Swimmer Segmentation and Pose Estimation using SAM and YOLO

Sai Preeth Adwuala, BSEE
Department of Electrical Engineering
San Jose State University
Email: saipreeth.aduwala@sjsu.edu

***Abstract*—This paper explores the challenges and solutions in developing a system for swim pose estimation using YOLOv8 and the Segment Anything Model (SAM). Accurate swimmer segmentation and pose estimation can enhance performance evaluation and technique optimization in swimming. The project integrates YOLOv8 for swimmer detection and SAM for precise segmentation. Initial results show promising segmentation performance, but pose estimation faces significant challenges due to occlusions, dynamic aquatic environments, and non-standard swimmer orientations. Future work will focus on improving pose estimation through advanced model fine-tuning and aquatic-specific data augmentation.**

***Index Terms*—YOLOv8, pose estimation, SAM, segmentation, swimming biomechanics, computer vision.**

I. Introduction

A. Motivation

Swimming biomechanics analysis is crucial for improving performance by optimizing body posture and minimizing drag. However, analyzing swimming poses is challenging due to environmental factors such as water refraction, bubbles, splashes, and the horizontal orientation of swimmers, which existing pose estimation models fail to handle effectively. This project aims to overcome these challenges by leveraging state-of-the-art models like YOLOv8 and SAM to create a robust swimmer segmentation and pose estimation pipeline.

B. Objectives

1) **Segmentation:** Accurately separate swimmers from the pool background using SAM.
2) **Pose Estimation:** Identify key body joints and their movements to analyze swimming techniques.
3) **Real-Time Feasibility:** Ensure the pipeline operates efficiently for real-time feedback.

C. Significance of the Work

This project provides a foundation for enhancing swimmer performance analysis through automated computer vision techniques. Unlike traditional methods requiring controlled environments, the proposed approach seeks to adapt to real-world swimming conditions. By addressing the limitations of existing models, this research contributes to the broader field of sports analytics and biomechanics.

II. Design and Methodology

A. System Architecture

The proposed system consists of two primary phases:

1) **Segmentation Phase:** YOLOv8 detects swimmer bounding boxes, and SAM generates masks to isolate swimmers from the background.
2) **Pose Estimation Phase:** MediaPipe Pose estimation model is used for keypoint detection, yielding key body joint positions for analysis.

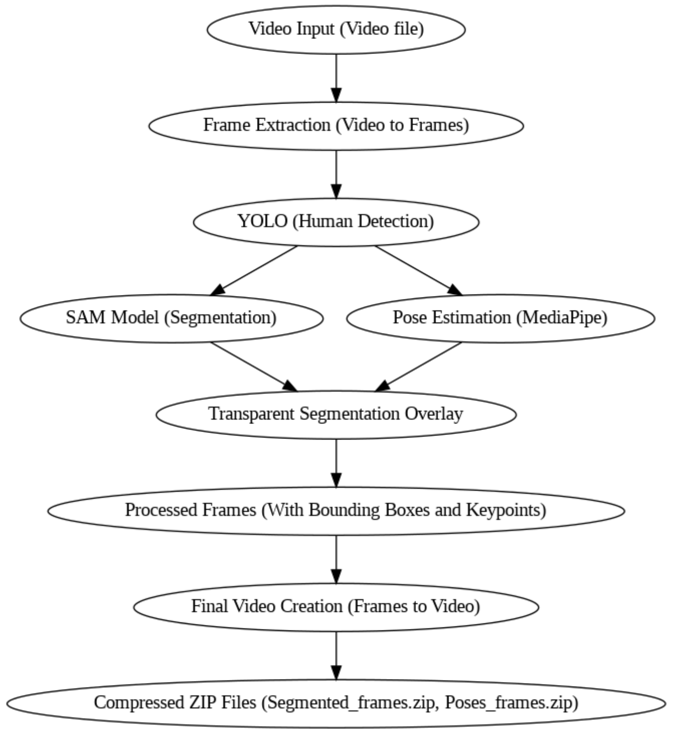
Video Input (Video file)
Frame Extraction (Video to Frames)
YOLO (Human Detection)
SAM Model (Segmentation)
Pose Estimation (MediaPipe)
Transparent Segmentation Overlay
Processed Frames (With Bounding Boxes and Keypoints)
Final Video Creation (Frames to Video)
Compressed ZIP Files (Segmented_frames.zip, Poses_frames.zip)

Fig. 1. Proposed system architecture for swimmer segmentation and pose estimation.

B. Algorithm Workflow

The workflow for video processing and analysis is as follows:

1) **Task 1 (Segmentation):**
   - Extract frames from the input video.

- YOLOv8 detects swimmers in each frame and generates bounding boxes.
- SAM uses the bounding boxes to generate precise segmentation masks of swimmers.
- Segmented frames are used to generate a new video with the swimmer isolation visualized.

2) **Task 2 (Pose Estimation):**
- Use the segmented frames from Task 1 as input to MediaPipe Pose estimation.
- Extract keypoints for body joints (e.g., shoulders, elbows, knees) from each frame.
- Store the pose estimation results as images for further analysis.

