An Intelligent Program to Assist in Swim Training and Technique Optimization Utilizing Machine Learning and Motion Tracking

Daniel C. Cao[1], Yu Sun[2]

[1]Los Osos High School, 6001 Milliken Ave, Rancho Cucamonga, CA 91737
Danccao333@gmail.com
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768
softcom.lab.cpp@gmail.com

ABSTRACT

This program addresses the shortfall in advanced technology application in swim training by introducing an innovative method that combines swimmer videos with AI for optimizing swimming starts. Utilizing common devices like smartphones for recording, alongside OpenCV for video processing and MediaPipe for accurate landmark detection, it navigates challenges in video input and output efficiently. The effectiveness of this approach was validated through two experiments focusing on the technology's accuracy and its impact on training quality. Results highlighted the technology's positive contribution to swim training, emphasizing its potential to revolutionize training techniques. This non-invasive, accessible solution emphasizes critical race aspects and offers scope for continuous innovation and improvement, demonstrating significant promise for enhancing athletic training across all levels.

KEYWORDS

Swim Training Technology, AI Landmark Detection, Pose Estimation, Innovation in Swim Training

1. INTRODUCTION

Swimming is a popular sport filled up with a diverse group of people from recreational swimmers to world-class athletes. However, the sport lacks modern day equipment; there are few reliable and efficient analysis systems that can be used to optimize technique. Today, swimming relies heavily on traditional coaching methods and there are significant challenges in obtaining accurate real-time data. The lack of access to advanced technologies in swimming is partly due to the fact that there is a technological gap between swimming and other sports. For instance, the usage of line-scan systems in track and field are able to accurately time athletes completely automatically. On the contrary, swimming touchpads used to measure times have common errors: the touchpad can fail to start, stop, and fail to accurately measure times when swimmers don't hit the touchpad with enough force [1].

Technological factors such as these can be huge determinants of an athlete's success. Oftentimes, races can come down to a hundredth of a second. In fact, Michael Phelps' seventh Olympic gold medal was won over Milorad Cavic by a hundredth of a second. The importance of technology in swimming has many aspects. First of all, technology that analyzes technique can have a huge impact on improving efficiency. Since races can come down to a hundredth of a second, every action matters, even the movements that aren't commonly associated with swimming—reaction time, the angle at which one pushes off the wall, how wide an athlete opens their hands—optimizing these could result in major improvements in both physical performance and race times. Studies show that elite swimmers can even train visual perception in relay swimming in order to gain a faster reaction time [2].

From Olympians to recreational swimmers, the implementation of technology can affect a wide group of people [3]. Using technology that can catalyze training results is crucial for every level of swimming, however many individuals may not have access to expensive equipment or resources to use technology. This is why having technology that is readily available to use with everyday devices is important to train swimmers of all levels into their full potential.

A solution is to combine swimming technique with a program that analyzes dive techniques through pose estimation with AI to break down swimming technique. In the past, pose estimation has been used for counting strokes and stroke classification [4]. Additionally, most technology makes use of either wearable sensors or recorded video [5][6][7][8]. Videos are often the simplest and easiest ways to obtain data, however there is a lack of extensive analysis. Since this program doesn't require the usage of any professional camera equipment and utilizes the input of a video for the resulting output, it will be available for most people in the modern day to use. Additionally, this program will not have a huge impact on traditional coaching methods; it does not completely change how a swimmer trains and only seeks to provide further information that analyzes a swimmer's technique, and it is up to the athlete and their coach to determine what they can do with that information.

Because this program can be used by a wide number of people, it is a better solution than other methods that are used to increase the performance of swimmers. In the past, methods such as new tech suits have created controversy in the sport: swimmers wearing Speedo's new Fastskin at the Sydney 2000 Olympics won eighty-three percent of all medals [3]. Additionally, using technology that provides an unfair advantage can also impact all athletes of an entire nation. For example, Chinese swimmers in the 1990's who used performance enhancing drugs still affect the stereotypes of Chinese athletes today [9]. Methods that are only available to certain groups can create an unfair advantage, but a program that can be easily run on a mobile device acts only to assist swimmers in their technique by giving them more information, and allows them to train more effectively.

