장의 그림 데이터만 가지고 있는 본 프로젝트 상황에서도 그림끼리 비교가 가능하다고 판단하였기 때문에 해당 프로젝트에 SIFT를 이용해서 특징점을 추출하여 해당하는 명화를 식별하고 그에 해당하는 명화 정보를 출력하고자 한다.  해당 프로젝트의 목적은 다음과 같다.  주변에 보이는 명화들을 이미지로 인식하여 그림의 작품명, 화가, 작품 설명 등 그림 정보 출력을 목표로 한다. | | |
| **개발 툴** | **<개발 툴>**  **[[1]](#footnote-1)OpenCV** : OpenCV(Open Source Computer Vision)은 실시간 [컴퓨터 비전](https://ko.wikipedia.org/wiki/%EC%BB%B4%ED%93%A8%ED%84%B0_%EB%B9%84%EC%A0%84)을 목적으로 한 프로그래밍 [라이브러리](https://ko.wikipedia.org/wiki/%EB%9D%BC%EC%9D%B4%EB%B8%8C%EB%9F%AC%EB%A6%AC)이다. 원래는 [인텔](https://ko.wikipedia.org/wiki/%EC%9D%B8%ED%85%94)이 개발하였다. 실시간 이미지 프로세싱에 중점을 둔 라이브러리이다. [인텔](https://ko.wikipedia.org/wiki/%EC%9D%B8%ED%85%94) [CPU](https://ko.wikipedia.org/wiki/%EC%A4%91%EC%95%99_%EC%B2%98%EB%A6%AC_%EC%9E%A5%EC%B9%98)에서 사용되는 경우 속도의 향상을 볼 수 있는 [IPP](https://ko.wikipedia.org/wiki/IPP)(Intel Performance Primitives)를 지원한다. 이 라이브러리는 [윈도](https://ko.wikipedia.org/wiki/%EB%A7%88%EC%9D%B4%ED%81%AC%EB%A1%9C%EC%86%8C%ED%94%84%ED%8A%B8_%EC%9C%88%EB%8F%84), [리눅스](https://ko.wikipedia.org/wiki/%EB%A6%AC%EB%88%85%EC%8A%A4) 등에서 사용 가능한 [크로스 플랫폼](https://ko.wikipedia.org/wiki/%ED%81%AC%EB%A1%9C%EC%8A%A4_%ED%94%8C%EB%9E%AB%ED%8F%BC)이며 [오픈소스](https://ko.wikipedia.org/wiki/%EC%98%A4%ED%94%88%EC%86%8C%EC%8A%A4) [BSD 허가서](https://ko.wikipedia.org/wiki/BSD_%ED%97%88%EA%B0%80%EC%84%9C) 하에서 무료로 사용할 수 있다. OpenCV는 [TensorFlow](https://ko.wikipedia.org/wiki/TensorFlow) , [Torch](https://ko.wikipedia.org/wiki/Torch) / [PyTorch](https://ko.wikipedia.org/wiki/PyTorch) 및 [Caffe](https://ko.wikipedia.org/w/index.php?title=Caffe&action=edit&redlink=1)의 [딥러닝](https://ko.wikipedia.org/wiki/%EB%94%A5%EB%9F%AC%EB%8B%9D) [프레임워크](https://ko.wikipedia.org/wiki/%ED%94%84%EB%A0%88%EC%9E%84%EC%9B%8C%ED%81%AC)를 지원한다  저장소 : <https://github.com/opencv/opencv>  **c++** : 2019 버전을 사용한다.  **<개발환경 및 설정 방법>**  Visual Studio 2019용으로 OpenCV 3.4.1 빌드하기  Extra 묘듈 contrib 포함.  비주얼 스튜디오 2019의 64비트를 이용했는데 처음 install때 opencv\_world부분에서 계속 오류(protoc 버전이 낮음으로 인해)가 나서 world기능을 빼고 다시 시도했지만 가장 중요한 모듈인 xfeatures2d부분에서 또 오류가 났다.  원인은 추가 종속성에서 xfeatures2dxxxd.lib를 넣어주지 않았기 때문이었다.  