2023 전기 졸업과제

중간보고서

- 실재감 증대/유지를 위한 증강현실 시각 효과 연구개발 -

팀 번호	16	참여 인원	전기컴퓨터공학부 정보컴퓨터공학전공 201724597 천형주
팀명	드루와 유니티의 숲		
분과명	인공지능		
과제명	실재감 증대/유지를 위한 증강현실 시각효과 연구개발		정보컴퓨터공학부 202055536 민예진
지도교수	이명호 교수님		

목차

1. 요구조건 및 제약사항 분석에 대한 수정사항

1) 요구조건

화면상의 물체를 선택하여 해당 물체를 없애는 inpainting 모델의 성능을 비교하는 실험을 진행한다.

① 카메라에서 받아오는 이미지를 image segmentation 결과와 함께 출력

② 화면 상에서 클릭을 함으로써 없앨 대상을 선택하고 마스킹 이미지 생성

③ 마스킹 이미지를 이용하여 선택한 대상을 화면 없애는 inpainting

2) 제약 사항 분석을 통한 수정사항

① 프로그램이 작동하는 플랫폼 변경

VR 기기에서 passthrough 기능은 확인하였지만 컬러 이미지와 윤곽의 이미지가 일치하지 않음, 또 passthrough된 화면을 미러링할 수 없었기에 VR기기의 화면을 이용해 image segmentation과 inpaintg을 실행할 수 없기에 웹캠이나 스마트폰 카메라를 이용하기로 함

② 실시간으로 대상을 없앰

실험 환경에서 Inpainting의 결과물을 실시간으로 처리에 한계가 있음을 깨닫고 inpainting 모델들의 성능을 비교하는 방법으로 진행

Inpainting 모델은 결과물의 정확도 및 속도를 고려하여 선정 예정

③ 사용자가 추가한 객체 추적

기존에는 모델에 있는 라벨이 아닌 사용자가 직접 추가한 라벨로 물체를 감지하고 클릭으로 선택할 수 있도록 설계 및 구현

④ Inpainting의 마스킹 이미지 의존도

모델에 따라 마스킹 이미지의 범위가 segmentation의 결과보다 넓어야 할 수 있음. 이때 OpenCV의 메소드를 통해 dilatation 등의 전처리 진행

2. 설계 상세화 및 변경 내역

① OpenCV를 이용하여 카메라 버퍼를 가져옴

② Object Tracking & Image Segmentation

- OpenCV를 Object traking & image segmentation 결과를 화면 상에 출력
- 화면에서 클릭한 위치를 바탕으로 객체를 선택

③ Inpainting 수행

- Segmentation으로 얻은 masking 이미지를 통해 클릭한 객체 없앰
- 모델 성능 향상을 위해 masking 이미지 보정

④ 모델 튜닝 및 경량화

- 모델 성능 향상을 위해 파라미터 튜닝 및 경량화 후 3번 과정 반복

3. 갱신된 과제 추진 계획

<table>
<tr><td colspan="2">5월</td><td colspan="4">6월</td><td colspan="4">7월</td><td colspan="5">8월</td><td colspan="2">9월</td></tr>
<tr><td>4주</td><td>5주</td><td>1주</td><td>2주</td><td>3주</td><td>4주</td><td>1주</td><td>2주</td><td>3주</td><td>4주</td><td>1주</td><td>2주</td><td>3주</td><td>4주</td><td>5주</td><td>1주</td><td>2주</td></tr>
<tr><td colspan="2">착수 보고서</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="4">유니티 파이썬 연동</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2"></td><td></td><td></td><td colspan="6">화면상 객체 선택과
Image Segmentation 구현</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2"></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="4">Inpainting 실험</td><td></td><td></td><td></td></tr>
<tr><td colspan="2"></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="4">Yolov8 커스텀 이미지
데이터셋 구성</td><td></td><td></td><td></td></tr>
<tr><td colspan="2"></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="3">최종 발표
보고서 준비</td></tr>
</table>

4. 구성원별 진척도

천형주	Yolo v7을 이용한 웹캠을 이용한 image segmentation 구현 Yolo v8과 OpenCV를 이용하여 객체를 직접 선택해 image segmentation하는 코드 구현
민예진	Inpainting 결과 시간 측정 보고서 작성 및 정리

5. 보고 시점까지의 과제 수행 내용 및 중간 결과

1) Segmentation 결과물

```python
    def show(self, p):
        """Display an image in a window using OpenCV imshow()."""
        im0 = self.plotted_img
        if platform.system() == 'Linux' and p not in self.windows:
            self.windows.append(p)
            cv2.namedWindow(str(p), cv2.WINDOW_NORMAL | cv2.WINDOW_KEEPRATIO)  # allow window resize (Linux)
            cv2.resizeWindow(str(p), im0.shape[1], im0.shape[0])

        mouse_class = MouseGesture()
        cv2.imshow(str(p), im0)
        global clicked_x
        global clicked_y
        cv2.setMouseCallback(str(p), mouse_class.on_mouse, param=im0)
        #print("좌표 : x : {} y : {}".format(clicked_x,clicked_y) )
        cv2.waitKey(500 if self.batch[3].startswith('image') else 1)  # 1 millisecond
```