2. Challenges

In order to build the project, a few challenges have been identified as follows.

2.1. Getting the pose data

One problem that needs to be addressed for the successful creation of this program is getting the pose data itself and obtaining meaningful measurements automatically from a video. Since this program relies heavily on mapping poses accurately, it is crucial that the artificial intelligence that maps the pose is accurate. This problem could be solved by utilizing OpenCV and Mediapipe. OpenCV will allow the computer to "read" the images/videos that are fed to it, and Mediapipe's existing pose landmark can be used for detection to generate points on the video and output data. Since Mediapipe's pose landmark detection is already trained, it will be accurate in detecting these points.

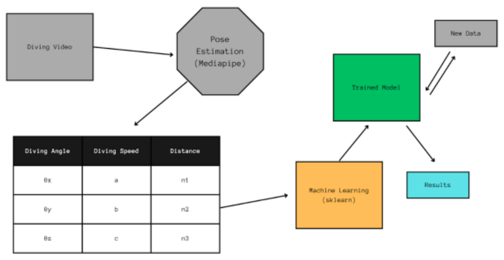
2.2. The RGB and BGR image formats

Another thing that would need to be considered is the RGB and BGR image formats used by Mediapipe and OpenCV respectively, and only by converting OpenCV's BGR into RGB will the needed output be created. A second problem that could be present is the issue of taking out the landmark data and creating a visualization of the numbers. The program needs to output a familiar image in order for it to be understood effectively and for it to be used to enhance swimming techniques. This could be done by using openCV to draw on the image; graphical overlays on the 2D image or video could create a visual representation that will be familiar to read. A familiar model would be the best representation of the data, as it will allow users to make the connections between the output model and the swimmer in the video, giving them more information for further analysis.

2.3. How to use the resulting data

Additionally, one of the biggest concerns is how to use the resulting data. Even with the pose estimation correctly utilized, there needs to be a way to represent the data for athletes in order for the program to be successfully used. A way to solve this would be to implement overlays on the model itself, such as the overlay of the diving angle that is used. Since kinematics is a large part of projectile movement, there could be angles that are better for the swimming start, and users could use this to optimize their technique. Another way to solve this problem would be to add logs, allowing users to see past attempts to compare and contrast different starts.

3. Solution



Diving Angle	Diving Speed	Distance
θx	a	n1
θy	b	n2
θz	c	n3

Figure 1. Flowchart showing the process of how the program will be created

In order for the program to function correctly, there needs to be three things to be taken into account. The three main things that this program links together is a video taken from a recording device, a pose estimation AI, and an analysis engine. First of all, there needs to be a functioning recording apparatus. We decided to use an iPhone camera in this scenario, as phone cameras are easily accessible and it is easy to transfer data to a computer quickly, however any recording device can be used. Once the video has been recorded, the data needs to be transferred into the computer; whether this is through a USB drive, cloud drive, or other method does not matter. This video will be placed into a video folder inside of the program for easy access. The first component of this program is a crucial part; the video makes up all of the input that our program will be using to output data values. Our program uses Python for the programming language, openCV to allow the computer to “read” the video, and Mediapipe’s pose estimation for video analysis. We chose to use Python because it allowed us to access the Mediapipe library. This is important as the accuracy of the data depends on the model that is outputted by the program, and Mediapipe has a well-trained pose estimation AI that can be used to access landmarks in order to get important points that are used for future analysis. By combining Mediapipe and OpenCV, we obtain data that can then be imported into the third component: the analysis engine. The analysis engine takes the data, arm and leg angles, launch angle, and distance between landmarks to predict the velocity and displacement of the swimmer.

Component A: Camera + OpenCV

The video that is used for input is crucial to the program’s output. The video will be recorded on a camera attached to a tripod, in the same location every time. This strengthens the accuracy of our data by making factors consistent. The video input depends on OpenCV in order to be read and analyzed. OpenCV will be used to open the video and extract frame data, and its computer vision algorithms will allow for the utilization of tracking objects and pose recognition.

