벤처 캡스톤 디자인 2조 결과보고서

<table>
<tr><td>작품 명</td><td colspan="6">YOLO 기반의 머신러닝을 사용하여, 물체인식을 통한 음성 알림 서비스를 제공하는 시각장애인용 '스마트 목걸이'</td></tr>
<tr><td>팀 명</td><td colspan="2">라이프 가이드</td><td>지도 교수</td><td colspan="3">소속 : 정보통신공학과
성명 : 권 기 현 (인)</td></tr>
<tr><td>제작 기간</td><td colspan="6">2021년 3월 3일 ~ 2021년 6월 10일</td></tr>
<tr><td rowspan="7">팀 구성
및
분담 내용</td><td>소 속</td><td>학년</td><td>학번</td><td>성명</td><td>분담 역할</td><td>참여도
(총100%)</td><td>직급</td></tr>
<tr><td>정보통신
공학과</td><td>4</td><td>201823001</td><td>맹하늘</td><td>프로젝트 총괄, YOLO 가중치 파일 제작, raspberyy pi 개발 환경 구축, 하드웨어와 구성도·구조도 설계 및 제작, PPT 제작 및 결과 보고서 작성</td><td>16.7%</td><td>팀장</td></tr>
<tr><td>정보통신
공학과</td><td>4</td><td>201824089</td><td>김하얀</td><td>raspberry pi 클라이언트 환경 구성(ros, catkin, melodic, opencv 등), raspberry pi 센서 연결(LiDAR, Camera), 버스번호판 데이터 음성출력(tts), server 소스코드 제작</td><td>16.7%</td><td>팀원</td></tr>
<tr><td>정보통신
공학과</td><td>4</td><td>201824413</td><td>김현정</td><td>라즈베리파이 소스코드 제작, Client 소스코드 제작, YOLO 가중치 파일 제작, 위험물체 인식 데이터셋 제작, ROS 환경 구성, 라즈베리파이 센서 연결</td><td>16.7%</td><td>팀원</td></tr>
<tr><td>정보통신
공학과</td><td>4</td><td>201720769</td><td>신선영</td><td>하드웨어와 구성도·구조도 설계 및 계획, IoT 시스템 구현, 발표, YOLO 가중치 파일 제작, 버스 번호판 데이터셋 제작, 발표</td><td>16.7%</td><td>팀원</td></tr>
<tr><td>정보통신
공학과</td><td>4</td><td>201822181</td><td>장하영</td><td>S/W 데이터 처리, 보고서 작성, server 소스코드 제작, raspberry pi 클라이언트 환경 구성, raspberry pi 센서 연결 (LiDAR, Camera), 라즈베리 파이 환경 구성, 외관 제작 및 디자인</td><td>16.7%</td><td>팀원</td></tr>
<tr><td>정보통신
공학과</td><td>4</td><td>201822437</td><td>최유진</td><td>버스번호 인식 후 음성출력 (tts), 버스번호판 데이터셋 제작, YOLO 가중치 파일 제작, 보고서 작성, 라즈베리 파이 클라이언트 환경 구성, 외관 제작 및 디자인</td><td>16.7%</td><td>팀원</td></tr>
<tr><td>작품 제작
개 요</td><td colspan="6">우리나라 전체 장애인 인구는 약 2,517,000명, 그중 시각장애인은 2019년 기준 약 253,000명이라고 한다. 그들의 수에 비하여 시각장애인을 위한 시설이 제대로 갖춘 장소를 찾기는 어렵다. 일상생활에 어려움을 겪는 시각장애인들에게 현재보다 더 나은 기회와 환경을 제공하고자 이와 같은 아이디어를 구상했다. YOLO 기반의 머신러닝을 활용하여 현재 대중교통상황과 도로 위의 상황을 이미지 데이터에서 음성 데이터로 변환하여 출력하고, 라이다 센서와 카메라로 얻은 시각적 정보를 수학적 계산과 YOLO 기반의 머신러닝을 통해 주변의 물체 및 물체와의 거리를 탐지하여 구체적인 상황 설명을 목표로 한다. 시각 정보의 활용이 어려운 사람들에게 음성 정보를 제공함으로써 외부에서의 다양한 기회와 낯선 장소에서의 활동 영역의 확장이 가능한 시스템을 제작했다.</td></tr>
</table>

