졸업 작품 보고서

지능형 안전 관리 시스템

2024년 12월

서경대학교 컴퓨터공학과

전영민

목차

1. 서론

1-1 배경 및 필요성

현대 사회는 급속한 산업화와 도시화로 인해 각종 안전사고 위험에 노출되어 있다. 특히 화재, 붕괴, 폭발 등의 사고는 대규모 인명 피해와 재산 손실을 초래할 수 있으며, 이러한 사고를 예방하고 효과적으로 대응하기 위한 안전 관리 시스템의 중요성이 더욱 강조되고 있다. 기존의 안전 관리 시스템은 주로 CCTV, 화재 감지 센서 등의 장비를 활용하여 사고를 감시하고, 담당자가 이를 직접 확인하여 대응하는 방식으로 운영되어 왔다. 그러나 이러한 시스템은 몇 가지 한계점을 가지고 있다. 첫째, 사람의 지속적인 관찰과 수동적인 대응에 의존하기 때문에 실시간으로 사고를 감지하고 신속하게 대응하기 어렵다. 둘째, 담당자의 주의력 부족이나 실수로 인해 사고를 놓칠 가능성이 존재한다. 셋째, 대규모 시설이나 복잡한 환경에서는 효율적인 감시 및 관리가 어렵다.[1] 최근 인공지능, 컴퓨터 비전, 딥러닝 기술의 발전은 이러한 한계점을 극복하고 보다 효율적이고 지능적인 안전 관리 시스템 구축을 가능하게 하고 있다. 특히 객체 인식, 행동 분석, 예측 기술을 활용하여 실시간으로 위험 상황을 감지하고 자동으로 대응하는 시스템 개발에 대한 연구가 활발히 진행되고 있다. 본 연구에서는 이러한 기술적 발전을 바탕으로, YOLO 모델[2]을 이용한 화재 감지와 Mediapipe Pose Solution[3] 및 GRU 모델[4]을 이용한 사람의 행동 추정 기능을 갖춘 지능형 안전 관리 시스템을 개발하고자 한다. 이를 통해 화재 발생 및 넘어짐과 같은 위험 상황을 실시간으로 감지하고, 자동으로 경고 메시지를 전송하여 신속한 대응을 가능하게 함으로써 안전사고 예방 및 피해 최소화에 기여하고자 한다.

1-2 해결하고자 하는 문제

본 연구에서는 기존 안전 관리 시스템의 한계점을 극복하고, 보다 효율적이고 지능적인 안전 관리를 가능하게 하는 시스템을 개발하고자 한다. 이를 위해 다음과 같은 문제들을 해결하고자 한다.

- 실시간 화재 감지: 화재는 초기 단계에서의 신속한 감지 및 진압이 매우 중요하다. 기존의 화재 감지 센서는 연기나 열을 감지하는 방식으로, 화재 발생 후 일정 시간이 지나야 감지가 가능하다는 단점이 있다. 본 연구에서는 YOLO 모델을 이용하여 영상 데이터를 실시간으로 분석하고 화재를 조기에 감지하는 시스템을 구현하여, 초기 대응 시간을 단축하고 피해를 최소화하고자 한다.
- 사람의 행동 추정 및 위험 상황 감지: 산업 현장이나 공공장소에서 발생하는 넘어짐, 쓰러짐 등의 사고는 심각한 부상으로 이어질 수 있다. 본 연구에서는 Mediapipe Pose Solution을 이용하여 사람의 골격 정보를 추출하고, GRU 모델을 이용하여 행동 패턴을 분석함으로써 넘어짐과 같은 위험 상황을 실시간으로 감지하는 시스템을 구현하고자 한다.
- 효율적인 경고 시스템: 위험 상황 발생 시 신속한 대응이 이루어질 수 있도록 효율적인 경고 시스템을 구축해야 한다. 본 연구에서는 SMTP 모듈을 이용하여 관리자에게 이메일을 전송하는 경고 시스템을 구현하여, 사고 발생 시 즉각적인 대응이 가능하도록 지원하고자 한다.

본 연구는 위 문제들을 해결함으로써, 인공지능 기반의 지능형 안전 관리 시스템을 구축하고, 안전사고 예방 및 피해 최소화에 기여하는 것을 목표로 한다.

1-3 작품 개발 목적

본 연구는 앞서 제시된 문제들을 해결하고, 인공지능 기술을 활용하여 효율적이고 지능적인 안전 관리 시스템을 구축하는 것을 목표로 한다. 구체적인 연구 목표는 다음과 같다.

- YOLO 모델 기반의 실시간 화재 감지 시스템 개발: YOLO v5 모델을 이용하여 영상 데이터에서 화재를 실시간으로 감지하는 시스템을 개발한다. 다양한 환경에서 촬영된 화재 영상 데이터를 학습시켜 모델의 정확도와 신뢰성을 향상시킨다.
- Mediapipe 및 GRU 모델 기반의 행동 추정 시스템 개발: Mediapipe Pose Solution을 이용하여 사람의 골격 정보를 추출하고, GRU 모델을 이용하여 행동 패턴을 분석하여 넘어짐과 같은 위험 상황을 감지하는 시스템을 개발한다. 다양한 행동 패턴을 학습시켜 모델의 정확도를 높이고 오탐을 최소화한다.
- FFmpeg 기반의 실시간 데이터 전송 및 처리 시스템 구현: 라즈베리파이 보드에 연결된 웹캠에서 촬영된 영상 데이터를 FFMPEG를 이용하여 파이썬 코드로 실시간 전송하고 처리하는 시스템을 구현한다. 효율적인 데이터 전송 및 처리를 통해 시스템의 성능을 향상시킨다.
- SMTP 모듈 기반의 이메일 경고 시스템 개발: 화재 감지 또는 위험 행동 감지 시, SMTP 모듈을 이용하여 관리자에게 이메일로 경고 메시지를 전송하는 시스템을 개발한다. 이메일에는 사고 발생 시간, 위치 정보 등을 포함하여 신속한 상황 파악 및 대응을 지원한다.

