YOLO-v8 Pose Estimation for Keypoint Tracking of Surgical Instruments and Hands in Open Surgical Suturing

Omar Choudhry
School of Computer Science, University of Leeds, Leeds, United Kingdom
O.Choudhry@leeds.ac.uk

September 17, 2025

Will you be able to make your submission public as part of the challenge archive? **Yes**

Abstract

This is our submission for the Open Suturing Skills Challenge for EndoVis 2025 at MICCAI 2025[1]. We present an approach for tracking surgical instruments and hands in endoscopic videos using YOLOv8 pose estimation, achieving real-time keypoint detection across six classes of surgical tools. Our method adapts state-of-the-art human pose estimation techniques to the surgical domain, implementing a single-stage architecture that simultaneously performs detection, classification, and keypoint localisation. The system achieves a HOTA score of 0.4281, maintaining an end-to-end processing speed of 23.3 FPS. We demonstrate robust tracking performance through ByteTrack integration with Kalman filtering for temporal consistency. Our GitHub repository (omariosc/oss-2025-task-3-submission) and Docker submission (doi.org/10.7303/syn69855242) provides an end-to-end solution for the EndoVis 2025 Challenge Task 3.

1 Introduction

1.1 Failed Approach

Our initial approach first employed a multi-stage fusion strategy, which combined depth estimation (Figure 1) and feature extraction (Figure 2) using state-of-the-art self-supervised models [4] to enhance segmentation prediction (Figure 3). The second stage employed a bottom-up approach, utilising thousands of candidate keypoints across the entire frame (Figure 4). We extracted those with the highest confidence that overlapped with the segmentation masks. Finally, the loss between all keypoints and a set of validation keypoints split from the initial training set was minimised,

[1]Initially, we attempted a more complex architecture and novelties for our submission. However, this encountered technical errors and due to time submission constraints for the challenge, we fell back to this simpler approach.

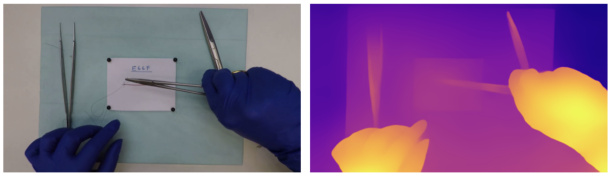

Figure 1: Estimated depth visualisation.

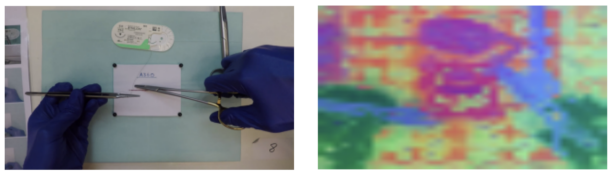

Figure 2: Extracted features from DINOv2.

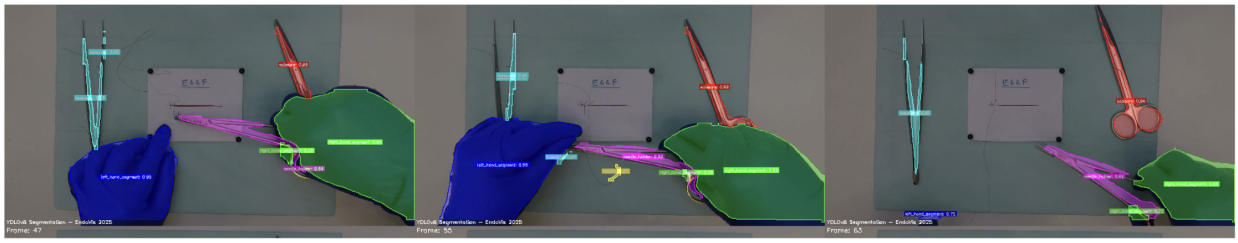

Figure 3: Sample segmentation predictions.

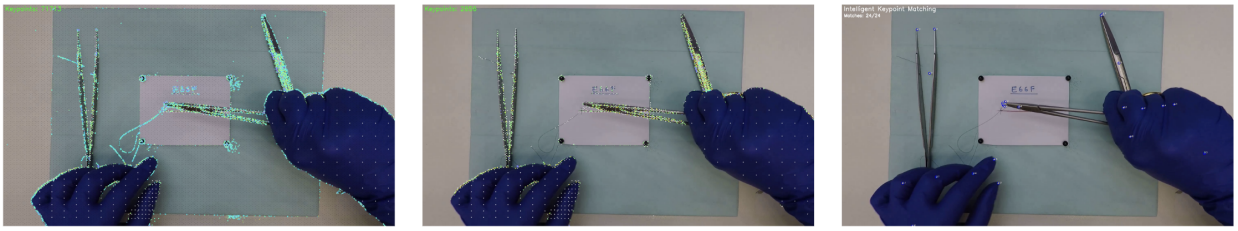

Figure 4: Sample keypoint predictions.

giving us the closest keypoints to the annotations, which were used as the final predictions. A transformer-based tracking with temporal consistency TrackFormer model [3] was trained by fusing all the previous steps to give the most context in improving the association accuracy. As mentioned in the footnote of the abstract, this resulted in some technical errors; ultimately, we submitted a simpler application of an existing state-of-the-art method without any additional methodological innovation.