작품 제작 결과 요약	YOLO 알고리즘을 통하여 dataset으로 딥러닝 학습을 진행하였다. 이 학습을 통하여 결과로 얻은 가중치 파일을 사용하여 도로 위에서 자전거, 자동차 등의 위험 물체를 인식하였다. 이 상황에서 미리 지정한 알맞은 정보를 음성으로 출력하였다. 또한, 충전이 가능한 배터리를 장착하여 휴대가 가능하도록 하였다.
추후 진행 사항	YOLO 알고리즘을 활용하기 위해 dataset의 수를 증가시켜 정확도를 높인다. 또한, 앱 개발을 통해 음성 안내 지도를 제작하여 User Interface를 제공한다.

- 목 차 -

1. 개발 동기 및 목적, 필요성

현재 우리나라 시각장애인은 2019년 기준 약 25만 3천 명이다. 시각장애인들은 활동하는 데에 많은 제약이 따르며 특히 외출 시 이동하는 도중에 다양한 위험에 노출된다. 예를 들어 도보가 없는 차도로 걸어가거나 인도에 오토바이, 자전거, 사람 등이 갑작스레 나타나는 경우 피하기 힘들다. 이 외에도 건물의 출입문의 위치를 찾는 데에 어려움을 느끼고 있다. 이러한 상황들을 줄이고자 시각장애인들에게 눈이 되어 줄 스마트 목걸이를 제작하려 한다.

시중 제품의 문제점을 보완하고 새로운 기능을 추가하여 제작한다. 현재 시각장애인을 위해 시각정보를 대신 전달해 줄 수 있는 기술 중 화이트 아이(White eye)가 있다. 화이트 아이(White eye)는 지팡이에 3차원의 공간인식 인터페이스 장치를 접목시켜 개발한 것으로 지팡이에 부착된 센서가 주변의 물체들을 먼저 인식해 진동으로 알려줌으로써 장애인들의 부상 위험을 낮출 수 있는 장점이 있다. 우리는 이 방식에 정확한 위험 물체 감지 기능을 추가하여 시각장애인의 주도적인 이동을 통한 편의를 제공한다.

2. 설계 구성요소

목표설정	- 시각장애인이 외출 시, 위험에 노출되지 않도록 도움을 주는 제품 개발을 목표로 한다.
합성	- 제품에 카메라, Lidar를 부착하여 사용자의 정면으로 다가오는 물체를 인식할 수 있게 하였고, 이를 음성으로 출력하여 사용자가 위험을 감지하고 피할 수 있게 하였다.
분석	- 제품은 크게 하드웨어 부분과 소프트웨어 부분으로 나누어 설계한다.
제작	- 케이스에 카메라와 Lidar 센서를 넣어 거치대에 장착하고, 라즈베리파이4를 하드웨어에 부착하여 완성한다.
시험	- 시험 결과 기존에 사용하려 했던 카메라의 성능이 좋지 않아 물체 인식이 정확하지 않다. - 시험 시 사용한 케이스는 막혀있어 부품이 발열 시에 쉽게 온도가 내려가지 않았다. 또한 부피가 커 사용에 불편함이 있다. - Lidar 센서는 거리를 비교적 정확하게 인식하였다.
평가	- 물체 인식이 정확하게 되도록 향상된 성능의 카메라로 교체한다. - 부품에 쿨러를 장착하고 케이스를 뒷부분이 뚫려있는 것으로 교체한다.
결과도출	- 평가를 통해 인지한 부족했던 부분을 보안하여 제품의 정확도를 높인 제품을 제작했다.

3. 설계 제한적/기능적 요구 조건, 학습 성과

1) 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
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 발생 시 저장된 이미지 데이터를 통하여 CCTV의 역할 수행
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, camera와 같은 지정된 표준의 장치를 사용