본 연구를 통해 개발된 지능형 안전 관리 시스템은 기존 시스템의 한계점을 극복하고, 다양한 환경에서 안전사고 예방 및 피해 최소화에 기여할 것으로 기대된다.

2. 관련 연구

2-1 유사한 연구, 제품, 서비스

최근 인공지능 기술의 발전과 함께 화재 감지 및 행동 인식 분야에서 다양한 연구 및 제품, 서비스가 개발되고 있다.

2-1-1 화재 감지 분야

영상 기반 화재 감지

- Vision AI[5]: Google Cloud에서 제공하는 Vision API는 이미지 및 비디오에서 객체를 감지하고 분류하는 기능을 제공하며, 화재 감지에도 활용될 수 있다.
- FireNet[6]: CNN 기반의 화재 감지 모델로, 초기 화재의 특징을 학습하여 정확도를 높이는 데 중점을 두고 있다.
- YOLOv5: 객체 감지 알고리즘으로, 실시간으로 이미지 및 비디오에서 객체를 감지할 수 있으며, 화재 감지에도 효과적으로 활용될 수 있다.

센서 기반 화재 감지

- IoT 기반 화재 감지 센서: 온도, 연기, 가스 등을 감지하는 센서를 통해 화재 발생을 감지하고, IoT 네트워크를 통해 경고 메시지를 전송하는 시스템이 개발되고 있다.

2-1-2 행동 인식 분야

- OpenPose[7]: 오픈 소스 기반의 실시간 다중 사람 포즈 추정 라이브러리로, 사람의 관절 위치를 정확하게 추출하여 다양한 행동 분석에 활용될 수 있다.
- PoseNet[8]: TensorFlow.js에서 제공하는 사람 포즈 추정 모델로, 웹 브라우저에서 실시간으로 사람의 포즈를 추정할 수 있다.
- Mediapipe: Google에서 개발한 오픈 소스 프레임워크로, 다양한 컴퓨터 비전 작업을 위한 솔루션을 제공한다. 본 연구에서는 Mediapipe의 Pose solution을 사용하여 사람의 골격 정보를 추출한다.

2-2 졸업작품의 차별성

본 연구는 기존 연구 및 제품, 서비스와 비교하여 다음과 같은 차별성을 가진다.

- 다중 위험 상황 감지: 화재 감지와 사람의 행동 추정 기능을 결합하여 화재, 넘어짐 등 다양한 위험 상황을 동시에 감지하고 대응할 수 있다.
- 실시간 알림 기능: SMTP 모듈을 이용하여 위험 상황 발생 시 즉시 이메일로 알림을 전송하여 신속한 대응을 가능하게 한다.
- 확장 가능성: 다양한 센서 및 기능을 추가하여 시스템을 확장하고, 다양한 환경에 적용할 수 있다.

본 연구는 이러한 차별성을 통해 기존 시스템의 한계를 극복하고, 보다 효율적이고 지능적인 안전 관리 시스템을 구현하고자 한다.

3. 시스템 설계 및 개발

3-1 요구사항 분석

3-1-1 기능적 요구사항

- 화재 감지:

실시간으로 영상 데이터를 분석하여 화재 발생 여부를 정확하게 감지해야 한다.
다양한 환경에서 발생하는 화재를 감지할 수 있도록 모델을 학습해야 한다.
화재 감지 시, 화면에 경고 메시지를 표시하고 알람을 발생시켜야 한다.

- 행동 추정:

실시간으로 사람의 골격 정보를 추출하고 행동 패턴을 분석해야 한다.
넘어짐, 쓰러짐 등 위험 행동을 정확하게 인식해야 한다.
위험 행동 감지 시, 화면에 경고 메시지를 표시하고 알람을 발생시켜야 한다.

- 경고:

화재 또는 위험 행동 감지 시, 관리자에게 이메일로 경고 메시지를 전송해야 한다.
경고 메시지에는 사고 발생 시간, 위치 정보 등을 포함해야 한다.

- 데이터 전송 및 처리:

라즈베리파이 보드에서 촬영된 영상 데이터를 실시간으로 파이썬 코드로 전송해야 한다.
FFMPEG를 이용하여 효율적인 데이터 전송 및 처리를 수행해야 한다.

3-1-2 비기능적 요구사항

- 성능:

실시간 처리: 영상 데이터를 실시간으로 처리하고, 빠른 응답 속도를 제공해야 한다.
정확도: 화재 및 위험 행동을 정확하게 감지하고 오탐을 최소화해야 한다.
안정성: 시스템은 안정적으로 동작해야 하며, 오류 발생을 최소화해야 한다.

- 사용자 인터페이스:

간편한 사용: 시스템 사용법이 간편하고 직관적이어야 한다.
효율적인 정보 제공: 시스템 상태 및 감지 결과를 효율적으로 표시해야 한다.