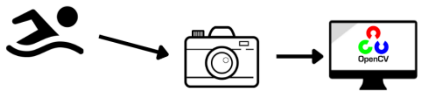
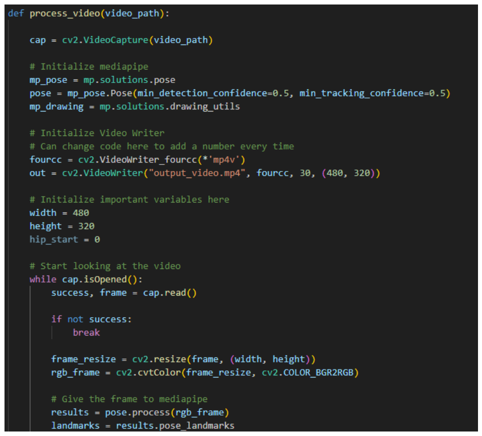


Figure 2. UI demonstration of program and Component A



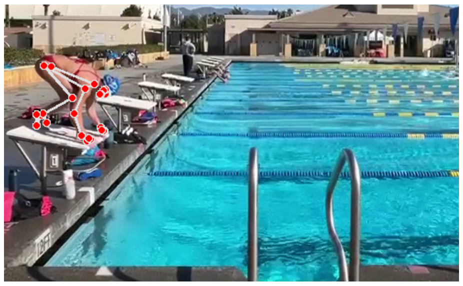
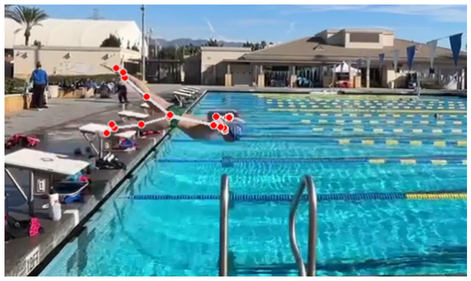
```
def process_video(video_path):

    cap = cv2.VideoCapture(video_path)

    # Initialize mediapipe
    mp_pose = mp.solutions.pose
    pose = mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5)
    mp_drawing = mp.solutions.drawing_utils

    # Initialize Video Writer
    # Can change code here to add a number every time
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    out = cv2.VideoWriter("output_video.mp4", fourcc, 30, (480, 320))

    # Initialize important variables here
    width = 480
    height = 320
    hip_start = 0

    # Start looking at the video
    while cap.isOpened():
        success, frame = cap.read()

        if not success:
            break

        frame_resize = cv2.resize(frame, (width, height))
        rgb_frame = cv2.cvtColor(frame_resize, cv2.COLOR_BGR2RGB)

        # Give the frame to mediapipe
        results = pose.process(rgb_frame)
        landmarks = results.pose_landmarks
```

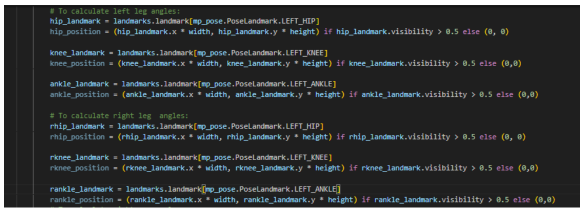
Figure 3. Code sample of process_video function

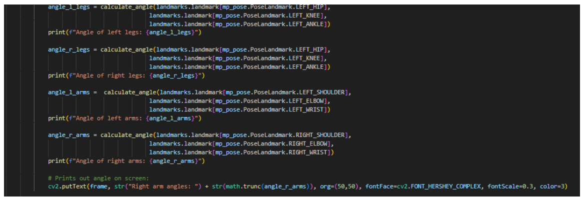
The process_video function is one of the most key functions in this program. This function is called in the main.py file where it runs through a video that is inputted by the user. OpenCV acts to analyze the inputted video and pass it onto Mediapipe. First, a Video Capture object is created in order to capture video frames from the file. Then a Video Writer object is created for the intention of saving the video. The width and height of the video are set to be 480 and 320 respectively as lowering the resolution results in less pixels needing to be accessed, reducing the computational load to process and manipulate data. The dimensions of the video are set to be variables for easy access. Before the video frames can be transferred to Mediapipe, one more thing needs to be done. This function also includes resizing and recoloring the data. Resizing the frame allows for consistency, which is important for the accuracy of the model. Additionally, because OpenCV codes videos in blue-green-red (BGR) format, the colors need to be recolored to fit Mediapipe's red-green-blue (RGB) format.

