**캡스톤 디자인 I**

**종합설계 프로젝트**

|  |  |
| --- | --- |
| **프로젝트 명** | *Auto blur with object detection* |
| **팀 명** | *Bblur* |
| **문서 제목** | 계획서 |

|  |  |
| --- | --- |
| **Version** | 1.6 |
| **Date** | 2019-04-16 |

|  |  |
| --- | --- |
| **팀원** | 김용욱 (조장) |
| 김대희 |
| 이나영 |
| 권보경 |
| 채승훈 |

|  |
| --- |
| **CONFIDENTIALITY/SECURITY WARNING**  이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 “Auto blur with object detection”를 수행하는 팀 “Bblur”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “Bblur”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다. |

**문서 정보 / 수정 내역**

|  |  |
| --- | --- |
| **Filename** | AutoBlurWithObjectDetection.doc |
| **원안작성자** | 김용욱,김대희,이나영,권보경,채승훈 |
| **수정작업자** | 김용욱,김대희,이나영,권보경,채승훈 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 수정날짜 | 대표수정자 | Revision | 추가/수정 항목 | 내 용 |
| 2019-03-07 | 전원 | 1.0 | 최초 작성 | 목차 및 개요 초안 작성 |
| 2019-03-08 | 전원 | 1.1 | 1차 수정 | 추가 자료조사 및 내용 추가 |
| 2019-03-11 | 전원 | 1.2 | 2차 수정 | 추가 자료조사 및 내용 수정 |
| 2019-03-12 | 전원 | 1.3 | 3차 수정 | 초안 최종 수정 |
| 2019-04-14 | 김용욱 | 1.4 | 4차 수정 | 피드백 반영 및 내용 수정 |
| 2019-04-15 | 이나영 | 1.5 | 5차 수정 | 변경된 내용 추가 |
| 2019-04-18 | 김용욱 | 1.6 | 6차 수정 | 내용 수정 |

**목 차**

[1 개요 4](#_Toc6518158)

[1.1 프로젝트 개요 4](#_Toc6518159)

[1.2 추진 배경 및 필요성 5](#_Toc6518160)

[1.2.1 기존의 개발된 시스템 현황 6](#_Toc6518161)

[1.2.2 기존의 개발된 시스템의 문제점 혹은 개발할 시스템의 필요성 6](#_Toc6518162)

[2 개발 목표 및 내용 7](#_Toc6518163)

[2.1 목표 7](#_Toc6518164)

[2.2 연구/개발 내용 7](#_Toc6518165)

[2.2.1 영상 파일 입력 및 출력 7](#_Toc6518166)

[2.2.2 딥러닝을 활용한 Object Detection(객체 인식) 8](#_Toc6518167)

[2.2.3 YOLO 모델 사용과 학습 9](#_Toc6518168)

[2.2.4 블러 처리 10](#_Toc6518169)

[2.2.5 결과물 출력 10](#_Toc6518170)

[2.2.6 어플리케이션 구현 10](#_Toc6518171)

[2.3 개발 결과 11](#_Toc6518172)

[2.3.1 시스템 기능 요구사항 11](#_Toc6518173)

[2.3.2 시스템 비기능(품질) 요구사항 11](#_Toc6518174)

[2.3.3 시스템 구조 12](#_Toc6518175)

[2.3.4 결과물 목록 및 상세 사양 12](#_Toc6518176)

[2.4 기대효과 및 활용방안 12](#_Toc6518177)

[3 배경 기술 13](#_Toc6518178)

[3.1 기술적 요구사항 13](#_Toc6518179)

[3.2 현실적 제한 요소 및 그 해결 방안 13](#_Toc6518180)

[3.2.1 하드웨어 13](#_Toc6518181)

[3.2.2 소프트웨어 13](#_Toc6518182)

[4 프로젝트 팀 구성 및 역할 분담 14](#_Toc6518183)

[5 프로젝트 비용 14](#_Toc6518184)

[6 개발 일정 및 자원 관리 15](#_Toc6518185)

[6.1 개발 일정 15](#_Toc6518186)

