REPORT 

팀 프로젝트 제안서 : 틀린그림 찾기

오픈소스SW개론(002) 3조

**목차**

1. [**개요**](#A1)
2. [**프로젝트 목표**](#A2)
3. [**주제 및 기능**](#A3)
4. [**구현 내용 및 방법**](#A4)
5. [**역할 분담**](#A5)
6. [**개발 일정**](#A6)
7. [**세부 개발 과정**](#A7)

3조 프로젝트 제안서

17011648 배정준, 17011695­ 김동역

프로젝트 주소: <https://github.com/GalaxyOverMe/FindTheDifference>

**1.** **개요**

3조는 OpenCV와 python GUI를 이용한 틀린그림찾기를 구현하였습니다.

배정준은 OpenCV를 이용한 백엔드를 담당하고

김동역은 wxPython을 이용한 프론트엔드를 담당하였습니다.

|  |  |
| --- | --- |
|  |  |
|  | |
| 틀린그림 찾기 **[1]** | |

1.1게임 장르

최종 결정한 장르: 틀린그림찾기(find the difference)

선택한 이유: 틀린그림찾기는 남녀노소 누구나 쉽게 즐길 수 있는 게임입니다. 틀린그림 찾기는 오랜 역사를 자랑하지만 파생된 variation들은 특별히 다른 유형을 띄지 않고 original form을 고수합니다. 이러한 틀린그림찾기를 컴퓨터로 접근해보고 싶었습니다.

1.2 시장조사 / 벤치마킹

구글 플레이스토어 누적 다운로드 횟수 100만회 이상 틀린그림찾기 게임 20개를 분석하였습니다

스누피 버전, 디즈니 버전, 뽀로로 버전, 유저의 점수로 IQ를 측정해주는 틀린그림찾기.

캐릭터, 컨셉이 바뀔 뿐 전반적인 게임형태는 동일합니다.

|  |  |  |
| --- | --- | --- |
|  | → |  |
| 일반적인 틀린그림찾기 구현 과정 | | |
|  | → |  |
| 컴퓨터로 구현할 틀린그림찾기 과정 | | |

기존 틀린그림찾기는 미리 준비된 틀린그림 찾기를 해결하는 방식으로 진행됩니다. 그러나 준비된 이미지가 아닌, 사용자의 이미지를 넣어도 문제를 만들어준다면 틀린그림찾기의 새로운 패러다임을 열 수 있을 것입니다.

**2. 프로젝트 목표**

- 미리 준비한 그림이 아닌 사용자가 입력한 이미지를 이용하여 즉석에서 틀린그림 찾기를 완성하여 즐길 수 있다.

**3. 주제 및 기능**

3.1 주제

이번 프로젝트에서는 틀린그림찾기 생성에 주목합니다. 불필요한 기능은 추가하지 않고 original form을 따릅니다. 제한된 시간안에 두 그림의 다른 부분을 모두 찾아내면 됩니다.

3.2 기능

사전에 준비되지 않은 이미지로 틀린그림 찾기를 생성하여 출제합니다.

**4. 구현 내용 및 방법 (파란색 글씨의 방법을 선택하였습니다.)**

크게 **객체 탐지**(Obejct Detection), **객체 삭제**(Object Deletion)의 2단계 과정으로 요약됩니다.

**객체 탐지**는 엣지 검출(Edge Detection)과 그룹핑으로 세분화됩니다. 그룹핑 방법에는 KNN알고리즘(K-Nearest Neighbors)과 Contour 방법을 사용해보고 좋은 결과를 도출하는 방법을 선택합니다.

**객체 삭제**는 주변색 동화(assimilate)방법 또는 Image Inpainting 방법을 시도해보고 좋은 결과를 도출하는 방법을 선택할 것입니다.**[2][3]**

|  |  |  |
| --- | --- | --- |
| 3d |  | A critical survey of state-of-the-art image inpainting quality assessment  metrics - ScienceDirect |
|  |
| ↓ | Image Inpainting |
|  | Facial Inpainting | Papers With Code |
| 2d |  |
|  |
| AI를 이용한 Image Inpainting |
| 주변 색 동화 | Image Inpainting |

**5. 역할분담**

백엔드 (배정준) : 원본 이미지를 입력받으면 틀린그림을 제작하여 프론트엔드에게 제공, 추가로 제공하는 정보(틀린그림의 메타데이터(좌표, 개수))

프론트엔드 (김동역) : 백엔드로부터 받은 그림 2가지를 이용하여 GUI를 구현

1. **개발 일정**

9주차(4월 26일 ~ 5월 2일) : 프로젝트 수행

백엔드: 객체 탐지(Object Detection) 과정 확립, 객체 삭제(Object Deletion) 과정 확립

프론트엔드: wxPython GUI 학습(Window)

10주차(5월 3일 ~ 5월 9일) : 프로젝트 수행

백엔드: 객체 탐지: Canny Detection 개선(Gaussian Filter -> Bilateral Filter)

객체 삭제: 사각형 -> 다각형

프론트엔드: wxPython GUI 학습(Widget)

