Project Kitty

Pose Estimation and Corrective Analytics for Swimming

Naina Bhalla
Roll Number: 240674

May 24, 2025

1 Introduction

Sport: Swimming (Freestyle, Butterfly, Backstroke, Breaststroke)

Swimming is a technically demanding sport where minor flaws in technique significantly impact performance and injury risk. Real-time feedback on body posture, kick consistency, hand entry, and breathing symmetry is vital for improvement.

Project Repository: `https://github.com/naina-bhalla/ProjectKitty.git`

2 System Architecture and Model Design

The system consists of three primary modules:

2.1 Data Scraping and Preprocessing:

Automated scraping and standardization of swimming videos from Youtube.

2.1.1 Video Scraping:

1. Searching YouTube for videos of specific swimming strokes (freestyle, butterfly, backstroke, breaststroke) using targeted keyword queries.
2. Downloading relevant videos in high quality.
3. Organizing the videos into stroke-specific folders (videos/freestyle/, videos/butterfly/, etc.), making the dataset structured and ready for further processing like segmentation and pose analysis.

2.1.2 Video Standardization Pipeline using FFmpeg and Multiprocessing :

This standardizes each swimming video to:

1. A **fixed resolution** of 640×360 pixels
2. A **target frame rate** of 30 frames per second
3. A consistent format (H.264 video codec, AAC audio codec)

Such preprocessing is essential for:

1. Reliable pose estimation and action segmentation
2. Reducing computation by down scaling unnecessarily large videos
3. Preventing model biases due to inconsistent input formats

2.2 Keypoint Extraction and Action Segmentation:

2.2.1 Keypoint Extraction:

Using MediaPipe's pose model, the system processes each video frame and extracts 32 3D keypoints representing major human joints.

- Converts each frame to RGB.
- Extracts keypoints per frame.
- Appends None if no landmarks are found.

2.2.2 Motion detection

Each of the following functions operates on a sliding window of frames (`kps_window`) and detects whether a certain type of motion is occurring.

a. is_overall_posture

- Measures shoulder-to-hip orientation stability.
- Returns true if the angular standard deviation is below a threshold.

b. is_kick_motion

- Analyzes vertical ankle movement variance.
- High variance indicates active kicking.

c. is_breathing_motion

- Checks x-axis distance changes between shoulders.
- Breathing causes lateral torso expansion detectable through shoulder movement.

d. is_hand_entry Verifies if the wrist dips below nose level in multiple frames.

2.2.3 Feature Engineering

The compute_features function extracts the following features from a 30-frame window:

- **Ankle Y Variance**: Consistency of vertical kicking.
- **Velocity Smoothness**: Difference in wrist velocities (indicating symmetry).
- **Shoulder Symmetry**: Lateral movement balance.
- **Elbow Angle Variance**: Arm movement control.
- **Hip Angle Variance**: Postural stability and body rotation.

Each sub-feature is calculated via helper functions:

- compute_velocity
- compute_symmetry
- compute_angle_variance

2.2.4 Classification

The classify_form() function applies rule-based thresholds on the extracted features, returning qualitative labels:

Motion Type	Feature Used	Label Criteria
Kick	Ankle Y Variance	> 0.005 = Inconsistent, else Consistent
Breathing	Shoulder Symmetry	> 0.05 = Biased, else Balanced
Hand Entry	Velocity & Symmetry	Smooth, Uneven, or Asymmetric
Overall Posture	Elbow & Hip Angle Variance	Stable if both variances < 0.1

The rules are inspired by biomechanical literature:

- [1]Biomechanics of Swimming: Symmetry and Coordination (2016)

2.2.5 Segmentation and Output

The segment_and_classify() function performs the full pipeline:

- Slides a window over video frames (30-frame window, 10-frame stride).
- Detects motion type.
- Computes features.
- Classifies form.
- Saves results to a structured JSON file for each detected window.

The final portion of the script (gather_video_paths) recursively scans the standardized_videos folder and extracts paths for batch processing, categorized by stroke style (e.g., freestyle, butterfly).

2.3 Motion Classification Training and Evaluation

To enable automatic classification of specific swimming actions (motions) such as *kick*, *breathing*, *hand entry*, and *overall posture*, developed a supervised machine learning pipeline that trains and evaluates multiple classifiers using keypoint-derived features.

2.3.1 Data Loading and Preprocessing

Each labeled action clip was preprocessed to extract pose features and stored as a JSON file with the format:

- features: a dictionary of numerical pose metrics (e.g., angles, distances between joints).
- label: ground truth class (e.g., "good", "bad").
- motion: one of the predefined motion categories.

The load_dataset function iterates over all stroke styles (e.g., freestyle, backstroke) within the dataset directory, loading only those samples that:

1. Belong to the current motion under analysis.
2. Contain complete data (i.e., both features and label are present).

The resulting feature vectors (`X`) and labels (`y`) are aggregated for each motion type.