2) 학습성과

지식응용	수학, 기초과학, 공학지식과 정보기술을 응용 할 수 있는 능력
	지식은 강의를 통한 일방적 습득이 불가능하므로, 작품 설계 실습을 통해 강의 내용을 이해하고 본인의 것으로 만들 수 있는 기회를 제공 받음
분석실험	자료를 이해하고 분석할 수 있는 능력 및 실험을 계획하고 수행 할 수 있는 능력
	raspberry pi와 제한된 환경에서 라이다 센서의 성능실험을 진행하며 다양한 각도로 문제를 해결할 수 있으며 과제의 결과를 도출
설계능력	현실적 제한조건을 반영하여 시스템 및 관련 프로그램을 설계 할 수 있는 능력
	- raspberry Pi의 패킷 설치 과정, YOLO 알고리즘에 대한 이해도 향상 - 오류 대처 능력 향상 - 제한된 비용을 효율적으로 사용하여 제품 제작
문제해결	전자정보통신공학 문제를 인식하며, 이를 공식화하고 해결할 수 있는 능력
	기존 설계 방식은 raspberry zero w를 이용하여 구현하는 것이었으나 Lidar 센서 설치 과정에서 발생하는 cpu부족 문제를 해결하기 위해 raspberry Pi 3, raspberry Pi 4도 함께 사용
공학 실무	전자정보통신공학 실무에 필요한 기술, 방법, 도구 등을 사용할 수 있는 능력
	프로젝트를 통해 기본 개념과 원리를 이해하고 그것을 적용, 체득할 수 있는 기회를 얻음
팀워크	복합 학제적 팀에서 구성원의 역할을 해낼 수 있는 팀웍 능력
	역할을 나누어 프로젝트를 진행하는 과정에서 자신의 역할이 아닌 다른 부분에서도 도움을 주며 협력
의사소통	효과적으로 의사를 전달 할 수 있는 능력
	회의 내에서 서로의 의견을 조율하고, 각각의 의견에 대한 피드백을 통해 상대방의 역할과 자신의 문제점 보완

4. 개념설계 및 상세설계

1) 개념설계

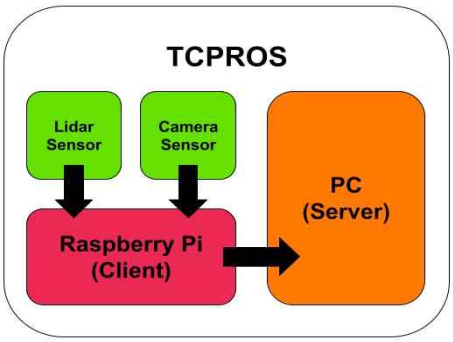


그림 1. 시스템 구성도

① 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

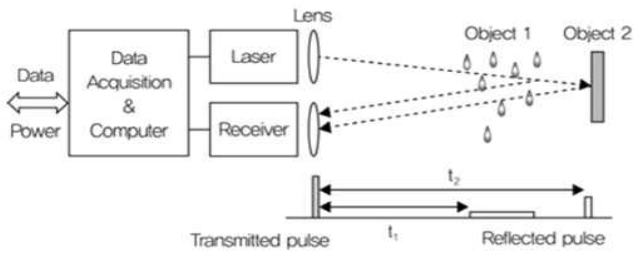


그림 2. ToF 센서

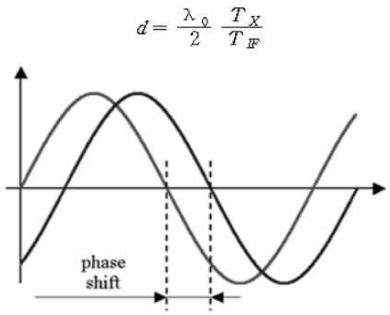


그림 3. Phase Shift 방식

② 소프트웨어

- 도보 위 위험 물체 식별 기능

· Camera에서 얻은 real time 영상을 사전에 학습된 가중치 파일을 사용한 YOLO 알고리즘을 통하여 위험 물체를 식별
· 학습된 가중치 파일은 이미지 파일 35만 장 중 1만 4천여 장을 선별하여 제작
· Lidar Sensor를 사용하여 이미지에서 식별된 위험 물체와 Lidar Sensor와의 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

· Topic 방식의 통신을 위해 메시지 파일 작성, Server/Client 노드 작성, 실행된 노드들의 통신 상태 확인 등의 과정을 거침

(ROS를 사용하여 TCP/IP 방식의 TCPROS 프로토콜로 Topic 방식으로 통신)