Component B: Pose Estimation

The second component uses Mediapipe in order to generate data from the video frames. First of all, the video is sent as frames to Mediapipe using OpenCV. Mediapipe's pose estimation is specifically being used; this library allows for the detection of landmarks on human bodies in image or video by using machine learning models with single videos. Specific landmarks will be marked in order to detect not poses, but techniques and angles in the swimming start.



Figures 4 and 5: Pose results with landmarks drawn on two different swimmers in the start position (4) and in the position after the launch (5)


```
# To calculate left leg angles:
hip_landmark = landmarks.landmark[mp_pose.PoseLandmark.LEFT_HIP]
hip_position = (hip_landmark.x * width, hip_landmark.y * height) if hip_landmark.visibility > 0.5 else (0, 0)

knee_landmark = landmarks.landmark[mp_pose.PoseLandmark.LEFT_KNEE]
knee_position = (knee_landmark.x * width, knee_landmark.y * height) if knee_landmark.visibility > 0.5 else (0,0)

ankle_landmark = landmarks.landmark[mp_pose.PoseLandmark.LEFT_ANKLE]
ankle_position = (ankle_landmark.x * width, ankle_landmark.y * height) if ankle_landmark.visibility > 0.5 else (0,0)

# To calculate right leg  angles:
rhip_landmark = landmarks.landmark[mp_pose.PoseLandmark.LEFT_HIP]
rhip_position = (rhip_landmark.x * width, rhip_landmark.y * height) if rhip_landmark.visibility > 0.5 else (0, 0)

rknee_landmark = landmarks.landmark[mp_pose.PoseLandmark.LEFT_KNEE]
rknee_position = (rknee_landmark.x * width, rknee_landmark.y * height) if rknee_landmark.visibility > 0.5 else (0,0)

rankle_landmark = landmarks.landmark[mp_pose.PoseLandmark.LEFT_ANKLE]
rankle_position = (rankle_landmark.x * width, rankle_landmark.y * height) if rankle_landmark.visibility > 0.5 else (0,0)
```



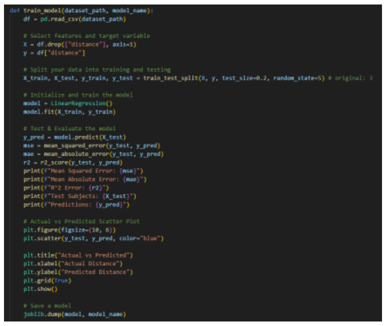
```
angle_l_legs = calculate_angle(landmarks.landmark[mp_pose.PoseLandmark.LEFT_HIP],
                               landmarks.landmark[mp_pose.PoseLandmark.LEFT_KNEE],
                               landmarks.landmark[mp_pose.PoseLandmark.LEFT_ANKLE])
print(f"Angle of left legs: {angle_l_legs}")

angle_r_legs = calculate_angle(landmarks.landmark[mp_pose.PoseLandmark.LEFT_HIP],
                               landmarks.landmark[mp_pose.PoseLandmark.LEFT_KNEE],
                               landmarks.landmark[mp_pose.PoseLandmark.LEFT_ANKLE])
print(f"Angle of right legs: {angle_r_legs}")

angle_l_arms =  calculate_angle(landmarks.landmark[mp_pose.PoseLandmark.LEFT_SHOULDER],
                               landmarks.landmark[mp_pose.PoseLandmark.LEFT_ELBOW],
                               landmarks.landmark[mp_pose.PoseLandmark.LEFT_WRIST])
print(f"Angle of left arms: {angle_l_arms}")

angle_r_arms = calculate_angle(landmarks.landmark[mp_pose.PoseLandmark.RIGHT_SHOULDER],
                               landmarks.landmark[mp_pose.PoseLandmark.RIGHT_ELBOW],
                               landmarks.landmark[mp_pose.PoseLandmark.RIGHT_WRIST])
print(f"Angle of right arms: {angle_r_arms}")

# Prints out angle on screen:
cv2.putText(frame, str("Right arm angles: ") + str(math.trunc(angle_r_arms)), org=(50,50), fontFace=cv2.FONT_HERSHEY_COMPLEX, fontScale=0.3, color=3)
```

