Leiden University

MSc in Computer Science

Skeletal Tracking in Speed Skating for Automated Stroke Detection

Name:	Sasank Amavarapu
Student ID:	s4047303
Date:	29/09/2025
1st supervisor:	Dr. Arno Knobbe
2nd supervisor:	Dr. Willem Jan Palenstijn

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Einsteinweg 55
2333 CC Leiden
The Netherlands

Abstract

This thesis investigates the feasibility of extracting stroke cycles and their phase timings from ceiling mounted Apex camera recordings in long-track speed skating. An end-to-end pipeline is developed that combines state-of-the-art pose estimation (Mask R-CNN with a ResNeXt-101 FPN backbone), robust temporal refinement, and homography-based multi-camera fusion to derive cyclic one-dimensional signals for phase detection. Using manually annotated ground truth, the Right Wrist–Right Ankle (RW–RA) distance was identified as the most reliable signal for capturing stroke cycles. Two training regimes were evaluated: Regime A, which validated the pipeline on a single skater and reproduced phase transition timings with errors within one frame for crouched phases and three to four frames for arm phases; and Regime B, which pooled data across multiple skaters and substantially improved generalisation, achieving robust detection across athletes, higher pose accuracy in varied conditions, and more precise phase identification. The results establish a proof-of-concept that demonstrates both the feasibility and scalability of automatic phase extraction, opening avenues for more generalisable models with richer datasets and for practical coaching applications through interpretable, cycle-level analytics.

Acknowledgements

I express my sincere gratitude to my supervisor for his steady support, invaluable feedback, and patience that have been instrumental in completing this thesis. I am also deeply thankful to my family and friends for their unwavering encouragement throughout this journey. Finally, I dedicate this work to my grandparents, whose guidance and inspiration have played a profound role in shaping my life.

Contents

Abstract . . . 1
Acknowledgements . . . 1

1 Introduction **3**
1.1 Context and Motivation . . . 3
1.2 Ceiling mounted Apex cameras . . . 3
1.3 Technical Approach (Bird's-Eye View) . . . 4
1.4 Problem Statement . . . 4
1.4.1 Research Questions . . . 4

2 Related Work **6**

3 Preliminaries **7**
3.1 Pose Estimation Model . . . 7
3.1.1 Detectron2 Framework . . . 7
3.1.2 Focus on Relevant Keypoints . . . 8
3.2 Main-Skater Selection Logic . . . 8
3.3 Temporal Smoothing Algorithms . . . 11
3.3.1 Outlier Detection . . . 11
3.3.2 Gap Filling and Interpolation . . . 12
3.3.3 Advanced Filtering Approaches . . . 12
3.4 Multi-Camera Fusion and Phase Detection . . . 13
3.4.1 Multi-Camera Fusion . . . 13
3.4.2 Phase Detection Signal . . . 15
3.4.3 Peak and Trough-Based Segmentation . . . 15
3.5 Evaluation Metrics . . . 15
3.5.1 Pose Accuracy Metrics . . . 15
3.5.2 Phase Detection Metrics . . . 16

4 Datasets and Evaluation Protocol **18**
4.1 Dataset Overview . . . 18
4.2 Implementation & Experimental Setup . . . 18
4.2.1 Hardware . . . 18
4.2.2 Software Stack . . . 18
4.2.3 Regime A (Skater-1 only) . . . 19
4.2.4 Regime B (Multi-skater pooled) . . . 19

5 Ground-truth Analysis: Choosing the Phase Signal **21**
5.1 Candidate Signals . . . 21
5.2 Baseline Phase Detection with Ground-truth . . . 22
5.2.1 Distinguishing Big and Small Extrema . . . 23
5.2.2 Mapping to Phase Categories . . . 23
5.2.3 Baseline Error from Ground-truth . . . 24

6 Regime A : Pipeline with Skater-1 Model **27**
6.1 Model Predictions . . . 27
6.1.1 Validation and Fold Selection . . . 27

6.1.2 Bounding Box Detection: Pre-trained vs Fine-tuned . . . 27
6.1.3 Best Fold's Evaluation Across Data Splits . . . 28
6.2 Temporal Smoothing . . . 29
6.2.1 Z-score Cleaning . . . 29
6.2.2 Alternative Filters: Kalman and RANSAC . . . 31
6.3 Fusion and Phase Detection . . . 31
6.4 Generalisation to Skater-2 and Skater-3 . . . 34
6.5 Interim Conclusion (Regime A) . . . 35

7 Regime B : Pipeline with Multi-Skater Model 37
7.1 Model Predictions . . . 37
7.1.1 Validation and Fold Selection . . . 37
7.1.2 Bounding Box Detection: Pre-trained vs Fine-tuned . . . 38
7.1.3 Best Fold's Evaluation Across Data Splits . . . 38
7.2 Temporal Smoothing . . . 39
7.2.1 Outlier detection procedure . . . 39
7.3 Fusion & Phase Detection (Skater-1) . . . 43
7.3.1 RW–RA signal (Regime B predictions) . . . 43
7.3.2 Phase-level errors . . . 43
7.4 Interim Conclusion (Regime B) . . . 45

8 Conclusion 46
8.1 Summary of Findings . . . 46
8.2 Key Contributions . . . 46
8.3 Significance and Impact . . . 46
8.4 Limitations and Future Work . . . 47
8.5 Applications . . . 47

Chapter 1

Introduction

1.1 Context and Motivation

Long-track speed skating is contested on a 400 m oval, across distances from 500 m to 10,000 m, with athletes typically specialising by distance and physiology. Small performance differences are influenced by training programmes, venue characteristics, skating techniques, and even altitude, which is why analytics is used routinely to seek marginal gains in preparation and race strategy [10, 14]. Data-driven studies for elite Dutch teams have shown how systematic analysis produces actionable and interpretable insights for training and performance optimisation [10].

Within a race, competitive advantage often comes from the efficiency with which a skater accelerates into and through the corners [9]. Understanding stroke cycles and their phases at the frame level opens the door to linking the on-ice technique to outcomes [24, 16]. The goal in this work is to make these phase-level features available automatically from video, so that they can feed coaching feedback and future analytics [16, 15].

Manual cycle-by-cycle annotation of stroke phases is labour intensive and inconsistent between raters [16]. An automated pipeline that delivers stroke boundaries and phase labels per frame can replace much of this effort, allowing faster feedback loops and a more systematic evaluation of technique [16]. In the broader speed-skating analytics literature, the value lies in interpretable and actionable outputs that coaches can trust, and our goal is to provide such cycle-level features from routine training videos [10].

1.2 Ceiling mounted Apex cameras

This study uses a fixed set of Apex ceiling-mounted cameras that view one of the corners from above [3]. Previous work on this system established the calibration and homography pipeline that aligns each camera with the rink plane. This study adopts the same concept and implementation. The top-down geometry yields overlapping but complementary views that can be stitched into a continuous corner trajectory in a common world frame.

Compared with athlete-worn IMUs or eye-level broadcast angles, overhead video is non-intrusive for skaters and reduces occlusion from rink-side clutter or other athletes [3]. In our data, right-side joints are notably less occluded from the Apex vantage, making them suitable for deriving clean cyclic signals for phase timing. This thesis extends the earlier work on trajectory tracking on Apex camera recordings, which focused on identifying the path and speed of a skater through the corner [3].

This study utilises recordings from Apex ceiling-mounted cameras at the rink corner, with example frames shown in Figure 1.1. These illustrate the variability in exposure conditions across different skaters, which motivates the need for robustness in the proposed pipeline. The recordings form the empirical basis for all subsequent analyses.

Several factors make phase timing from overhead video non-trivial. Keypoints such as wrists and ankles are small relative to the frame. Partial views and multiple skaters are observed, and the exposure varies between recordings (see Figure 1.1, where different skaters are shown left to right under noticeably

Figure 1.1: Illustration of exposure variability across recordings. From left to right: Skater 1, Skater 2, and Skater 3. Differences in lighting conditions affect image appearance and motivate brightness augmentation during training.

different lighting conditions) [19, 21, 10]. The scale also differs by camera, so the pixel displacements are not directly comparable until transformed [3]. Finally, the bird's-eye perspective changes the apparent motion patterns across the view [19, 21, 5]. These challenges motivate robust pose estimation, main-skater selection, and careful normalisation before phase detection.

1.3 Technical Approach (Bird's-Eye View)

The study processes the Apex video frame by frame using a pose estimation framework based on deep learning. Convolutional neural networks (CNN) have become the standard for human keypoint localisation, and in this work the architecture is Mask R-CNN [6], instantiated with a ResNeXt-101 (32×8d) backbone [22] and a Feature Pyramid Network (FPN) [13], implemented through the Detectron2 library [21]. Detections belonging to the intended athlete are selected deterministically using geometric rules. The per-frame keypoints are then *linked across time* to form joint trajectories, which are refined by robust z-score outlier rejection using the median absolute deviation (MAD) and short-gap interpolation [11]. Trajectories from multiple Apex cameras are re-projected to the rink plane via homographies and stitched by a monotonic angular coordinate [3]. A one-dimensional cyclic signal derived from inter-joint distances is lightly smoothed (Savitzky–Golay) and its peaks and troughs map to phase timings for evaluation [17].

1.4 Problem Statement

Within this pipeline, the objective is to obtain the stroke cycles and the timing of their respective phases from the routine training videos recorded by the Apex system. The study focusses on detecting a single athlete per frame, producing stable keypoints, and analysing their trajectories for phase timing.

1.4.1 Research Questions

1. *Keypoint detection.* How well does a pre-trained Mask R-CNN with ResNeXt-101 FPN perform on Apex footage for joint localisation, and how does performance change when fine-tuned on a pooled multi-skater dataset?
2. *Tracking and smoothing.* Does robust z-score cleaning with short-gap interpolation remove spikes due to misclassification of joints, while preserving trajectory shape better than generic filters for these recordings?