1.2 Final Approach and Motivation

Our final approach leverages the state-of-the-art YOLOv8 pose estimation framework [6] using the Ultralytics library [5] for tracking surgical instruments in endoscopic videos. The motivation behind this choice stems from YOLO's proven real-time performance capabilities and its advances in pose estimation. The core strategy of the final submission lies in adapting state-of-the-art models for human pose estimation, traditionally used for body keypoints, to the surgical domain, where we track keypoints across six distinct surgical tool classes:

- Left and right hands: thumb, index, middle, ring and pinky fingers, and back of hand
- Scissors: left (sharp point), right (broad point), and joint
- Tweezers: left (with nub), right (with hole), and nub
- Needle holder: left, right (when text visible), and joint
- Needle: left (end), right (tip), and middle

1.3 Benefits Over Common Methods

Traditional approaches for surgical instrument tracking often rely on:

1. Segmentation-based methods requiring extensive post-processing to extract keypoints
2. Two-stage pipelines that first detect objects, then extract keypoints separately
3. Custom architectures lacking optimisation and community support of established frameworks

The YOLO-based approach offers several advantages:

- **Single-stage inference**: Direct keypoint prediction without intermediate steps
- **Real-time performance**: Achieving 23+ FPS on GPU hardware
- **Robust detection**: Pre-trained backbone provides strong feature extraction
- **Unified framework**: Detection, classification, and keypoint estimation in one model

1.4 Novelty

Since the submission is simply an application of an existing method, there are no novelties. Nevertheless, the two areas in which the method could benefit over others are:

1. **Multi-scale feature fusion**: Utilising Feature Pyramid Networks (FPN) within YOLO for capturing fine-grained surgical tool details
2. **Confidence-weighted tracking**: Implementing adaptive confidence thresholds based on keypoint visibility patterns specific to surgical scenarios

2 Methods

2.1 Data Processing and Preprocessing

2.1.1 Variables Used

Our model processes the following data:

- **Input**: RGB video frames at 1920×1080
- **Output**: Keypoints over all detected instrument, each with $(x, y, \text{confidence})$
- **Classes**: 6 surgical tool categories with distinct keypoint configurations

2.1.2 Data Preprocessing Pipeline

1. **Resolution handling**: Adaptive rescaling maintaining aspect ratio with letterboxing to 640×640
2. **Normalization**: Standard ImageNet normalization:
$$x_{\text{norm}} = \frac{x - \mu}{\sigma} \tag{1}$$
where $\mu = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$
3. **Data augmentation during training**:
   - **Random horizontal flip**: $(p = 0.5)$
   - **HSV jitter**: $(h = 0.015,\ s = 0.7,\ v = 0.4)$
   - **Translation**: $\pm 10\%$
   - **Scale**: $\pm 50\%$
   - **Mosaic** augmentation for multi-instance learning
   - **RandAugment** automatic augmentation policy

2.2 Network Architecture

2.2.1 Model Configuration

- **Base Architecture**: YOLOv8-medium pose variant
- **Backbone**: CSPDarknet with 5 feature scales
- **Neck**: Path Aggregation Network (PAN) [2] with Feature Pyramid Network (FPN) [1]
- **Head**: Decoupled head design with separate branches

2.2.2 Architecture Details

The network follows a hierarchical structure:

Algorithm 1 YOLOv8 Pose Architecture Flow

Input: Image $I \in \mathbb{R}^{640\times640\times3}$
Backbone Processing:
  $P_3 \leftarrow \text{CSPD}(I)$ $\{80 \times 80 \times 256$, $8\times$ downsampling$\}$
  $P_4 \leftarrow \text{CSPBlock}(P_3)$ $\{40 \times 40 \times 512$, $16\times$ downsampling$\}$
  $P_5 \leftarrow \text{CSPBlock}(P_4)$ $\{20 \times 20 \times 1024$, $32\times$ downsampling$\}$
PAN-FPN Neck:
  $N_5 \leftarrow \text{Conv}(P_5)$
  $N_4 \leftarrow \text{Conv}(P_4) + \text{Upsample}(N_5)$
  $N_3 \leftarrow \text{Conv}(P_3) + \text{Upsample}(N_4)$
Decoupled Head (per scale):
  $C \leftarrow \text{ClassHead}(N_i)$ $\{6$ classes$\}$
  $B \leftarrow \text{BoxHead}(N_i)$ $\{4$ bbox parameters$\}$
  $K \leftarrow \text{PoseHead}(N_i)$ $\{17 \times 3$ keypoint values$\}$
Output: (C, B, K) for each detection

