Final Deliverable

Smart Internz Externship in Artificial Intelligence

Title	VirtualEye - LifeGuard for Swimming Pools to Detect Active Drowning using IBM Cloud
Team Members	Shruthi Rekha R 20BCE2702 Kavitha S 20BCE0001 Roshni Priya S 20BCE2184 Dhanusha Srimathi A 20BCE2449

1 Introduction

1.1 Overview

VirtualEye is an innovative solution aimed at enhancing safety in swimming pools. Serving as a lifeguard, it employs cutting-edge computer vision and artificial intelligence technologies to swiftly identify active drowning incidents. The VirtualEye website provides an in-depth look into this life-saving system, showcasing its features and functionality. By constantly monitoring the pool area, VirtualEye can quickly detect distress signals, alerting pool staff or authorities to take immediate action. With its real-time detection capabilities, VirtualEye aims to prevent accidents and save lives, making swimming pools a safer environment for everyone.

1.2 Purpose

A virtual lifeguard in a swimming pool utilizes advanced technologies, such as computer vision and motion sensors, to detect potential drowning incidents. Its purpose is to provide continuous surveillance, prompt detection, and improved response times to enhance swimmer safety and save lives. By analyzing factors like body movements and underwater behavior, the virtual lifeguard can identify individuals in distress or exhibiting signs of drowning. It acts as an additional layer of safety, complementing human lifeguards, and offers constant vigilance, especially in crowded or challenging monitoring situations. The virtual lifeguard aims to overcome the limitations of human supervision and provide reliable surveillance for optimal pool safety.

2 Literature Survey

2.1 Existing problem

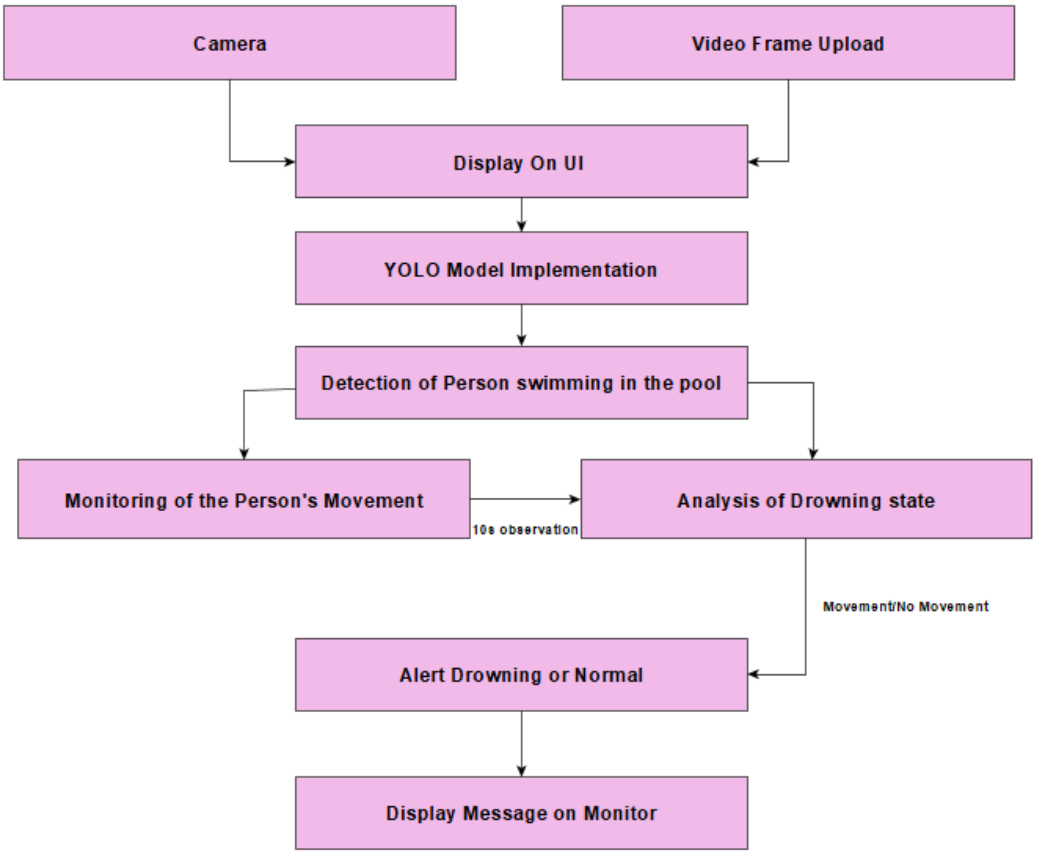
Swimming pool drowning detection faces challenges due to reliance on human supervision, limitations in detecting quick and silent incidents, false alarms, lack of real-time monitoring, and high costs for advanced technologies. Lifeguards may be distracted or fatigued, making constant monitoring difficult. Traditional detection systems may not promptly identify drownings, while false alarms disrupt pool activities. Real-time monitoring is often lacking, and cost can be prohibitive for widespread adoption. Addressing these challenges requires developing reliable AI-powered systems that can quickly detect distress signals and anomalous behaviors. Integration of multiple sensors and intelligent algorithms can improve accuracy and reduce false alarms. Training and awareness programs for lifeguards and pool operators are also crucial. By overcoming these challenges, swimming pools can become safer environments, preventing drowning incidents and saving lives.

2.2 Proposed solution

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	VirtualEye - LifeGuard for Swimming Pools To Detect Active Drowning.
2.	Idea / Solution description	The project aims to improve swimmer safety by detecting the accurate pulse rate of individuals in swimming pools. This information is sent as an alert to lifeguards, enabling them to predict changes in pulse rate and take prompt action to prevent drowning accidents, especially for beginners.
3.	Novelty / Uniqueness	Active drowning detection in swimming pool
4.	Social Impact / Customer Satisfaction	In case of an incident it is possible to extract and store the videos.
5.	Business Model (Revenue Model)	Can generate revenue from direct customers,like Lifeguard and collaborate with maritime sector and other swimming pool authorities.
6.	Scalability of the Solution	It helps the LifeGuard for earlier prediction of drowning.

3 Theoretical Analysis

3.1 Block diagram



4 Experimental Investigations

We have gone through many research papers that are related to this project, summary of those experimental analysis are given below

Paper 1:

Title: The effect of lifeguard experience upon the detection of drowning victims in a realistic dynamic visual search task

Year of Publication: 2018

Author: Laxton, Crundall, D

Summary: The results of this current study have found the predicted advantage for lifeguards in spotting and responding to drowning targets in a swimming pool situation. They identified both active and passive drowning targets more frequently and more quickly than control participants,
which suggest that experience and/or training have positively influenced the visual search and target processing skills of this specialist group. Lifeguards also appear to have a higher threshold for responding to a drowning target. This may reflect their greater sensitivity to visual cues that discriminate between drowning and normal swimming. Additionally, lifeguards may be more aware of the dangers of committing to a potentially drowning target. Once a response is initiated in a pool situation (e.g. entering the water to rescue the drowning swimmer) the lifeguard is limited in their ability to spot secondary drowning targets. Thus lifeguards may need greater evidence before responding, though this did not negatively impact on their time
to respond when they chose to do so.

Paper 2:

Title: An Automatic Video-based Drowning Detection System for Swimming Pools Using Active Contours

Year of Publication: 2016

Author: Nasrin Salehi, Maryam Keyvanara, Seyed Amirhassan Monadjemmi

Summary: In this paper, they provided a method to robust human tracking and semantic event detection within the context of a video surveillance system capable of automatically detecting drowning incidents in a swimming pool. In the current work, an effective background detection that incorporates prior knowledge using HSV color space and contour

detection enables swimmers to be reliably detected and tracked despite the significant presence of
water ripples. This algorithm was able to detect all the drowning conditions along with the exact position of the drowning person in the swimming pool and had an average detection delay of 1.53 seconds, which is relatively low compared to the needed rescue time for a lifeguard operation.