3. *Stitching and signal choice.* After homography based stitching, which one-dimensional signal best captures stroke cyclicity?
4. *Phase timing accuracy.* What phase-timing accuracy is achievable relative to manual annotations?

The remainder of this thesis is organised as follows. Chapter 2 reviews related work on pose estimation and its application to sports performance analysis, with a particular focus on speed skating. Within Chapter 3, Sections 3.1–3.5 introduce the methodological building blocks of the pipeline, including the pose estimation framework, main-skater selection, temporal smoothing, multi-camera fusion, and evaluation metrics. Chapter 4 describes the datasets, recording conditions, and experimental setup used throughout the study. Chapter 5 establishes a ground-truth baseline for phase detection, identifying the most reliable one-dimensional signal. Chapter 6 presents Regime A, in which the model is trained and evaluated on one skater and tested for generalisation on others. Chapter 7 extends this design to Regime B, where pooled multi-skater training is used to improve robustness and performance across athletes. Finally, Chapter 8 summarises the findings, highlights key contributions, discusses limitations, and outlines practical applications through a prototype analytics dashboard.

Chapter 2

Related Work

Human pose estimation has developed rapidly since the introduction of DeepPose, which first demonstrated the feasibility of using deep neural networks for direct joint localisation from images [19]. Subsequent architectures such as Mask R-CNN [6], enhanced with ResNeXt backbones [22] and Feature Pyramid Networks (FPN) [13], achieved state-of-the-art performance on large-scale benchmarks like Microsoft COCO [12]. Toolkits such as Detectron2 [21] have made these advances broadly accessible, enabling their transfer to domain-specific applications such as sports.

In sports analytics, pose estimation has been applied to a variety of cyclic domains where efficiency is linked to movement phases. Work in swimming, rowing, and running has shown that temporal joint trajectories and derived signals can provide interpretable features such as cycle duration, symmetry, and intra-cycle coordination, supporting both performance monitoring and coaching feedback [15]. In speed skating, sports analytics has traditionally focused on performance parameters derived from sensors and video, with studies showing how systematic data collection can contribute to marginal gains at the elite level [10, 9]. These results highlight the potential of pose-derived signals to bridge biomechanics and outcome-oriented analysis.

Within skating, most vision-based studies have relied on frontal or side-view recordings. Park et al. (2023) used 2D joint keypoints from broadcast-style angles to measure stroke cycles in speed skating [16], while Zhu et al. (2024) proposed a ground-reaction-force model to study stroke phases [24]. Although these approaches demonstrate the potential of computer vision for phase detection, they remain limited by occlusions and clutter inherent to rink-side perspectives.

Closer to the present work, de Rooij (2024) introduced the ceiling-mounted **Apex** camera system and developed a homography pipeline to track skater trajectories and speeds through a corner [3]. That study established the value of overhead views for non-intrusive, occlusion-resistant tracking, but it did not extend to joint-level phase analysis.

The present thesis builds directly on this line. Instead of focussing on trajectories alone, it integrates modern keypoint detectors with multi-camera fusion and temporal refinement to extract stroke phases automatically.

Chapter 3

Preliminaries

3.1 Pose Estimation Model

Human pose, or skeletal estimation, consists of pinpointing the "keypoints" or the skeletal joints in the human body[19]. These keypoints provide a structured representation of the biophysical movements (see Figure 3.1). In speed skating, tracking the movements of these joints, particularly wrists, elbows, shoulders, hips, knees, and ankles, will allow a more detailed understanding of the skeletal movements of a skater and the study of how specific motions could help improve the overall performance of a skater, such as higher speeds during corners.

3.1.1 Detectron2 Framework

Architecture

The pose estimation model used in this study is based on *Mask R-CNN*, an extension of Faster R-CNN that introduces additional branches, for example, segmentation and human keypoint detection [6]. The implementation is provided by *Detectron2*, an open-source library developed by Facebook AI Research that offers modular and efficient implementations of state-of-the-art object detection and segmentation architectures [21].

The mask R-CNN architecture can be divided into four main components, as illustrated in Figure 3.2. The **backbone** extracts multiscale feature maps from the input frame. The **Region Proposal Network (RPN)** generates candidate regions of interest, where objects are likely to be present. The **RoIAlign** operation ensures precise spatial alignment when combining the features of these regions. Finally, the network contains **three task-specific branches**: for predicting segmentation masks, regressing bounding box coordinates, and classifying object categories. This modular design enables simultaneous detection, segmentation, and keypoint estimation within a unified framework [6].

We adopt the `keypoint_rcnn_X_101_32x8d_FPN_3x` configuration from Detectron2 [21], which instantiates a ResNeXt-101 (32×8d) backbone [22] combined with a **Feature Pyramid Network (FPN)** [13]. FPN builds semantically strong multi-scale features via a top-down pathway with lateral connections, which improves localisation of small structures such as keypoints. ResNeXt increases the representational capacity efficiently through grouped convolutions, outperforming comparable ResNet models at similar complexity. He et al. report consistent COCO gains for Mask R-CNN when using FPN and stronger backbones. Particularly, their ablation results show that ResNeXt-101-FPN surpasses ResNet-101-FPN, motivating our choice for fine-tuning [6, Table 1].

The network was initially pre-trained on the Microsoft COCO dataset [12], which provides annotations for bounding boxes and 17 human keypoints across a large-scale image corpus. These weights were then used for transfer learning to fine-tune the model on the skating dataset, whose aerial, top-down view images differ from the standard frontal or side view images in COCO. This fine-tuning step adapts the model to the specific visual characteristics of speed-skating frames, while leveraging the representational power of the pre-trained backbone. For each input frame, the model outputs bounding boxes and 17 COCO-style keypoints (nose, eyes, ears, shoulders, elbows, wrists, hips, knees, and ankles), each represented by (x, y) pixel coordinates and a confidence score.

Figure 3.1: Example frame from Apex camera 3, annotated with 12 skating-specific keypoints and a bounding box. Visible keypoints are marked as circles, while occluded joints (left hip and left knee) are denoted with crosses. These annotations form the basis for subsequent analyses.

3.1.2 Focus on Relevant Keypoints

In the context of speed skating, the skeletal joints most relevant for capturing bio-physical movements are the wrists, elbows, shoulders, hips, knees, and ankles. Accordingly, the annotation and training procedures were restricted to this subset of 12 keypoints, adapted from the full set of 17 COCO keypoints. This targeted selection ensures that the framework is fine-tuned to major joints directly influencing skating performance. Furthermore, as discussed in the phase detection component of this study, particular emphasis is placed on the right ankle and right wrist, which serve as critical indicators to identify cyclic phases of motion.

Downstream Filtering Ideally, the experimental setup would have ensured that each video sequence contained only a single skater performing the corner. However, since the recordings were collected in a proof-of-concept setting during regular training sessions, multiple athletes often appeared in the background. As the fine-tuned model was trained to recognise speed skaters, it produced detections not only for the target skater but also for other individuals present in the frame. To address this, a downstream filtering step was introduced to isolate the main skater of interest, ensuring that subsequent analysis was based exclusively on a consistent trajectory of the intended athlete rather than being confounded by spurious detections of other skaters.

In summary, Detectron2 with a Mask R-CNN backbone was adapted to predict the most relevant 12 joints for speed skating. While the model outputs detections for all visible persons, a downstream filtering step ensures that subsequent analysis is restricted to the intended skater.

3.2 Main-Skater Selection Logic

Since multiple skaters often appear in the same frame (see Figure 3.3), it is necessary to establish a robust logic to consistently identify the athlete of interest. Because the model is trained to predict skaters both when they enter and exit the frame, the training set also includes partial detections. As a result, the detector may assign similar confidence scores to the full skater and to a boundary skater, and in some cases the boundary skater can even receive a marginally higher confidence score. This makes it unreliable

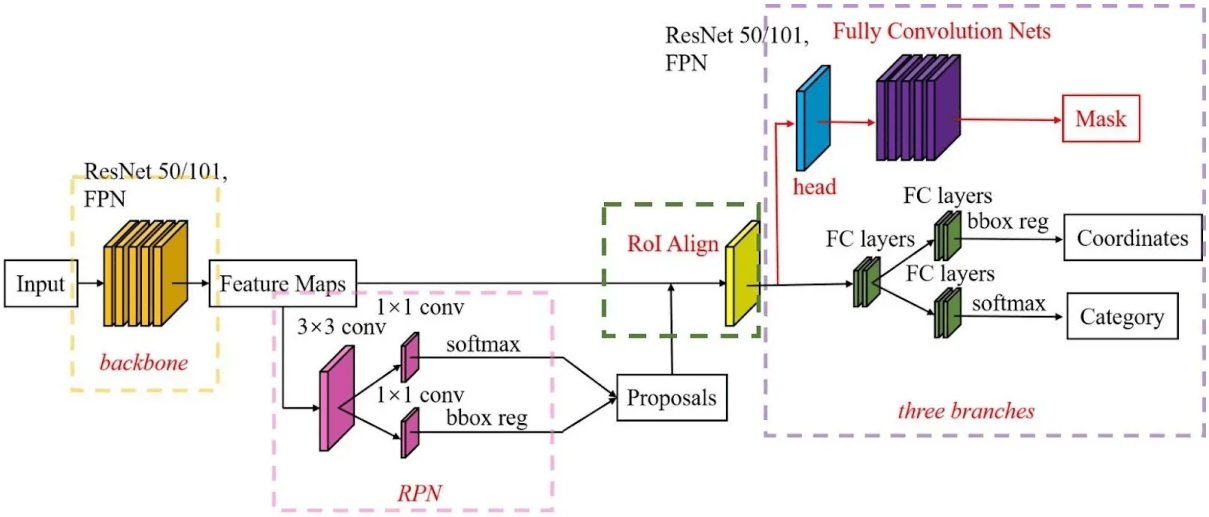

Figure 3.2: Schematic overview of the Mask R-CNN framework, consisting of backbone, RPN, RoIAlign, and three output branches (mask, coordinates, category). Adapted from a blog illustration; the architecture itself is formally described in [6].

to rely solely on confidence-based selection.