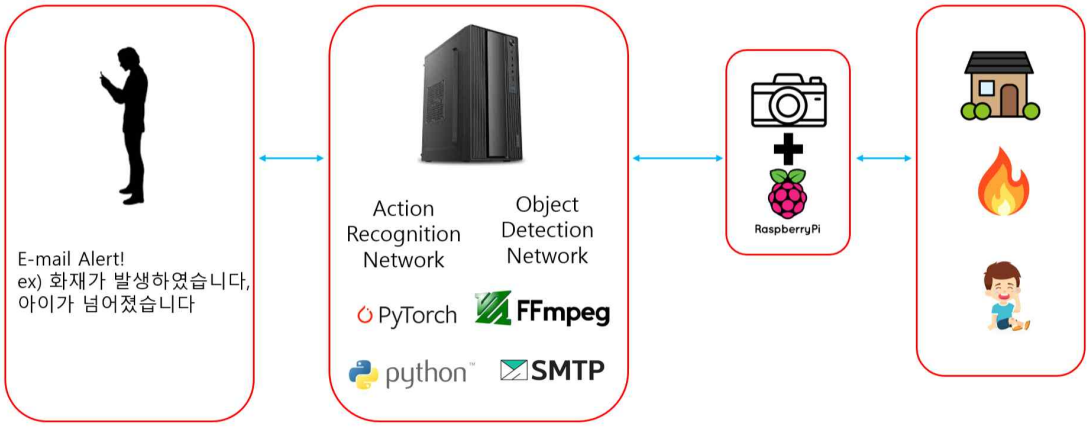
- 보안:

데이터 보호: 개인 정보 및 시스템 데이터를 안전하게 보호해야 한다.
무단 접근 방지: 시스템에 대한 무단 접근을 차단해야 한다.

- 유지보수:

용이성: 시스템 유지보수가 용이해야 하며, 문제 발생 시 신속하게 해결할 수 있어야 한다.
확장성: 향후 기능 추가 및 시스템 확장이 용이해야 한다.

3-2 시스템 아키텍처



3-3 개발도구 및 환경

본 연구에서는 지능형 안전 관리 시스템 개발을 위해 다음과 같은 개발 도구 및 환경을 사용하였다.

3-3-1 하드웨어

- 라즈베리파이 4 Model B
- 웹캠
- 메인 프레임 컴퓨터(CPU: Intel i7-12700H, GPU: RTX 3070 Ti laptop, RAM: 32GB)

3-3-2 소프트웨어

운영체제:

- 라즈비안 OS (Debian 10 Buster)
- Windows (10, 11)

프로그래밍 언어:

- Python 3.10

라이브러리:

- OpenCV: 영상 처리 및 컴퓨터 비전 라이브러리
- PyTorch: 딥러닝 프레임워크 (YOLOv5, GRU 모델 학습 및 실행)
- Mediapipe: 사람의 포즈 추정 및 행동 분석 라이브러리
- FFMPEG: 영상 데이터 전송 및 처리 도구
- SMTP: 이메일 전송 프로토콜

3-3-3 개발 환경

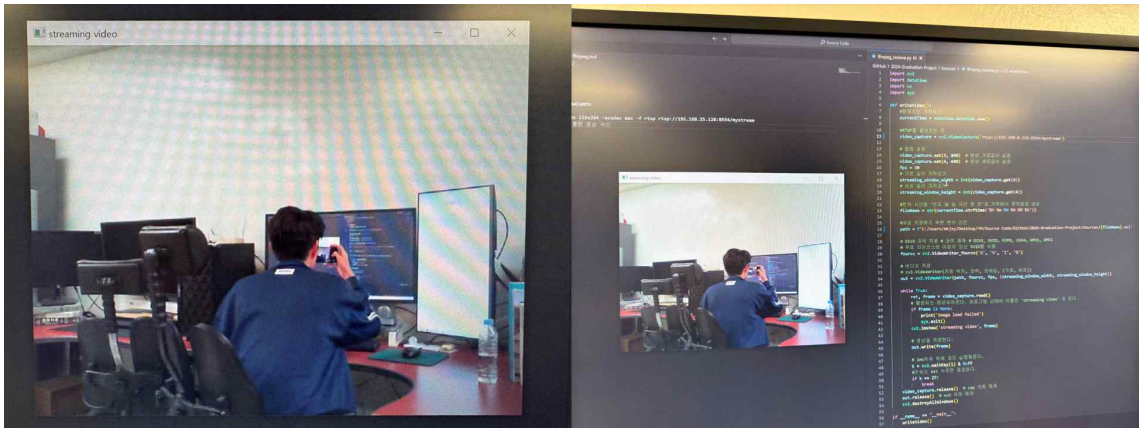
Visual Studio Code - 코드 작성 및 디버깅
Git - 버전 관리 시스템

위와 같은 개발 도구 및 환경을 사용하여 지능형 안전 관리 시스템을 구현하였다.

4. 결과

4-1 시스템 실행, 테스트 결과

4-1-1 라즈베리파이 보드 + 웹캠 연동



본 연구에서는 라즈베리파이 보드에 연결된 웹캠에서 촬영된 영상 데이터를 실시간으로 메인 프레임 컴퓨터로 전송하여 처리하기 위해 FFMPEG를 이용한 RTSP 스트리밍 방식을 사용하였다.

RTSP 서버 구축

리눅스 환경에서 간편하게 RTSP 서버를 구축할 수 있게 해주는 mediamtx 라이브러리를 사용하여 라즈베리파이 보드 상에서 RTSP 서버를 구축하였다.

FFMPEG 스트리밍

FFMPEG를 사용하여 웹캠으로부터 실시간 영상을 읽고, RTSP 서버로 스트리밍하는 기능을 구현하였다.

