Comprehensive Code Report: Fish Metrics Estimation Pipeline

`FishEstimatorWithWeight_Modified.py`

Prepared by: Yash

June 30, 2025

1 Introduction

This report documents the architecture, usage, features, and limitations of the script `FishEstimatorWithWeight_Modified.py`, which serves as the primary driver in a fish detection and measurement pipeline. The system reads input video files, detects fish instances using a YOLO model, estimates their physical dimensions using ArUco markers, and predicts weights using a pre-trained polynomial regression model. The output includes an annotated video along with overlayed statistics and summaries.

2 Execution Command

To execute the script, the following command is used:

```
python .\FishEstimatorWithWeight_Modified.py
    --model .\best.pt
    --output ..\Final_outputs\C10.mp4
    --video ..\Training_data-videos\C10.mp4
```

3 Command Line Arguments

Below is a list of configurable parameters supported by the script:

video_path (str)	Path to the input video file. **(Required)**
output_path (str)	Path to save the output annotated video. *(Optional)*
max_width (int)	Maximum width for output video resolution. Default: `1280`
max_height (int)	Maximum height for output video resolution. Default: `720`
output_fps (int)	Frame rate for output video. Default: `15`
frame_skip (int)	Process every Nth frame to save time. Default: `2`

jpeg_quality (int)	Quality of JPEG compression (1–100). Default: `70`
crf_quality (int)	CRF value for H.264 compression (18–30). Default: `28`
skip_no_marker_frames (bool)	Skip frames without detectable ArUco markers. Default: `True`
show_video (bool)	Display the video during processing. Default: `True`
batch_process (bool)	Use memory-efficient batch processing. Default: `True`

4 Core Features

1. **Video Compression Pipeline**
   - Resolution downscaling
   - Frame skipping
   - JPEG compression
   - Final output encoded using H.264 with CRF control
2. **Fish Detection**
   - Uses YOLOv12s (`best.pt`) trained on annotated datasets
   - Detects fish instances per frame
3. **Length and Width Estimation**
   - Uses ArUco markers as real-world scale references
   - Converts pixel dimensions to centimeters
4. **Weight Estimation via Regression**
   - Polynomial regression model trained on `LengthWeightData.xlsx`
   - Equation:

$$\text{Weight} = 0.0129 + 1.0643 \cdot \text{Length} + 0.00008 \cdot \text{Length}^2$$

   - Performance: Mean Squared Error = 0.00085, $R^2 = 0.99925$
5. **Output Video**
   - Annotated with bounding boxes, length, width, estimated weight
   - Summary slide at the end showing statistics

5 Limitations

- Input video quality may suffer from low resolution, glare, or shakiness.
- No frame-to-frame object identity tracking (fish IDs not preserved).
- RoboFlow auto-annotations used in training may be inconsistent.

6 Suggested Improvements

1. **Object Tracking**
   - Integrate tracking algorithms like **SORT**, **Deep SORT**, or **ByteTrack** to persist fish IDs across frames.
2. **Fish Count Logic**
   - Define a confidence threshold (e.g., 0.6)
   - Use statistical mode of fish counts from valid frames
   - Example: 60% of frames show $\geq$3 fish $\rightarrow$ count = 3
3. **Behavior Analysis**
   - With tracking enabled, extract trajectories and perform behavioral analytics (speed, clustering, etc.)
4. **Freshness Estimation (Future Scope)**
   - Utilize cues like eye clarity, skin tone, and posture to assess freshness

7 Directory Structure

```
project_root/
 Build/
    FishEstimatorWithWeight_Modified.py
    best.pt
    best_optimized.onnx
    poly_transformer2.pkl
    weight_model_poly2.pkl
 Detection-Model-Training/
    runs/
    training_data.yolov12/
    fish_detection_train.py
    yolo12s.pt
 Final_outputs/
 Training_data-videos/
 Weight-Estimation-LR_model/
```

8 Component Breakdown

Build/

Main execution directory.

- `FishEstimatorWithWeight_Modified.py` – Pipeline script
- `best.pt` – Trained YOLOv12s model
- `best_optimized.onnx` – Auto-converted ONNX model for inference

- `poly_transformer2.pkl` – Polynomial transformer for weight estimation
- `weight_model_poly2.pkl` – Final trained regression model

Detection-Model-Training/

Contains training scripts and model checkpoints.

- Training assumes a single species (e.g., *White Leg Shrimps*)
- Detection model: `yolo12s.pt`
- Datasets labeled using RoboFlow

Weight-Estimation-LR_model/

Contains regression model artifacts.

- Input: `LengthWeightData.xlsx`
- Outputs:
  - `poly_transformer2.pkl` – Polynomial feature transformer
  - `weight_model_poly2.pkl` – Trained regression model
- Usage:
  1. Transform raw features using polynomial transformer
  2. Pass to regression model for weight estimation

Training_data-videos/

- Raw videos provided by ICAR during implementation
- Excluded from version control (`.gitignore`)

Final_outputs/

- Stores processed and annotated output videos

9 Conclusion

The `FishEstimatorWithWeight_Modified.py` script consolidates multiple stages of fish detection, compression, and estimation into a unified and optimized workflow. While the current implementation produces accurate results, enhancements such as object tracking and behavior analytics will significantly boost the system's real-time usability and scientific insight generation.

Note: For a complete understanding, please refer to the supplementary files: `Project_Report.pdf`

For walkthroughs, collaboration, or further review, feel free to reach out to the author.