In an ideal experimental setup, the target athlete could have been distinguished by an unobtrusive (colour-based) marker on their apparel, making automatic identification straightforward even in the presence of other skaters. In the absence of such a setup, a custom selection logic was developed that is robust enough to consistently isolate the main skater in all six video sequences used in this study.

Skater Selection Criteria

In order to isolate the main skater from multiple detections in a frame, we define the following criteria and formalise them into a selection algorithm.

Definitions:

- **Corner inclusion:** Given two axis-aligned boxes $a = (x_1^a, y_1^a, x_2^a, y_2^a)$ and $b = (x_1^b, y_1^b, x_2^b, y_2^b)$, define

$$\mathrm{CornerIn}(a, b) \;=\; \bigvee_{(x,y)\in\{\text{corners of } a\}} \big(x_1^b < x < x_2^b \;\wedge\; y_1^b < y < y_2^b\big),$$

  which is true if any corner of a lies strictly inside b.

- **Bounding-box centre and image midline:** For a box a, the vertical coordinate of its centre is $c_y(a) = \frac{1}{2}\big(y_1^a + y_2^a\big)$. Let H denote the height of the image, and the *horizontal* midline is at $y = H/2$.

- **Confidence score:** Each detection i has a score $s_i \in [0, 1]$ (Detectron2 keypoint R-CNN), indicating the confidence that the box contains a skater.

- **Sets used by the algorithm:** $\mathcal{U}$ - detections already assigned to a cluster ("visited"); $\mathcal{G}$ - the list of clusters; g - a single cluster (set of indices); C - one retained candidate per cluster; $k_g = \arg\max_{i\in g} s_i$ - the index kept from cluster g.

- **Notation for sets:** $\emptyset$ denotes the empty set; [ ] denotes an empty list (used for storing clusters). Let $\mathbb{I} = \{\, i \mid B_i \text{ is a detected box in the frame} \,\}$ be the index set of all detections. Returning $\emptyset$ in the algorithm means that no main skater is selected for that frame.

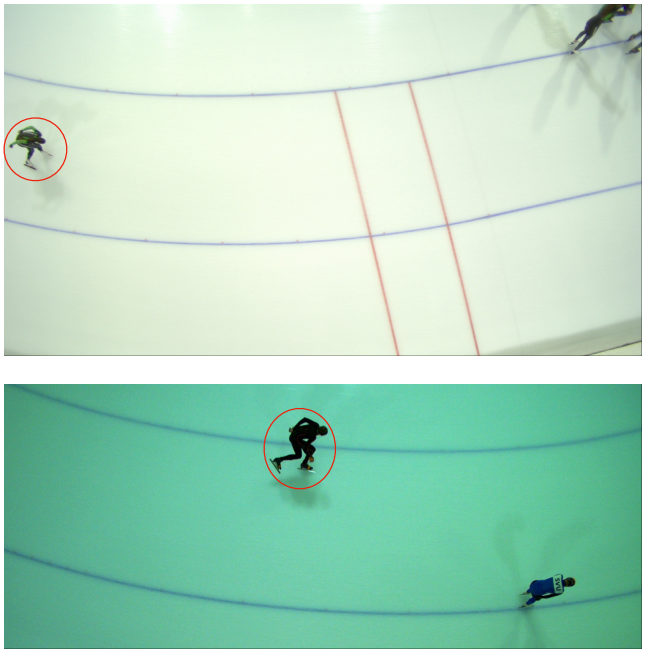

Figure 3.3: Example frames showing the presence of multiple skaters, with the main-skater marked in the red circle. The top image (Skater-1) illustrates a typical corner view where distractors appear near the track boundary, while the bottom image (Skater-2) shows an obstructing skater directly on the track. The Skater-2 image has been slightly enhanced for clarity, as the original capture had lower exposure. These cases motivate the need for a deterministic main-skater selection procedure.

Algorithm 1: Main-skater selection logic. The default procedure applies to all recordings of Skater 1 and Skater 3, and to five of the six cameras of Skater 2. A fixed override is applied for Skater 2 under Apex camera 4, where the reference line is shifted from $H/2$ to $H/4$.

Input: Detections with boxes $\{B_i\}$, scores $\{s_i\}$; image height H
Output: Index $i^\star$ of the main skater, or $\varnothing$
if *no boxes* **then**
  return $\emptyset$; `// no selection`
$\mathbb{I} \leftarrow \{\, i : B_i \text{ exists} \,\}$; `// index set of detections`
$\mathcal{G} \leftarrow [\,]$;
$\mathcal{U} \leftarrow \emptyset$; `// clusters; visited set`
foreach $i \in \mathbb{I} \setminus \mathcal{U}$ **do**
  $g \leftarrow \{i\}$;
  foreach $j \in \mathbb{I} \setminus \{i\}$ **do**
    if CornerIn(B_i, B_j) ***or*** CornerIn(B_j, B_i) **then**
      add j to g;
  append g to $\mathcal{G}$; mark all g in $\mathcal{U}$;
$C \leftarrow \{\, k_g = \arg\max_{i \in g} s_i \mid g \in \mathcal{G} \,\}$;
$c_y^{\text{ref}} \leftarrow H/2$;
return $i^\star = \arg\min_{k \in C} \left| \frac{y_1^{(k)} + y_2^{(k)}}{2} - c_y^{\text{ref}} \right|$;

Explanation of the algorithm:

- If no detections are present in the frame, the algorithm returns $\emptyset$, which means that no main skater is selected.
- Let $\mathbb{I}$ be the index set of all the detections in the frame. Detections are grouped into clusters: Two

boxes are grouped if a corner of one lies inside the other.

- From each cluster $g \subseteq \mathbb{I}$, the highest score detection k_g is retained. The set of retained candidates is C.
- The main skater is chosen from C as the candidate whose bounding-box centre is closest to the *horizontal* midline $y = H/2$ (with a fixed $H/4$ reference for Skater-2, Apex camera 4).

By applying these rules, a single trajectory is consistently extracted for the main skater, providing a stable basis for all downstream evaluation and phase detection. For most recordings, including all six Apex camera recordings of Skater-1 and Skater-3, and five of the six recordings of Skater-2, this procedure is sufficient. However, in the case of Skater-2 under Apex camera 4, the assumption that the main skater lies closest to the image midline does not hold (see Figure 3.3). In this view, the main skater appears in the upper part of the frame, while another skater nearer to the midline would otherwise be selected. To address this, a fixed override is applied where the reference line is shifted from $H/2$ to $H/4$, ensuring that the correct skater is consistently selected. This adjustment is treated as part of the deterministic preprocessing logic rather than an experimental choice.

3.3 Temporal Smoothing Algorithms

Tracking joint trajectories across video frames inevitably introduces noise and occasional discontinuities due to imperfect detections, such as misclassification between left and right sides of the same joint. To ensure that the downstream analysis is based on stable and physiologically possible signals, several smoothing and interpolation techniques are commonly used. This section outlines the concepts of outlier detection, short-gap interpolation, and advanced filtering methods.

3.3.1 Outlier Detection

Deviation-Based Identification

To detect impossible jumps in joint trajectories, we examine the frame-to-frame displacement of each joint j,

$$\Delta x_t^{(j)} = x_t^{(j)} - x_{t-1}^{(j)}, \qquad \Delta y_t^{(j)} = y_t^{(j)} - y_{t-1}^{(j)}.$$

Since each camera view has its own pixel scale, raw displacements cannot be directly compared between cameras. To enable a consistent outlier treatment while still operating in image coordinates, these displacements are normalised by the diagonal of the main skater's bounding box in frame t,

$$d_t = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}, \qquad \Delta x_t^{*(j)} = \frac{\Delta x_t^{(j)}}{d_t}, \quad \Delta y_t^{*(j)} = \frac{\Delta y_t^{(j)}}{d_t}.$$

This produces a scale-invariant measure of joint motion that can be applied uniformly across trajectories from all cameras. The transformation into world coordinates is performed only after the smoothing stage.

To robustly identify outliers, we compare these normalised displacements against the global median and *median absolute deviation* (MAD) across the sequence [11]. The MAD for a variable $z = \{z_1, \ldots, z_T\}$ is defined as

$$\mathrm{MAD}(z) = \mathrm{median}\big(\, |z_i - \mathrm{median}(z)| : i = 1, \ldots, T \,\big).$$

A displacement is flagged if

$$\left| \frac{\Delta x_t^{*(j)} - \mathrm{median}(\Delta x^{*(j)})}{\mathrm{MAD}(\Delta x^{*(j)})} \right| > k \quad \text{or} \quad \left| \frac{\Delta y_t^{*(j)} - \mathrm{median}(\Delta y^{*(j)})}{\mathrm{MAD}(\Delta y^{*(j)})} \right| > k,$$

with k a chosen threshold (e.g. $k = 11$). Using this method, joint predictions that deviate strongly from the expected temporal continuity are treated as outliers and subsequently corrected.

Iterative Outlier Cleaning

Based on the deviation measure defined above, the cleaning procedure was applied iteratively with additional safeguards to ensure robustness. The key steps are the following.