코드 예시

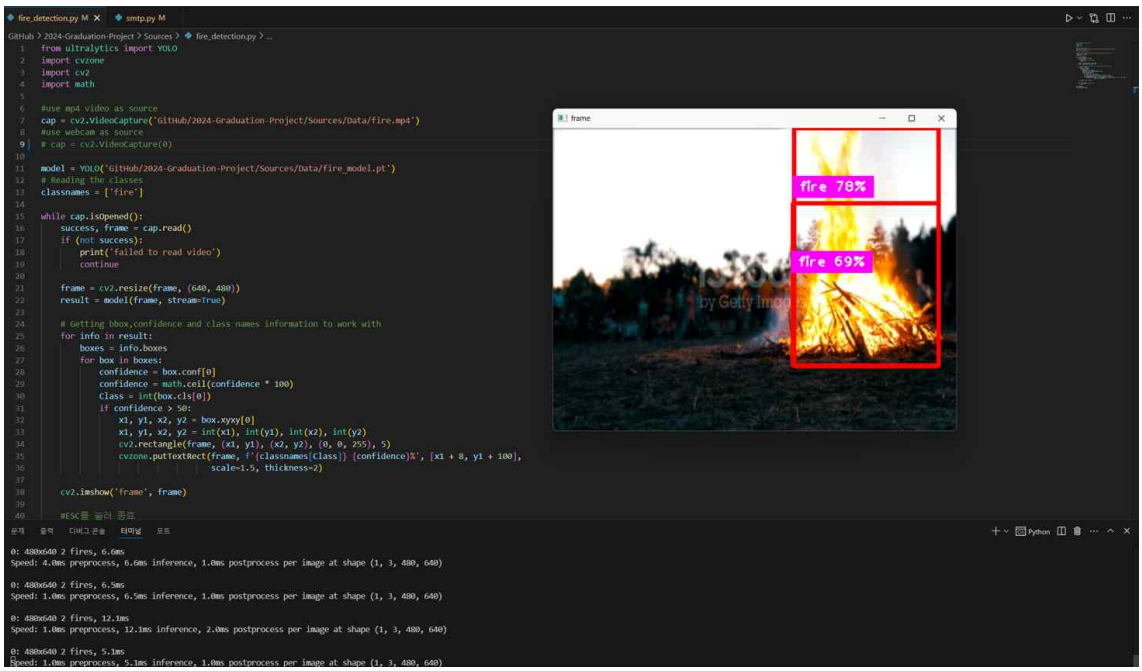
```
ffmpeg -i /dev/video0 -vcodec libx264 -acodec aac -f rtsp rtsp://192.168.25.128:8554/mystream
```

위와 같이 라즈베리파이 보드와 웹캠을 연동하고 FFMPEG를 사용하여 RTSP 통신을 구축함으로써, 메인 프레임 컴퓨터에서 실시간으로 영상 데이터를 수신하고 처리할 수 있도록 구현하였다.

4-1-2 메인 프레임 컴퓨터

라즈베리파이로부터 송출받은 실시간 영상에 대하여 화재를 감지하고 사람의 동작을 추정하여 위험 상황 발생 여부를 추론한다. 위험 상황이 발생할 경우 사용자에게 알림(E-mail)을 전송하고 실시간 추론 현황을 시각화하는 기능을 구현하였다.

4-1-3 화재 감지 모델



본 연구에서는 Yolov5 모델을 사용하여 영상에서 화재를 실시간으로 감지하는 기능을 구현하였다. Yolov5는 객체 감지 알고리즘으로, 빠른 속도와 높은 정확도를 가지고 있어 실시간 화재 감지에 적합하다.

모델 학습

화재 감지 모듈을 구현하는데 사용한 Yolov5 모델은 화재를 감지하기 위한 화염 및 연기 객체에 대하여 사전학습 된 가중치를 사용하였다.

화재 감지 모듈 구현

영상 전처리: 입력 영상을 Yolov5 모델에 맞는 크기로 조정하였다. 필요에 따라 밝기 조정, 노이즈 제거 등의 전처리 과정을 수행하였다.

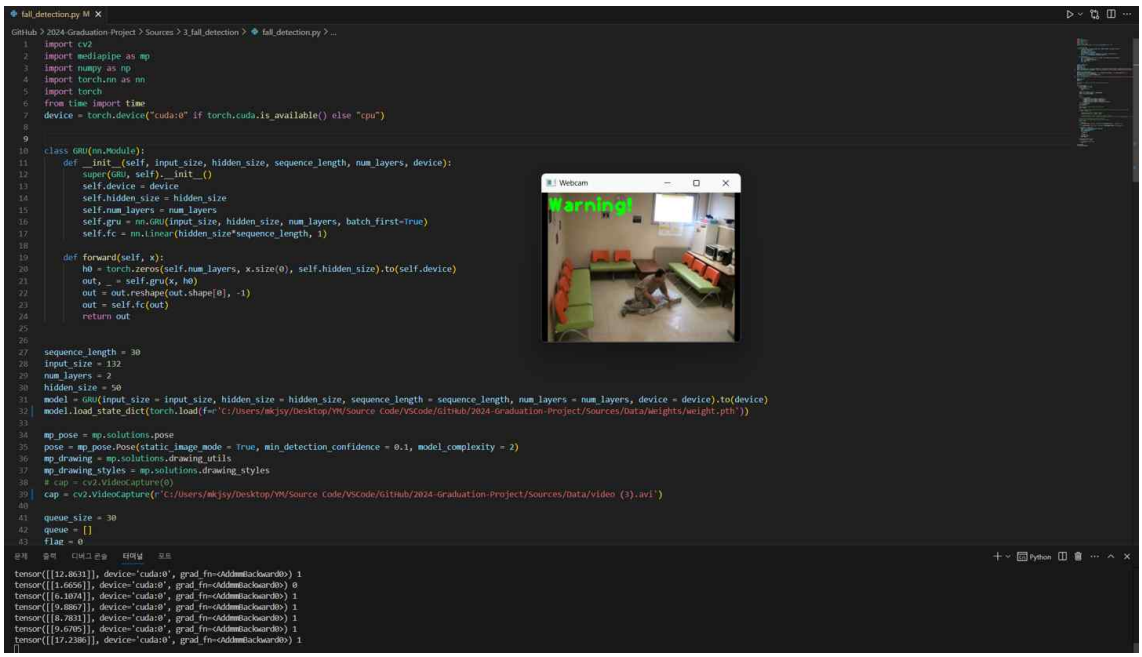
객체 감지: Yolov5 모델을 사용하여 입력 영상에서 화재 객체를 감지하였다. 감지된 객체의 bounding box, confidence score, 클래스 정보를 획득하였다.