Figure 6 and 7: Code sample for Mediapipe landmarks (6) and angle analysis (7)

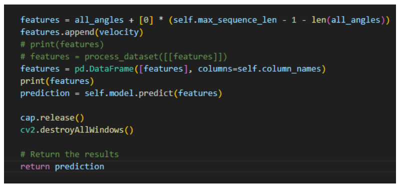
The coding for the Mediapipe analysis is important to create the data. One of the most important pieces of data that is to be obtained from the video frames is the angles of various limbs. Because the swimming start relies so heavily on technique and power, specific angles will allow for a more optimized dive. The code calculates two important angles: legs and arms. To create an angle, three points need to be selected. For the legs, these three points are the hip, knee, and ankle landmarks and positions. These landmarks are acquired and transferred to two other functions: the calculate_angle and calculate_distance functions. These functions take the three landmarks and convert them to a numerical value. For example, there are angles for both the right and left arms and legs; both of these angles are printed and updated every frame. These are also overlaid onto the video itself using openCV's putText function. The angle of the arms are represented using the right arm, as both arms are symmetrical in the start, but the leg angles are shown separately.

Component C: Analysis Methodology

The analysis methodology uses a trained machine in order to predict and output data. Specific landmarks that are mapped in MediaPipe are used to determine a few data points that can be used for the AI engine. These data values include all of the key angles of the swimmer's dive along with the velocity of the swimmer. The AI engine takes all this as a csv file and selects features from the DataFrame, splitting it into testing and training sets, and then outputs more information.



```
def train_model(dataset_path, model_name):
    df = pd.read_csv(dataset_path)

    # Select features and target variable
    X = df.drop(["distance"], axis=1)
    y = df["distance"]

    # Split your data into training and testing
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5) # original: 3

    # Initialize and train the model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Test & Evaluate the model
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f"Mean Squared Error: {mse}")
    print(f"Mean Absolute Error: {mae}")
    print(f"R^2 Error: {r2}")
    print(f"Test Subjects: {X_test}")
    print(f"Predictions: {y_pred}")

    # Actual vs Predicted Scatter Plot
    plt.figure(figsize=(10, 6))
    plt.scatter(y_test, y_pred, color="blue")

    plt.title("Actual vs Predicted")
    plt.xlabel("Actual Distance")
    plt.ylabel("Predicted Distance")
    plt.grid(True)
    plt.show()

    # Save a model
    joblib.dump(model, model_name)
```



```
features = all_angles + [0] * (self.max_sequence_len - 1 - len(all_angles))
features.append(velocity)
# print(features)
# features = process_dataset([[features]])
features = pd.DataFrame([features], columns=self.column_names)
print(features)
prediction = self.model.predict(features)

cap.release()
cv2.destroyAllWindows()

# Return the results
return prediction
```

Figures 8 and 9: Code samples for training the AI and testing the AI

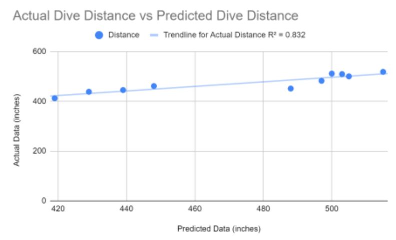
The AI engine is composed of two parts: a trainer file and a tester file. The trainer file takes all the data values mentioned above as a csv file and selects features from the DataFrame, splitting it into testing and training sets. Additionally, a column containing the respective true distances for each swimmer in the video list is added to the csv. 80% of this information is used to train the AI, while 20% of this information is placed into the testing set. The csv file (here it is named swimming.csv) is converted into a model, outputting the model as a .pkl, which will then be used for the tester. The tester file analyzes one video based on the videos that are used to train it. It first plays out the video that the user selects to be analyzed and grabs all of the angles and velocities to compare it to what the AI engine has learned. On completion, it adds all of this into the DataFrame features, which is then used to predict the displacement of the dive. The displacement of the dive is outputted as the prediction in the same unit of measure used in the distance column (in this case, inches).