Iterative flagging: Compute robust z-scores of the deviation measure and flag all frames with $|z| > k$ (here $k = 11$). Blank flagged frames and repeat the process until no new outliers are detected or a maximum number of passes is reached.

Run promotion: If two or more consecutive frames are flagged, the entire contiguous run is blanked. This avoids partially corrupted segments remaining in the trajectory.

Guarded horizontal steps: For non-adjacent gaps, horizontal displacements are replaced by the median of adjacent steps. This prevents spurious large Δx values across long discontinuities.

Boundary preservation: At the edges of a blanked region, the first valid frame before the error and the first valid frame after the error are always preserved. This ensures that the trajectory connects back smoothly to normal motion. In extreme scenarios (e.g., errors at the very start or end of the sequence), only one boundary exists; in such cases, that boundary frame is retained and the missing side is linearly extrapolated from the available neighbour.

3.3.2 Gap Filling and Interpolation

Linear Interpolation for Short Gaps

Once outliers are removed, short gaps, consecutive missing values of length ≤ 4 frames, appear in the temporal trajectories of the joints. If both the start and end of such a gap are known (say frames t_1 and t_2 with $t_2 > t_1$), the missing values for any frame $t \in (t_1, t_2)$ are filled by linear interpolation:

$$p_t^{(j)} = \frac{t_2 - t}{t_2 - t_1} p_{t_1}^{(j)} + \frac{t - t_1}{t_2 - t_1} p_{t_2}^{(j)},$$

where $p_t^{(j)} = (x_t^{(j)}, y_t^{(j)})$ is the 2D coordinate of joint j at frame t.

At the extremes of the sequence, a short gap may have only one bounding endpoint. In such cases, we apply *linear extrapolation* based on the nearest two valid frames:

- If the missing values occur before the first valid frame t_1, the trajectory is extrapolated backward using the line defined by $p_{t_1}^{(j)}$ and $p_{t_1+1}^{(j)}$.
- If the missing values occur after the last valid frame t_2, the trajectory is extrapolated forward using the line defined by $p_{t_2-1}^{(j)}$ and $p_{t_2}^{(j)}$.

This strategy preserves temporal continuity while avoiding the introduction of artificial oscillations. Because gaps are typically short (≤ 4 frames), linear interpolation provides a simple and effective way to restore joint trajectories without introducing bias.

3.3.3 Advanced Filtering Approaches

Savitzky–Golay Filtering

This technique smooths a time series by fitting a polynomial of degree d within a sliding temporal window of size w [17]. For each frame t, the joint trajectory is replaced by the value predicted by the local least-squares polynomial:

$$\tilde{p}_t^{(j)} = \arg\min_{q \in \mathcal{P}_d} \sum_{s=t-\lfloor w/2 \rfloor}^{t+\lfloor w/2 \rfloor} \left\| p_s^{(j)} - q(s) \right\|^2,$$

where $\mathcal{P}_d$ denotes the set of polynomials of degree d. This filter reduces high-frequency noise while preserving local trends, such as peaks and slopes. This smoothing is essential to identify the cyclic motions of the strokes in skating.

Kalman Filtering

Kalman filtering treats the trajectory of a joint as the hidden state of a linear dynamical system [8]. For each frame t, the state update is

$$x_t = Ax_{t-1} + w_t, \qquad p_t^{(j)} = Hx_t + v_t,$$

where A is the state transition matrix, H the observation matrix, and w_t, v_t represent the process and observation noise (typically Gaussian). The Kalman filter recursively alternates between *prediction* (using the motion model) and *update* (using the observed detection), generating an estimate $\hat{p}_t^{(j)}$ that balances temporal smoothness and measurement fidelity.

RANSAC Polynomial Fitting

RANSAC (Random Sample Consensus) is a robust regression technique that fits a polynomial trajectory to joint positions while discarding outliers [4]. Given the set of detections $\{p_t^{(j)}\}$, the procedure iteratively:

1. selects a random subset of frames,
2. fits a polynomial $q(s)$ of degree d,
3. identifies inliers as points within a tolerance ϵ of $q(s)$,
4. retains the model if it maximises the number of inliers.

The final trajectory is the polynomial fit to all inliers:

$$\tilde{p}_t^{(j)} \approx q^*(t),$$

where q^* is the best-supported polynomial. This method is effective in eliminating sporadic large errors while preserving the overall motion trend.

In practice, the choice of smoothing strategy depends on the nature of the gaps and the level of noise. Linear interpolation and extrapolation are well suited for very short gaps (at most a few frames), where continuity can be restored without introducing noticeable bias. Savitzky–Golay filtering is effective when high-frequency jitter must be suppressed while still preserving the cyclic peaks of motion. Kalman filtering provides a principled approach when balancing noisy detections with expected motion dynamics, particularly in longer sequences. Finally, the RANSAC polynomial fitting is most appropriate when occasional large misdetections occur, as it explicitly rejects outliers during model fitting.

Together, these approaches form a toolbox of techniques to ensure that joint trajectories are stable and biomechanically meaningful.

3.4 Multi-Camera Fusion and Phase Detection

In order to analyse a complete skating pass over the corner from multiple ceiling-mounted cameras, detections must be transformed into a common world coordinate frame, stitch overlapping trajectories into a continuous path, and derive cyclic motion signals from which skating phases can be identified.

3.4.1 Multi-Camera Fusion

Homography and Distortion Correction

The six ceiling-mounted cameras provide overlapping but distinct perspectives of the skating turn. To express all trajectories in a common world coordinate system, each camera view is corrected for lens distortion and aligned to the ice-rink plane using a homography transformation. This transformation relates the corresponding points in the image coordinates and the world coordinates through a perspective mapping:

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix},$$

where H is the homography matrix estimated from calibration markers placed on the ice. The concept and implementation follow the procedure established in earlier work on this camera system [3],

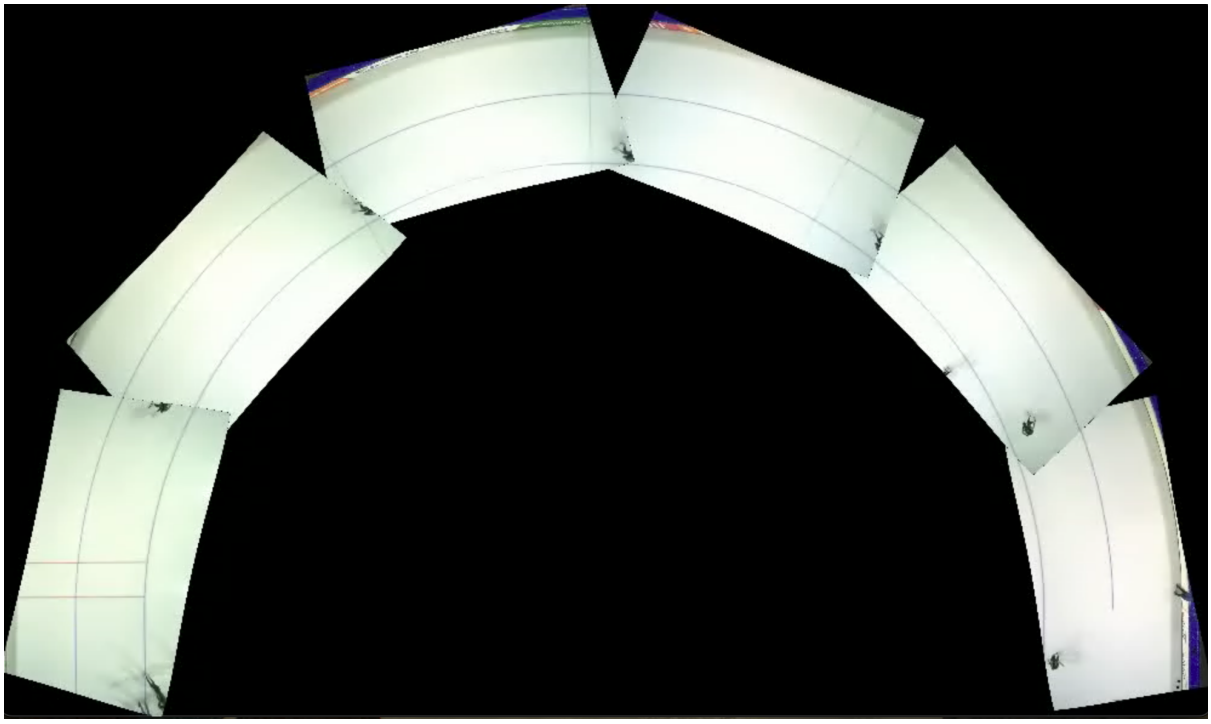

Figure 3.4: Stitching of the six Apex camera views into a continuous top-down representation of the turn. Each camera segment overlaps slightly with its neighbours; overlap removal is later performed based on angular position θ.

where distortion correction and homography were applied to translate image-space detections into world coordinates. Once transformed, trajectories from all cameras can be stitched together into a unified representation.

Trajectory Stitching Across Cameras

After re-projection into world coordinates, each of the six cameras provides a partial view of the skater's trajectory through the turn. To reconstruct a complete sequence, the per-camera trajectories are concatenated in temporal order, creating a stitched path that spans the full corner. This stitching step ensures that the motion is expressed in a single continuous coordinate system rather than in six disjoint image planes (see Figure 3.4).

Transformation to Angular Coordinates

While stitched trajectories are initially described in Cartesian coordinates (x, y), it is more convenient to represent motion along the corner, in terms of an angular variable. The midpoint of the shoulders is used as a proxy for the skater's body position, and an instantaneous centre of rotation is estimated for the corner. The angular position is then defined as

$$\theta_t = \arctan 2\big(y_t - y_c,\ x_t - x_c\big),$$

where (x_c, y_c) denotes the estimated centre of the turn. This transformation provides a monotonic reference along the arc, which is robust to differences in camera perspective.