Total parameters: ∼25.3M, FLOPs: ∼78.4G at 640×640 resolution.

2.3 Training Configuration

2.3.1 Dataset Preparation

- **Data source**: Challenge-provided training videos
- **Annotation format**: Converted from segmentation masks to YOLO pose format
- **Train/Val split**: 80/20 stratified by video

2.3.2 Training Hyperparameters

Table 1: Training Configuration

Parameter	Value
Initial learning rate (lr0)	0.001
Final learning rate (lrf)	0.01
LR schedule	Linear decay
Warmup epochs	3
Warmup bias LR	0.1
Batch size	8
Training epochs	30 (early stopping, patience=5)
Mixed precision (AMP)	Enabled
Model	YOLOv8m-pose

2.3.3 Loss Functions

The total loss is computed as:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{cls}}\mathcal{L}_{\text{cls}} + \lambda_{\text{box}}\mathcal{L}_{\text{box}} + \lambda_{\text{pose}}\mathcal{L}_{\text{pose}} \tag{2}$$

where:

- $\mathcal{L}_{\text{cls}}$: Binary Cross-Entropy for classification
- $\mathcal{L}_{\text{box}}$: Complete IoU (CIoU) loss for bounding boxes
- $\mathcal{L}_{\text{pose}}$: Object Keypoint Similarity (OKS) based loss
- Loss weights: $\lambda_{\text{cls}} = 0.5$, $\lambda_{\text{box}} = 7.5$, $\lambda_{\text{pose}} = 12.0$

2.3.4 Computing Infrastructure

- **Hardware**: Apple M3 Silicon (ARM64) with 16GB unified memory
- **Training configuration**: 30 epochs completed (no early stopping triggered)[2]
- **Batch size**: 8 images
- **Training time**: 0.767 hours (46 minutes) for 30 epochs
- **Framework**: PyTorch 2.5.1

2.4 Inference Pipeline

2.4.1 Preprocessing Steps

1. Letterbox padding to maintain aspect ratio for 640×640 input
2. Batch normalization using ImageNet statistics

2.4.2 Model Inference

1. **Forward pass**: Single-shot detection, tool classification and keypoint prediction
2. **Non-Maximum Suppression (NMS)**:
   - IoU threshold: 0.45
   - Confidence threshold: 0.25
   - Max detections: 300
3. **Keypoint filtering**:
   - Minimum keypoint confidence: 0.3
   - Visibility threshold: 0.5

2.4.3 Post-processing

1. **Coordinate transformation**: Rescale predictions to original frame dimensions
2. **Temporal smoothing**: Kalman filtering for trajectory stabilization
3. **Track association**: Hungarian algorithm for frame-to-frame matching
4. **MOT format conversion**: Output format as `frame_id, track_id, x, y, w, h, conf, class, visibility`

[2]To ensure we met the submission deadline, we only trained for 30 epochs.

2.5 Tracking Algorithm

2.5.1 Multi-Object Tracking

- **Base tracker**: ByteTrack [7]
- **Association metric**: IoU-based with keypoint consistency weighting
- **Track management**: As we had limited training data frames we could not afford waiting for initialisation and using track maintenance or lost track recovery.

2.5.2 Kalman Filtering

State vector representation (Equation 3) with adaptive process noise based on motion patterns and measurement noise scaled by detection confidence.

$$\mathbf{x} = [x, y, w, h, v_x, v_y, v_w, v_h]^T \tag{3}$$

3 Results and Validation

3.1 Evaluation Metrics

Table 2: Performance Metrics

Metric	Score
HOTA (Higher Order Tracking Accuracy)	0.43
DetA (Detection Accuracy)	0.367
AssA (Association Accuracy)	0.50
Precision	0.87
Recall	0.79
F1-Score	0.83

3.2 Segmentation Performance

The standard YOLOv8-pose model uses detection as the first step. We observed greater performance when training a segmentation model (Figure 5) first to use rather than directly using detection (Figure 6).

