시각 장애인을 위한 스마트 벨트

벤처 캡스톤 디자인 2조 정보통신공학과

201823001 맹 하늘 201720769 신 선영

201824089 김 하얀 201822181 장 하영

201824413 김 현정 201822437 최 유진

목차

개발 동기 및 필요성

시각 장애인들이 활동할 때에 많은 제약과 외출 시 많은 위험 존재

- 갑작스러운 장애물이 등장할 때, 회피 어려움
- 동시에 여러 대의 버스 정차 시, 정확한 위치 구별 불가

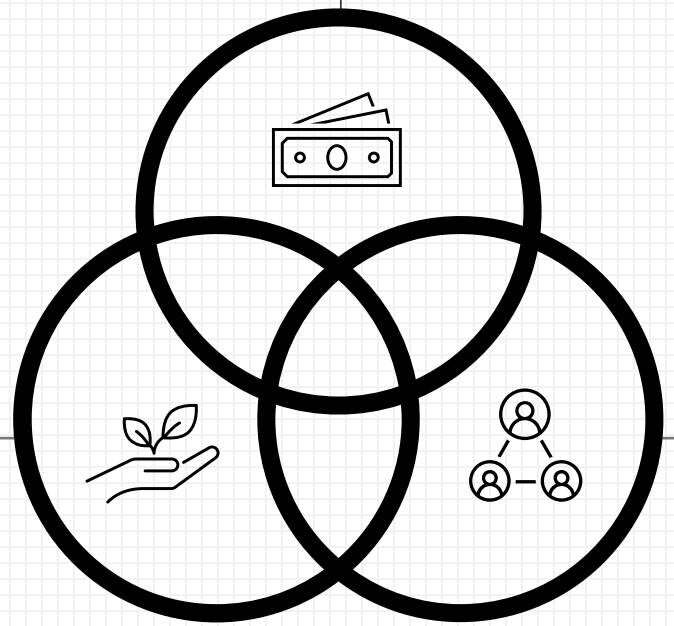
화이트 아이(White eye): 지팡이에 3차원의 공간인식 인터페이스 장치를 접목시켜 개발한 상품