Overlap Removal Using Angular Position

Consecutive camera views typically contain overlapping temporal segments. To avoid duplication, these overlaps are resolved based on the monotonic progression of θ. For any region where two cameras cover the same angular range, the segment with higher-quality predictions (e.g. fewer missing joints or outliers) is retained, while the redundant frames are discarded. This angle-based trimming ensures that the stitched trajectory progresses smoothly along the turn without discontinuities or duplicated intervals.

3.4.2 Phase Detection Signal

Cyclic Signals from Joint Distances

The motion of a skater's stroke is cyclic in nature, and phase detection can be used to identify repetitive patterns in joint trajectories. One way to extract such patterns is to reduce the two-dimensional joint coordinates into a one-dimensional signal that oscillates with the skater's stroke cycle. Candidate signals include inter-joint distances, vertical displacements, or joint angles, each of which may highlight different biomechanical aspects of the motion. The choice of signal will be examined in the next chapter, where we compare alternatives and select the most robust metric for our dataset.

3.4.3 Peak and Trough-Based Segmentation

Cycle Boundary Identification

Once a cyclic signal has been established, skating phases can be identified by detecting local maxima and minima. Formally, given a signal $s(t)$ derived from joint trajectories, peaks and troughs are detected using prominence-based criteria, as implemented in `scipy.signal.find_peaks` from the SciPy library [20], to ensure that only salient oscillations are retained. The intervals between successive extrema correspond to motion cycles, which can then be subdivided into phase categories such as arm forward, arm backward, crouched after forward pass or crouched after backward pass. This principle provides the foundation for phase detection, with the detailed mapping rules, and evaluation is deferred to the methodology section.

These definitions establish the basis for detecting the cyclic phases of strokes in speed skating at the turn of the ice rink from keypoint trajectories. The specific choice of signal and thresholds, as well as the mapping of extrema to annotated phases, are detailed in the methodology chapter.

3.5 Evaluation Metrics

To quantitatively evaluate both pose estimation and phase detection, we define a set of metrics that capture spatial accuracy, temporal stability, and segmentation consistency. These definitions are given here as references for the subsequent methodology and results.

3.5.1 Pose Accuracy Metrics

Intersection-over-Union (IoU):

To evaluate the accuracy of person detection, we compute the Intersection-over-Union (IoU) between predicted and ground-truth bounding boxes. Let B^{pred} and B^{GT} denote the predicted and ground-truth boxes respectively. The IoU is defined as

$$\text{IoU}(B^{\text{pred}}, B^{\text{GT}}) \;=\; \frac{\text{area}(B^{\text{pred}} \cap B^{\text{GT}})}{\text{area}(B^{\text{pred}} \cup B^{\text{GT}})}.$$

An IoU of 1 indicates a perfect match, while lower values indicate poorer alignment. For evaluation, detections are often thresholded by IoU, e.g. a detection is considered correct if IoU > 0.5. In this study, IoU serves as a detection-level quality measure complementing the joint-based metrics, with both the continuous IoU values and thresholded correctness used where appropriate.

Mean Per Joint Position Error (MPJPE):

For a given joint j at frame t, let $p_t^{(j)} = (x_t^{(j)}, y_t^{(j)})$ denote the predicted position and $\hat{p}_t^{(j)}$ the ground-truth. The per-joint error is defined as the Euclidean distance

$$e_t^{(j)} = \left\| p_t^{(j)} - \hat{p}_t^{(j)} \right\|_2 .$$

The MPJPE is the mean error across all joints and frames:

$$\text{MPJPE} = \frac{1}{TJ} \sum_{t=1}^{T} \sum_{j=1}^{J} e_t^{(j)}.$$

Normalised MPJPE: A scale-invariant variant divides each error by the diagonal of the main skater's bounding box at that frame d_t:

$$\tilde{e}_t^{(j)} = \frac{e_t^{(j)}}{d_t}, \qquad \text{MPJPE}_{\text{norm}} = \frac{1}{TJ} \sum_{t=1}^{T} \sum_{j=1}^{J} \tilde{e}_t^{(j)}.$$

Percentage of Correct Keypoints (PCK):

A prediction for joint j at frame t is considered correct if

$$e_t^{(j)} \leq \tau \cdot d_t,$$

where d_t is the diagonal length of the main skater's bounding box at frame t, and τ is a fixed tolerance (e.g. 0.05 or 0.10). PCK is the fraction of correct joints:

$$\text{PCK} = \frac{\#\{\, e_t^{(j)} \leq \tau d_t \,\}}{TJ}.$$

Normalised PCK: Equivalently, using $\tilde{e}_t^{(j)} = e_t^{(j)}/d_t$, a joint is correct if $\tilde{e}_t^{(j)} \leq \tau$.

Worst-Joint 95th Percentile Error:

For each joint j, compute the 95th percentile of its error distribution $\{e_t^{(j)}\}_t$. The worst-joint error is defined as the maximum over the joints:

$$\text{Worst-Joint P95} = \max_j \ \text{P95}\big(\{e_t^{(j)}\}_t\big).$$

Normalised Worst-Joint P95: Errors are divided by d_t before computing percentiles:

$$\text{Worst-Joint P95}_{\text{norm}} = \max_j \ \text{P95}\big(\{\tilde{e}_t^{(j)}\}_t\big).$$

Temporal Jitter:

To assess temporal stability, we measure the frame-to-frame variation of each joint:

$$\text{Jitter}^{(j)} = \frac{1}{T-1} \sum_{t=2}^{T} \left\| p_t^{(j)} - p_{t-1}^{(j)} \right\|_2.$$

The reported jitter is the average over all joints. Lower jitter indicates smoother trajectories. *Normalised Jitter:* Normalisation by d_t yields

$$\tilde{\delta}_t^{(j)} = \frac{\|p_t^{(j)} - p_{t-1}^{(j)}\|_2}{d_t}, \qquad \text{Jitter}_{\text{norm}} = \frac{1}{J(T-1)} \sum_{j=1}^{J} \sum_{t=2}^{T} \tilde{\delta}_t^{(j)}.$$

For MPJPE, PCK, Worst-Joint P95, and Jitter, we report both absolute and normalised variants. While the absolute metrics quantify raw pixel-level accuracy, the normalised versions (scaled by the bounding-box diagonal) provide scale-invariant evaluation. These normalised metrics are particularly important for cross-dataset comparisons, where frame resolution, camera setup, or skater size may differ.

3.5.2 Phase Detection Metrics

Mean Absolute Error in Angular Domain:

For each annotated phase boundary, let θ^{GT} and θ^{pred} denote the ground-truth and predicted angular positions. The angular error is

$$\text{MAE}_\theta = \frac{1}{N} \sum_{i=1}^{N} \left| \theta_i^{\text{pred}} - \theta_i^{\text{GT}} \right|,$$

averaged over N boundaries.

Mean Absolute Error in Frame Index:

Because each frame t is associated with an angular position θ_t, the angular error can be mapped back to the frame domain. Thus, MAE_θ can equivalently be reported as an average misalignment in frames, obtained by converting the angular deviation into the corresponding number of frames. This does not constitute a separate metric, but rather a re-expression of MAE_θ for interpretability.

In summary, pose accuracy metrics quantify the spatial precision and temporal stability of joint trajectories, while phase detection metrics evaluate the alignment and overlap of predicted phases with the ground truth. Together, these measures provide a comprehensive framework for assessing the quality of the proposed pipeline.

This chapter introduced the foundational components required for analysing speed-skating motion from multi-camera video. The pose estimation framework and the selection logic for isolating the main skater were described, along with temporal smoothing techniques to obtain stable joint trajectories. The transformation and stitching of multi-camera views were detailed, and the evaluation metrics that will be used throughout the study were defined. Together, these preliminaries provide the conceptual foundation for the methodology. The next chapter documents the recordings from the Apex cameras, the evaluation protocol, and implementation details that anchor the experiments.

Chapter 4

Datasets and Evaluation Protocol

This chapter documents the recordings from the Apex cameras, the scope of applicability for each skater, and the protocols followed in training, evaluation, and reproducibility. It outlines the experimental setup for model training and inference, including the comparison of pre-trained and fine-tuned models in two training regimes. Evaluation is carried out up to model predictions for all skaters, and extended to smoothing and phase detection only where the data permits.

4.1 Dataset Overview

Table 4.1: Pipeline applicability for the three skaters, showing available cameras' recordings and which stages of the pipeline (model predictions, smoothing, and phase detection) could be performed. Ticks indicate availability, while crosses indicate stages that could not be applied, with reasons noted.

Skater	Cameras Available	Model Predictions	Smoothing	Phase Detection
Skater-1	All cameras	✓	✓	✓
Skater-2	All cameras (cropped)	✓	✓	✗ (no homography)
Skater-3	cam-2 only	✓	✓	✗ (single-camera only)

This subsection summarises the applicability of the pipeline to the available recordings for each skater. As shown in Table 4.1, Skater-1 is used throughout the entire pipeline, from model predictions through to phase detection, while Skater-2 is restricted to predictions and smoothing due to cropped recordings that invalidate homography. Skater-3, available only from a single camera, serves as a supplementary case and is likewise limited to predictions and smoothing.

4.2 Implementation & Experimental Setup

This subsection outlines the hardware, software stack, and training regimes used in the experiments.

4.2.1 Hardware

The experiments were conducted on NVIDIA GPUs in the Google Colab environment. Model training was performed on an A100 GPU, which provides sufficient memory and compute capability to handle the fine-tuning of high-capacity models such as the ResNeXt-101 backbone used here. All inference experiments, including the evaluation of validation and test splits, were executed on T4-class GPUs, which offer lower memory but are adequate for prediction tasks once models are trained, significantly decreasing inference costs.