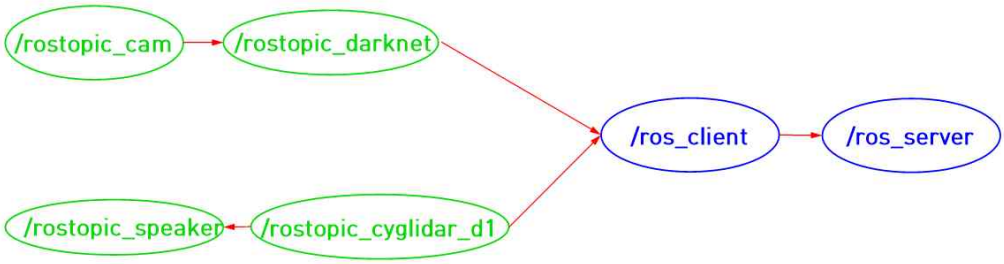


그림 4. RQT 그래프

2) 상세설계

① 하드웨어 기능

- Raspberry Pi zero w

· 배터리 사용이 가능한 휴대용 마이크로프로세서

· Raspberry Pi zero와 달리 특별한 추가 모듈설치 없이 Bluetooth와 Wifi를 사용한 통신이 가능

· 학습된 YOLO의 weight파일을 기반으로 이미지데이터에서 원하는 정보를 식별

· 모바일과 Bluetooth로 통신하여 위치와 상황에 적합한 정보를 송수신

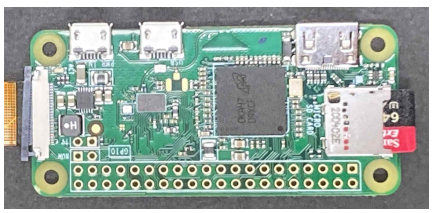


그림 5. Raspberry Pi zero w

- Raspberry Pi 4B

· 1.5GHz 쿼드코어 CPU와 60H에서 4k 영상을 지원하며 최대 메모리는 4GB까지 탑재 가능

· 학습된 YOLO의 weight 파일을 기반으로 이미지 데이터에서 원하는 정보를 식별

· Bluetooth 사용을 위해 필요한 라이브러리 설치 필요

· micro HDMI 사용, USB Type-C 케이블로 전원공급 가능

- TFmini plus Lidar Sensor

· 크기 35mm X 21mm X 18.5mm, 무게 12g 정도로 휴대에 용이

· 측정 가능 범위는 0.1m~12m까지 가능

· 레이저가 반사되어 돌어오는 시간을 계산하여 주변 사물의 거리를 측정

- 2D/3D Dual Solid-State ToF LiDAR

· 37.4mm x 37.4 mm x 24.5 mm의 크기와 28g의 무게로 휴대에 용이
· 2D로 사용 시 200mm~8000mm의 측정이 가능하며 120도의 시야각을 갖고 15Hz의 속도 측정 가능
· 레이저 센서와 같은 원리로 빛이 돌아오는 시간을 계산하여 전방의 물체 유무를 측정

- LI-USB30-P031X

· 2592 x 1944, 1920 x 1080, 800 x 600, 640 x 480의 동영상 저장 가능
· 26mm x 26mm x 38mm의 크기, 22g의 무게로 휴대에 용이
· 글로벌 셔터 기능을 갖춘 Aptina MT9M021 Color / Mono CMOS 센서를 기반으로 720p HD RAW 데이터 실시간 스트리밍이 가능

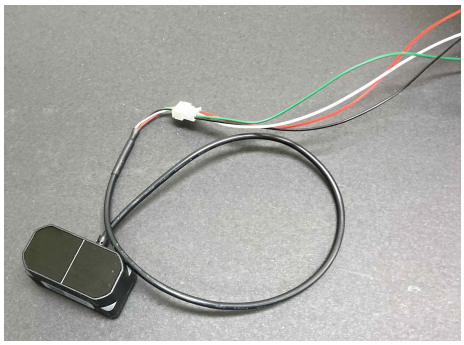


그림 6. Lidar 센서

CygLiDAR	측정 범위 (Detection range)	2D : 200mm ~ 8,000mm 3D : 50mm ~ 2,000mm (*DRM)
	오차 범위 (Distance accuracy)	±1%
	측정 분해능 (Resolution (mm 단위 측정))	2D : 1° (Angle) 3D : 160 x 60 (Pixel)
	시야각 (FOV : Field of View)	2D/3D Horizontal : 120° 3D Vertical : 65°
	파장 (Wavelength)	Laser Diode : NIR 808nm LED : NIR 808nm
	측정 속도 (Measuring speed)	2D : 15Hz 3D : 15Hz
Raspberry pi zero W	1GHz, single-core CPU	
	512MB RAM	
Camera	5MP OV5647 sensor	
	2592 x 1944 still picture resolution	
	Support 1080p30, 720p60 and 640x480p60/90 video record	
	Dimension: 60mm x 11.5mm x 5mm	
	CCD size: 1/4inch	
	Field of View: 72.4 degree	