xxx는 opencv의 버전으로 처음에 가장 최신 버젼인 4.3.0으로 설치했다가 오류의 원인을 찾기 위해 버젼을 3.4.1로 낮췄다.  SIFT를 지원하는 opencv 버전에 추가 기능(xfeatures2d등)을 제공하는 opencv\_contrib를 cmake프로그램으로 추가한다.   1. CMake 설치 2. OpenCV 3.4.1 다운(<https://github.com/opencv/opencv/releases>) 3. OpenCV 3.4.1 contrib 다운   (<https://github.com/opencv/opencv_contrib/releases>)   1. CMake 설정  * OpenCV 소스코드 위치 지정 * 빌드 결과물 저장될 위치 지정 * 프로젝트 생성될 generator 설정에서 Visual Studio 16 2019 선택 후 Finish * 옵션들은 설정해줍니다.   <체크해제 목록>  BUILD\_PERF\_TESTS  BUILD\_TESTS  BUILD\_JAVA  BUILD\_PACKAGE  WITH\_1394  WITH\_GSTEAMER  WITH\_LAPACK  WITH\_VTK  <체크 목록>  OPENCV\_ENABLE\_NONFREE  BUILD\_opencv\_world(OpenCV 모듈을 하나의 파일로 만들어줌)  <옵션 위치 지정>  OPENCV\_EXTRA\_MODULES\_PATH : opnecv-sources/opencv-\_contrib-3.4.10/modules  CMAKE\_INSTALL\_PREFIX(빌드 결과물이 복사되는 위치) : opencv-3.4.10/build  search창에 ‘python’ 검색 -> 자동설정     * Generate 후 빌드하기 위해서 open project  1. Visual Studio 빌드  * 구성 release로 변경 후 INSTALL 파일 빌드 * 구성 Debug 변경 후 INSTALL 파일 빌드  1. 프로젝트 설정  * 플랫폼 x64로 변경 후 속성에서 구성을 ‘모든 구성’으로 변경 * C/C++ 추가 포함 디렉터리에 헤더파일 위치 입력   (opencv-3.4.10\build\include)   * 링커 추가 라이브러리 디렉터리에 OpenCV 라이브러리 파일 위치 입력 (opencv-3.4.10\build\x64\vc16\lib) * 디버깅 환경 항목에 DLL 파일 위치 입력   (PATH=c:\opencv-3.4.10\build\x64\vc16\bin;%PATH%)   * 구성을 ‘활성(Debug)’로 바꾼 후 링커-입력-추가종속성 항목 맨 앞에 opencv-world3410.lib; 입력 * 구성을 ‘Release’로 바꾼 후 링커-입력-추가종속성 항목 맨 앞에 opencv-world3410.lib; 입력 | | |
| **주요 기능 설명** | 실내, 창문, 보는, 앉아있는이(가) 표시된 사진  자동 생성된 설명  C:\Users\JSW\Desktop\6.JPG  Web GALLERY OF ART(wga.hu) 사이트는 48000여개의 이미지를 저장하고 있는데 이에 대한 정보(작가, 작품 이름, 제작 기간, 기법, 실제 크기, 보관 장소)를 엑셀과 csv파일로 제공하고 있다.   * <https://www.wga.hu/> -   wga.hu에서 파일을 다운받고 화가별로 분류된 폴더 안에는 csv파일을 있고 이미지가 작품명의 알파벳순으로 1.jpg부터 저장되어 있음. 여기에서 이미지를 다운받을 수 있는 파이썬 코드를 이용해 csv파일에 저장되어 있는 각각의 그림의 URL(jpg 파일로 연결된다.)을 읽어 매칭을 하기 위한 실질적인 이미지를 다운받는다.  Sift는 이미지의 크기나 회전에 불변하는 특징점들을 추출해서 매칭을 시키는 알고리즘이다.  불변하는 특징점들을 추출해서 찍은 사진과 대응이 되는 부분을 찾은 후 그림과 같이 매칭을 시키도록 하였다.  SIFT는 아래의 단계에 걸쳐 이미지의 특징점을 추출하고 매칭한다.    [[2]](#footnote-2)이를 통해 그려진 특징점 각각에는 특징점의 위치, 특징점의 영향 범위에 대한 반경, 그리고 회전 시 특징점을 식별할 수 있는 각도값으로써의 특징점 방향을 알 수 있다. 특징점으로써의 keypoint를 추출했다면, 이 keypoint를 가지고 각 keypoint의 식별자를 계산 해야 한다. 