[6.2 일정별 주요 산출물 15](#_Toc6518187)

[6.3 인력자원 투입계획 16](#_Toc6518188)

[6.4 비 인적자원 투입계획 17](#_Toc6518189)

[7 참고 문헌 17](#_Toc6518190)

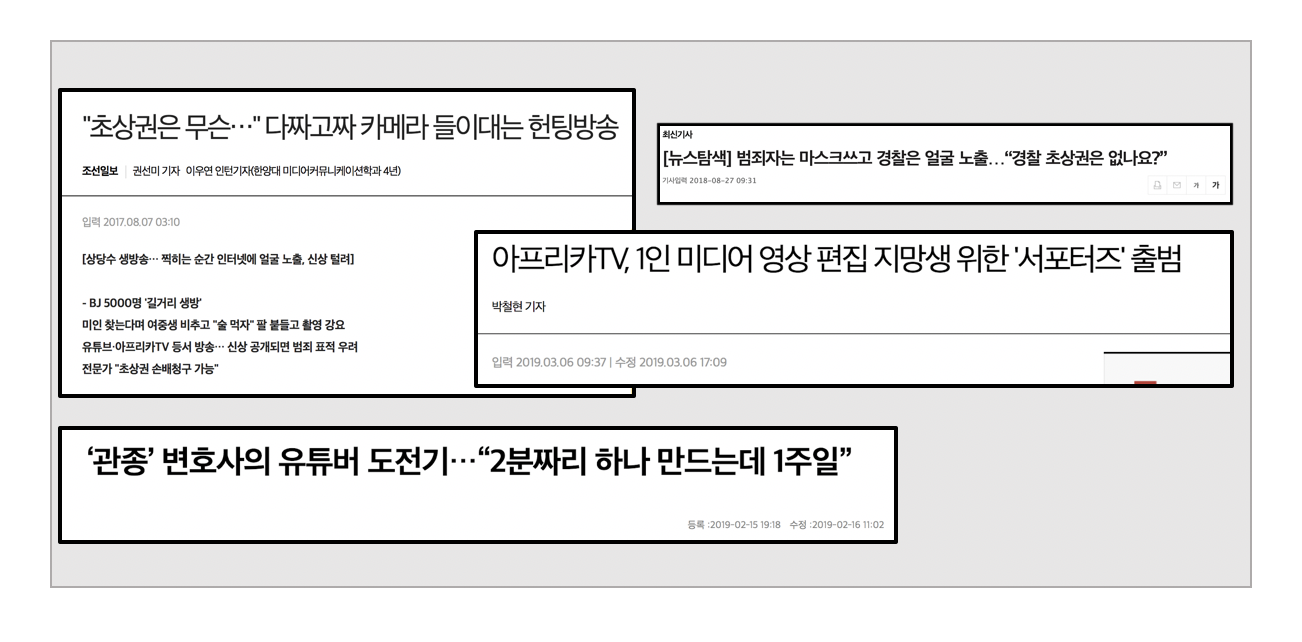
# **개요**

## 프로젝트 개요

이 프로그램은 초상권 보호를 위하여 입력 영상을 받아 특정인을 제외한 얼굴을 블러 처리해준다.

사용자가 이미지 파일 또는 영상 파일 또는 웹캠을 입력으로 넣는다. 프로그램은 입력을 OpenCV 라이브러리를 이용하여 프레임마다 나눈다. 그 나눈 프레임들을 딥 러닝 기반의 오브젝트 디텍션 기법인 YOLO를 이용하여 얼굴 좌표들을 얻는다. 그리고 얻은 좌표에 대해 Face\_recognition 라이브러리를 이용하여 특징 값을 추출하고 학습되어 있는 특징과 비교한 후, 학습되지 않은 대상에 대하여 블러 또는 모자이크 처리한다. 따라서 사용자는 입력한 이미지 또는 영상 파일에 대하여 편집된 사진 또는 영상을 얻을 수 있다. 추가적으로 입력 파일에서 브랜드 로고를 탐색하여 자동으로 블러 처리하는 기능을 제공한다.