표 1. YOLO 기반 물체 검출 실험에 사용된 환경

② 소프트웨어 기능

- Client(Raspberry Pi)

pkg : detector_cam.py

```
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')  #weights, cfg파일 경로

    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])


                    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()
```

- **Server(PC)**

pkg : rosserial_server.py

```
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')  #weights, cfg파일 경로

    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])


                    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()
```

5. 과제 해결 방안 및 과정

각 기술을 구현하는데 필요한 프로그램과 소스코드들을 조사한 후, 프로그램을 설치하고 소스코드를 실행하는 과정에서 몇 가지 오류가 발생하여 수정하고 해결하였다.

1) 목걸이 구조의 문제점

초기 구상은 raspberry pi zero W를 사용하여 경량화한 목걸이 형태의 모습이었으나, 착용 후 이동 시 흔들림이 심하여 카메라를 통한 객체 인식이 어려워졌다. 그래서 흔들림을 줄이기 위해 몸과 밀착될 수 있는 벨트 형태로 대체 제작하여 사용자가 이동 시에도 물체 인식을 정확히 할 수 있도록 하였다.

2) Raspberry pi zero W 사양

ROS를 설치하는 데 필요한 사양에 비해 Raspberry pi zero W의 CPU 사양이 낮아 Raspberry pi 4로 대체하였다. 앞서 제작 형태를 목걸이에서 벨트로 바꾸었기 때문에 경량화를 위해 선택하였던 Raspberry pi zero W를 사용하지 않아도 문제가 없었다.

3) 버스번호 인식

현재 우리나라의 차량 번호판은 개인정보로 분류되어 data 수집이 어려워 외국 차량번호판 237장의 이미지 dataset을 사용하여 yolo로 학습을 진행하였다. 하지만 국내 시내버스 번호판과 외국의 차량 번호판은 규격이 달라 인식에 어려움이 있었다. 국내 버스 번호판을 임의로 생성하였으나 만일 버스정류장에 두 대 이상의 버스가 도착 시에 버스 번호판이 보이지 않아 사용이 어렵다는 문제가 있었다. 최종적으로 직접 수집한 약 60개의 data의 버스 번호를 노선 별로 정면과 측면으로 2개의 클래스를 설정하여 dataset을 학습시켰고, 정확도가 50%인 결과를 도출해낼 수 있었다. 이는 더 많은 data를 모아 학습시키면 정확도가 더 향상될 것으로 예상된다.

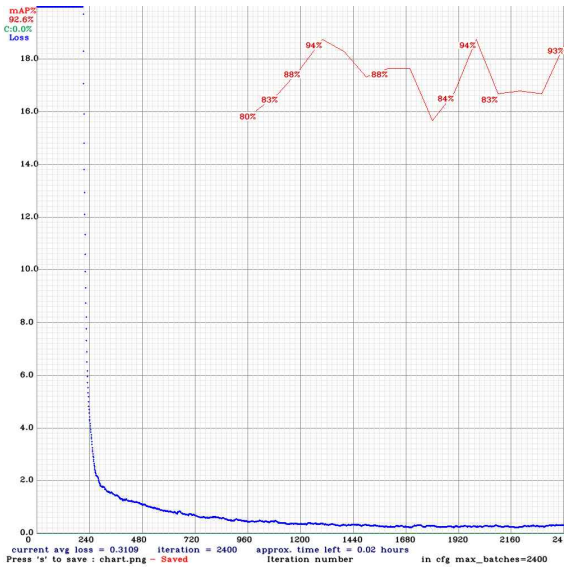


그림 7. YOLO 가중치 학습

Class Balance	Classification result (No)
License Plate	237

표 2. Class Type

그림 8. 사업용 대형 등록 번호판 기준

2) 위험 물체 인식

① 도보 위 위험 물체 dataset으로 YOLO 가중치파일 제작

- darknet을 사용하여 11개 클래스와 15,000개의 이미지 dataset을 학습
- YOLO 가중치파일 학습 결과[표3]