4.2.2 Software Stack

The implementation relied on a Python-based software stack. Model training and inference were performed using PyTorch and the Detectron2 framework. The handling of image and video frames was

managed with OpenCV, while NumPy and Pandas were used for numerical computation and dataset organisation. SciPy was employed for signal processing tasks such as Savitzky–Golay smoothing and peak detection, and scikit-learn was used for RANSAC-based post-processing. Matplotlib was used for generating plots and visualisations. To ensure reproducibility, random seeds were fixed across all libraries (PyTorch, NumPy, and scikit-learn), and exact library versions and configuration files were archived with the experimental notebooks. Roboflow was used for annotation management, creating augmentations, and dataset export in COCO format, ensuring consistency between training and evaluation splits.

4.2.3 Regime A (Skater-1 only)

This regime was based on the Skater-1 dataset alone, with the breakdowns summarised in Table 4.2. The setup can be described as follows:

- **Training set (50%):** 297 frames, expanded to 880 samples through augmentation (brightness variation ±20%).
- **Validation set (23%):** 134 frames, used to monitor performance across five folds of cross-validation.
- **Test set (27%):** 162 frames, held out for final evaluation of the chosen model.
- **Cross-validation:** The training set was split into five folds and the model corresponding to the fold with the best performance on the validation set was retained as the final model.
- **Comparison:** The selected fine-tuned model was evaluated on the test set and compared against the pre-trained Detectron2 baseline (without fine-tuning) on the same split.

This design allows the effect of fine-tuning to be quantified under controlled conditions, ensuring that improvements can be directly attributed to transfer learning rather than dataset bias.

This choice of augmentation was motivated by differences in the raw video conditions across skaters: Skater-2's recordings contain noticeably darker frames, while Skater-3's recordings appear lighter. As illustrated in Figure 1.1, such exposure variability is inherent to the dataset. Introducing brightness variation during training therefore helps the model generalise better across different illumination levels, anticipating the variability encountered when extending beyond Skater-1.

Table 4.2: Dataset splits used in the two training regimes. Ratios are computed with respect to the raw (pre-augmentation) dataset size of 593 images.

Split	**#images**	**Ratio (%)**
Train	297 (880 after augmentation)	50.1%
Validation	134	22.6%
Test	162	27.3%

4.2.4 Regime B (Multi-skater pooled)

This regime was based on all the datasets, which pooled annotated frames across Skater-1, Skater-2, and Skater-3 where available. The breakdown of this dataset is summarised in Table 4.3. The setup can be described as follows:

- **Training set (50%):** 614 frames, expanded to 1825 samples through augmentation (brightness variation ±20%).
- **Validation set (25%):** 307 frames, used to monitor performance during training.
- **Test set (25%):** 307 frames, held out for final evaluation.
- **Cross-validation:** The training set was split into five folds, following the same strategy as in Regime A. The fold with the best performance on the validation set was retained as the final model.
- **Comparison:** The fine-tuned model selected under this regime was evaluated on the test set and compared against the pre-trained Detectron2 baseline (without fine-tuning) on the same split.

The use of brightness augmentation was retained even though the dataset already combined all skaters. This decision was motivated by the need to further improve robustness to environmental variations such as illumination changes across recordings, thereby enhancing the generalisation capacity of the fine-tuned model.

Regime A isolates the effect of fine-tuning within a single-skater dataset, allowing improvements over the pre-trained baseline to be quantified under tightly controlled conditions. Regime B extends this design to a pooled, multi-skater setting, thereby testing the generalisation of the model under more diverse recording conditions. Both regimes follow the same five-fold cross-validation strategy, with the best-performing fold retained for evaluation against the pre-trained baseline on a held-out test set. Having established the datasets, the augmentation strategy, and the training protocols, the next section presents the results obtained under Regime A, beginning with model predictions for Skater-1.

Table 4.3: Dataset splits used in Regime B (multi-skater pooled). Ratios are computed with respect to the raw (pre-augmentation) dataset size of 1228 images.

Split	**#images**	**Ratio (%)**
Train	614 (1825 after augmentation)	50.0%
Validation	307	25.0%
Test	307	25.0%

Evaluation metrics are defined in Chapter 3 (Section 3.5) and are applied here to quantify model predictions and, where applicable, phase detection accuracy (as illustrated in Section 5, ahead).

Having defined the datasets, the augmentation strategy, and the training regimes, the next chapter establishes a ground-truth baseline for phase detection using manual annotations.

Chapter 5

Ground-truth Analysis: Choosing the Phase Signal

This chapter analyses ground-truth keypoint annotations to determine a one-dimensional signal that reliably captures the cyclic skating stroke, establishing the canonical phase signal under ideal conditions before introducing model predictions. Ground-truth annotations are manually labelled joint coordinates, and they serve as the reference against which model outputs are evaluated. By basing the first analysis entirely on these annotations, the effect of detection errors is excluded, allowing a clean assessment of which signals best capture stroke cyclicity.

Building on Section 3.4 (world-coordinate reprojection, overlap pruning, and definition of the angular position θ), recordings of Skater-1 across all six cameras are used to evaluate candidate distance metrics to derive a compact one-dimensional signal suitable for phase detection based on peaks and troughs. The evaluation is performed exclusively on ground-truth annotations to provide a baseline for how well each candidate captures the cyclic structure of the stroke.

The centre of the fitted circle, estimated from the midpoint of the skater's shoulders, was located at $(-0.38, -1.66)\,\mathrm{m}$, and the resulting flattened angular range covered $\theta_{\mathrm{flat}} \in [0.000, 3.279]\,\mathrm{rad}$ for the analysed turn.

5.1 Candidate Signals

To determine which signal best captures the cyclic nature of the stroke, several one-dimensional candidates were derived from the ground-truth keypoints. These included inter-joint distances such as Left Wrist–Right Wrist (LW–RW) and Right Wrist–Right Ankle (RW–RA), as well as distances from joints to midpoints, namely RW–MidW (Right Wrist to Hip Midpoint), RA–MidW (Right Ankle to Hip Midpoint), RW–BM (Right Wrist to Bounding-box Midpoint), and RA–BM (Right Ankle to Bounding-box Midpoint). These options follow the phase-signal notion introduced in Section 3.4.2.

The focus was placed specifically on the right side joints, as they are less prone to occlusion from the perspective of the Apex cameras (see Figure 3.1). Among the available right side joints, shoulder, elbow, wrist, hip, knee, and ankle, the shoulder and hip exhibit little differential motion, while elbows and knees, although mobile, tend to move close to the torso, making their trajectories less distinctive and more frequently occluded. Comparitively, the wrists and ankles demonstrate a greater degree of free-range motion away from the body, producing clearer oscillatory patterns that are more suitable for capturing the cyclic nature of skating strokes.

Figure 5.1 illustrates these candidate signals across all six cameras for Skater-1. The horizontal axis represents the flattened angular position θ_{flat} (in radians) along the fitted skating turn, while the vertical axis represents the raw Euclidean distance in world units for each metric. Each colour corresponds to a different camera segment after pruning overlapping frames.

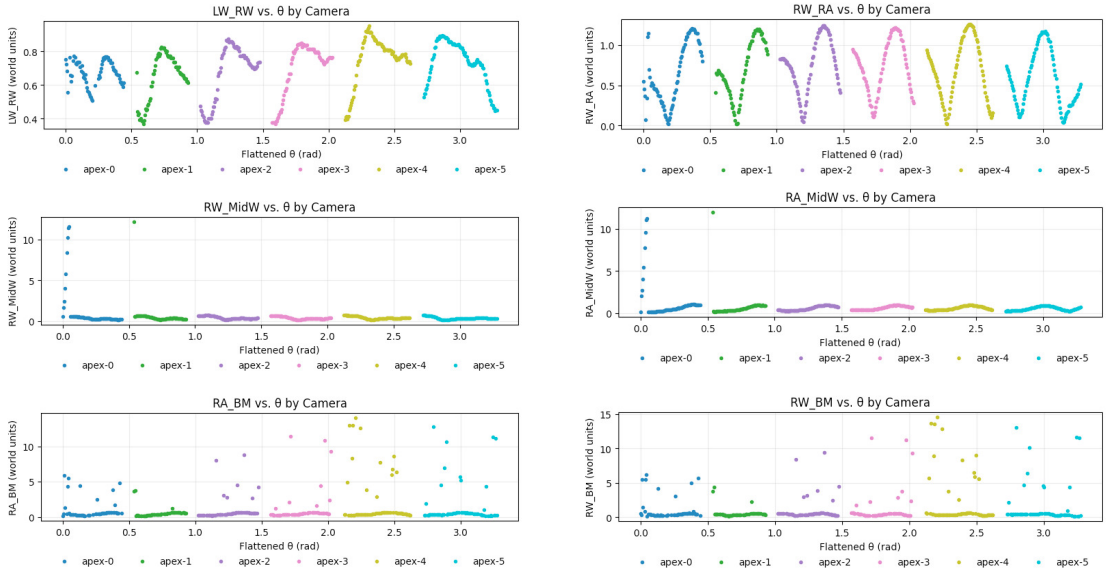

Figure 5.1: Comparison of one-dimensional signals derived from ground-truth keypoints across all six Apex cameras for Skater-1. The x-axis denotes the flattened angular position θ_{flat} in radians, and the y-axis shows the corresponding raw euclidean distance (world units) for each metric. Each coloured trace corresponds to one Apex camera segment after overlap removal. The RW–RA distance shows clear oscillatory patterns, while other signals remain flat or noisy.

Finding from Ground-Truth Analysis

After removing overlapping frames via the homography pipeline, all candidate signals were evaluated under comparable conditions. Among them, RW–RA exhibited clear cyclic oscillations without requiring any further trimming.