11주차(5월 10일 ~ 5월 16일) : 백엔드-프론트엔드 협업 시작

객체 탐지 : Canny Detection -> Morphologic Edge 방법으로 최종 결정

객체 삭제 : ImageInpainting 방법으로 최종 결정

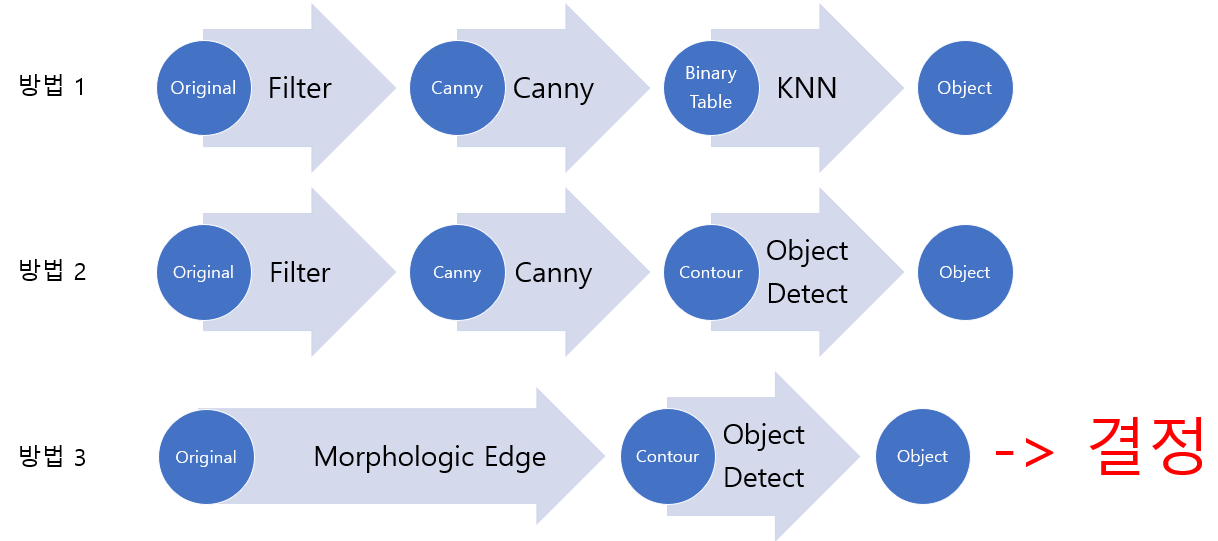
12주차(5월 17일 ~ 5월 23일) : 프로젝트 중간 발표

13주차(5월 24일 ~ 5월 30일) : 프로젝트 수행

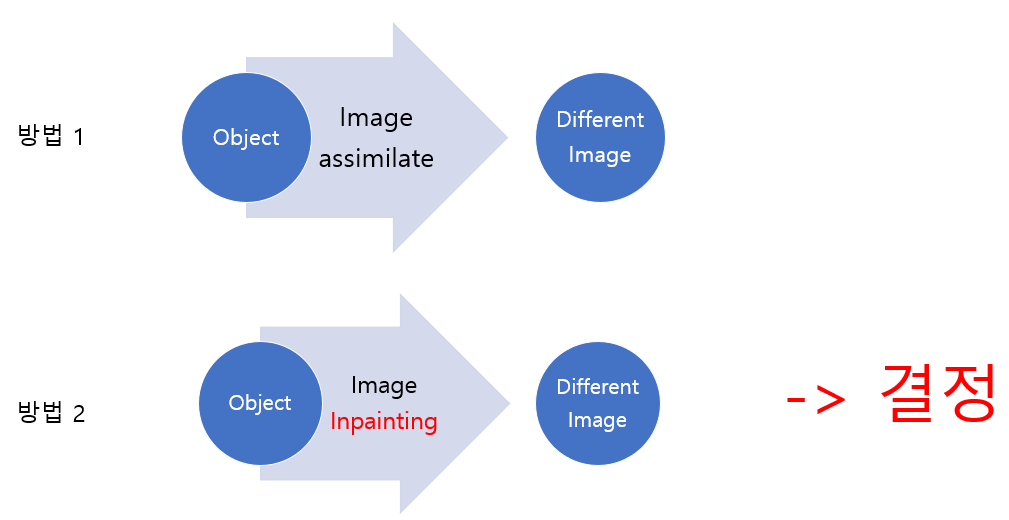
14주차(5월 31일 ~ 6월 6일) : 프로젝트 최종 발표

1. **세부 개발 과정**

**7.1전략 변경**

**객체 탐지  
**

**객체 삭제**

****

3 big problem with before Idea

Object Detection:

ㆍ Too many computation in KNN

ㆍ Morphologic Edge can find more Object than Canny.

Object Deletion:

ㆍ Poor result in image assimilate(absorb neighbor)

Canny 와 Morphology 비교

|  |
| --- |
|  |
| Canny 방법 |
|  |
| Morphologic Edge방법 |

Morphologic Edge 방법이 Canny 방법보다 더 많은 Object를 찾아내는 것을 확인할 수 있다.

백엔드 :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| 원본 그림 | Filter 적용 | | Contour 표시 | |
|  | | | | |
| 퀴즈 후보를 생성 | | 원본에서 지워질 부분 | | Different Image |

Object Detection **: 원본 그림** -> Morphologic Edge -> **Contour[[1]](#footnote-1) 표시** -> **폐곡선을 하나의 Object로 인식**

Object Deletion : 인식한 Object를 삭제

1주차

1. 상세 시나리오  
   1. UI / 제공정보(화면에 어떤 정보를 제공할 것인가?)

2. 조작방법

1. 요구사항 정의서
2. 모듈화
3. 그룹핑
4. 사용자 매뉴얼

Reference

**[1]** 니즈폼<https://birdmom.nizform.com/view.htm?fid=76386&Div=47>

**[2]** <https://www.sciencedirect.com/science/article/abs/pii/S1047320317301803>

**[3]** <https://paperswithcode.com/task/facial-inpainting>

1. Contour : 같은 색 강도를 연결한 선, canny image를 입력으로 받는다. [↑](#footnote-ref-1)