→ 위험 물체 감지 기능 추가하여 시각장애인에게 편의 제공

설계 제한적/기능적 요구 조건

경제

기본 지원금 내에서 사용



환경

환경에 유해하지 않은
부품으로 작품을 설계

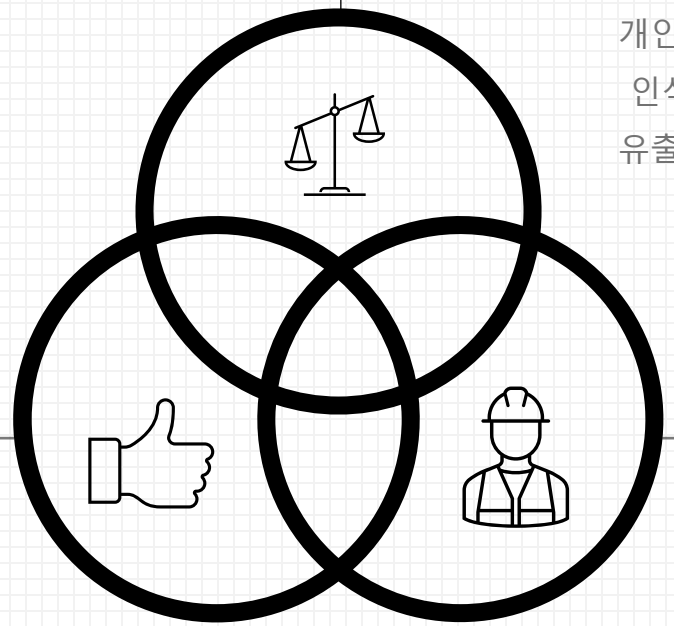
사회

카메라에 인식된 이미지 정보를
음성데이터로 제공하여
시각장애인의 사회적 참여와 활동 범위 확장

설계 제한적/기능적 요구 조건

윤리

학습을 위한 Dataset 이미지 파일이
개인 정보를 침해하지 않아야 함
인식을 위한 이미지 데이터와 사용자 정보가
유출되지 않아야 함



미학

사용자가 스트랩을 착용했을 때
불편함이 없도록 설계

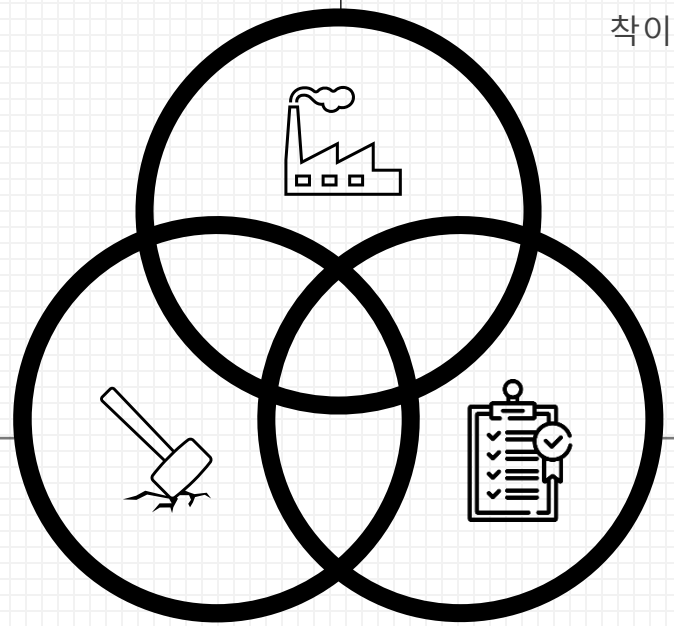
보건 및 안전

거리에서 발생할 수 있는 사용자에 대한
위험 상황을 다양하게 설계하여
사고 예방과 사고 발생 시 저장된 이미지
데이터를 통하여 CCTV의 역할 수행