화재 판단: confidence score가 특정 임계값 이상이고 클래스가 'fire'인 객체를 화재로 판단하였다.

코드 예시

```python
from ultralytics import YOLO
import cvzone
import cv2
import math

#use mp4 video as source
cap = cv2.VideoCapture(r'C:/Users/mkjsy/Desktop/YM/Source Code/VSCode/GitHub/2024-Graduation-Project/Sources/Data/fire.mp4')
#use webcam as source
# cap = cv2.VideoCapture(0)

model = YOLO(r'C:/Users/mkjsy/Desktop/YM/Source Code/VSCode/GitHub/2024-Graduation-Project/Sources/Data/fire_model.pt')
# Reading the classes
classnames = ['fire']

while cap.isOpened():
    success, frame = cap.read()
    if (not success):
        print('failed to read video')
        continue

    frame = cv2.resize(frame, (640, 480))
    result = model(frame, stream=True)

    # Getting bbox,confidence and class names information to work with
    for info in result:
        boxes = info.boxes
        for box in boxes:
            confidence = box.conf[0]
            confidence = math.ceil(confidence * 100)
            Class = int(box.cls[0])
            if confidence > 50:
                x1, y1, x2, y2 = box.xyxy[0]
                x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 5)
                cvzone.putTextRect(frame, f'{classnames[Class]} {confidence}%', [x1 + 8, y1 + 100],
                                   scale=1.5, thickness=2)

    cv2.imshow('frame', frame)

    #ESC를 눌러 종료
    if cv2.waitKey(5) & 0xFF == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

4-1-4 사람 동작 인식 모델



본 연구에서는 사람의 행동을 인식하고 넘어짐과 같은 위험 상황을 감지하기 위해 Human Action Recognition 모델을 학습시키고 추론 모듈을 구축하였다. 이를 위해 Mediapipe Pose Solution과 GRU 모델을 사용하였으며, 구체적인 구현 내용은 다음과 같다.

데이터 수집 및 전처리

- 데이터 수집: 학습 데이터는 넘어짐 동작을 포함하는 공개 데이터셋(Le2i)을 활용하였다. 다양한 각도, 조명, 배경에서 촬영된 데이터를 포함하여 모델의 일반화 성능을 높였다.
- 데이터 전처리: Mediapipe Pose Solution을 사용하여 각 프레임에서 사람의 골격 정보(keypoint)를 추출하였다. 추출된 keypoint 좌표를 정규화하고, 프레임 간의 차이를 계산하여 GRU 모델의 입력으로 사용하였다.

GRU 모델 학습

- 모델 구축: GRU 모델은 시퀀스 데이터 학습에 효과적인 순환 신경망(RNN) 모델이다. 입력층, GRU 층, 출력층으로 구성된 GRU 모델을 구축하였다.
GRU 층은 여러 개의 GRU 셀을 포함하며, 이전 시간 단계의 정보를 기억하고 현재 입력과 결합하여 행동을 예측한다.
- 모델 학습: 수집된 데이터를 사용하여 GRU 모델을 학습시켰다.
손실 함수로는 BCEWithLogitsLoss를 사용하였고, 최적화 알고리즘으로는 Adam optimizer를 사용하였다.

추론 모듈 구현

실시간 골격 정보 추출: 웹캠에서 입력되는 영상 데이터에서 Mediapipe Pose Solution을 사용하여 실시간으로 사람의 골격 정보를 추출하였다.
행동 분류: 추출된 골격 정보를 GRU 모델에 입력하여 행동을 분류하였다.

GRU 모델은 입력된 골격 정보 시퀀스를 분석하여 넘어짐 동작을 예측한다.
위험 상황 감지: 넘어짐으로 분류된 경우, 위험 상황으로 판단하고 경고 메시지를 출력한다. 넘어짐 감지 정확도를 높이기 위해 일정 시간 동안 연속적으로 넘어짐으로 분류될 경우에만 경고를 발생시키도록 구현하였다.

코드 예시

1) 학습 데이터 전처리 코드

```python
import mediapipe as mp
import os
import cv2
import pandas as pd
import natsort
import numpy as np

mp_pose = mp.solutions.pose
pose = mp_pose.Pose(static_image_mode = True, min_detection_confidence = 0.1, model_complexity = 2)
mp_drawing = mp.solutions.drawing_utils

def PointExtractor(route):
        count = 0
        for r in os.listdir(route):
            print(count)
            df = pd.DataFrame()

            tmp_path = os.listdir(route + "/" + r)
            tmp_path = natsort.natsorted(tmp_path)
            for i in tmp_path[-90:]:
                img = cv2.imread(route + "/" + r + "/" + i)
                rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                result = pose.process(rgb_img)
                x = []

                if not result.pose_landmarks:
                    continue

                try:
                    for k in range(33):
                        x.append(result.pose_landmarks.landmark[k].x)
                        x.append(result.pose_landmarks.landmark[k].y)
                        x.append(result.pose_landmarks.landmark[k].z)
                        x.append(result.pose_landmarks.landmark[k].visibility)
                except AttributeError:
                    x = np.zeros(132)
                tmp = pd.DataFrame(x).T
                df = pd.concat([df, tmp])