그림 1 results.py

Yolo v8 모델을 이용한 image segmentation 을 위한 코드 분석 결과, 모델을 실행했을 때 OpenCV 를 이용하여 결과창을 띄우는 것을 확인했다. 결과를 표시하는 화면에서 화면상에서 클릭한 위치의 좌표를 저장하는 코드를 추가해주었다.

```python
        global clicked_x
        global clicked_y
        if self.args.save or self.args.show:  # Add bbox to image
            plot_args = {
                'line_width': self.args.line_width,
                'boxes': self.args.boxes,
                'conf': self.args.show_conf,
                'labels': self.args.show_labels,
                'cx':clicked_x,   #화면에서 클릭된 x좌표
                'cy':clicked_y}   #화면에서 클릭된 y좌표
            if not self.args.retina_masks:
                plot_args['im_gpu'] = im[idx]
            self.plotted_img = result.plot(**plot_args)
```

```python
        if pred_masks and show_masks:
            if img_gpu is None:
                img = LetterBox(pred_masks.shape[1:])(image=annotator.result())
                img_gpu = torch.as_tensor(img, dtype=torch.float16, device=pred_masks.data.device).permute(
                    2, 0, 1).flip(0).contiguous() / 255
            idx = pred_boxes.cls if pred_boxes else range(len(pred_masks))
            annotator.masks(pred_masks.data, colors=[colors(x, True) for x in idx], im_gpu=img_gpu)

            label = -1 #아무 물체도 선택하지 않을 상태

            for i in range(len(pred_masks.data)):
                mask = pred_masks.data[i].cpu().numpy()  # 마스크 데이터를 넘파이 배열로 변환 (GPU에서 계산된 경우 .cpu()를 사용하여 CPU로 이동)
                #print(mask.shape)
                if(mask[cy][cx] == 1):
                    label = i
                    break

            print(label) #선택한 물체의 라벨 번호를 출력
```

그림 2 predict.py

모델을 돌려 결과를 예측하는 코드가 predict.py에 있는데 pred_masks의 값에 image segmentation된 물체들의 라벨이 숫자로 분류되고 있었다. results.py에서 저장한 좌표 값을 활용하기 위해 cx, cy 변수를 선언하고, 값을 받아와 좌표의 마스크 이미지 값이 1이 되

는 라벨을 출력하도록 코드를 추가해주었다.

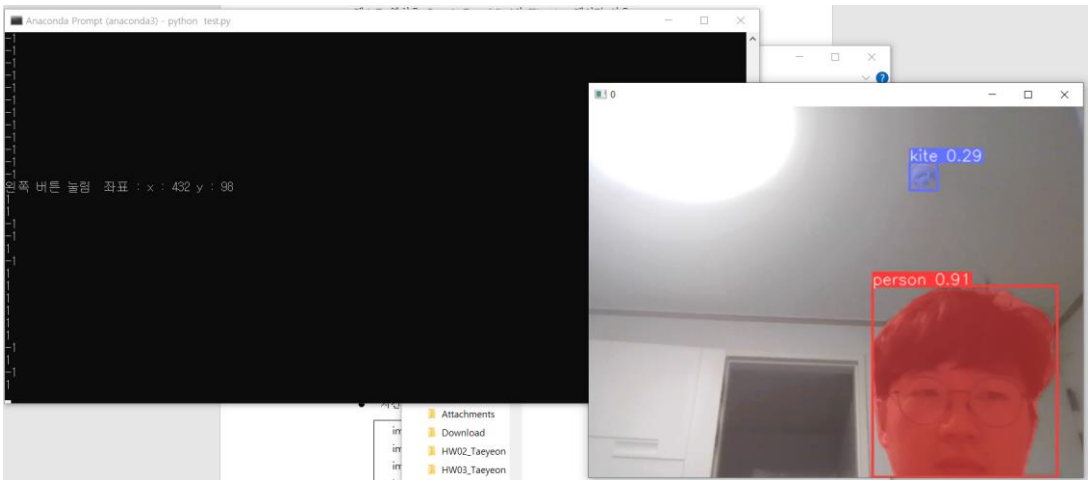


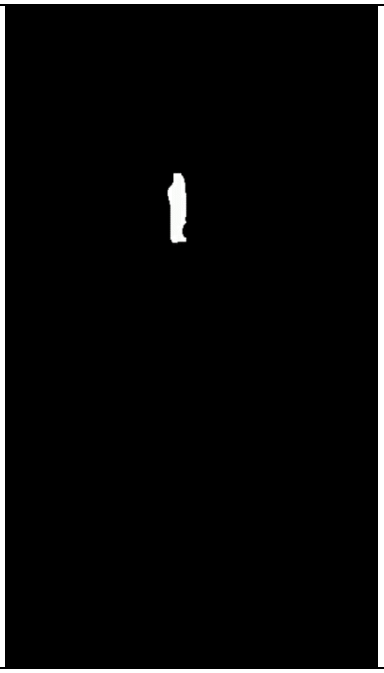
그림 3 결과 - 클릭한 물체의 라벨인 1이 출력

2) Inpainting 결과물

테스트 영상은 Google DeepMind의 Kinetics 데이터 이용

- 원본 / 마스킹 이미지 / Inpainting 결과

  마스킹 이미지는 dilatation연산으로 마스크 이미지를 확장함.

		
원본	마스킹 이미지	결과 이미지

- 시간 측정

image0: 0.7143936157226562	image10: 0.36542439460754395
image1: 0.37601542472839355	image11: 0.36796021461486816
image2: 0.37412428855895996	image12: 0.3654937744140625
image3: 0.3762364387512207	image13: 0.36516380310058594
image4: 0.37516140937805176	image14: 0.3674349784851074
image5: 0.37312960624694824	image15: 0.3661477565765381
image6: 0.37821316719055176	image16: 0.36594223976135254
image7: 0.3764965534210205	image17: 0.3658921718597412
image8: 0.38724303245544434	image18: 0.3669912815093994
image9: 0.3748135566711426	image19: 0.3684370517730713