설계 제한적/기능적 요구 조건

생산성

시중에 저렴하게 구매 가능한 가슴 스트랩에 탈부착이 가능한 본체를 결합하여 쉽게 생산 가능



내구성

사용자가 몸에 부착하여 장시간 이동 시 손상이 발생하지 않게 설계

산업표준

Raspberry Pi 4, pi camera 와 같은 지정된 표준의 장치를 사용

개념설계

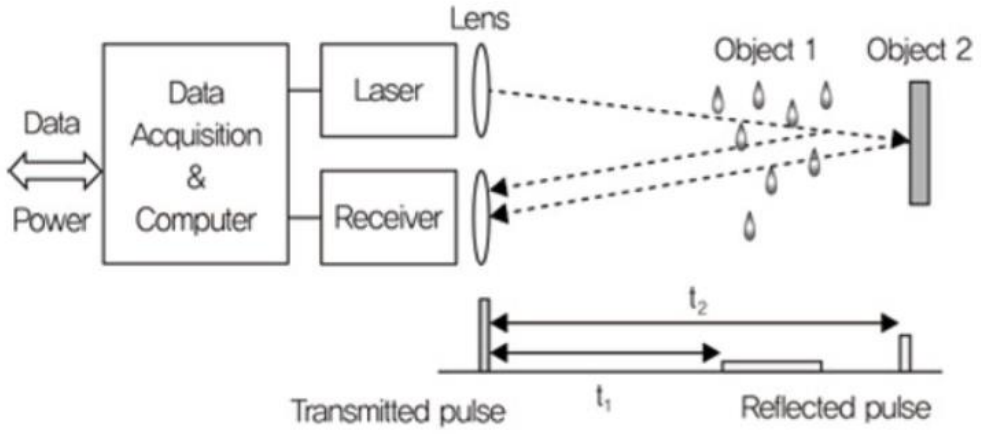
- 하드웨어

· 라즈베리파이에 USB Camera와 Lidar Sensor, GPIO에 Li-ion 배터리를 연결하여 Client 제작

· camera와 Lidar sensor 융합

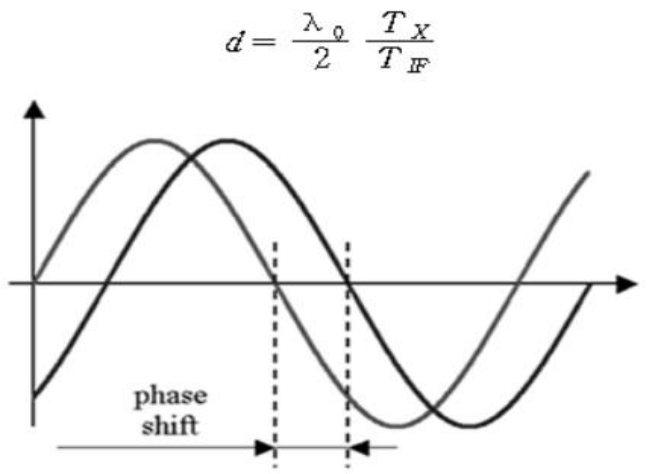
- ToF 센서

· 레이저가 펄스 신호를 방출하여 측정 범위 내에 있는 물체들로부터의 반사 펄스 신호들이 수신기에 도착하는 시간을 측정함으로써 거리를 측정하는 방식



- Phase Shift

· 특정 주파수를 가지고 연속적으로 변조되는 레이저 빔을 방출하고 측정 범위 내에 있는 물체로부터 반사되어 되돌아오는 신호의 위상 변화량을 측정하여 시간 및 거리를 계산하는 방식



개념설계

- 소프트웨어

- 도보 위 위험 물체 식별 기능

· Camera에서 얻은 real time 영상을 사전 학습된 가중치파일을 사용한 YOLO 알고리즘을 통하여 위험물체를 식별

· 학습된 가중치 파일은 이미지 파일 35만장 중 1만 4천 여장을 선별하여 제작하였다.

· Lidar Sensor를 사용하여 이미지에서 식별된 위험물체와 사용자와의 거리 측정

· 측정한 값을 PC로 전송하여 구체적인 값으로 확인

· ROS melodic, OpenCV, darknet 등을 라즈베리 파이에 설치하여 RViz라는 시각화 도구를 통해 측정된 값을 확인.

· 위험 물체와의 거리가 2m 미만일 경우 Speaker를 통하여 위험 경고 출력

- Server(PC)와 Client(Raspberry Pi) 데이터 통신

· ROS에서는 'Topic'방식을 사용하여 통신

· 수신 측을 'Server', 송신 측을 'Client'라고 부름

상세설계

- 하드웨어

① Raspberry Pi 4

② TFmini plus Lidar Sensor

③ 2D/3D Dual Solid-State ToF LiDAR

④ LI-USB30-P031X

⑤ PC

- 소프트웨어

- 도보 위 위험 물체 식별 기능

① Client(Raspberry Pi)

② Server(PC)

작품 사진

[작품의 앞면]

[작품의 뒷면]

작품 사진

[작품 착용 예시]

상세설계

① Client(Raspberry Pi)

```
from SerialClient import *
import sys
import cv2
import numpy as np
from cv_bridge import CvBridge, CvBridgeError
import roslib
import rospy
import imp
import thread
import multiprocessing
from serial import *
import StringIO
from std_msgs.msg import Time
from rosserial_msgs.msg import *
from rosserial_msgs.srv import *
import diagnostic_msgs.msg
import errno
import signal
import socket
import struct
import time

class detector_cam(object):

    def __init__(self):

        self.bridge_object = CvBridge()
        self.image_sub = rospy.Subscriber("/image_raw",Image,self.camera_callback)

    def camera_callback(self,data):
        try:
            # We select bgr8 because its the OpneCV encoding by default
            image = self.bridge_object.imgmsg_to_cv2(data, desired_encoding="bgr8")
        except CvBridgeError as e:
            print(e)

    def get_output(model):
        layer_names = model.getLayerNames()
        output_layers = [layer_names[i[0]-1]for i in model.getUnconnectedOutLayers()]
        return output_layers

    def draw_preds(img, class_id, confidence, x, y, x_plus_w, y_plus_h):

        label = str(classes[class_id])
        color = COLORS[class_id]
        cv2.rectangle(img, (x,y), (x_plus_w,y_plus_h), color, 2)
        cv2.putText(img, label, (x-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
        image=img[y:y_plus_h,x:x_plus_w]

def main():
    detector_cam_object = detector_cam()
    rospy.init_node('detector_node', anonymous=True)
    classes = None

    with open('/home/hy/catkin_ws/src/darknet_ros/cfg/obj.names', 'r') as f: #names 파일 경로
        classes = [line.strip() for line in f.readlines()]

    COLORS = np.random.uniform(0, 255, size=(len(classes), 3))
    scale_factor = 0.00392
    model = cv2.dnn.readNet('/home/hy/catkin_ws/src/darknet_ros/cfg/yolov3-tiny_best.weights','/home/hy/catkin_ws/src/darknet_ros/cfg/yolov3-tiny.cfg')
```