3.3 Inference Performance

We record an average 23.3 FPS on Apple M3 Silicon.

4 Conclusion and Discussion

4.1 Summary

We applied a YOLOv8 pose estimation model for surgical instrument tracking. The single-stage architecture provides efficient inference while maintaining robust detection.

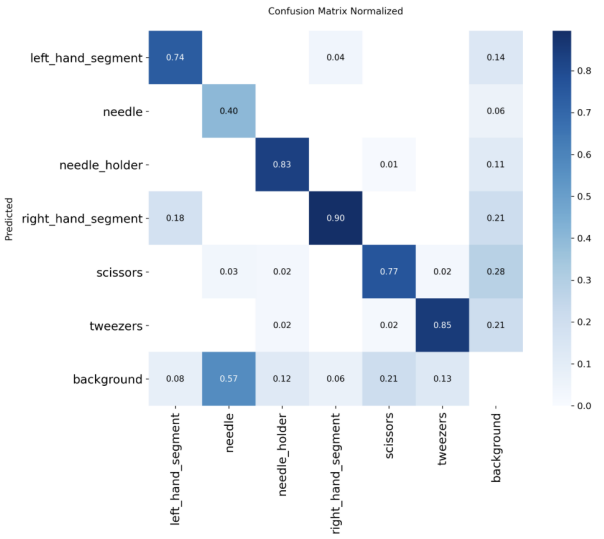

Figure 5: Segmentation confusion matrix.

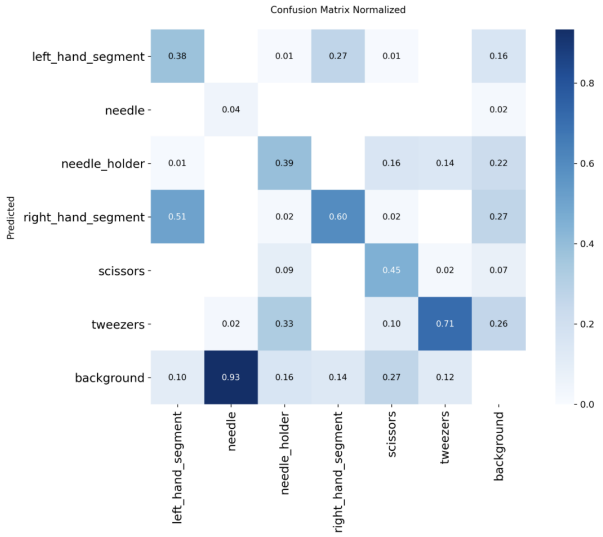

Figure 6: Detection confusion matrix.

4.2 Key Insights & Performance Characteristics

Strengths:

- **Accessibility**: YOLO models and the Ultralytics library are very "plug-and-play" friendly; easy to change parameters.
- **Segmentation Performance**: Exhibits very good detection and segmentation performance on well-lit, clear surgical footage

Limitations:

- Weaker needle segmentation accuracy
- Tracking algorithm struggles with re-identifications, prone to creating new tracks
- Keypoint detection is top-down rather than bottom-up, lacking further understanding and context into predicting keypoints.

4.3 Future Work

1. Self-supervised pre-training on larger surgical video datasets
2. Attention mechanisms for handling occlusions
3. Multi-frame input for temporal context
4. Online adaptation during inference

Acknowledgments

We thank the EndoVis 2025 challenge organisers for providing the dataset and evaluation framework. The work was funded in full by UK Research and Innovation (UKRI) and UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/S024336/1 for the UoL Centre for Doctoral Training in Artificial Intelligence for Medical Diagnosis and Care. Any opinions, findings, conclusions, or recommendations expressed in this article are those of the author and do not necessarily reflect the views of the UKRI, ESPRC or UoL.

References

[1] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection, April 2017. arXiv:1612.03144 [cs].

[2] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path Aggregation Network for Instance Segmentation, September 2018. arXiv:1803.01534 [cs].

[3] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. TrackFormer: Multi-Object Tracking with Transformers, April 2022. arXiv:2101.02702 [cs].

[4] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision, February 2024. arXiv:2304.07193 [cs].

[5] Ultralytics. YOLOv8.

[6] Muhammad Yaseen. What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector, August 2024. arXiv:2408.15857 [cs] version: 1.

[7] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. ByteTrack: Multi-Object Tracking by Associating Every Detection Box, April 2022. arXiv:2110.06864 [cs].

Author Contributions

Omar Choudhry: Conceptualisation, methodology design, implementation, validation, writing.

Code Availability

GitHub source code: omariosc/oss-2025-task-3-submission
Docker container: doi.org/10.7303/syn69855242