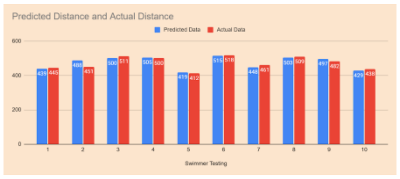
4. Experiment

4.1. Experiment 1

This program brings up some blind spots. One of them is the accuracy of the AI engine. In other words, how well does the system map out the data compared to reality?

This can be tested through an experiment testing the predicted results versus the actual results. In order to ensure accurate data for this experiment, a few components need to be set into place. First, a specific camera angle and location will need to be kept consistent throughout training the ML model. This is to reduce the variability in displacement (changing the displacement would affect velocity, and thus the ML model itself). The camera angle selected was a 90 degree angle from the right-hand side of the swimmer. The camera was kept the same distance from the swimmer for every video to keep the data consistent. The swimmer would dive when sent off and would hold a streamline underwater to glide as far as they could. The moment where the swimmer surfaced was when their distance was recorded. A sample of 10 swimmers was taken from a population of 15 swimmers in the most advanced group at a swim club to reduce variability.



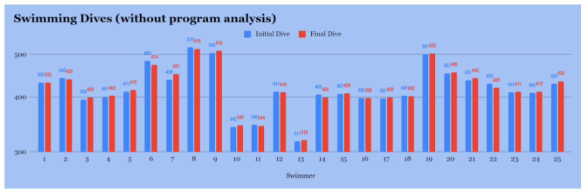


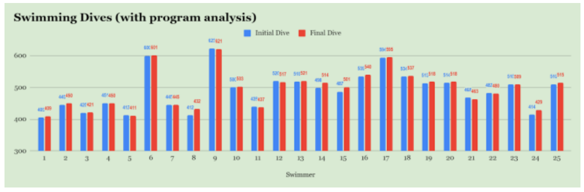
Figures 10 and 11: Scatter chart with predicted data values (10) Column chart showing distances per swimmer (11)

The charts show the data values of the dives, with predicted results being the results that the AI engine outputs, and actual results being the distances that were measured in real life. On average, the AI engine predicted a distance that was 11.2 inches off the actual distance. The regression line created from the data is linear and positive. These show that the AI engine was somewhat accurate in its predictions of the dives based on the data it was given. However, an issue was that the AI engine often outputted a predicted dive that was less than the real distance—this happened 6 times as opposed to 4 times where it outputted a greater value—which may influence how swimmers are trained. The goal of the AI Engine is to take the data and make the dive as optimal as possible, as higher predictions would mean that the dive could be better, and predictions that are around the same would mean that the dive technique is already somewhat ideal. These results could be caused by the training of the AI Engine; it was only trained with a few videos from a video list, and more videos would definitely improve its predictions.

4.2. Experiment 2

Another important area that needs to be tested is how effective this system was at improving athlete training. To test this, an experiment was set up with a sample of 50 swimmers selected completely randomly. Half of the swimmers were unknowingly put into a control group of people who didn't use the system. The other half of the swimmers were put into another group of people who did use the system. Both groups were blinded, and didn't know that the experiment was happening. The groups were each given advice by the same coach to reduce variability in suggestions based on the usage of the system or not. Each swimmer attempted a dive and their distance was recorded. After receiving advice about their dive (the control group got simple advice, while the other group analyzed their dive according to the program), the swimmers tried again and their new distance was recorded.





Figures 12 and 13: Column charts showing the before and after of dive distances after training with/without the program

The data shows that swimmers who utilized the program generally had more improvement than swimmers who didn't utilize the program. For swimmers who didn't utilize the program, 15 of them improved their distance, 2 stayed the same, and 8 did worse by a small amount. Regarding the changes in distance, swimmers who improved their dive generally improved an average of 3.8 inches, and swimmers who didn't improve generally had a dive that was, on average, 3.4 inches less than their initial one. On the other hand, of the swimmers who did utilize the program 16 of them improved their distance, 1 stayed the same, and 8 did worse by a small amount. Regarding this group's distance changes, swimmers who improved their dive generally improved an average of about 6.56 inches, and swimmers who didn’t improve generally dove about 1.89 inches less. On average, swimmers who utilized the program had more improvement; factors that could affect this conclusion are the advice that the coach is giving the swimmers, how well the swimmers respond to the advice, and the increased data that is given to the swimmers by the program.