## 추진 배경 및 필요성

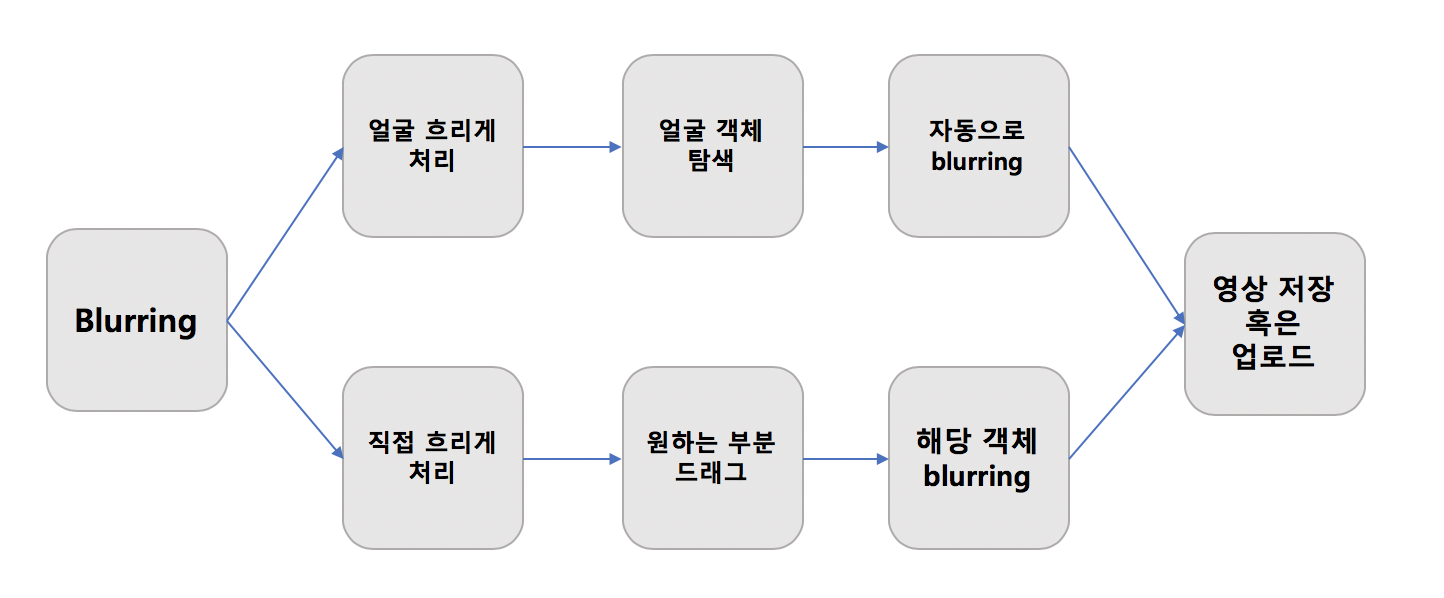


**그림 1 초상권 및 편집 프로그램 관련 기사**

유튜브를 필두로 한 영상 시장은 최근까지 지속적으로 크게 성장하고 있다. 그런데 시장이 커지고 있는 만큼 문제점도 발생하고 있다. 길거리를 지나가다 얼굴이 찍혀 바로 실시간 방송에 노출되는 문제가 많아지고 있다. 또는 행인의 얼굴이 찍혀 있는 영상을 편집 없이 업로드하는 문제도 있다. 이는 당사자 동의가 없을 경우 명백한 초상권 침해다. 또한 네이버 거리뷰, 다음 로드뷰 등에서도 초상권 보호를 필요로 한다. 인터넷과 디지털 기술의 발달로 동영상 콘텐츠의 파급력과 영향력이 점차 커져가는 현재에는, 더욱이 초상권이라는 권리의 개념을 제대로 인식하고 보호해야 한다.