2.3.2 Model Training and Evaluation

For each motion category, we split the dataset into training and test sets (80:20) using stratified random sampling. Three machine learning models were trained and evaluated:

- **Support Vector Machine (SVM)**: Effective in high-dimensional spaces, especially when paired with standardized features.
- **Random Forest Classifier**: A robust model suitable for noisy or imbalanced datasets.
- **Multi-Layer Perceptron (MLP)**: A simple feedforward neural network trained for 1000 iterations.

Feature standardization was applied to SVM and MLP models using StandardScaler in a Pipeline.

The training and evaluation were repeated for each of the four motions. The following steps were performed:

1. Model training using the training set.
2. Prediction and evaluation using the test set.
3. Output of precision, recall, F1-score, and accuracy metrics.
4. Saving the trained model (.pkl format) for future deployment.

2.3.3 Example Output

For each motion, the script prints:

- Motion type (e.g., ==== KICK ====)
- Label distribution to check for class imbalance.
- Classification report per model (SVM, Random Forest, MLP).

All trained models are stored in the `models/` directory with filenames formatted as motion_ModelName.pkl— (e.g., kick_SVM.pkl—).

2.3.4 Classification Results by Motion Type

Kick

Label Distribution:
consistent_kick: 950
inconsistent_kick: 1414

Model	Accuracy	Observations
SVM	73%	Better recall for inconsistent kicks.
Random Forest	100%	Perfect accuracy; potential overfitting.
MLP	99%	Excellent generalization, nearly perfect F1-scores.

Breathing

Label Distribution:
balanced_breathing: 445
breathing_bias_detected: 245

Model	Accuracy	Observations
SVM	99%	High recall and precision across both classes.
Random Forest	100%	Perfect performance.
MLP	99%	Comparable to Random Forest, slight recall drop on minority class.

Hand Entry

Label Distribution:
smooth_hand_entry: 1281
unstable_hand_entry: 461
asymmetric_hand_entry: 415

Model	Accuracy	Observations
SVM	98%	Strong results across all classes.
Random Forest	100%	Excellent class separation.
MLP	100%	High precision and recall for each class.

Overall Posture

Label Distribution:
stable_posture: 460
unstable_posture: 281

Model	Accuracy	Observations
SVM	99%	Excellent generalization.
Random Forest	100%	Perfect classification.
MLP	99%	Robust across both posture labels.

2.3.5 Summary

- **Random Forest** consistently achieved 100% accuracy across all motion types, suggesting it fits the data extremely well. It suggests overfitting.
- **MLP** provided strong generalization performance (98–99% accuracy), with consistently high precision and recall, making it a reliable option.
- **SVM**, while effective, lagged slightly behind the other models, especially for more imbalanced datasets (e.g., `kick`).

3 Deployment and Inference Pipeline

The trained models are deployed in a pipeline that processes new swimming videos and overlays real-time feedback. The steps are:

1. **Model Loading:** All trained models for each motion type (`kick`, `breathing`, `hand entry`, `overall posture`) are loaded from the chosen_models/ directory.
2. **Video Processing:** Each frame of the input video is passed through the MediaPipe pose estimator to extract keypoints.
3. **Sliding Window Analysis:** For every window of frames, features are computed and all models are run in parallel to predict the motion quality.
4. **Feedback Overlay:** The predicted label and actionable feedback for each motion are overlaid on the video frame using OpenCV, with enhanced visibility via outlined text and background rectangles.
5. **Output Generation:** The annotated video is saved for review and further analysis.

4 Environment Setup

Recommended Environment:

- `Python 3.8+`
- `mediapipe 0.10.11`

- `opencv-python`
- `numpy`
- `scikit-learn`
- `joblib`
- `yt-dlp`
- `ffmpeg` (system package)

Installation Steps:

```
# Clone the repository
git clone https://github.com/yourusername/project-kitty.git
cd project-kitty

# (Optional) Create a virtual environment
python3 -m venv venv
source venv/bin/activate

# Install Python dependencies
pip install -r requirements.txt

# Install FFmpeg (Ubuntu/Debian)
sudo apt-get install ffmpeg

# Or on Mac (Homebrew)
brew install ffmpeg
```

5 How to Run the Pipeline

1. **Download Videos:**

   ```
   python src/video_scraper.py
   ```

2. **Standardize Videos:**

   ```
   python src/video_standardizer.py
   ```

3. **Extract Keypoints and Segment:**

   ```
   # Open and run in Jupyter/ VSCode
   src/keypoint_extractor.ipynb
   ```

4. **Train Models:**

```
# Open and run in Jupyter/VSCode
src/model_training.ipynb
```

5. **Run Inference and Overlay Feedback:**

```
python src/analyzer.py --input test.mp4 --model_dir
  chosen_models/ --output output.avi
OR
python src/analyzer.py --input test.mp4 --output output.avi
```

References

- [1]Biomechanics of Swimming: Symmetry and Coordination (2016)