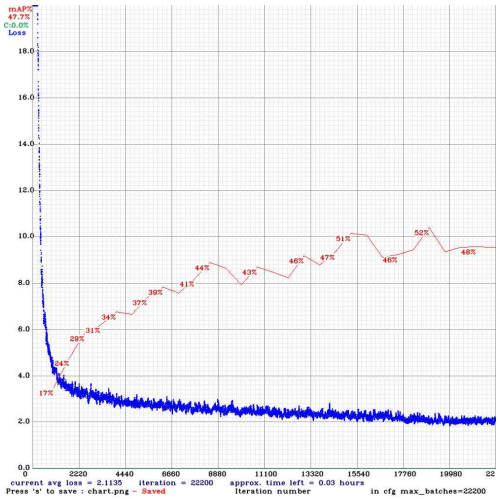


그림 9. YOLO 가중치 학습

Class Balance	Classification result (No)
car	64,399
pedestrain	10,806
trafficLight-Red	6,870
trafficLight-Green	5,465
truck	3,623
trafficLight	2,568
biker	1,864
trafficLight-RedLeft	1,751
trafficLight-GreenLeft	310
trafficLight-Yellow	272
trafficLight-YellowLeft	14

표 3. Class Type

② YOLO가중치 파일을 사용하여 위험 물체 탐지 시 음성 경고 시스템 제작

- gtts 패키지를 사용하여 class 탐지 시 bluetooth로 연결된 스피커에서 음성으로 출력

detector_warning_gtts.py

```
import cv2
import argparse
import numpy as np
from gtts import gTTS
from PIL import Image
import pytesseract
import os

def check_num(text, thenum):
    if(thenum == text) : #thenum in text
        tts = gTTS(text= OCR_text, lang='en')
        tts.save("anpr.mp3")
        os.system("mpg321 anpr.mp3")
        print(OCR_text)
        cv2.imshow('crop',image)

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]
    #cv2.imshow('crop',image)
    OCR_text = pytesseract.image_to_string(image, lang = 'eng')
    print(OCR_text)
    check_num(OCR_text, thenum)

image = cv2.VideoCapture(0)
classes = None
with open('/home/pi/ex1/darknet/busapp/tiny/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/pi/ex1/darknet/busapp/tiny/yolov3-tiny_final.weights',
'/home/pi/ex1/darknet/busapp/tiny/yolov3-tiny.cfg')  #weights, cfg파일 경로
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])

                    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))
```

```
    cv2.imshow('result', img)
    if cv2.waitKey(1) == ord('q'):
        break

cv2.destroyAllWindows()
```

③ 라즈베리파이 서버 통신

- 라즈베리파이에서 YOLO가중치 파일을 사용하여 객체 탐지 시, 극심한 발열 발생 및 종료되는 현상 발생
- 라즈베리파이 카메라를 통해 얻은 이미지 데이터를 서버로 송신하여 실습

rpi_detecor_warning_gtts.py

```
import cv2
import socket
import numpy as np
import argparse
from gtts import gTTS
from PIL import Image
import pytesseract
import os

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('xxxx.xxxx.xxxx.xxxx', xxxx))

cam = cv2.VideoCapture(0)
encode_param = [int(cv2.IMWRITE_JPEG_QUALITY), 90]

while True:

####add detector_warning_gtts.py source code####
    result, frame = cv2.imencode('.jpg', frame, encode_param)
    data = numpy.array(frame)
    stringData = data.tostring()
    s.sendall((str(len(stringData))).encode().ljust(16) + stringData)

cam.release()
```

④ ROS Melodic 환경 구성

- Lidar Sensor값을 측정하기 위해 Raspberry Pi zero W에 ROS 설치
- zero W의 CPU 부족으로 인해 Raspberry pi 4로 교체
- SD 카드에 ubuntu LTS 18.04.5 버전을 구운 후 ROS-melodic과 openCV 설치
- Raspberry Pi 4에 ROS 설치한 후 Lidar와 연결하려고 했으나 구매한 lidar 회사에서 제공한 오픈소스와 lidar USB 드라이버 설치에 오류 발생
- ROS를 더 지원해 줄 수 있는 가상머신 VirtualBox를 다운받아 시도.