Two experiments were run in order to test the accuracy of the program and its impacts on swim training. The first experiment concerned the AI Engine’ s ability to accurately predict the dive distance. Data from 10 swimmers of high ability were sent to be analyzed in the AI Engine, which was able to predict somewhat accurately the distance of the swimmer given the angles of their dive, and was off by 11.2 inches on average. This experiment demonstrated that the AI Engine had potential to predict with a lot higher accuracy, as it was trained with a few videos. It also demonstrates the effectiveness of pose estimation in the sport, as the angles at which a swimmer sets up on a block has a high effect on how far they go. The second experiment concerned training using the data from the program. It involved blinding two groups of 25 swimmers, divided into control and experimental groups. The control group received traditional coaching and dove with new information. Their dives before and after the coaching was recorded. This same process was repeated with the experimental group, however they received the annotated video of their dive, information outputted by the program, and coaching before they attempted to dive a second time. The experimental group showed higher improvement compared to the control group, demonstrating that the program was somewhat successful in refining how a swimmer can train.

5. Related work

Methodology A: Inertial Sensors

Inertial sensors are novel methods of analysis consisting of microelectromechanical systems, accelerometers, and gyroscopes [10]. These inertial sensors are attached to the swimmer, which in turn, obtains specific data. This solution is fairly effective to produce specific analysis. Additionally, it provides analysis to the four strokes, as all of them can be reduced into different locomotion phases between walls [11]. However, because of the overall difficulty in obtaining accurate data in aquatic environments, there are some challenges that this solution faces. Sensors are difficult to be placed in efficient places. Sometimes, a specific swimsuit is required in order to get it in the correct place, however this may be of annoyance to the swimmer, as it is quite bulky. Even single inertial sensors placed on lateral sides of swimmers were found to be uncomfortable [11]. This device also ignores the device’s influence on coaching practice; commercial systems seem to be more geared towards recreational swimmers who want more data.

Since the pose estimation uses a video to record the swimmers and accurately provides them information based on a pose-estimation, there is no need for extra sensors to be placed onto the swimmer that may cause discomfort and interfere with their swimming technique.

Methodology B: Other Uses of Pose Estimation

Pose estimation can also be used to analyze the strokes, however analysis for starts and turns are underdeveloped because the technology usage is lacking. This pose estimation is able to estimate what stroke a swimmer is doing, and creates recordings that can be used to quantitatively assess performance [12]. However, not much else can be accounted for without further analysis, and while the data could be used for training, there is no further information given past this model. It also ignores swimming starts (or dives), a key part to swimming, as at any point in the water, it is off the start where an athlete is the fastest in the water. In fact, once in the water, the swimmer never gains velocity, but will decelerate until they hit the speed that they will maintain for the rest of the distance [13]. Because our solution focuses on the start, specific training methods could be utilized with the analysis data that is outputted. This also allows the athlete to focus on one of the most integral parts of their race.

Methodology C: Accelerometers

Accelerometers are an interesting piece of technology that can be used to determine the velocities of swimmers. It can be used to measure the push-off velocities of swimmers off the walls, which is crucial to the race as it is off the start and turns where the swimmer will be the fastest in the entire race [14]. Additionally, accelerometers can be used to recognize swimming styles, stroke count, and intensity of exercise [15]. However, this relies on sensors that bring up the same issue as the inertial sensors. Accelerometers are taped on the swimmers, often at the lower back position, which could result in discomfort while swimming. However, this method does end up calculating the acceleration and maximum push-off velocity successfully. Since the video solution is less invasive than attaching to the athlete's body, it allows them to swim the same way that they usually do without discomfort.