이에 더해서 방송심의 관한 규정에 의하면 특정 방송에서는 브랜드 로고를 노출시키는 형태의 광고는 금지되고 있고, 현재 이 작업을 브랜드 로고에 검은 테이프를 붙이거나 편집할 때 직접 블러 처리를 하고 있다. 이러한 작업을 수작업으로 하여서 인력의 낭비가 심하고 그리고 규정을 위반하면 과징금 또는 제재를 받는다.

### **1.2.1 기존의 개발된 시스템 현황**



**그림 2 유사한 프로그램 ‘유튜브’의 처리 단계**

동영상 블러 처리 기능을 가진 편집 프로그램 중에 ‘유튜브편집기'가 있다. 이 에디터는 동영상을 삽입하여 얼굴을 자동으로 추적하거나, 직접 설정하여 흐리게 처리하는 두 가지 방식이 있다. 전자는 동영상 내에서 얼굴에 해당하는 객체를 탐색한 후 자동으로 블러 처리하는 방식이다. 후자는 사용자가 해당 동영상에서 블러 처리를 할 사물을 드래그하여 설정하여 블러 처리를 하는 방식이다. 이렇게 처리한 동영상 파일은 업로드하거나, 저장할 수 있다.

### **1.2.2 기존의 개발된 시스템의 문제점 혹은 개발할 시스템의 필요성**

위의 유사 프로그램은 동영상을 입력으로 받아 사용자의 필요에 따라 자동 혹은 수동으로 설정하는 방식이다. 자동으로 처리하는 방식은 처음에 직접 하나하나 클릭을 해야하고 해당 객체가 다른 사물과 겹치면 추적기가 벗어날 수 있으며, 화면이 전환되는 구간에서 블러 처리를 하지 않을 부분까지 적용된다. 그뿐만 아니라 수동으로 처리하면 각 단계별로 출력물에 대하여 확인하는 과정이 필요로 하여 이에 따른 인력과 시간이 많이 소요된다는 문제점이 있다. 반면에 우리 프로그램은 이와 다르게 사진 또는 동영상 파일을 받아 얼굴을 인식하고 바로 블러 처리해주는 기능으로 덜 수동적이고 빠르게 사용자의 요구를 만족할 수 있고, 기존의 프로그램보다 동영상 편집하는 데에 소요되는 시간이 적고 인력 낭비 또한 해결할 수 있다.

# **개발 목표 및 내용**

## 목표

이 프로그램은 입력(실시간 웹캠, 이미지 파일, 영상 파일)에서 특정인을 제외한 얼굴을 블러 처리하여 출력해준다. 또한 모바일 어플리케이션을 개발해서 이미지나 영상 파일에 대해 서버를 통한 블러 처리를 실행할 수 있도록 한다. 추가적으로 브랜드 로고를 인식하여 블러 처리된 결과물을 생성하는 것을 목표로 한다.

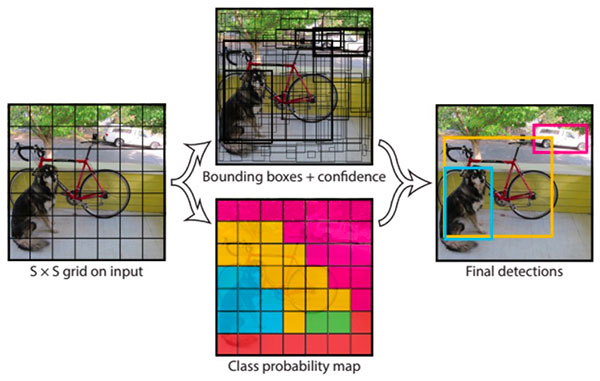
## 연구/개발 내용

## 2.2.1 영상 파일 입력 및 출력

사용자가 지정한 파일을 입력으로 받으면 OpenCV 라이브러리를 이용하여 입력을 프레임마다 나누어 수정할 수 있도록 구현한다. 또한 프레임마다 나누어 수정한 영상의 이미지들을 다시 영상으로 출력해준다.