키포인트와 식별자의 정보를 모두 얻은 후 이 키포인트와 식별자를 조합하여 특징점을 사용하여 서로 다른 이미지 사이의 동일한 위치를 식별할 수 있다.  실내, 테이블, 앉아있는, 컴퓨터이(가) 표시된 사진  자동 생성된 설명  사용자가 사진을 찍으면 이미지를 분석한 후 매칭을 시켜 데이터로부터 일치하는 그림을 찾는다. 위의 사진을 보면 이미지가 일치할 경우 이미지 데이터와 사진 사이에 매칭 포인트들이 많아지고, 그에 따라 화면에 나타나는 초록색 사각형의 넓이가 가장 커진다. 이를 이용해서 저희가 가지고 있는 그림 이미지와 매칭을 해보고 그 중에서 원본 사진에 사각형의 면적이 가장 크게 나타나는 그림 이미지를 일치한다고 판단하였다. 오른쪽 그림을 보면, 왼쪽 위에 있는 사진이 가지고 있는 데이터셋이고, 오른쪽이 찍은 사진이다. 찍은 사진에서 사각형이 그려지며 일치하는 그림을 찾게 된다.  C:\Users\JSW\Desktop\2.JPG  위의 그림은 일치하는 그림에서의 결과값이다.  사각형의 area를 계산해서 가장 큰 area를 갖는 이미지를 일치하다고 판단하였다.    일치하면 해당 그림의 인덱스와 csv파일을 매칭시켜 csv파일에서 ,로 구분이 된 각각의 그림의 command 창에 출력하였다.  csv파일에 올라와 있는 다음과 같은 정보들을 출력하도록 하였다.  작가이름, 연혁, 작품명, 제작기간, 기법+사이즈, 보관장소, 타임라인 | | |
| **최종**  **결과물 소개** | C:\Users\JSW\Desktop\3.JPG  C:\Users\JSW\Desktop\4.JPG  먼저 찍은 사진을 이미지 데이터와 비교한다.  이 후 콘솔창에 사진 이미지 파일의 이름을 입력 받는다.  C:\Users\JSW\Desktop\1.JPG  스크린샷, 컴퓨터이(가) 표시된 사진  자동 생성된 설명  그 다음 이 찍은 사진과 매칭이 되는 그림의 정보를 csv 파일로부터 읽어와 콘솔 창에 출력하였다.  C:\Users\JSW\Desktop\6.JPG  만약 찾고자 하는 그림이 데이터 셋에 없다면, 매칭 포인트의 수가 매우 적어져, 그 결과로 그려지는 사각형의 넓이도 마찬가지로 줄어들 것이다.  몇 가지 사진으로 테스트를 해 본 결과 이미지를 찾을 수 없다면 그려지는 사각형의 면적이 1000을 넘지 않는다는 것을 확인하였다.  따라서 찾고자 하는 그림이 데이터 셋에 없으면 화면과 같이 해당 그림을 찾을 수 없다는 문구를 출력하였다. | | |
| **구현과정에서**  **발생한 문제점 및 해결과정** | 초기 프로젝트 구현 방식을 잘못 설정하여 사용 알고리즘과 개발환경을 바꾸는 일이 있었다. 또한 구현과정에서도 몇몇 문제점이 있었다.  구현 과정에서 발생한 문제점은, opencv 오픈 소스를 구현할 때 초기 설정하는 부분에서 오류가 많이 나서 여기서 문제점을 해결하는데 시간을 가장 많이 할애했다.  밑은 발생한 오류들이다.   * 처음 install때 protocol 버전이 낮음으로 인해 오류 발생 * 가장 중요한 모듈인 xfeatures2d부분에서 오류가 발생   사용할 오픈 소스 코드에서 가장 중요한 부분인 xfeatures2d 헤더 부분에서 오류가 발생해 추가 종속성을 넣어주고, opencv의 낮은 버전인 3.4.1을 다운받는 등 이외에도 계속해서 발생하는 오류를 잡아갔다. | | |
| **결론** | * 이미지 매칭을 통해 올바른 그림이 매칭이 되는 것을 확인 * 매칭이 된 이미지의 정보를 출력하는데 성공. * 실행 시간을 줄이도록 개선 * 사용자가 보다 실용적으로 사용하도록 개선 | | |

1. "OpenCV," *위키백과*, n.d. 수정, 2020년 6월 27일 접속, https://ko.wikipedia.org/wiki/OpenCV. [↑](#footnote-ref-1)
2. "Python과 OpenCV – 33 : SIFT(Scale-Invariant Feature Transform)을 이용한 이미지의 특징점 추출," *GIS DEVELOPER*, n.d. 수정, 2020년 6월 27일 접속, http://www.gisdeveloper.co.kr/?p=6779. [↑](#footnote-ref-2)