            df.to_csv('test%d.csv' % count)
            count += 1

if __name__ == '__main__':
    for j in range(1,13):
        num = j
        path = r"/home/youngmin/disk_b/datasets/HumanPose(Video)/sit/3-%d" % num
        destination = r"/home/youngmin/YM/My/csv/3-%d" % num
        if not os.path.exists(destination):
            os.makedirs(destination)
        os.chdir(destination)
        PointExtractor(path)
```

2) GRU 모델 구축 코드

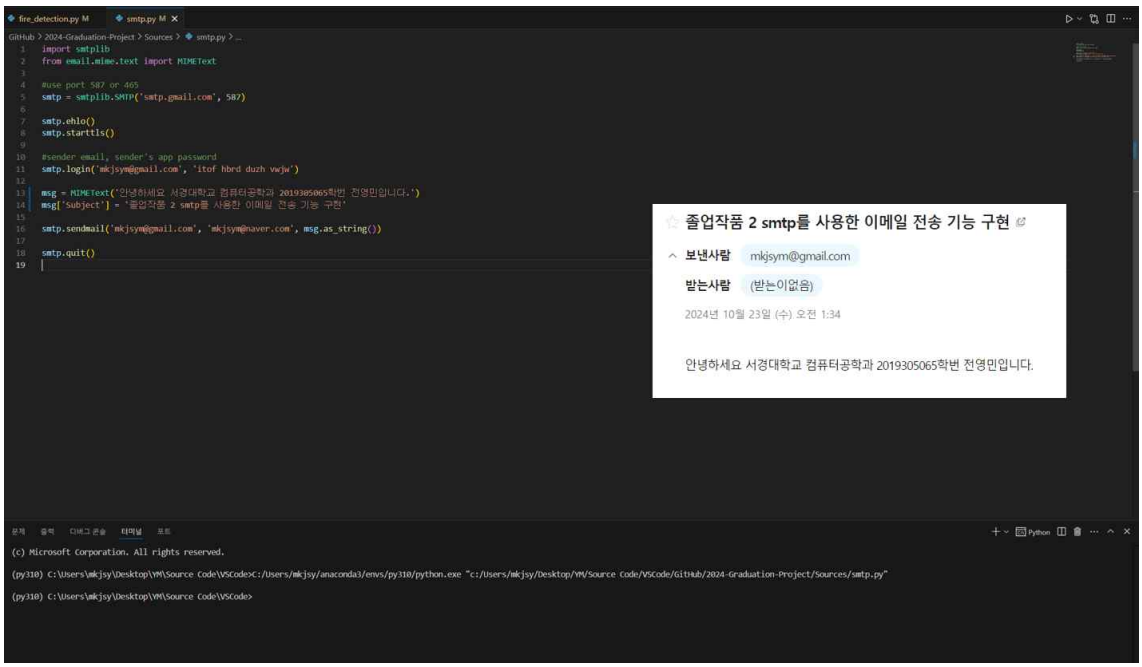
```python
class GRU(nn.Module):
    def __init__(self, input_size, hidden_size, sequence_length, num_layers, device):
        super(GRU, self).__init__()
        self.device = device
        self.hidden_size = hidden_size
        self.num_layers = num_layers
        self.gru = nn.GRU(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size*sequence_length, 1)

    def forward(self, x):
        h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).to(self.device)
        out, _ = self.gru(x, h0)
        out = out.reshape(out.shape[0], -1) # <- state 추가
        out = self.fc(out)
        return out
```

3) GRU 모델 학습 코드

```python
model = GRU(input_size = input_size, hidden_size = hidden_size, sequence_length = sequence_length, num_layers = num_layers, device = device).to(device)
#model.load_state_dict(torch.load(f=r'C:/Users/mkjsy/Desktop/YM/Source Code/VSCode/SHM/My/savepoint.pth'))
criterion = nn.BCEWithLogitsLoss()
optimizer = optim.Adam(model.parameters(), lr = 1e-3)

loss_graph = []
n = len(train_loader)
for epoch in range(num_epochs):
    running_loss = 0.0

    for i, data in enumerate(train_loader, 0):
        seq, target = data
        out = model(seq)
        loss = criterion(out, target)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        running_loss += loss.item()

    loss_graph.append(running_loss/n)
    if (epoch + 1) % 100 == 0:
        print('[epoch: %d] loss: %.4f' %(epoch, running_loss/n))

torch.save(obj=model.state_dict(), f=save_path)
```

위와 같이 Human Action Recognition 모델을 학습시키고 추론 모듈을 구축하여 지능형 안전 관리 시스템에서 사람의 행동을 인식하고 넘어짐과 같은 위험 상황을 감지하는 기능을 구현하였다.

4-1-5 위험 상황 알림 기능



본 연구에서는 화재 또는 넘어짐과 같은 위험 상황 발생 시, 관리자에게 즉시 알림을 전송하여 신속한 대응을 가능하게 하는 기능을 제공한다. 이를 위해 SMTP 모듈을 이용하여 이메일 알림 기능을 구현하였다.

SMTP 모듈 설정

SMTP 서버 정보 설정: Gmail, 네이버 메일 등 이메일 서비스 제공업체의 SMTP 서버 주소, 포트 번호, 사용자 ID, 비밀번호 등을 설정한다.

이메일 전송

- SMTP 서버 연결: 설정된 SMTP 서버 정보를 사용하여 SMTP 서버에 연결한다.
- 이메일 전송: 생성된 이메일 메시지를 관리자에게 전송한다.
- 예외 처리: 네트워크 오류 등으로 인해 이메일 전송이 실패할 경우, 오류 메시지를 기록하고 재전송을 시도하거나 다른 알림 방식을 활용한다.

코드 예시