Paper 3:

Title: Computer Vision Enabled Drowning Detection System

Year of Publication: 2021

Author: Upulie Handalage, Nisansali Nikapotha, Chanaka Subasingha, Tereen Prasanga

Summary: Safety is paramount in all swimming pools. The current systems expected to address the problem of ensuring safety at swimming pools have significant problems due to their technical aspects, such as underwater cameras and methodological aspects such as the need for human intervention in the rescue mission. The use of an automated visual-based monitoring system can help to reduce drownings and assure pool safety effectively. This study introduces a revolutionary technology that identifies drowning victims in a minimum amount of time and dispatches an automated drone to save them. Using convolutional neural network (CNN) models, it can detect a drowning person in three stages. Whenever such a situation like this is detected, the inflatable tube-mounted self-driven drone will go on a rescue mission, sounding an alarm to inform the nearby lifeguards. The system also keeps an eye out for potentially dangerous actions that could result in drowning. This system's ability to save a drowning victim in under a minute has been demonstrated in prototype experiments' performance evaluations. In this research paper they used CNN model but in our paper we used yolo model for better accuracy in predicting drowning.

Paper 4:

Title: Deep Learning and Vision-Based Early Drowning Detection

Year of publication: 2023

Author: Maad Shatnawi, Frdoos Albreiki, Ashwaq Alkhoori, Mariam Alhebshi

Summary: Drowning is one of the top five causes of death for children aged 1–14 worldwide. According to data from the World Health Organization (WHO), drowning is the third most common reason for unintentional fatalities. Designing a drowning detection system is becoming increasingly necessary in order to ensure the safety of swimmers, particularly children. This paper presents a computer vision and deep learning-based early drowning detection approach. We utilized five convolutional neural network models and trained them

on our data. These models are SqueezeNet, GoogleNet, AlexNet, ShuffleNet, and ResNet50. ResNet50 showed the best performance, as it achieved 100% prediction accuracy with a reasonable training time. When compared to other approaches, the proposed approach performed exceptionally well in terms of prediction accuracy and computational cost. In this paper they used cnn and transfer learning model to improve the accuracy but we used yolo which is a pretrained model for better object detection and prediction.

Paper 5:

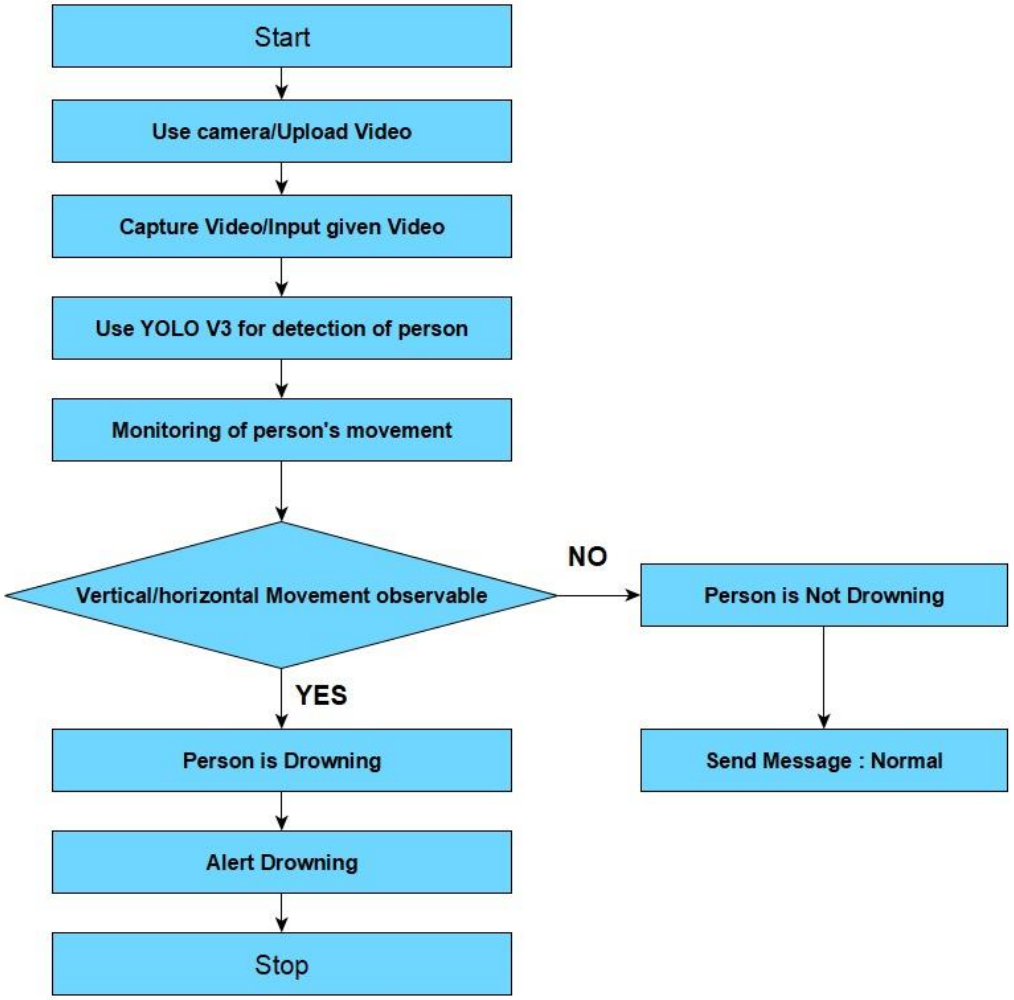
Title: An Automatic Video-based Drowning Detection System for Swimming Pools Using Active Contours

Year of publication: 2016

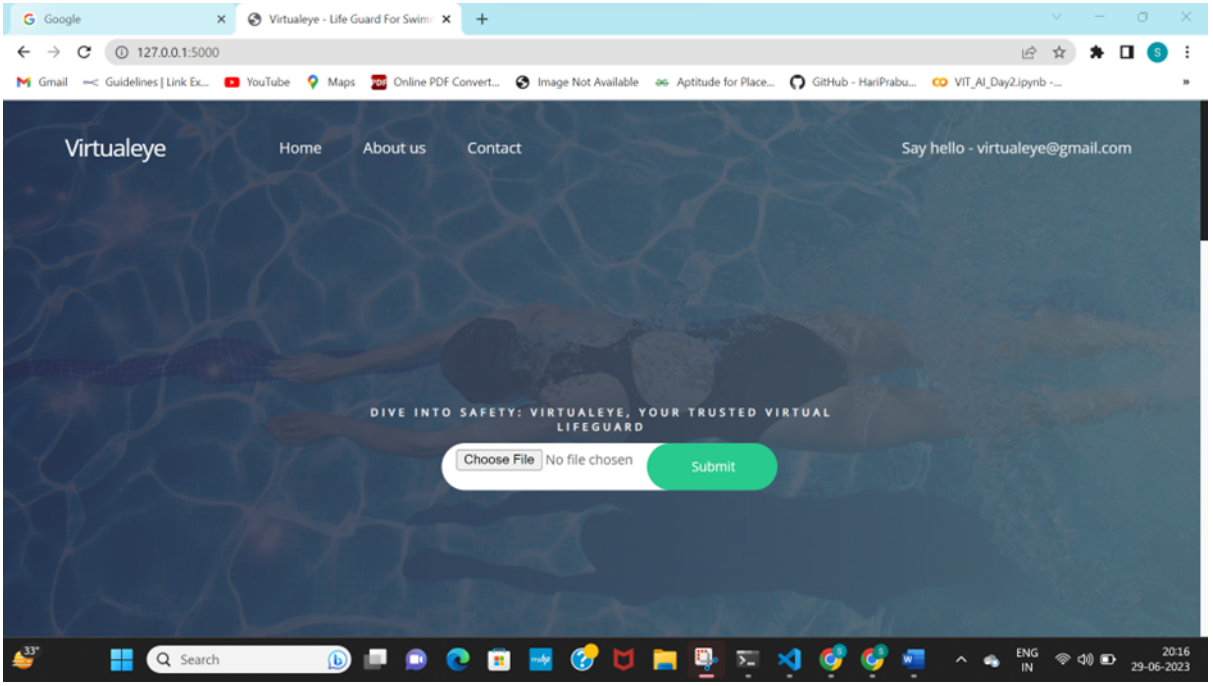
Author: Nasrin Salehi and Maryam Keyvanara and Seyed Amirhassan Monadjemmi.