### **2.2.2 딥러닝을 활용한 Object Detection(객체 인식)**

영상의 특정 영역을 탐색하여 특정 얼굴 또는 브랜드 로고를 찾아 블러 처리를 자동화하는 프로그램을 개발하기 위해서는 이미지의 어떤 부분이 얼굴인지 또는 브랜드 로고 인지 파악하는 기능이 필요하다. 따라서 우리는 기존의 영상처리 기술보다 더 효율적이라고 알려져 있는 딥러닝을 활용한 Object Detection에 주목했다. Object detection은 특정한 이미지에서 물체가 존재하는지 인식, 분류하고, 또한 인식한 Object가 이미지 상 어디에 위치하는지를 탐색하는 알고리즘이다. 아래 이미지는 Object detection 모델 중 하나인 YOLO가 이미지 탐색에 대해 어떻게 접근하는지 대략적으로 나타낸다.

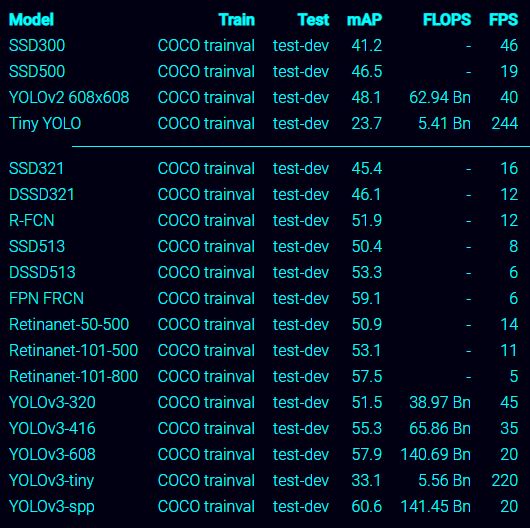


**그림 3 YOLO 이미지 탐색 과정**

이 알고리즘은 2013년의 R-CNN부터 시작되어 Fast R-CNN, Faster R-CNN, YOLO v1, SSD 등을 거쳐 현재까지 발전해왔다. 그중 Faster R-CNN 까지는 속도 면에서 영상에서 전통적인 프레임 기준인 30fps에 많이 미달하는 모습을 보여줬다. 하지만 YOLO라는 모델부터 TITAN X 그래픽카드를 사용하여 실행했을 때 최대 45fps까지도 나오기 시작했다. 이는 실시간 영상에도 적용할 수 있을 정도다. 따라서 우리가 원하는 영상 처리를 자동화하는 프로그램에서 Object detection을 사용하면 효율적으로 개발할 수 있다고 판단했다.

### **2.2.3 YOLO 모델 사용과 학습**

이 프로그램은 실시간 영상은 물론이고, 영상의 모든 프레임의 이미지를 일일이 탐색해야 하므로 기본적으로 빠른 속도를 필요로 한다. 그래서 아래의 비교표와 같이 최근 Object detection 문제에서 좋은 성능을 갖고 있는 YOLO v3 모델을 사용하기로 했다.



**그림 4 YOLO 와 다른 모델의 속도 비교**

모델에서 필요한 기능은 이미지 상 사람의 얼굴 위치 또는 브랜드 로고의 위치를 찾는 것이다. 따라서 모델을 사용할 때 목표를 얼굴 또는 브랜드 로고로 제한하고 각 프레임 이미지들을 대상으로 탐색한다. 또한 이 신경망 모델에는 이미 학습되어 있는 매개변수가 있다. 하지만 성능 테스트를 했을 때 사람의 얼굴이나 사람 전체를 잘 탐지하지 못하는 결과가 나올 수 있다. 그렇게 되면 개발 목적에 맞도록 데이터 셋을 활용하여 모델을 세밀하게 학습시키는 fine-tuning을 할 계획이다.

### **2.2.4 블러 처리**

**그림 5 이미지 블러 처리**

Object detection을 사용해서 목표의 적절한 이미지 상 좌표를 얻는다. 얼굴에 대한 블러 처리를 하는 경우에는Face Recognition 라이브러리를 사용하여 그 좌표 내의 얼굴에서 특징을 추출하고, 미리 저장해둔 사진 얼굴들의 특징들과 비교하여 다른 사람이라고 인식하면 블러 처리해준다. 브랜드 로고에 대하여 블러 처리를 하는 경우에는 얻은 좌표내의 특징과 학습시켜 놓은 데이터와 비교하여 브랜드 로고로 인식하면 블러 처리를 해준다. 블러 처리는 OpenCV에서 제공하는 픽셀 값들을 정렬한 후에 픽셀의 중간 값으로 채워버리는 medianBlur를 이용하여 진행한다.