```python
import smtplib
from email.mime.text import MIMEText

senderEmail = 'mkjsym@gmail.com'
senderPW = 'itof hbrd duzh vwjw'
receiverEmail = 'mkjsym@gmail.com'

def sendMessage(content, title):
    #use port 587 or 465
    smtp = smtplib.SMTP('smtp.gmail.com', 587)

    smtp.ehlo()
    smtp.starttls()

    #sender email, sender's app password
    smtp.login(senderEmail, senderPW)

    msg = MIMEText(content)
    msg['Subject'] = title

    smtp.sendmail(senderEmail, receiverEmail, msg.as_string())

    smtp.quit()


#use port 587 or 465
smtp = smtplib.SMTP('smtp.gmail.com', 587)

smtp.ehlo()
smtp.starttls()

#sender email, sender's app password
smtp.login('mkjsym@gmail.com', 'itof hbrd duzh vwjw')

msg = MIMEText('안녕하세요 서경대학교 컴퓨터공학과 2019305065학번 전영민입니다.')
msg['Subject'] = '졸업작품 2 smtp를 사용한 이메일 전송 기능 구현'

smtp.sendmail('mkjsym@gmail.com', 'mkjsym@naver.com', msg.as_string())

smtp.quit()
```

위와 같이 SMTP 모듈을 이용하여 위험 상황 발생 시 관리자에게 이메일 알림을 전송하는 기능을 구현하였다. 이를 통해 관리자는 위험 상황에 신속하게 대응하고 피해를 최소화할 수 있다.

4-2 결과 평가 및 처음 목표 달성 여부

본 연구에서 개발한 지능형 안전 관리 시스템의 성능을 평가하고 연구 목표 달성 여부를 확인하기 위해 다음과 같은 실험을 수행하였다.

목표 달성 여부

- 실시간 화재 감지: Yolov5 모델을 사용하여 실시간으로 화재를 감지하는 목표를 달성하였다.
- 넘어짐 감지: Mediapipe Pose Solution과 GRU 모델을 사용하여 넘어짐을 감지하는 목표를 달성하였다.
- 실시간 알림: SMTP 모듈을 사용하여 위험 상황 발생 시 실시간으로 이메일 알림을 전송하는 목표를 달성하였다.

본 연구에서는 위와 같은 실험 결과를 통해 개발된 지능형 안전 관리 시스템이 화재 및 넘어짐과 같은 위험 상황을 효과적으로 감지하고, 실시간으로 알림을 제공하여 안전사고 예방에 기여할 수 있음을 확인하였다.

5. 토론

5-1 결과 분석

본 연구에서 개발한 지능형 안전 관리 시스템의 실험 결과를 분석한 결과, 다음과 같은 사실을 확인하였다.

- 화재 감지: Yolov5 모델은 다양한 환경에서 발생하는 화재를 효과적으로 감지했지만, 조명 변화, 연기, 그림자 등의 요인으로 인해 오탐이 발생하는 경우가 있었다. 특히, 야간이나 어두운 환경에서 촬영된 영상에서 오탐율이 높게 나타났다. 이는 학습 데이터셋에 야간 환경의 화재 영상이 충분히 포함되지 않았기 때문으로 판단된다. 향후 다양한 조명 조건에서 촬영된 화재 영상 데이터를 추가하여 모델을 학습시킴으로써 오탐율을 줄일 수 있을 것으로 예상된다.
- 행동 인식: Mediapipe Pose Solution과 GRU 모델을 결합한 행동 인식 시스템은 넘어짐을 정확하게 감지했지만, 빠른 움직임, 복잡한 배경, 부분적인 가려짐 등으로 인해 오류가 발생하는 경우가 있었다. 특히, 여러 사람이 동시에 움직이는 경우, 사람의 움직임을 정확하게 추적하고 분석하는 데 어려움이 있었다. 이는 Mediapipe Pose Solution의 성능 한계와 GRU 모델의 학습 데이터 부족으로 인한 것으로 판단된다. 향후 다양한 환경에서 촬영된 행동 영상 데이터를 추가하고, 더욱 정교한 행동 분석 알고리즘을 적용하여 인식 정확도를 높일 수 있을 것으로 예상된다.

본 연구에서 개발한 지능형 안전 관리 시스템은 화재 및 넘어짐과 같은 위험 상황을 효과적으로 감지하고 실시간으로 알림을 제공하여 안전사고 예방에 기여할 수 있음을 확인하였다. 하지만, 실제 환경에 적용하기 위해서는 앞서 언급한 개선 사항들을 반영하여 시스템의 성능과 안정성을 더욱 향상시키는 것이 필요하다.

6. 결론

6-1 졸업작품 전체 요약

본 연구에서는 인공지능 기술을 활용하여 화재 및 넘어짐과 같은 위험 상황을 실시간으로 감지하고, 이를 통해 안전사고를 예방하고 피해를 최소화할 수 있는 지능형 안전 관리 시스템을 개발하였다. 시스템은 라즈베리파이 4 Model B 보드와 웹캠을 기반으로 구축되었으며, YOLOv5 모델을 사용하여 영상에서 화재를 감지하고, Mediapipe Pose solution과 GRU 모델을 사용하여 사람의 행동을 분석하여 넘어짐을 감지한다. FFmpeg를 이용하여 웹캠에서 촬영된 영상 데이터를 파이썬 코드로 실시간 전송하고, 위험 상황 감지 시 SMTP 모듈을 통해 관리자에게 이메일로 경고 메시지를 전송한다. 본 시스템은 저렴한 비용으로 구축이 가능하며, 실시간으로 위험 상황을 감지하고 알림을 제공하여 효율적인 안전 관리를 가능하게 한다. 또한, 향후 다양한 센서 및 기능을 추가하여 시스템을 확장하고, 다양한 환경에 적용할 수 있는 가능성을 제시한다.