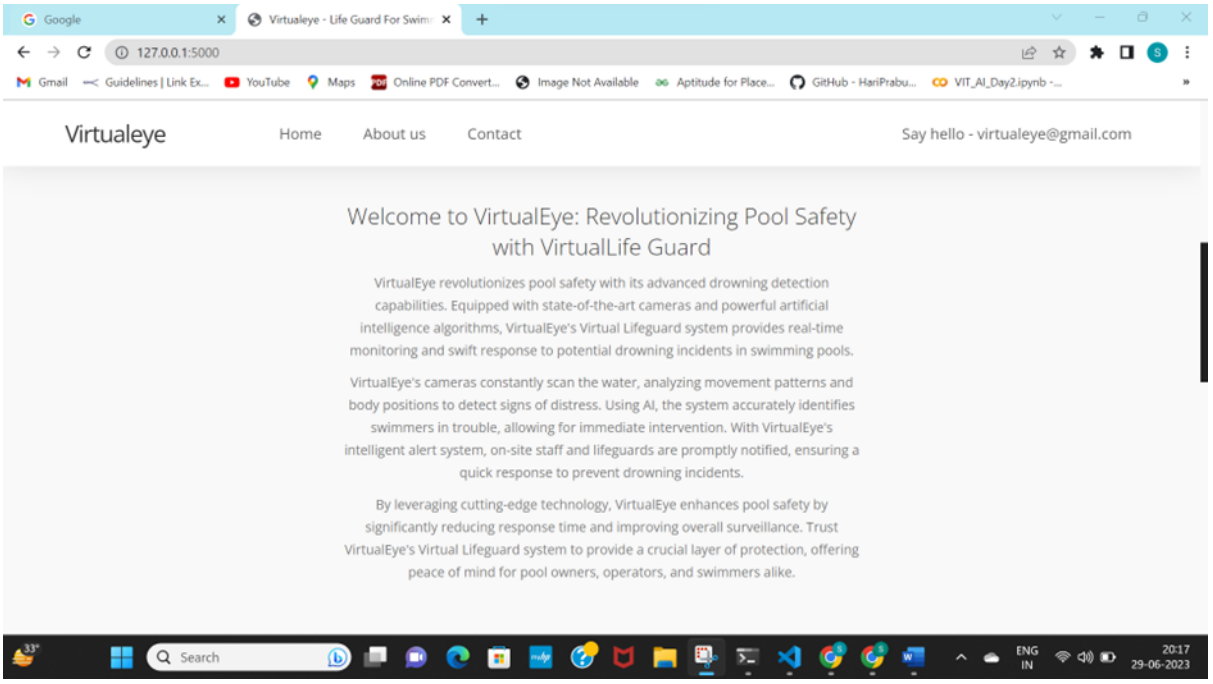
This paper presents a real-time drowning detection method for swimming pools using HSV color space analysis and contour detection. The system employs video surveillance and image processing techniques to detect and track individuals in swimming pools. By applying thresholding in the HSV color space, the algorithm segments the region of interest in each video frame. Contour detection is then utilized to track the movement of the detected individuals. The system can identify drowning persons in indoor swimming pools and promptly alert lifeguards if someone is missing for a specific duration.The proposed method addresses the challenges posed by water ripples, shadows, and splashes by relying on the HSV color space for segmentation and contour-based tracking. The system converts each video frame to the HSV color space and applies thresholding to obtain a binary image where the swimmer appears as a white region.Contour detection is employed to identify contours in the binary image, and the contour with the largest area, representing the swimmer, is selected and tracked across consecutive frames. The system continuously updates the contour and tracks the swimmer's movement.Experiments conducted using recorded video sequences from swimming pools demonstrate the system's high accuracy in detecting drowning persons with minimal false alarms. Real-time tracking is achieved, and the maximum reported alarm delay is 2.6 seconds, ensuring reliability for rescue and resuscitation purposes.In conclusion, this paper presents a real-time drowning detection method based on HSV color space analysis and contour detection. The proposed system showcases precise detection and tracking of drowning persons in swimming pools, offering potential for enhancing safety and security in such environments.