### **2.2.5 결과물 출력**

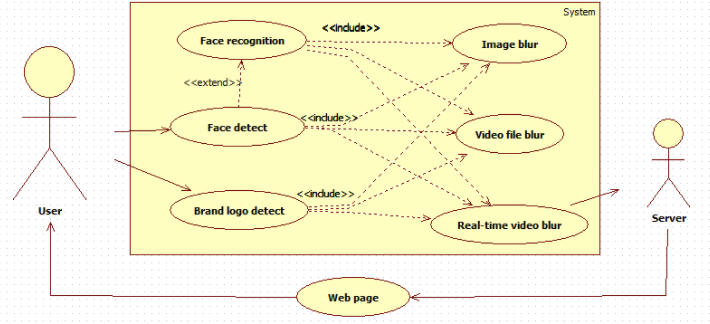
사용자가 객체 탐색 기준을 쉽게 선택할 수 있도록 하고, 입력할 영상 파일을 쉽게 지정할 수 있는 등의 기능을 포함하는 UI를 구현한다. 실시간 웹캠 영상에 대해서는 사용자 PC를 RTSP서버로 사용해서 사용자 PC의 웹캠 영상을 AWS 서버로 전송한 뒤, 블러링 프로세스를 거친 후에 웹 페이지를 통해 처리된 영상을 재생한다. 사용자는 본인 pc에서 웹 페이지에 접속해서 영상을 볼 수 있다.

### **2.2.6 어플리케이션 구현**

전문적으로 영상을 다루는 사람이 아닌 일반인들도 간단하게 사용할 수 있는 안드로이드용 어플리케이션이다. 벚꽃축제, 콘서트, 식당 등 사람들이 많은 장소에서 찍은 사진을 바로 sns에 올릴 때 유용할 것이라고 판단하여 추가적으로 개발을 진행하게 되었다. 블러 처리를 할 서버를 따로 두어 어플리케이션이 무겁지 않게 하고, 카메라 기능을 추가하여 사진을 찍는 과정을 포함한 전 과정을 어플 내에서 해결할 수 있도록 한다.

## 개발 결과

## 2.3.1 시스템 기능 요구사항



**그림 6 사용자 기능 요구사항**

사용자는 객체 탐색 기준을 얼굴 또는 브랜드 로고로 선택 할 수 있다. 얼굴을 선택했을 경우는 탐색되는 모든 얼굴을 블러 처리 할 수 있고, 학습시킨 얼굴을 제외하여 블러 처리 할 수 있다. 이미지 파일, 영상 파일, 실시간 웹캠을 입력으로 선택할 수 있다. 실시간 웹캠의 경우 사용자 PC에서부터 서버로 영상을 전송하고, 서버가 블러 처리한 영상을 웹페이지에 재생하여 제공한다.

### **2.3.2 시스템 비기능(품질) 요구사항**

1. 기준에 따라 꼼꼼한 블러 처리를 하기 위해 해당 영상에서 찾고자 하는 대상의 인식의 정확도가 높아야 한다.
2. 인식을 할 때 해당 영상에서 얼굴의 정면, 측면 모두 인식할 수 있어야 한다.
3. 사진 또는 영상의 화질이 인식할 수 있을 정도의 화질이 되어야한다.