The other signals remained noisy or flat and would have needed additional heuristic pruning, which risks discarding genuine information. RW–LW also showed some degree of cyclicity, but the peaks and troughs were less distinctive, with the pattern breaking down in the first camera segment. Moreover, in practice, RW–LW is more prone to misclassification when applied to model predictions, since the left wrist is often occluded in the Apex-camera perspective. Taken together, these factors make RW–RA the most robust and generalisable choice for deriving a one-dimensional cyclic signal.

5.2 Baseline Phase Detection with Ground-truth

After establishing RW–RA as the principle one-dimensional signal, the next step is to reproduce the manually annotated phases using this signal. The RW–RA distance derived from ground-truth keypoints is analysed across the turn, and peaks and troughs are detected using prominence-based criteria (Section 3.4.3).

Before detecting peaks and troughs, the RW–RA signal was smoothed using a Savitzky–Golay filter (see Section 3.3.3) to reduce local jitter while preserving the oscillatory structure. Different window sizes (3, 5 and 7) were tested. As shown in Figure 5.2, a window size of 3 provided sufficient smoothing without introducing noticeable distortions. Larger windows increasingly shifted the peak locations of the extrema. For this reason, a window size of 3 was adopted in all subsequent experiments.

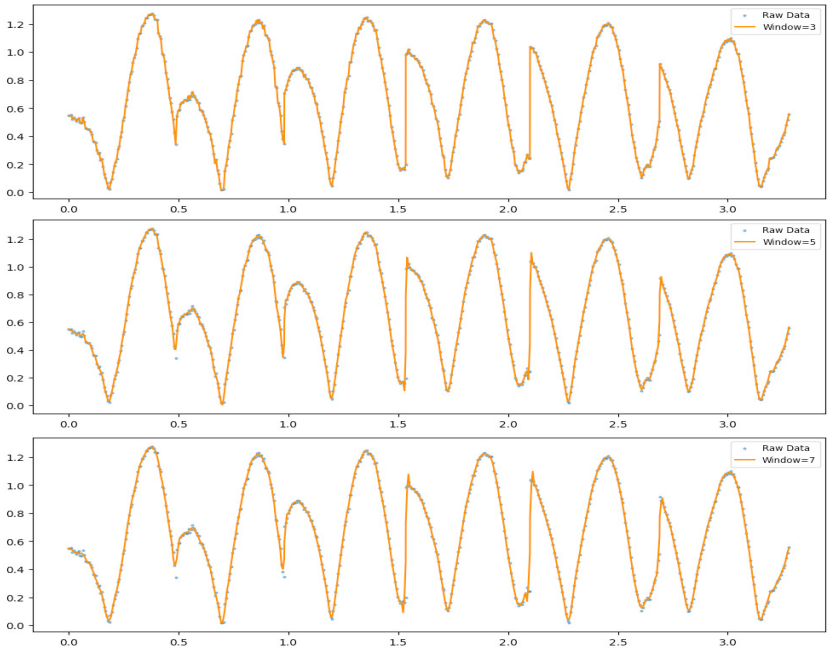

Figure 5.2: Comparison of Savitzky–Golay filtering of the RW–RA signal using window sizes 3 (top), 5 (middle), and 7 (bottom). The x-axis denotes the flattened angular position θ_{flat} in radians along the fitted turn, and the y-axis shows the RW–RA distance in world units. Blue points represent the raw signal, while the orange lines show the smoothed output at each window size. A smaller window size ($w = 3$) preserves peaks and troughs without distortion, whereas larger windows begin to shift peak positions.

5.2.1 Distinguishing Big and Small Extrema

Once the RW–RA signal is smoothed and its extrema are identified, the next step is to separate *big* from *small*, in the peaks and troughs. This classification reflects the alternating motion of the skating stroke and allows for a consistent mapping to annotated phases.

For peaks, each candidate maximum is compared against its immediate neighbours. A peak higher than both its neighbours is labelled as a **big peak**, while smaller intermediate maxima are labelled as **small peaks**. Edge cases at the start or end of the sequence are classified relative to the nearest available neighbour.

For troughs, the classification depends on their adjacent peaks. A trough following a big peak is labelled as a **small trough**, while a trough following a small peak is labelled as a **big trough**. This ensures that each cycle is represented by an alternating sequence of big and small extrema, consistent with the push–recovery rhythm of skating.

Figure 5.3 illustrates this logic on the smoothed RW–RA signal. The big extrema are represented as circles (red for peaks, blue for troughs), while the small extrema are represented as triangles (red for peaks, blue inverted for troughs).

5.2.2 Mapping to Phase Categories

With extrema classified, they can be mapped onto the manually annotated phase categories to evaluate how well the RW–RA signal reproduces the ground-truth segmentation. These categories reflect successive positions of the skater during a stroke, illustrated in Figure 5.4. From left to right, the snapshots show the transition from *Crouched Forward* into *Arm Forward*, then into *Crouched Backward*, and finally into *Arm Backward*.

The four annotated categories are defined as follows:

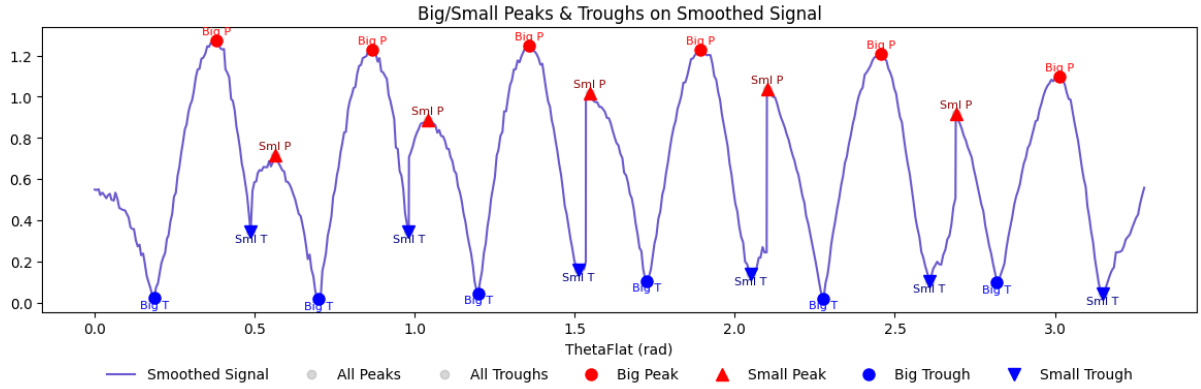

Figure 5.3: Classification of big and small extrema on the smoothed RW–RA signal. The x-axis denotes the flattened angular position θ_{flat} in radians, and the y-axis shows the RW–RA distance in world units. Big peaks (red circles) correspond to the forward extension of the stroke, small peaks (red triangles) to the backward extension, big troughs (blue circles) to the crouched forward phase, and small troughs (blue inverted triangles) to the crouched backward phase.

Figure 5.4: Illustration of the annotated phases for Skater-1 from Apex camera 3. From left to right: Arm Backward → Crouched Forward → Arm Forward → Crouched Backward.

- **Crouched Forward:** corresponds to a *big trough*. The skater is in a low crouched position at the beginning of the stroke cycle, with arms compressed close to the torso and preparing for the forward extension.
- **Arm Forward:** corresponds to a *big peak*, when the forward-swinging arm reaches maximum extension, coinciding with a strong forward push in the stroke cycle.
- **Crouched Backward:** corresponds to a *small trough*, when the skater returns to a crouched position after the forward extension, arms retracting towards the torso and preparing for the backward swing.
- **Arm Backward:** corresponds to a *small peak*, when the backward-swinging arm reaches maximum extension behind the body, completing the cycle before returning to the crouched forward position.

With this mapping established, the alignment between signal-derived phases and ground-truth annotations can now be quantified using the metrics defined in Section 3.5.

5.2.3 Baseline Error from Ground-truth

The alignment quality can first be quantified by comparing predicted and ground-truth boundaries at the individual segment level. Table 5.1 summarises the error for each annotated phase, reported as the difference between the predicted angular position and the ground-truth midpoint. Across all phases,

Table 5.1: Phase-level errors between predicted and ground-truth boundaries for Skater-1 across all cameras. Start and end keys are given in the form cam-i_*frame*, denoting the frame index within the corresponding camera. Errors are reported as the absolute difference between the predicted angular position and the midpoint of the annotated phase segment.

Phase	**Start Key**	**End Key**	**GT θ (rad)**	**Pred θ (rad)**	**Error (rad)**
Crouched Forward	cam-1_29	cam-1_33	0.179594	0.186249	0.006655
Arm Forward	cam-1_60	cam-1_64	0.357364	0.380487	0.023123
Crouched Backward	cam-1_88	cam-2_1	0.498645	0.487259	0.011386
Arm Backward	cam-2_15	cam-2_19	0.591702	0.562961	0.028740
Crouched Forward	cam-2_34	cam-2_36	0.694184	0.699980	0.005796
Arm Forward	cam-2_61	cam-2_63	0.849303	0.866716	0.017414
Arm Backward	cam-3_19	cam-3_21	1.082655	1.041085	0.041570
Crouched Forward	cam-3_38	cam-3_40	1.192852	1.198576	0.005724
Arm Forward	cam-3_62	cam-3_66	1.339248	1.356897	0.017650
Crouched Backward	cam-3_93	cam-3_95	1.516750	1.510718	0.006032
Arm Backward	cam-4_11	cam-4_13	1.546579	1.546862	0.000283
Crouched Forward	cam-4_40	cam-4_42	1.719829	1.725977	0.006148
Arm Forward	cam-4_66	cam-4_70	1.881433	1.893416	0.011983
Crouched Backward	cam-4_95	cam-4_97	2.050154	2.049772	0.000382
Arm Backward	cam-5_15	cam-5_17	2.106202	2.100474	0.005728
Crouched Forward	cam-5_43	cam-5_45	2.271377	2.277441	0.006064
Arm Forward	cam-5_69	cam-5_73	2.436908	2.455583	0.018675
Crouched Backward	cam-5_99	cam-5_101	2.613936	2.607860	0.006076
Arm Backward	cam-6_29	cam-6_31	2.697295	2.690601	0.006694
Crouched Forward	cam-6_49	cam-6_51	2.813016	2.818952	0.005936
Arm Forward	cam-6_79	cam-6_83	3.001111	3.013151	0.012040
Crouched Backward	cam-6_105	cam-6_107	3.149003	3.148761	0.000241

the mean absolute angular error was 0.011 rad. Given the global (across cameras) conversion factor of $\Delta\theta$/frame = 0.0059 rad/frame, this corresponds to an average misalignment of approximately two frames.