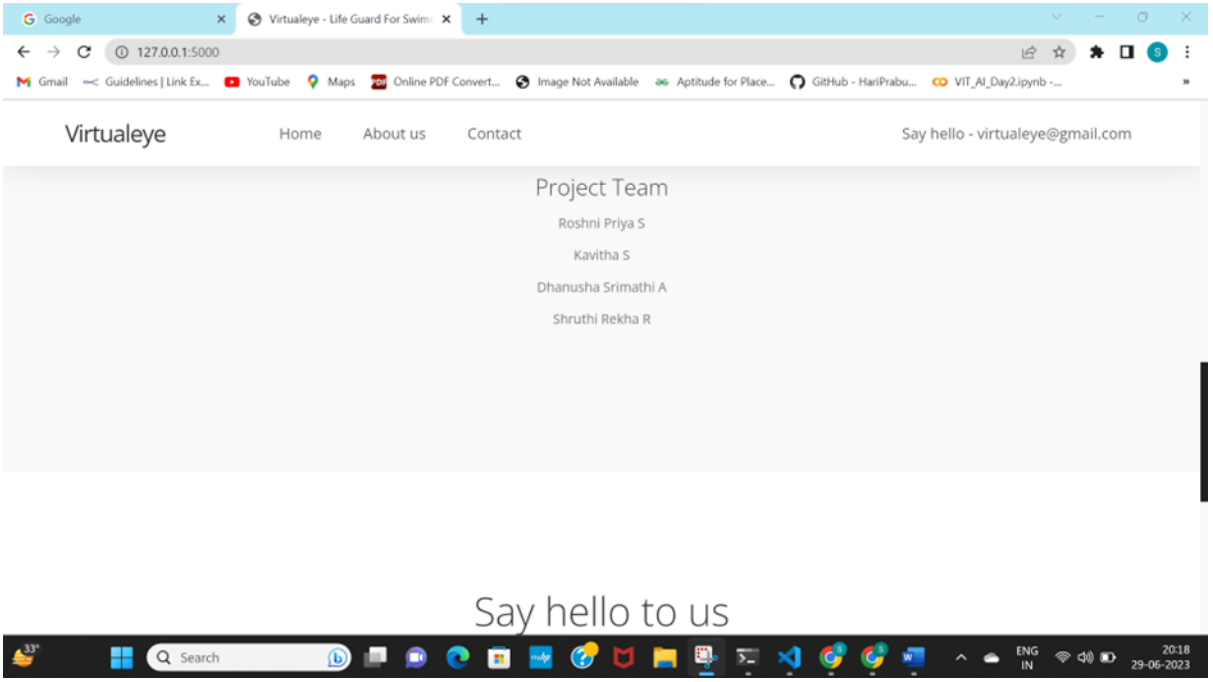
5 Flowchart

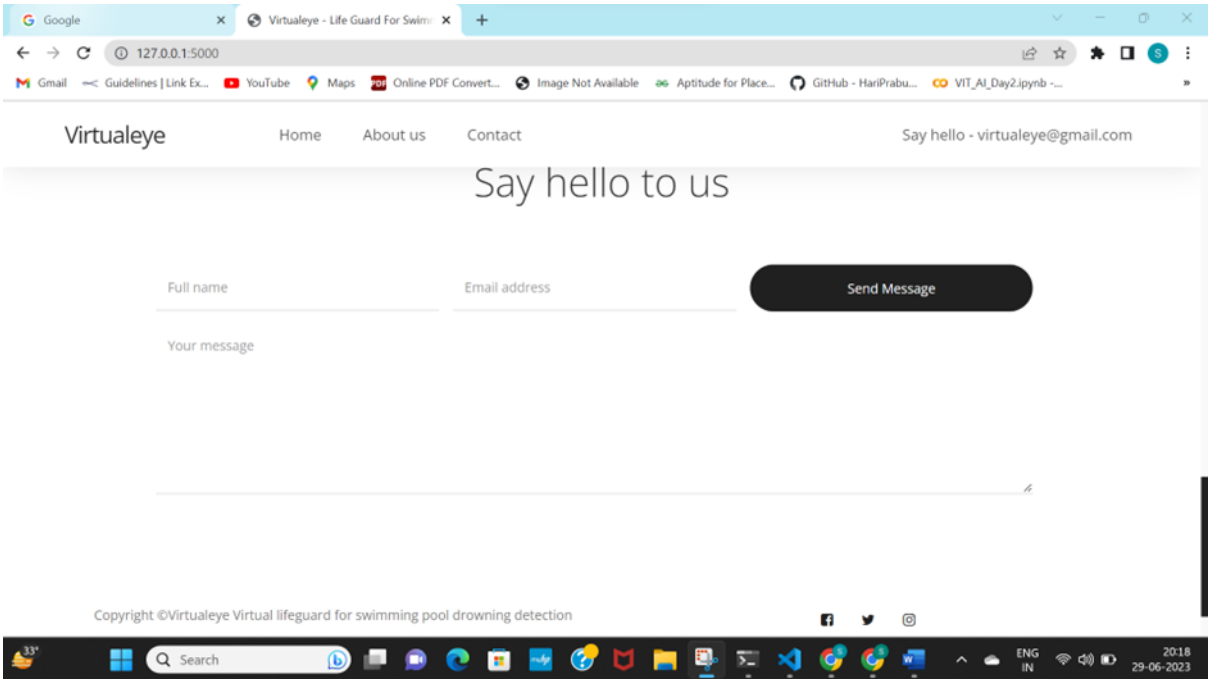

Start
Use camera/Upload Video
Capture Video/Input given Video
Use YOLO V3 for detection of person
Monitoring of person's movement
Vertical/horizontal Movement observable
NO
Person is Not Drowning
Send Message : Normal
YES
Person is Drowning
Alert Drowning
Stop

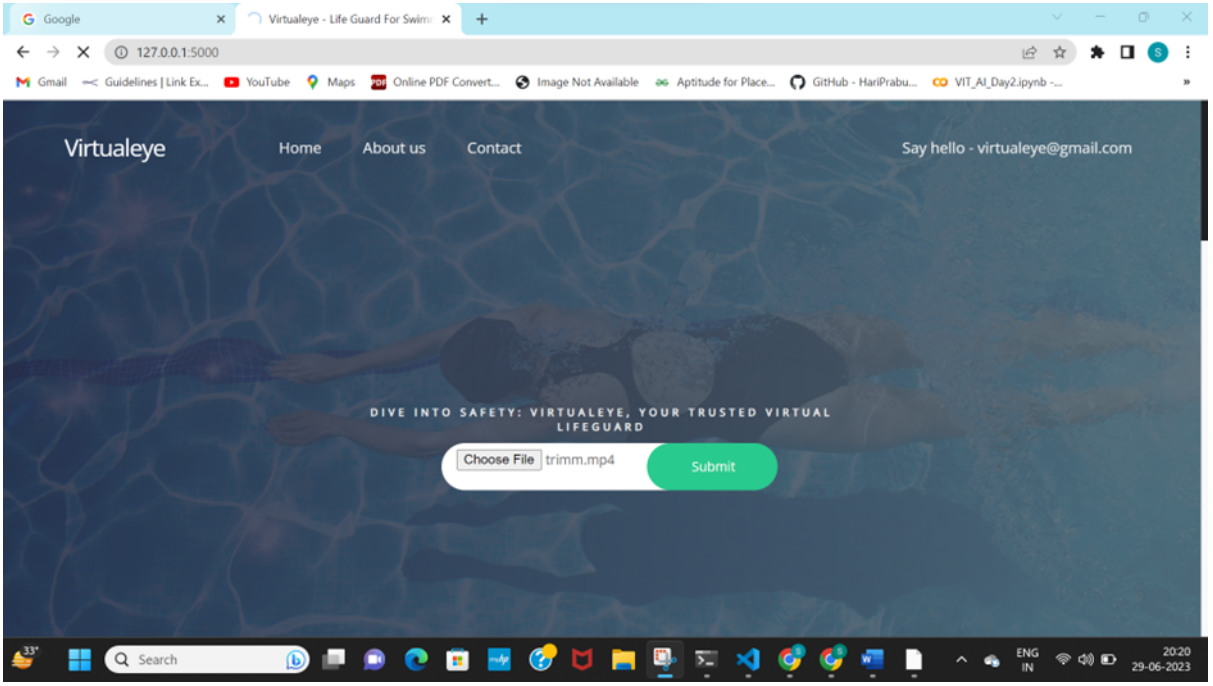
6 Result

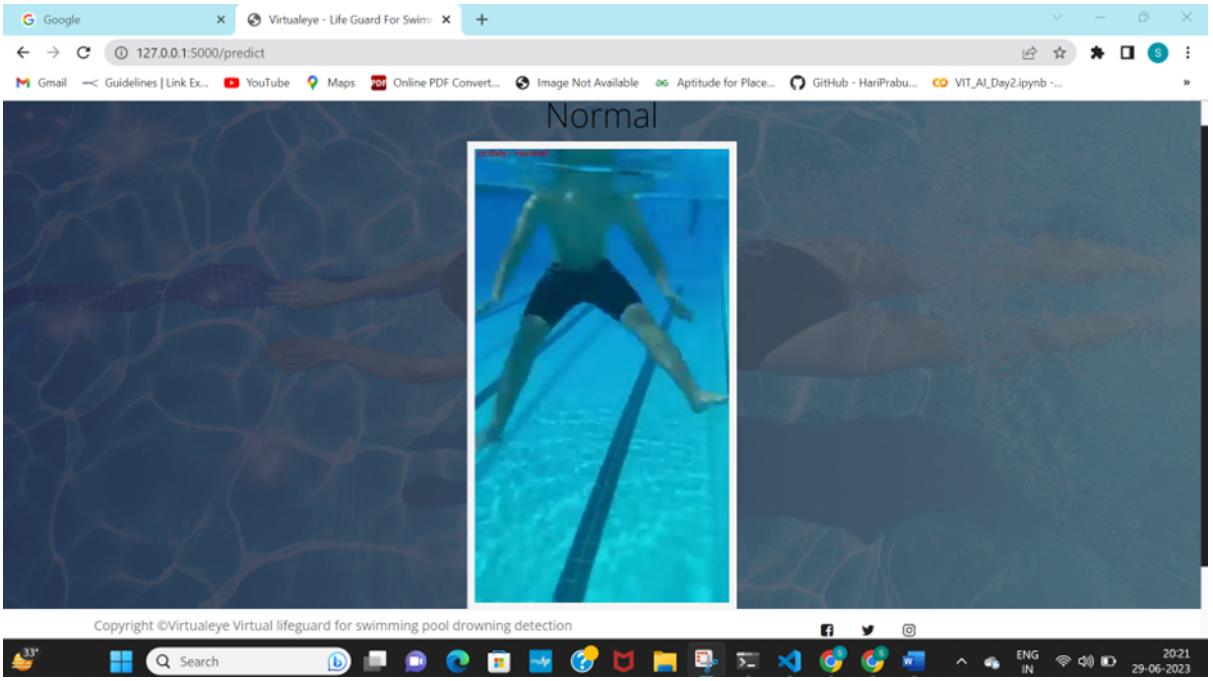

Google
Virtualeye - Life Guard For Swim
127.0.0.1:5000
Gmail
Guidelines | Link Ex...
YouTube
Maps
Online PDF Convert...
Image Not Available
Aptitude for Place...
GitHub - HariPrabu...
VIT_AI_Day2.ipynb -...
Virtualeye
Home
About us
Contact
Say hello - virtualeye@gmail.com
DIVE INTO SAFETY: VIRTUALEYE, YOUR TRUSTED VIRTUAL LIFEGUARD
Choose File
No file chosen
Submit
Search
ENG
IN
20:16
29-06-2023