6-2 졸업작품이 가지는 의의

본 연구에서는 Yolov5, Mediapipe, GRU 모델 등의 인공지능 기술을 활용하여 화재 및 넘어짐과 같은 위험 상황을 실시간으로 감지하는 지능형 안전 관리 시스템을 개발하였다. 라즈베리파이 4 Model B 보드와 웹캠을 기반으로 구축된 본 시스템은, 저렴한 비용으로 효율적인 안전 관리를 가능하게 한다는 점에서 큰 의의를 가진다.

본 연구의 주요 결과는 다음과 같다.

- 실시간 화재 감지: YOLOv5 모델을 통해 다양한 환경에서 발생하는 화재를 실시간으로 감지하는 시스템을 구현하였다.
- 넘어짐 감지: Mediapipe Pose solution과 GRU 모델을 이용하여 사람의 행동을 분석하고 넘어짐을 정확하게 감지하는 시스템을 구현하였다.
- 효율적인 경고 시스템: FFMPEG를 이용한 실시간 데이터 전송 및 SMTP 모듈을 이용한 이메일 경고 시스템을 구축하여 위험 상황 발생 시 즉각적인 대응을 가능하게 하였다.

본 연구는 다음과 같은 의의를 가진다.

- 안전 관리 시스템의 지능화: 인공지능 기술을 활용하여 기존 안전 관리 시스템의 한계를 극복하고, 보다 효율적이고 지능적인 시스템 구축에 기여하였다.
- 안전사고 예방 및 피해 최소화: 실시간 위험 상황 감지 및 경고 시스템을 통해 안전사고를 예방하고 인명 및 재산 피해를 최소화할 수 있다.
- 다양한 분야로의 적용 가능성: 본 시스템은 산업 현장, 공공장소, 가정 등 다양한 환경에 적용 가능하며, 화재 감지 및 넘어짐 감지 외에도 다양한 안전 관리 기능을 추가하여 확장할 수 있다.

본 연구를 통해 개발된 지능형 안전 관리 시스템은 안전 관리 분야의 발전에 기여할 수 있을 것으로 기대된다.

6-3 개선사항

본 연구에서 개발한 지능형 안전 관리 시스템은 기존 시스템에 비해 향상된 성능을 보이지만, 실제 환경 적용 및 상용화를 위해서는 다음과 같은 개선 및 추가 연구가 필요하다.

- 화재 감지 성능 향상:

다양한 환경 및 조건에서 촬영된 화재 영상 데이터를 추가적으로 학습시켜 YOLOv5 모델의 정확도를 향상시킨다.

연기, 불꽃, 열 등 다양한 화재 특징을 분석하여 화재 감지 알고리즘을 개선한다.

조명 변화, 그림자, 카메라 흔들림 등 외부 요인에 대한 모델의 강건성을 향상시킨다.

- 행동 인식 정확도 향상:

다양한 각도에서 촬영된 영상 데이터를 사용하여 Mediapipe Pose solution의 정확도를 높인다.

넘어짐, 쓰러짐 외에 다양한 위험 행동 (폭행, 침입 등)을 인식할 수 있도록 GRU 모델을 개선한다.

사람의 행동 인식 오류를 줄이기 위해 골격 정보 외에 다른 특징 (움직임 속도, 방향 등)을 추가적으로 활용한다.

- 시스템 안정성 및 효율성 향상:

라즈베리파이 보드의 성능 한계를 극복하기 위해 경량화된 모델 또는 하드웨어 가속 기술을 적용한다.

시스템의 실시간 처리 성능을 향상시키기 위해 코드 최적화 및 병렬 처리 기술을 적용한다.

시스템 장시간 운영 시 발생할 수 있는 오류 및 메모리 누수 문제를 해결한다.

- 다양한 기능 추가:

화재 및 넘어짐 감지 외에 객체 인식, 얼굴 인식, 이상 행동 감지 등 다양한 기능을 추가하여 시스템을 확장한다.

사용자 인터페이스를 개선하여 시스템 상태 및 감지 결과를 효과적으로 표시하고, 사용자 편의성을 높인다.

경고 시스템을 개선하여 이메일 외에 SMS, 카카오톡 등 다양한 채널을 통해 알림을 전송한다.

- 실제 환경 적용 및 평가:

실제 환경에서 시스템을 구축하고 장기간 운영하여 성능 및 안정성을 평가한다.

다양한 환경 및 조건에서 시스템 성능을 비교 분석하고, 개선 방안을 도출한다.

사용자 피드백을 수집하여 시스템을 개선하고 사용자 만족도를 높인다.

7. 참고 문헌

[1] 정인선, "서울 '공공감시CCTV' 14만여대…2년 새 두 배 급증 '세계 최상위권'" 한겨레, 2023년 5월 24일, https://www.hani.co.kr/arti/economy/it/1093078.html

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection." arXiv:1506.02640

[3] "Mediapipe 솔루션 가이드" https://ai.google.dev/edge/mediapipe/solutions/guide?hl=ko

[4] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling." arXiv:1412.3555

[5] "Vision API" https://cloud.google.com/vision/

[6] OlafenwaMoses, "FireNET." https://github.com/OlafenwaMoses/FireNET

[7] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, Yaser Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields." arXiv:1812.08008

[8] "PoseNet." https://www.tensorflow.org/lite/examples/pose_estimation/overview?hl=ko