상세설계

① Client(Raspberry Pi)

```
while image.isOpened():
    ret, img = image.read()
    if not ret:
        break

    Height, Width, _ = img.shape
    blob = cv2.dnn.blobFromImage(img, scalefactor=1/255., size=(416,416), swapRB=True)
    model.setInput(blob)
    outputs = model.forward(get_output(model))

    class_ids = []
    confidences = []
    bounding_boxes = []
    conf_threshold = 0.5
    nms_threshold = 0.4
```

```
for output in outputs:
    for detection in output:
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        if confidence > 0.5:
            center_x = int(detection[0]*Width)
            center_y = int(detection[1]*Height)
            w = int(detection[2] * Width)
            h = int(detection[3] * Height)
            x = center_x - w / 2
            y = center_y - h / 2
            class_ids.append(class_id)
            confidences.append(float(confidence))
            bounding_boxes.append([x,y,w,h])
```

상세설계

① Client(Raspberry Pi)

```
            indices =  cv2.dnn.NMSBoxes( bounding_boxes, confidences, conf_threshold, nms_threshold )

            for i in indices:
                i = i[0]
                box = bounding_boxes[i]
                x,y,w,h = box[0],box[1],box[2],box[3]
                draw_preds( img, class_ids[i], confidences[i], round(x), round(y), round(x+w), round(y+h))
                message = print(class_ids[i]+", "+ confidences[i]+", "+ round(x)+", "+ round(y)+", "+ round(x+w)+", "+ round(y+h))
                self.publisher = rospy.Publisher(self.topic, self.message, queue_size=10)
                self.publisher = rospy.Publisher(self.topic, self.image, queue_size=10)

    try:
        rospy.spin()
    except KeyboardInterrupt:
        print("Shutting down")
    cv2.destroyAllWindows()

if __name__ == '__main__':
    main()
```

상세설계

② Server(PC)

```
#include <ros/ros.h>
#include <image_transport/image_transport.h>
#include <opencv2/highgui/highgui.hpp>
#include <cv_bridge/cv_bridge.h>

#include <iostream>
#include "helper/FrameLoader.h"

using namespace std;
using namespace cv;

int main(int argc, char** argv)
{
  VideoCapture cap(0);

  if(!cap.isOpened()) {
   return -1;
  }
 ros::init(argc, argv, "detector_cam");
 ros::NodeHandle nh;
 image_transport::ImageTransport it(nh);
 image_transport::Publisher pub = it.advertise("/camera/image_raw", 10000);


 ros::Rate loop_rate(20); // fps

 while (nh.ok()) {
  Mat frame;
  cap >> frame;
  cout << "web cam frame size:" << frame.size().width << ", " << frame.size().height << endl;
  cout << "publish new frame" << endl;
  sensor_msgs::ImagePtr msg = cv_bridge::CvImage(std_msgs::Header(), "bgr8", frame).toImageMsg();
  pub.publish(msg);
  imshow("web cam", frame);
  waitKey(100);
  ros::spinOnce();
  loop_rate.sleep(); // to maintain te fps needed
 }
}
```

과제 해결 방안

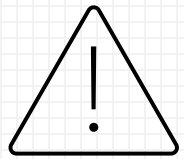
목걸이 구조의 문제점

Raspberry pi zero W를 사용하여 경량화한 목걸이 형태의 모습

→ 착용 후 이동 시 흔들림이 심하여 카메라를 통한 객체 인식이 어려움

해결 방안

> 흔들림을 줄이기 위하여 몸에 밀착될 수 있는 벨트 형태로 변경



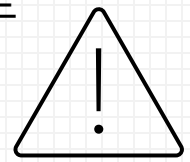
과제 해결 방안

Raspberry Pi zero W 사양

ROS를 설치 시, 필요한 사양에 비해 Raspberry Pi zero W의 CPU 사양이 낮음.