Google
Virtualeye - Life Guard For Swim
127.0.0.1:5000
Gmail
Guidelines | Link Ex...
YouTube
Maps
Online PDF Convert...
Image Not Available
Aptitude for Place...
GitHub - HariPrabu...
VIT_AI_Day2.ipynb -...
Virtualeye
Home
About us
Contact
Say hello - virtualeye@gmail.com
Welcome to VirtualEye: Revolutionizing Pool Safety with VirtualLife Guard
VirtualEye revolutionizes pool safety with its advanced drowning detection capabilities. Equipped with state-of-the-art cameras and powerful artificial intelligence algorithms, VirtualEye's Virtual Lifeguard system provides real-time monitoring and swift response to potential drowning incidents in swimming pools.
VirtualEye's cameras constantly scan the water, analyzing movement patterns and body positions to detect signs of distress. Using AI, the system accurately identifies swimmers in trouble, allowing for immediate intervention. With VirtualEye's intelligent alert system, on-site staff and lifeguards are promptly notified, ensuring a quick response to prevent drowning incidents.
By leveraging cutting-edge technology, VirtualEye enhances pool safety by significantly reducing response time and improving overall surveillance. Trust VirtualEye's Virtual Lifeguard system to provide a crucial layer of protection, offering peace of mind for pool owners, operators, and swimmers alike.
Search
ENG
IN
20:17
29-06-2023

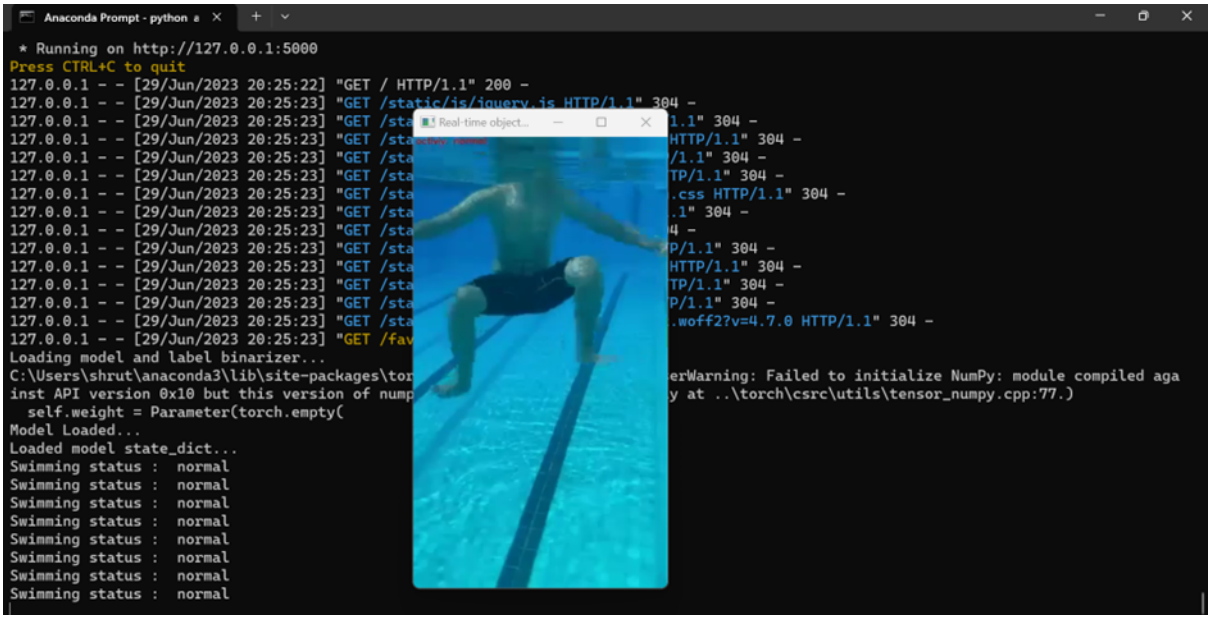

Google
Virtualeye - Life Guard For Swim
127.0.0.1:5000
Gmail
Guidelines | Link Ex...
YouTube
Maps
Online PDF Convert...
Image Not Available
Aptitude for Place...
GitHub - HariPrabu...
VIT_AI_Day2.ipynb -...
Virtualeye
Home
About us
Contact
Say hello - virtualeye@gmail.com
Project Team
Roshni Priya S
Kavitha S
Dhanusha Srimathi A
Shruthi Rekha R
Say hello to us
Search
ENG
IN
20:18
29-06-2023

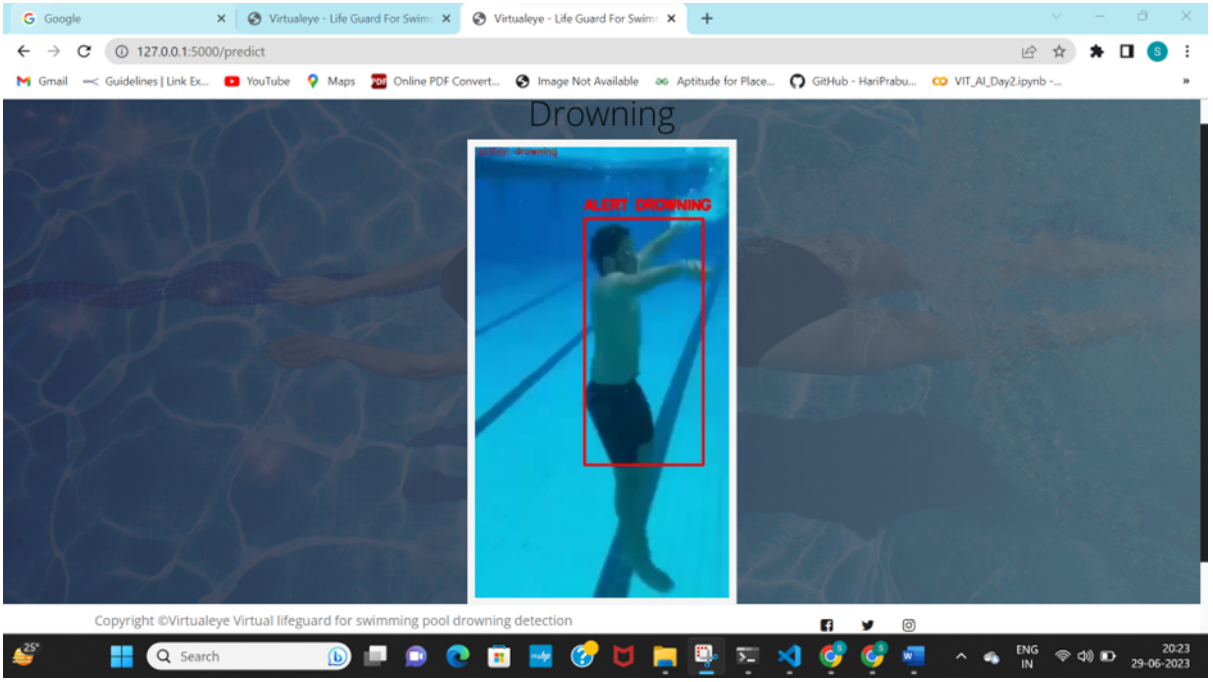

Google
Virtualeye - Life Guard For Swim
127.0.0.1:5000
Gmail
Guidelines | Link Ex...
YouTube
Maps
Online PDF Convert...
Image Not Available
Aptitude for Place...
GitHub - HariPrabu...
VIT_AI_Day2.ipynb -...
Virtualeye
Home
About us
Contact
Say hello - virtualeye@gmail.com
Say hello to us
Full name
Email address
Send Message
Your message
Copyright ©Virtualeye Virtual lifeguard for swimming pool drowning detection
Search
ENG
IN
20:18
29-06-2023