⑤ ROS 시리얼과 라즈베리 파이 간의 통신

- PC와 Raspberry Pi 간의 통신을 위해 Raspberry Pi에 다시 ROS-Melodic과 rosserial 패키지 설치.
- 클라이언트 전송 값 손실을 최소화하기 위해 Ubuntu 환경에 ROS 설치하였으나 시중에 제공되는

ROS open source는 ROS버전 Melodic이 아닌 Kinetic이었음.
- ROS는 Arduino를 지원하다보니 라즈베리파이 환경에 기존에 해결되지 않았던 Error 발생
→ ROS C++, Python 언어를 추가적으로 학습해 lidar 센서 연결 및 Open Source 수정작업 수행하여 Build Error 해결

6. 기대효과 및 향후 계획

1) 기대효과

기존의 Smart Cane(시각장애인을 위한 스마트 지팡이)에서 아이디어를 착안했다. 배터리를 부착하여 휴대가 가능하고, 이는 시각장애인들이 장시간 착용했을 때도 부담이 가지 않을 것이다. 시중에서 널리 유통되는 제품에 기술을 추가하여 변형한 만큼 시각장애인들에게 장소를 이동하는 데에 더 많은 편의를 제공할 수 있을 것이다. 또한 예상치 못한 사고 발생 시 저장된 이미지 데이터를 통하여 블랙박스의 역할을 수행할 것으로 기대한다.

시각장애인들을 위한 비슷한 제품들이 현재도 계속해서 개발되고 있다. 다양한 기술들이 접목되어 점점 발전된 시스템이 개발될 수 있도록 시각 장애인들의 삶의 질을 높여주는 효과를 기대할 수 있다.

2) 향후 계획

앞서 설계한 작품에 다음과 같은 기능들을 추가하면 더욱 시각장애인들의 외출에 도움이 될 것이라 예상된다.

먼저, 이미지 data의 수가 적어 학습시킨 data의 정확도가 낮은 문제를 해결한다. 국내 시내버스 노선별 번호를 인위적으로 생성하여 dataset을 제작한 후 학습시킨다. 이 학습시킨 YOLO 알고리즘을 이용하여 버스정류장에 연속적으로 버스가 도착했을 때 어떤 순서로 도착하였는지 알려주는 기능을 추가할 것이다.

또한, 앱 개발을 통해 User Interface를 제공하고 음성 안내 지도를 제작할 계획이다. 앱 내에는 경로 탐색과 경로 안내, 물체 식별 안내 등의 기능을 추가할 것이다.

7. 작품 결과 분석

YOLO 알고리즘을 통하여 dataset으로 딥러닝 학습을 진행하였다. 이 학습을 통하여 결과로 얻은 가중치 파일을 사용하여 도로 위에서 자전거, 자동차, 행인, 신호등, 트럭, 길의 구덩이 등의 위험 물체를 인식하였다. 이 상황에서 미리 지정한 알맞은 정보를 음성으로 출력하였다. 카메라를 통해 보이는 객체를 인식하고 위치의 값을 Lidar 센서의 렌즈 위치에 맞게 행렬 변환시켜 객체 간 거리를 측정하였다. 이 데이터 값을 클라이언트에서 서버로 전성하고, 서버로 전송한 결과를 서버의 gui를 통하여 출력하였다. 클라이언트에서는 미리 지정한 위험사항을 카메라와 라이다로 예측시 사용자에게 음성으로 위험을 인지할 수 있게 출력하였다. 서버와 클라이언트의 환경은 모두 우분투이며 ROS 시리얼을 통해 통신하였다. 하드웨어는 라즈베리 파이에 라이다 센서와 카메라 센서를 USB 포트로 연결하였다. 이 때, 각 부품은 크기가 작고 가벼운 라즈베리파이 제로 W, TFmini plus Lidar 센서, 파이 캠 등을 사용하여 이동시 불편함이 없도록 제작하였다. 또한, 충전이 가능한 배터리를 장착하여 휴대가 가능하도록 하였다.