### **2.3.3 시스템 구조**



**사진 7 시스템 구조**

### **2.3.4 결과물 목록 및 상세 사양**

|  |  |  |
| --- | --- | --- |
| **대분류** | **소분류** | **형식** |
| 결과물 | 사진 | 선택한 기준으로 대상을 블러 처리한 사진 |
| 결과물 | 영상 | 선택한 기준으로 대상을 블러 처리한 영상 |
| 결과물 | 실시간 웹캠 | 선택한 기준으로 대상을 블러 처리한 실시간 웹캠 |

## 기대효과 및 활용방안

이 프로그램을 실시간 방송 스트리밍이나 영상 편집 시 활용하면 초상권을 보호하는 데에 도움이 될 수 있다. 이전에는 뉴스나 정규방송 또는 1인 미디어 방송영상 등을 편집할 때, 초상권이나 사생활 침해를 유발할 수 있는 대상을 하나하나 직접 블러 처리를 해야 했다. 노출된 브랜드 로고 또한 직접 편집해야만 했다. 인터넷을 통해 쉽게 접근할 수 있는 포털 사이트의 스트리트뷰나 기업에서 편집해야 할 대량의 영상물들을 마찬가지로 많은 인력과 시간을 들여 편집해야 했다. 따라서 이 프로그램을 활용하면 원하는 대상을 자동으로 블러 처리하여 인력 낭비를 줄이고 시간을 절약하는 효과를 기대할 수 있다.

# **배경 기술**

## 기술적 요구사항

서버 환경 : AWS

개발 환경 : Python3.6 , OpenCV , Tensorflow, Keras, KMUSW 딥러닝클라우드, 안드로이드 스튜디오

사용기술 : YOLO V3 , Face\_recognition , Blurring

## 현실적 제한 요소 및 그 해결 방안

### **3.2.1 하드웨어**

프레임 이미지들을 YOLO를 사용해서 Object detecting 하고 얼굴 분류를 하고 블러 처리하는데 드는 시간은 GPU의 사양에 따라 상이하다. 좋은 성능을 가진 TITAN X GPU를 사용했을 때는 약 40FPS까지 실시간 이미지 연산이 가능하다.

### **3.2.2 소프트웨어**

이 프로그램에서는 최근 Object detection 관련하여 다른 모델에 비해 간단하고 빠른 속도를 갖고 있다고 알려져 있는 YOLO 모델을 사용한다. YOLO는 다른 object detection 모델들과 비교하였을 때 중간 크기나 큰 크기의 Object를 detecting 하는 것에 비하여 작은 Object를 detecting 하는 성능이 낮다는 단점을 가지고 있다. 그리고 얼굴 분류 관련하여서 Face recognition 라이브러리를 사용하는데 동양인 얼굴 인식률은 낮다는 단점과 자체적으로 가진 얼굴 탐색기능에는 Hog방식과 CNN 방식을 사용하여 매우 느리다. 따라서 얼굴 탐색에는 YOLO 모델을 사용하여 빠르게 Bounding box를 얻어내고, 그 얻어낸 결과로 classification을 진행할 것이다. 동양인 인식을 잘 못하는 것은 threshold 값을 조절하여 약간의 성능을 올릴 수 있다.

# **프로젝트 팀 구성 및 역할 분담**

|  |  |
| --- | --- |
| **이름** | **역할** |
| 김용욱 | Team Management, Object detection모델 연구와 사용 및 웹 개발 |
| 김대희 | Object detection 모델 연구와 사용 및 웹 개발 |
| 권보경 | 어플리케이션 및 서버 개발 |
| 이나영 | 어플리케이션 및 서버 개발 |
| 채승훈 | UX / UI |

# **프로젝트 비용**

|  |  |
| --- | --- |
| **항목** | **예상치 (MD)** |
| 아이디어 구상 | 100 |
| 영상 처리 | 50 |
| YOLO 모델 사용 및 학습 | 100 |
| 모바일 어플리케이션 및 서버 개발 | 100 |
| UI 구현 | 50 |
| 합 | 400 |