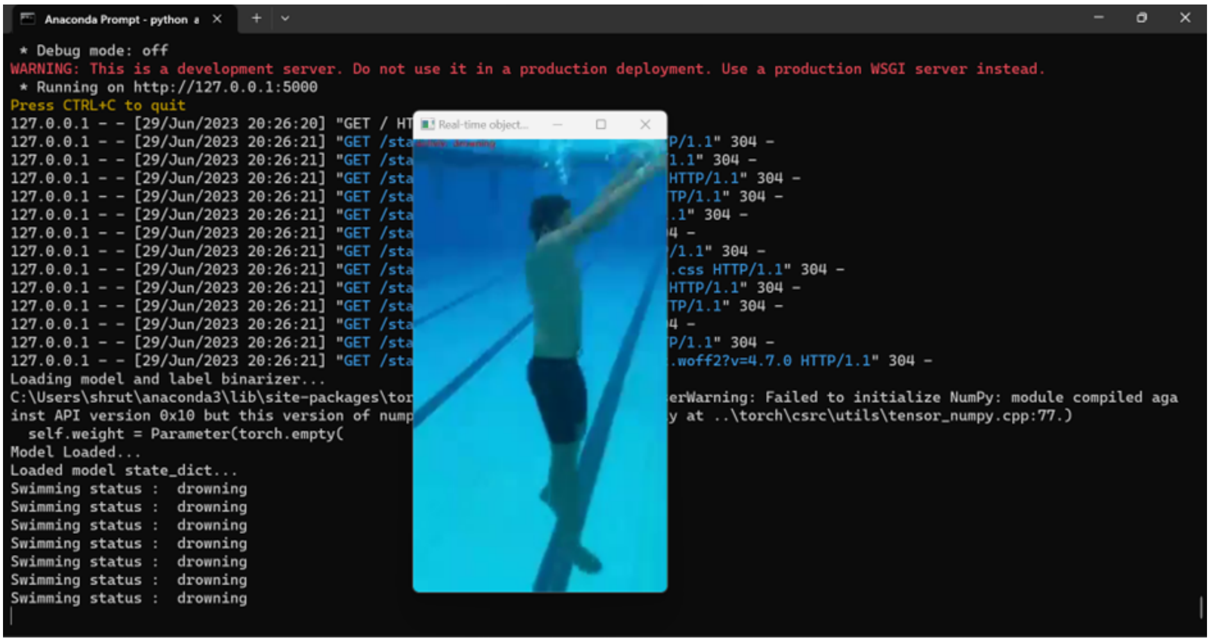



```
Model Loaded...
Loaded model state_dict...
Swimming status :  normal
Swimming status :  normal
Swimming status :  normal
Swimming status :  normal
Swimming status :  normal
Swimming status :  normal
Swimming status :  normal
Swimming status :  normal
```


Anaconda Prompt - python a
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [29/Jun/2023 20:25:22] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [29/Jun/2023 20:25:23] "GET /static/js/jquery.js HTTP/1.1" 304 -
127.0.0.1 - - [29/Jun/2023 20:25:23] "GET /sta
127.0.0.1 - - [29/Jun/2023 20:25:23] "GET /sta
127.0.0.1 - - [29/Jun/2023 20:25:23] "GET /sta
127.0.0.1 - - [29/Jun/2023 20:25:23] "GET /sta
127.0.0.1 - - [29/Jun/2023 20:25:23] "GET /sta
127.0.0.1 - - [29/Jun/2023 20:25:23] "GET /sta
127.0.0.1 - - [29/Jun/2023 20:25:23] "GET /sta
127.0.0.1 - - [29/Jun/2023 20:25:23] "GET /sta
127.0.0.1 - - [29/Jun/2023 20:25:23] "GET /sta
127.0.0.1 - - [29/Jun/2023 20:25:23] "GET /sta
127.0.0.1 - - [29/Jun/2023 20:25:23] "GET /sta
127.0.0.1 - - [29/Jun/2023 20:25:23] "GET /fav
Loading model and label binarizer...
C:\Users\shrut\anaconda3\lib\site-packages\tor
inst API version 0x10 but this version of nump
self.weight = Parameter(torch.empty(
Model Loaded...
Loaded model state_dict...
Swimming status : normal
Swimming status : normal
Swimming status : normal
Swimming status : normal
Swimming status : normal
Swimming status : normal
Swimming status : normal
Swimming status : normal
Real-time object...
erWarning: Failed to initialize NumPy: module compiled aga
y at ..\torch\csrc\utils\tensor_numpy.cpp:77.)


Google
Virtualeye - Life Guard For Swim
Virtualeye - Life Guard For Swim
127.0.0.1:5000/predict
Gmail
Guidelines | Link Ex...
YouTube
Maps
Online PDF Convert...
Image Not Available
Aptitude for Place...
GitHub - HariPrabu...
VIT_AI_Day2.ipynb -...
Drowning
ALERT DROWNING
Copyright ©Virtualeye Virtual lifeguard for swimming pool drowning detection
Search
ENG
IN
20:23
29-06-2023


Anaconda Prompt
self.weight = Parameter(torch.empty(
Model Loaded...
Loaded model state_dict...
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning


Anaconda Prompt - python
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
Real-time object...
Loading model and label binarizer...
Model Loaded...
Loaded model state_dict...
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning
Swimming status : drowning

7 Advantages & Disadvantages

7.1 Advantages

1. Continuous Monitoring: The Virtualeye system provides constant surveillance, eliminating the limitations of human supervision. It can monitor the pool area 24/7, ensuring a proactive approach to drowning detection.
2. Quick Detection: The system utilizes advanced technologies like computer vision and motion sensors to promptly detect signs of active drowning. It can identify distressed individuals in real-time, allowing for immediate response and potentially saving lives.
3. Accurate Distress Identification: The Virtualeye system's intelligent algorithms can accurately distinguish between normal pool activities and signs of distress. This reduces false alarms and ensures that lifeguards are alerted when there is a genuine risk.
4. Coverage in Challenging Areas: The system can monitor and detect drowning incidents even in challenging areas of the pool, such as deep ends or obscured sections. This extends the reach of lifeguards and improves overall safety coverage.
5. Nighttime Monitoring: The Virtualeye system can operate effectively during nighttime or low-light conditions, using infrared or thermal imaging capabilities. This ensures continuous monitoring, even when visibility is limited.