Three methodologies for technology in swimming were mentioned in this paper. Methodology A concerned inertial sensors, which was able to output specific stroke data, but faced challenges regarding the aquatic environments and comfortability of swimmers due to attaching the inertial sensors themselves. Methodology B concerned pose estimation, which was effective for stroke data, but lacked focus on starts and turns. Out of all the methods, B was the least invasive due to its video recording element. Methodology C involved the usage of accelerometers to determine swimmer velocity. However, this faced the same challenge present in Methodology A, as swimmers could face discomfort in sensor placement. Each of these approaches had unique advantages and challenges in what they tried to accomplish, and the pose estimation program created in this paper targets these and tries to combine them: it attempts to output analyzed data that is accurate while also utilizing a less invasive solution.

6. CONCLUSIONS

This project, focusing on enhancing swimming starts, currently analyzes only the dive and immediate follow-through. Expanding its scope to include underwater phases could provide a more comprehensive analysis of a swimmer's speed. Improvements are also needed in the machine learning (ML) model's accuracy. By incorporating data from a wider range of dives and swimmer body types, and adjusting for variables like starting block differences, the model can offer more precise feedback for technique refinement.

Although in its nascent stage, this program's application of pose tracking technology in swimming shows significant promise. It has the potential to offer detailed data for improving training methods and individual techniques, heralding advancements in the application of technology in sports.

REFERENCES

[1] Tanner, David A. "Sprint Performance Times Related to Block Time in Olympic Swimmers." Journal of Swimming Research 15 (2001).

[2] Conceição, Ana, et al. "Occlusion technique in swimming: a training method to improve exchange block time in swimming relays." The Journal of sports medicine and physical fitness 60.7 (2020): 957-964.

[3] Preston, Claire, and Daniel KN Johnson. "Fastest in the pool: the role of technological innovation on swimming record breaks." (2011).

[4] Costa, Joana, et al. "Framework for intelligent swimming analytics with wearable sensors for stroke classification." Sensors 21.15 (2021): 5162.

[5] Giulietti, Nicola, et al. "SwimmerNET: Underwater 2D Swimmer Pose Estimation Exploiting Fully Convolutional Neural Networks." Sensors 23.4 (2023): 2364.

[6] Xu, Shaofeng, and Somi Lee. "An inertial sensing-based approach to swimming pose recognition and data analysis." Journal of Sensors 2022 (2022): 1-12.

[7] Einfalt, Moritz, Dan Zecha, and Rainer Lienhart. "Activity-conditioned continuous human pose estimation for performance analysis of athletes using the example of swimming." 2018 IEEE winter conference on applications of computer vision (WACV). IEEE, 2018.

[8] Liu, Shuai. "Application Research of Artificial Intelligence in Swimming." Big Data Analytics for Cyber-Physical System in Smart City: BDCPS 2020, 28-29 December 2020, Shanghai, China. Springer Singapore, 2021.

[9] Galluzzi, David. "The Doping Crisis in International Athletic Competition: Lessons from the Chinese Doping Scandal in Women's Swimming." Seton Hall J. Sport L. 10 (2000): 65.

[10] Mooney, Robert, et al. "Inertial sensor technology for elite swimming performance analysis: A systematic review." Sensors 16.1 (2015): 18.

[11] Hamidi Rad, Mahdi, et al. "A novel macro-micro approach for swimming analysis in main swimming techniques using IMU sensors." Frontiers in Bioengineering and Biotechnology 8 (2021): 597738.

[12] Einfalt, Moritz, Dan Zecha, and Rainer Lienhart. "Activity-conditioned continuous human pose estimation for performance analysis of athletes using the example of swimming." 2018 IEEE winter conference on applications of computer vision (WACV). IEEE, 2018.

[13] Tor, Elaine, David L. Pease, and Kevin A. Ball. "Key parameters of the swimming start and their relationship to start performance." Journal of sports sciences 33.13 (2015): 1313-1321.

[14] Stamm, Andy, et al. "Determining maximum push-off velocity in swimming using accelerometers." Procedia Engineering 60 (2013): 201-207.

[15] Siirtola, Pekka, et al. "Efficient accelerometer-based swimming exercise tracking." 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM). IEEE, 2011.

[16] Blanksby, Brian, Lee Nicholson, and Bruce Elliott. "Swimming: Biomechanical analysis of the grab, track and handle swimming starts: an intervention study." Sports Biomechanics 1.1 (2002): 11-24.