# **개발 일정 및 자원 관리**

## 개발 일정

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **항목** | **세부내용** | **1월** | **2월** | **3월** | **4월** | **5월** | **6월** | **비고** |
| 요구사항분석 | 요구 분석 |  |  |  |  |  |  |  |
| 아이디어 구상 |  |  |  |  |  |  |  |
| 관련분야연구 | 주요 기술 연구 |  |  |  |  |  |  |  |
| 관련 시스템 분석 |  |  |  |  |  |  |  |
| 설계 | 시스템 설계 |  |  |  |  |  |  |  |
| 구현 | 영상 처리 |  |  |  |  |  |  |  |
| 모델 사용 및 학습 |  |  |  |  |  |  |  |
| 어플리케이션 개발 |  |  |  |  |  |  |  |
| UX / UI |  |  |  |  |  |  |  |
| 테스트 | 시스템 테스트 |  |  |  |  |  |  |  |

## 일정별 주요 산출물

|  |  |  |  |
| --- | --- | --- | --- |
| **마일스톤** | **개요** | **시작일** | **종료일** |
| 계획서 발표 | **산출물 :**   1. 프로젝트 수행 계획서 2. 프로젝트 기능 일람표 | ~ | 2019-03-15 |
| 설계 완료 | 시스템 설계 완료  **산출물 :**   1. 시스템 설계 사양서 | 2019-03-08 | 2019-03-20 |
| 중간 보고 | 입력받은 파일에 대해 탐색 후 좌표 획득  **산출물 :**   1. 프로젝트 1차 중간 보고서 2. 프로젝트 진도 점검표 3. 1차분 구현 소스 코드 | 2019-03-08 | 2019-04-19 |
| 구현 완료 | 입력받은 파일에 대해 탐색 후 좌표를 얻고 그 좌표에 대하여 분류하고 블러 처리 기능 구현  실시간 처리 기능 구현 및 웹 UI 구현  **산출물 :**   1. 각 기능 소스 코드 2. 서버 및 웹 페이지 3. 어플리케이션 | 2019-04-20 | 2019-05-20 |
| 테스트 | 시스템 통합 테스트  **산출물:** 데모 사진 또는 영상 | 2019-05-20 | 2019-05-25 |
| 최종 보고서 | 최종 보고  **산출물: 최종 보고서** | 2019-05-25 | 2019-05-30 |

## 인력자원 투입계획

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **이름** | **개발항목** | **시작일** | **종료일** | **총개발일(MD)** |
| 김용욱 | *Object detection 모델 연구와 사용 및 웹 개발* | 2019-03-09 | 2019-05-20 | 20 |
| 김대희 | *Object detection 모델 연구와 사용 및 웹 개발* | 2019-03-09 | 2019-05-20 | 20 |
| 권보경 | *모바일 어플리케이션 및 서버 개발* | 2019-03-09 | 2019-05-20 | 20 |
| 이나영 | *모바일 어플리케이션 및 서버 개발* | 2019-03-09 | 2019-05-20 | 20 |
| 채승훈 | *UX / UI* | 2019-03-09 | 2019-05-20 | 20 |

## 비 인적자원 투입계획

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **항목** | **Provider** | **시작일** | **종료일** | **Required Options** |
| aws | amazon | 2019-03-15 | 2019-05-31 |  |
| 개인 노트북 5대 | LG, Lenovo, Apple | 2019-01-15 | 2019-05-31 |  |
| KMU SW CLOUD | Kookmin univ | 2019-04-01 | 2019-05-31 |  |

# **참고 문헌**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **번호** | **종류** | **제목** | **출처** | **발행년도** | **저자** | **기타** |
| 1 | 논문 | YOLO: Unified, Real-Time Object Detection | pjreddie | 2018 |  |  |
| 2 | Web page | YOLO object detection with OpenCV | pyimagesearch | 2018 |  |  |
| 3 | Web page | Yolov3: An Incremental Improvement | tistory | 2018 |  |  |
| 5 | Web page | 이미지 Detection 문제와 딥러닝: YOLOv2로 얼굴인식하기 | blog | 2018 |  |  |
| 6 | Web page | html5 player for rtsp stream | Github | 2018 |  |  |
| 7 | Web page | Yolo-v3 and Yolo-v2 for Windows and Linux-How to train (to detect your custom objects) | github | 2019 |  |  |
| 8 | Web page | Face\_recognition | github | 2018 |  |  |