7.2 Disadvantages

1. Cost: Implementing the Virtualeye system can involve significant costs, including the installation of cameras, sensors, and the necessary infrastructure. This can be a barrier for smaller or budget-constrained swimming pool facilities.
2. Technical Limitations: The system's effectiveness can be affected by technical limitations, such as accuracy issues in detecting subtle signs of distress or challenges in differentiating between normal water activities and actual drowning incidents.
3. False Alarms: While efforts are made to minimize false alarms, there is still a possibility of false positives. Factors like splashing, playful activities, or sudden movements can trigger false alarms, which can cause disruptions and reduce trust in the system.
4. Reliance on Additional Staff: The Virtualeye system serves as a supplement to lifeguards but does not replace the need for human supervision. Adequate staffing and trained personnel are still required to respond to alerts generated by the system and perform rescue operations.
5. Maintenance and Updates: The system requires regular maintenance, updates, and calibration to ensure optimal performance. This can involve additional costs and ongoing technical support.

8 Applications

The Virtualeye virtual lifeguard system has specific applications in swimming pool environments for drowning detection:

1. Real-time Monitoring: The system continuously monitors the pool area in real-time, using computer vision and motion sensors to detect potential signs of distress or drowning. It analyzes various factors like body movements, underwater behavior, and anomalies to identify individuals who may be in trouble.

2. Prompt Alert Generation: Upon detecting a potential drowning incident, the Virtualeye system generates instant alerts or alarms to notify lifeguards and pool staff. These alerts can be sent to their mobile devices or displayed on a centralized monitoring system, enabling quick response and intervention.

3. Accurate Distress Identification: The advanced algorithms of the Virtualeye system help accurately distinguish between normal pool activities and distress situations. This reduces false alarms and ensures that lifeguards are alerted only when a genuine drowning risk is identified.

4. Coverage in Challenging Areas: The system can monitor and detect drowning incidents in challenging or hard-to-reach areas of the pool, such as deep ends, corners, or obscured sections. It provides comprehensive coverage and extends the reach of lifeguards to ensure all areas are monitored effectively.

5. Nighttime Monitoring: The Virtualeye system can operate effectively during nighttime or low-light conditions. It utilizes infrared or thermal imaging capabilities to detect body heat signatures, allowing for continuous monitoring even when visibility is limited.

6. Integration with Existing Safety Systems: The Virtualeye system can be integrated with other safety systems in the swimming pool environment, such as alarm systems, surveillance cameras, or automatic pool covers. This integration enhances the overall safety infrastructure and provides a comprehensive approach to drowning detection and prevention.

By focusing on drowning detection, the Virtualeye virtual lifeguard system offers real-time monitoring, prompt alerts, accurate distress identification, coverage in challenging areas, nighttime monitoring capabilities, and integration with existing safety systems. These applications aim to enhance drowning prevention efforts and ensure a safer swimming pool environment for all patrons.

9 Conclusion

Swimming is one of the best exercises that helps people to reduce stress in this urban lifestyle. Swimming pools are found larger in number in hotels, and weekend tourist spots and barely people have them in their house backyard. Beginners, especially, often feel it difficult to breathe underwater which causes breathing trouble which in turn causes a drowning accident. Worldwide, drowning produces a higher rate of mortality without causing injury to children. To overcome this conflict, a meticulous system is to be implemented along the swimming pools to save human life. By studying body movement patterns and connecting cameras to artificial intelligence (AI) systems we can devise an underwater pool safety system that reduces the risk of drowning. Usually, such systems can be developed by installing more than 16 cameras underwater and ceiling and analyzing the video feeds to detect any anomalies.

10 Future Scope

The future scope of virtual lifeguards for active drowning detection in swimming pools is promising and holds potential for further advancements:

1. Enhanced Detection Algorithms: Future developments can focus on refining the detection algorithms used by virtual lifeguards. This includes improving accuracy in identifying distress signals, distinguishing between normal and abnormal behaviors, and reducing false alarms.
2. Multi-Sensor Fusion: Combining data from multiple sensors, such as underwater cameras, depth sensors, and wearable devices, can provide a more comprehensive and accurate view of the pool environment. This integration can enhance the virtual lifeguard's capability to detect active drowning and respond effectively.
3. Integration with Pool Infrastructure: Virtual lifeguards can be integrated with pool infrastructure, such as automatic pool covers, water circulation systems, and alarm systems. This integration allows for coordinated responses, such as automatically closing the pool cover or activating alarms in the event of a detected drowning incident.
4. Remote Monitoring and Alert Systems: Advances in connectivity and IoT technologies can enable remote monitoring of swimming pools. Virtual lifeguards can transmit real-time data to remote monitoring centers or mobile devices, allowing for instant alerts and remote assistance in emergency situations.
5. Augmented Reality (AR) and Virtual Reality (VR): AR and VR technologies can be utilized to provide lifeguards with enhanced situational awareness. They can visualize the pool environment in real-time, receive alerts, and access critical information to make informed decisions and respond effectively.

By embracing these future developments, virtual lifeguards for active drowning detection in swimming pools have the potential to become more accurate, proactive, and integrated within the overall pool safety infrastructure, leading to safer swimming experiences and reduced drowning incidents.

11 Bibliography

[1] Laxton, V., & Crundall, D. (2018). The effect of lifeguard experience upon the detection of drowning victims in a realistic dynamic visual search task. Applied cognitive psychology, 32(1), 14-23.

[2] Salehi, N., Keyvanara, M., & Monadjemmi, S. A. (2016). An automatic video-based drowning detection system for swimming pools using active contours. Int. J. Image, Graph. Signal Process, 8(8), 1-8.

[3] Handalage, U., Nikapotha, N., Subasinghe, C., Prasanga, T., Thilakarthna, T., & Kasthurirathna, D. (2021, December). Computer Vision Enabled Drowning Detection System. In 2021 3rd International Conference on Advancements in Computing (ICAC) (pp. 240-245). IEEE.

[4]Shatnawi, M., Albreiki, F., Alkhoori, A., & Alhebshi, M. (2023). Deep Learning and Vision-Based Early Drowning Detection. Information, 14(1), 52.

[5]Salehi, N., Keyvanara, M., & Monadjemmi, S. A. (2016). An automatic video-based drowning detection system for swimming pools using active contours. Int. J. Image, Graph. Signal Process, 8(8), 1-8.