해결 방안

- 경량화를 위해 선택한 Raspberry Pi zero W를 선택한 것인데,
- 제작 형태를 목걸이에서 벨트로 바꾸었기 때문에 Raspberry Pi 4로 대체하여도 문제 없음.



과제 해결 방안

버스번호 인식

① 우리나라 차량 번호판 수집 어려움 → 외국 차량 번호판 237장 이미지 dataset 사용

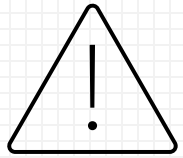
> 국내 시내버스 번호판과 외국 차량 번호판의 규격이 달라 인식 어려움.

② 국내 버스 번호판 임의 생성 → 버스정류장에 두 대 이상의 버스 도착 시, 버스 번호판이 보이지 않아 사용 어려움.

> 직접 수집한 약 60개의 data의 버스 번호를 노선 별로 정면과 측면 2개의 클래스를 설정하여 학습

③ 정확도가 50%인 결과 도출

> 더 많은 data를 모아 학습 시 정확도가 더 향상될 것으로 예상.



기대효과 및 향후 계획

기대효과

- 배터리 부착으로 휴대 가능
- 시중에서 널리 유통되는 제품에 기술을 추가·변형하여 시각장애인에게 장소를 이동하는 데에 더 많은 편의를 제공할 수 있을 것으로 예상

향후 계획

- 국내 시내버스의 노선별 번호를 인위적으로 생성하여 dataset을 제작한 후 학습하여, 버스정류장에 연속적으로 버스 도착했을 때 순서 알림 기능 추가.
- 앱 개발을 통한 User Interface 제공 및 음성 안내 지도 제작 예정

작품 결과 분석

- YOLO 알고리즘을 통하여 dataset으로 딥러닝 진행
- 결과로 얻은 가중치 파일을 사용하여 도로 위에서 위험 물체 인식
- 이 상황에서 미리 지정된 정보를 음성으로 출력
- 충전이 가능한 배터리를 장착하여 휴대 가능하도록 함.

역할분담

맹하늘	김하얀	김현정
프로젝트 총괄, 하드웨어와 구성도·구조도 설계 및 제작, S/W 데이터 처리, PPT 제작, yolo 가중치 파일 제작, 결과보고서 작성, raspberry pi 개발 환경 구축	raspberry pi 클라이언트 환경 구성(ros, catkin, melodic, opencv 등), raspberry pi 센서 연결(LiDAR, Camera), 버스번호판 데이터 음성출력(tts), server 소스코드 제작	라즈베리파이 소스코드 제작, Client 소스코드 제작, yolo 가중치 파일 제작, 위험물체 인식 데이터셋 제작, ros 환경 구성, 라즈베리파이 센서 연결

신선영	장하영	최유진
하드웨어와 구성도·구조도 설계 및 계획, IoT 시스템 구현, 발표, yolo 가중치 파일 제작, 버스 번호판 데이터셋 제작	S/W 데이터 처리, 보고서 작성, server 소스코드 제작, raspberry pi 클라이언트 환경 구성, raspberry pi 센서 연결(LiDAR, Camera), 라즈베리 파이 환경 구성, 앱 구동 환경 개발, 외관 제작 및 디자인	버스번호 인식 후 음성출력 (tts), 버스 번호판 데이터셋 제작, yolo 가중치 파일 제작, 보고서 작성

참고 문헌

- https://github.com/sid0312/ANPR
- https://github.com/AlexeyAB/darknet
- https://github.com/pjreddie/darknet
- https://github.com/mullue/lab-custom-model-anpr
- https://github.com/kairess/ANPR-with-Yolov4

Thank

You