8. 비용분석 및 팀원 간 역할분담

1) 비용 분석

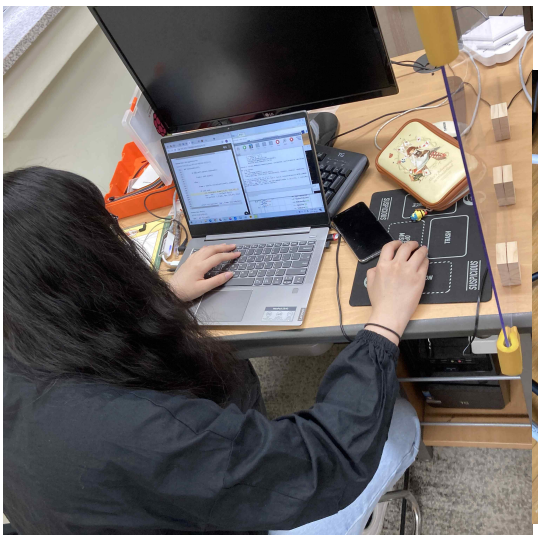
연번	품 명	수 량	단 가	금 액	요구목적 및 사용처
1	RaspberryPi Li-ion Battery Hat 5V	1	28,200	28,200	배터리
2	SD 64GB (샌디스크 익스트림 프로 MLC 64GB)	3	17,500	52,500	라즈베리 파이 SD카드
3	라즈베리 파이 제로 W	2	28,500	57,000	라즈베리 파이 제로 W
4	라즈베리 파이 제로, 제로 W용 5백만 화소 카메라	1	16,500	16,500	카메라
5	2D/3D Dual 라이다 거리 측정 센서 Solid-State ToF LIDAR	1	187,000	187,000	라이다 센서
6	Coms 14500 충전지 리튬이온배터리 - 800mAh	1	6,300	6,300	배터리
7	254mm 2x20 핀 분리 라즈베리 파이 제로 GPIO커넥터	1	8,300	8,300	GPIO 커넥터
8	RaspberryPi4 스타트 키드	1	124,300	124,300	라즈베리 파이4
9	TFmini Plus LIDAR	1	57,800	57,800	라이다 센서
10	라즈베리파이4 18650 5V 4A UPS 충전모듈	1	34,100	34,100	배터리
11	삼성 고방전 18650 리튬이온 전지	2	10,850	21,700	배터리
12	샤프 아두이노 적외선 거리 센서	1	15,900	15,900	적외선 센서
13	라즈베리파이 3B+/4B 쿨링팬	1	2,100	2,100	쿨링팬
14	액션캠/고프로 가슴 스트랩	1	10,580	10,580	하드웨어
	합 계	18	547,930	622,280	

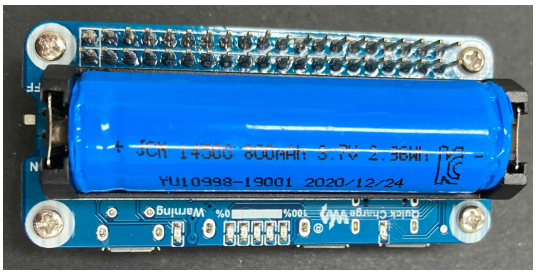
2) 역할 분담

구분	성명	역할
팀장	맹하늘	프로젝트 총괄, YOLO 가중치 파일 제작, raspberry pi 개발 환경 구축, 하드웨어와 구성도·구조도 설계 및 제작, PPT 제작 및 결과 보고서 작성
팀원	김하얀	raspberry pi 클라이언트 환경 구성(ros, catkin, melodic, opencv 등), raspberry pi 센서 연결(LiDAR, Camera), 버스번호판 데이터 음성출력(tts), server 소스코드 제작
팀원	김현정	라즈베리파이 소스코드 제작, Client 소스코드 제작, YOLO 가중치 파일 제작, 위험물체 인식 데이터셋 제작, ROS 환경 구성, 라즈베리파이 센서 연결
팀원	신선영	하드웨어와 구성도·구조도 설계 및 계획, IoT 시스템 구현, 발표, YOLO 가중치 파일 제작, 버스 번호판 데이터셋 제작, 발표
팀원	장하영	S/W 데이터 처리, 보고서 작성, server 소스코드 제작, raspberry pi 클라이언트 환경 구성, raspberry pi 센서 연결(LiDAR, Camera), 라즈베리 파이 환경 구성, 외관 제작 및 디자인
팀원	최유진	버스번호 인식 후 음성출력 (tts), 버스 번호판 데이터셋 제작, YOLO 가중치 파일 제작, 보고서 작성, 라즈베리 파이 클라이언트 환경 구성, 외관 제작 및 디자인

9. 작품사진 및 활동사진

그림 10. 차 번호 인식 음성출력 TTS






10. 참고문헌

· 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