Appendix

Source Code

DrownDetect.py

```
import cvlib as cv
from cvlib.object_detection import draw_bbox
import cv2
import time
import numpy as np
import sklearn
import joblib
import cv2
import torch
import torch.nn as nn
import torch.nn.functional as F
import numpy as np
import albumentations
from torch.utils.data import Dataset, DataLoader
```

```
from PIL import Image
import time
import argparse


# Define command-line arguments
parser = argparse.ArgumentParser()
parser.add_argument('--source', required=True, help='Video source file name')

# Parse command-line arguments
args = parser.parse_args()

lb = joblib.load('lb.pkl')
class CustomCNN(nn.Module):
    def __init__(self):
        super(CustomCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, 5)  # changed 3 to 1
        self.conv2 = nn.Conv2d(16, 32, 5)
        self.conv3 = nn.Conv2d(32, 64, 3)
        self.conv4 = nn.Conv2d(64, 128, 5)
        self.fc1 = nn.Linear(128, 256)
        self.fc2 = nn.Linear(256, len(lb.classes_))
        self.pool = nn.MaxPool2d(2, 2)
    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = self.pool(F.relu(self.conv4(x)))
        bs, _, _, _ = x.shape
        x = F.adaptive_avg_pool2d(x, 1).reshape(bs, -1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x


print('Loading model and label binarizer...')
lb = joblib.load('lb.pkl')
model = CustomCNN()
print('Model Loaded...')
model.load_state_dict(torch.load('model.pth', map_location='cpu'))
model.eval()
print('Loaded model state_dict...')
aug = albumentations.Compose([
    albumentations.Resize(224, 224),
```

```
    ])

t0 = time.time() #gives time in seconds after 1970

def detectDrowning(source):
    isDrowning = False
    fram=0
    #input from the camera
    cap = cv2.VideoCapture("videos/" + source)

    if (cap.isOpened() == False):
        print('Error while trying to read video')
    frame_width = int(cap.get(3))
    frame_height = int(cap.get(4))
    while(cap.isOpened()):

        status, frame = cap.read()

        # apply object detection
        bbox, label, conf = cv.detect_common_objects(frame)

        # if only one person is detected, use model-based detection
        if len(bbox) == 1:
            bbox0 = bbox[0]
            #centre = np.zeros(s)
            centre = [0,0]


            for i in range(0, len(bbox)):
                centre[i] =[(bbox[i][0]+bbox[i][2])/2,(bbox[i][1]+bbox[i][3])/2 ]

            centre =[(bbox0[0]+bbox0[2])/2,(bbox0[1]+bbox0[3])/2 ]

            start_time = time.time()
            model.eval()
            with torch.no_grad():
                pil_image = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
                pil_image = aug(image=np.array(pil_image))['image']
                if fram == 500:
                    break
                fram+=1
                pil_image = np.transpose(pil_image, (2, 0, 1)).astype(np.float32)
                pil_image = torch.tensor(pil_image, dtype=torch.float).cpu()
                pil_image = pil_image.unsqueeze(0)
```

```
            outputs = model(pil_image)
            _, preds = torch.max(outputs.data, 1)

        print("Swimming status : ",lb.classes_[preds])
        if(lb.classes_[preds] =='drowning'):
            isDrowning = True
        if(lb.classes_[preds] =='normal'):
            isDrowning = False
        out = draw_bbox(frame, bbox, label, conf,isDrowning)

    # if more than one person is detected, use logic-based detection
    elif len(bbox) > 1:
        # calculate the centroid of each bounding box
        centres = []
        for i in range(len(bbox)):
            bbox_i = bbox[i]
            centre_i = [(bbox_i[0] + bbox_i[2])/2, (bbox_i[1] + bbox_i[3])/2]
            centres.append(centre_i)

        # calculate the distance between each pair of centroids
        distances = []
        for i in range(len(centres)):
            for j in range(i+1, len(centres)):
                dist = np.sqrt((centres[i][0] - centres[j][0])**2 + (centres[i][1] -
centres[j][1])**2)
                distances.append(dist)

        # if the minimum distance is less than a threshold, consider it as drowning
        if len(distances) > 0 and min(distances) < 50:
            isDrowning = True
        else:
            isDrowning = False
        out = draw_bbox(frame, bbox, label, conf, isDrowning)

    else:
        out = frame

    # display output
    cv2.imshow("Real-time object detection", out)

    # press "Q" to stop
    if cv2.waitKey(1) & 0xFF == ord('q'):
        cap.release()
        cv2.destroyAllWindows()
```

```
        exit()

detectDrowning(args.source)
```

App.py

```
from flask import Flask, render_template, request
import cvlib as cv
from cvlib.object_detection import draw_bbox
import cv2
import time
import numpy as np
import sklearn
import joblib
import cv2
import torch
import torch.nn as nn
import torch.nn.functional as F
import numpy as np
import albumentations
from torch.utils.data import Dataset, DataLoader
from PIL import Image
import time
import argparse
import os
import shutil

app = Flask(__name__)


@app.route('/')
def demo():
    return render_template('index.html')


lb = joblib.load('lb.pkl')


class CustomCNN(nn.Module):
    def __init__(self):
        super(CustomCNN, self).__init__()
```

```
        self.conv1 = nn.Conv2d(3, 16, 5)  # changed 3 to 1
        self.conv2 = nn.Conv2d(16, 32, 5)
        self.conv3 = nn.Conv2d(32, 64, 3)
        self.conv4 = nn.Conv2d(64, 128, 5)
        self.fc1 = nn.Linear(128, 256)
        self.fc2 = nn.Linear(256, len(lb.classes_))
        self.pool = nn.MaxPool2d(2, 2)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = self.pool(F.relu(self.conv4(x)))
        bs, _, _, _ = x.shape
        x = F.adaptive_avg_pool2d(x, 1).reshape(bs, -1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x


@app.route('/predict', methods=['POST'])
def log():
    output_file = 'output.jpg'

    video_file = request.files['filename']  # Use request.files to access the uploaded file
    video_file.save('videos/' + video_file.filename)  # Save the uploaded file to 'videos' folder

    print('Loading model and label binarizer...')
    lb = joblib.load('lb.pkl')
    model = CustomCNN()
    print('Model Loaded...')
    model.load_state_dict(torch.load('model.pth', map_location='cpu'))
    model.eval()
    print('Loaded model state_dict...')
    aug = albumentations.Compose([
        albumentations.Resize(224, 224),
    ])
    t0 = time.time()

    # actual code
    webcam = cv2.VideoCapture("videos/" + video_file.filename)
    isDrowning = False
    fram = 0
    if not webcam.isOpened():
```

```
    print('Error while trying to read video')

while True:
    status, frame = webcam.read()

    if not status:
        break

    # apply object detection
    bbox, label, conf = cv.detect_common_objects(frame)
    # if only one person is detected, use model-based detection
    if len(bbox) == 1:
        bbox0 = bbox[0]
        #centre = np.zeros(s)
        centre = [0, 0]

        for i in range(0, len(bbox)):
            centre[i] = [(bbox[i][0]+bbox[i][2])/2,
                         (bbox[i][1]+bbox[i][3])/2]

        centre = [(bbox0[0]+bbox0[2])/2, (bbox0[1]+bbox0[3])/2]

        start_time = time.time()
        model.eval()
        with torch.no_grad():
            pil_image = Image.fromarray(
                cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
            pil_image = aug(image=np.array(pil_image))['image']
            fram += 1
            pil_image = np.transpose(
                pil_image, (2, 0, 1)).astype(np.float32)
            pil_image = torch.tensor(pil_image, dtype=torch.float).cpu()
            pil_image = pil_image.unsqueeze(0)
            outputs = model(pil_image)
            _, preds = torch.max(outputs.data, 1)

        print("Swimming status : ", lb.classes_[preds])
        if (lb.classes_[preds] == 'drowning'):
            isDrowning = True
            predict='Drowning'
        if (lb.classes_[preds] == 'normal'):
            isDrowning = False
            predict='Normal'
        out = draw_bbox(frame, bbox, label, conf, isDrowning)
```

```
            cv2.imwrite(output_file, out)

        # if more than one person is detected, use logic-based detection
        elif len(bbox) > 1:
            # calculate the centroid of each bounding box
            centres = []
            for i in range(len(bbox)):
                bbox_i = bbox[i]
                centre_i = [(bbox_i[0] + bbox_i[2])/2,
                            (bbox_i[1] + bbox_i[3])/2]
                centres.append(centre_i)

            # calculate the distance between each pair of centroids
            distances = []
            for i in range(len(centres)):
                for j in range(i+1, len(centres)):
                    dist = np.sqrt(
                        (centres[i][0] - centres[j][0])**2 + (centres[i][1] - centres[j][1])**2)
                    distances.append(dist)

            # if the minimum distance is less than a threshold, consider it as drowning
            if len(distances) > 0 and min(distances) < 50:
                isDrowning = True
            else:
                isDrowning = False
            out = draw_bbox(frame, bbox, label, conf, isDrowning)

            cv2.imwrite(output_file, out)

        else:
            out = frame

            cv2.imwrite(output_file, out)

        # display output
        cv2.imshow("Real-time object detection", out)

        # press "Q" to stop
        if cv2.waitKey(1) == 27:  # press Esc to exit
            break
    webcam.release()
    cv2.destroyAllWindows()
```

```
    destination_folder = 'C:/Users/shrut/OneDrive/Desktop/Drowning-Detection--master/static'
    shutil.copy(output_file, destination_folder)
    return render_template('predict.html', output=predict)


if __name__ == '__main__':
    app.run()
```