C. *Code Implementation*

The Python implementation for video processing, segmentation, and pose estimation is described below:

Listing 1. Your Code Caption

```
# Install necessary packages
%pip install ultralytics segment-anything mediapipe
import cv2
import os
import numpy as np
from ultralytics import YOLO
from segment_anything import SamPredictor,
    sam_model_registry
import mediapipe as mp

# Load YOLOv8 model and SAM
model = YOLO("yolov8n.pt")
DEVICE = torch.device('cuda' if torch.cuda.
    is_available() else 'cpu')
sam = sam_model_registry["vit_h"](checkpoint='
    sam_vit_h_4b8939.pth')
sam.to(device=DEVICE)
mask_predictor = SamPredictor(sam)

# Load MediaPipe Pose model
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(min_detection_confidence=0.5,
    min_tracking_confidence=0.5)

# Video processing: Task 1 - Segmentation
input_video = cv2.VideoCapture('input_video.mp4')
output_video = cv2.VideoWriter('segmented_output.mp4
    ',
                                cv2.
                                    VideoWriter_fourcc
                                    (*'mp4v'),
                                30, (640, 480))

while input_video.isOpened():
    ret, frame = input_video.read()
    if not ret:
        break
    results = model(frame)  # YOLOv8 detection
    boxes = results[0].boxes.xywh

    # Generate masks using SAM for each detected
        bounding box
    for box in boxes:
        mask_predictor.set_image(frame)
        masks, _, _ = mask_predictor.predict(box=box
            )
        # Overlay the mask on the original frame
        segmented_frame = np.multiply(frame, masks)
             # Apply mask to the frame
        output_video.write(segmented_frame)  # Write
             to output video

input_video.release()
output_video.release()

# Pose Estimation: Task 2
input_video = cv2.VideoCapture('segmented_output.mp4
    ')
pose_images = []

while input_video.isOpened():
    ret, frame = input_video.read()
    if not ret:
        break

    # Process each frame with MediaPipe Pose
    frame_rgb = cv2.cvtColor(frame, cv2.
        COLOR_BGR2RGB)
    result = pose.process(frame_rgb)

    # Draw pose landmarks on the frame
    if result.pose_landmarks:
        mp.solutions.drawing_utils.draw_landmarks(
            frame, result.pose_landmarks, mp_pose.
            POSE_CONNECTIONS)

    # Save pose images
    pose_images.append(frame)

input_video.release()
```

III. Simulation and Results

A. *Segmentation Performance*

The segmentation phase demonstrated high accuracy in isolating swimmers from pool backgrounds. Figure 3 shows an example of the segmented output video. However, issues such as occlusions from bubbles and splashes occasionally reduced segmentation quality.

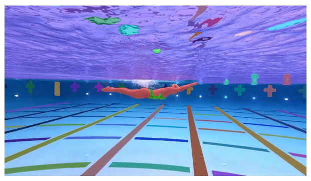
Fig. 2. Segmentation Output (SAM)

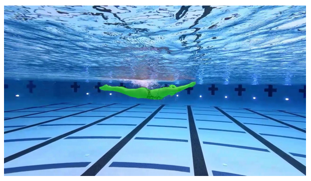
Fig. 3. Segmentation Output (YOLO and SAM)

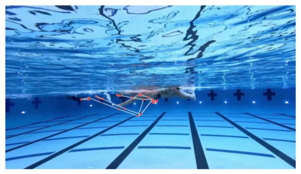
Fig. 4. Pose Estimation (No Detection)

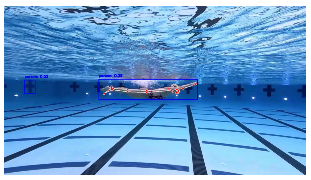
Fig. 5. Pose Estimation (Detection)

Fig. 6. Sample snippets of swimmer images

B. *Pose Estimation Challenges*

Initial testing of pose estimation revealed challenges in detecting joints due to:

- Horizontal orientations of swimmers.
- Occlusions caused by water dynamics.
- Limited training data for aquatic-specific scenarios.

Pose images are stored for each frame and further analyzed for swimmer joint positions.

C. Performance Metrics

- **Segmentation Accuracy:** Around 90% of swimmers were successfully segmented.
- **Pose Estimation Accuracy:** Around 20 % of the swimmer's poses were successfully estimated.

IV. Conclusion and Future Work

This work demonstrates the effectiveness of YOLOv8 and SAM for swimmer segmentation in real-world scenarios. While the segmentation phase is largely successful, significant challenges remain in pose estimation, especially in dynamic aquatic environments. Future efforts will focus on:

- Fine-tuning pose estimation models for underwater environments.
- Collecting and annotating a larger dataset of swimming videos.
- Integrating rotation correction for better keypoint detection in non-upright swimmers.

References

1) Meta AI, Segment Anything Model, *https://github.com/facebookresearch/segment-anything*.
2) YOLOv8 Documentation, *https://github.com/ultralytics/yolov8*.
3) MediaPipe Pose, *https://google.github.io/mediapipe/solutions/pose*.

Appendix

A. Code

The complete implementation is available in the following URL: Google Colab Notebook.

B. Results

The entire results of current execution are available in the following URL: Google Drive Folder.

C. Video Presentation

A demonstration of the system can be found at the following URL: YouTube Video.