While the aggregate error provides a baseline, examining phase-wise MAE can give more insight about where the signal aligns well and where it struggles. Table 5.2 reports the mean absolute error for each type of phase, in both radians and equivalent frame units.

To illustrate these results visually, Figure 5.5 shows the RW–RA signal generated from ground-truth keypoints with phase annotations overlaid. The shaded regions correspond to the GT-annotated intervals for each phase, while the coloured markers indicate the automatically detected extrema mapped to phase categories. The plot confirms that the RW–RA signal captures the cyclic structure of the stroke and aligns closely with the annotated phases, consistent with the errors reported in Tables 5.1 and 5.2.

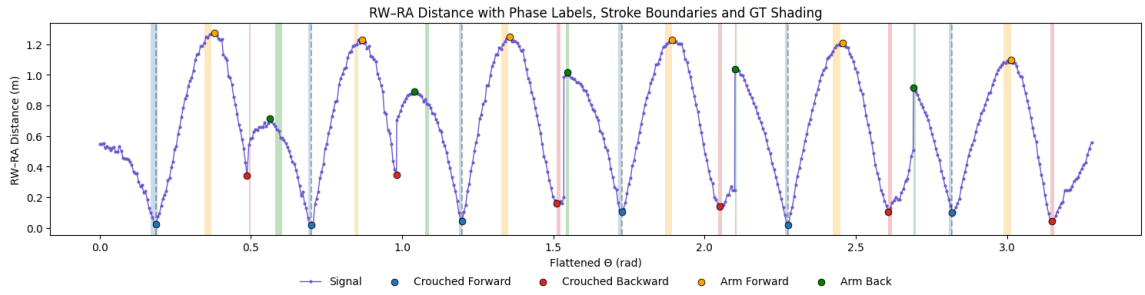

Figure 5.5: RW–RA signal derived from ground-truth keypoints with annotated phases. Vertical shading indicates the GT-annotated intervals, while coloured markers show detected extrema mapped to phases. The signal demonstrates clear cyclicity and close alignment with the annotated boundaries.

Table 5.2: Phase-wise mean absolute error (MAE) between predicted and annotated boundaries for Skater-1. Errors are reported both in angular units (radians) and in frame units, with the global average error per frame also shown.

Phase	**MAE (rad)**	**MAE (frames)**
Crouched Forward	0.00605	1.026
Crouched Backward	0.00482	0.818
Arm Forward	0.01681	2.851
Arm Backward	0.01660	2.815

Table 5.2 provides a phase-wise summary of mean absolute errors (MAE), reported both in radians and in frame units, together with the global average error per frame. The results indicate that crouched phases are reproduced with sub-frame accuracy, whereas arm phases show higher errors, typically around 2–3 frames.

The results confirm that the RW–RA signal captures the cyclic phases with high fidelity. The crouched phases (*Crouched Forward* and *Crouched Backward*) are reproduced with sub-frame to a single frame accuracy, while the arm phases (*Arm Forward* and *Arm Backward*) deviate by about three frames on average. This can be partly explained by annotation difficulty: the forward-extended phase is tricky to label precisely because skaters often hold the right arm in extension for several frames, producing a flatter peak in the RW–RA signal and increasing ambiguity in identifying the exact point of maximum extension. As a result, both human annotation and automated detection can show higher errors in this phase.

Design Decision #1. The ground-truth analysis of Skater-1 establishes RW–RA as the canonical one-dimensional signal for phase detection. It produced a clear oscillatory behaviour, aligned closely with annotated phases, and yielded the lowest baseline error among candidates. This decision is tied to Skater-1, where six-camera trajectories and valid homographies were available. For Skater-2, cropped recordings invalidate these homographies, so RW–RA in world coordinates cannot be applied. In that case, the evaluation is limited to the model-predicted keypoints in the image space, without phase detection.

Having established a GT baseline, we now turn to the dataset splits and training setup used to evaluate model-based predictions.

Chapter 6

Regime A : Pipeline with Skater-1 Model

This chapter presents Regime A, in which the model is trained exclusively on Skater 1 and evaluated on Skater 1, Skater 2, and Skater 3 for detection and keypoint accuracy. Temporal smoothing is assessed for all skaters, while multi-camera fusion and phase detection are reported only for Skater 1.

6.1 Model Predictions

6.1.1 Validation and Fold Selection

To identify the best-performing model, five-fold cross-validation was performed on the training set. Validation metrics follow Section 3.5. Table 6.1 summarises the results on the validation split (as defined in Table 4.2) across all folds. The metrics reported include mean per-joint position error (MPJPE) for all 12 keypoints and for the two critical joints (right wrist and right ankle), the percentage of correct keypoints at the thresholds 5% and 10% (PCK @ 0.05, PCK @ 0.10) and the 95th percentile error of the worst joint. All folds achieved nearly identical performance, with MPJPE in the range of 0.012–0.015 (normalised by the bounding box diagonal) and PCK values consistently above 0.98.

Table 6.1: Validation performance across the five folds for Skater-1 (Regime A). Metrics include detection rate, MPJPE normalised (all joints and RW/RA), PCK at 5% and 10%, and worst-joint 95th percentile error.

Fold	**Total**	**IoU $\geq$ 0.5**	**MPJPE$_n$ (K12)**	**MPJPE$_n$ (K2)**	**PCK@0.05**	**PCK@0.10**	**Worst-P95**
0	134	133	0.012	0.013	0.991	0.996	0.062
1	134	133	0.012	0.013	0.992	0.989	0.068
2	134	133	0.012	0.014	0.989	0.989	0.067
3	134	133	0.012	0.013	0.992	0.996	0.058
4	134	133	0.012	0.015	0.992	0.989	0.055

Given the stability of the models across folds, fold 3 with the lowest MPJPE and highest PCK values, also with a lower Worst-P95, was selected as the final model for downstream evaluation. This choice ensured that the subsequent test results represent the best-performing fine-tuned model while remaining representative of the overall cross-validation.

6.1.2 Bounding Box Detection: Pre-trained vs Fine-tuned

Accurate and consistent localisation of the main skater's bounding box (as defined by Algorithm 1) is essential for downstream stages such as temporal smoothing and phase detection. Table 6.2 compares bounding box detection performance of the pre-trained Detectron2 model against the fine-tuned model (fold 3, best choice from the previous step) across the train, validation, and test splits. The metrics include the number of undetected frames, cases with multiple predictions, detection rate, and mean intersection-over-union (IoU) with ground-truth bounding boxes.

Table 6.2: Bounding box detection performance of the pre-trained and fine-tuned models across train, validation, and test splits of Skater-1. Fine-tuned model consistently achieves perfect detection and high IoU, while the pre-trained model suffers from frequent missed detections and lower IoU.

Model	Split	#Images	Not Detected	Detection Rate	Mean IoU
Fine-tuned	Train	297	0	100.0%	0.975
	Valid	134	0	99.3%	0.961
	Test	162	0	100.0%	0.963
Pre-trained	Train	297	59	80.1%	0.742
	Valid	134	25	81.3%	0.762
	Test	162	26	84.0%	0.696

The results show that the pre-trained model is unreliable: nearly one-fifth of the frames in the training set and over 20% of frames in validation and test were not detected. Such gaps are problematic because downstream analysis requires continuous trajectories for temporal smoothing and phase segmentation. In contrast, the fine-tuned model achieved perfect detection across all frames in all splits, with a mean IoU consistently above 0.96. This shows that fine-tuning not only improves detection rate but also increases localisation accuracy, making it the only viable choice for subsequent stages of the pipeline.

6.1.3 Best Fold's Evaluation Across Data Splits

After selecting the best fold based on validation performance, the corresponding fine-tuned model was evaluated across training, validation, and test splits. Tables 6.3 and 6.4 report detection-level metrics (mean IoU, detection coverage) and keypoint-level metrics in normalised units (with the bounding-box diagonal) and in pixel space, respectively.

Table 6.3: Best fold: detection coverage and normalised keypoint metrics across splits (Skater-1). Normalisation is with respect to the per-frame main-skater bounding-box diagonal.

Split	#Images	Mean IoU	Det. Coverage	MPJPE$_n$ (K12)	MPJPE$_n$ (K2)	PCK@0.05 (12)	WorstP95$_n$
Train	297	0.975	100.0%	0.007	0.007	0.999	0.016
Valid	134	0.961	99.3%	0.012	0.012	0.992	0.065
Test	162	0.963	100.0%	0.014	0.015	0.986	0.104

Table 6.4: Best fold: pixel-space keypoint errors across splits (Skater-1). Pixel-space values complement the normalised metrics in Table 6.3.

Split	#Images	MPJPE$_{px}$ (K12)	MPJPE$_{px}$ (K2)	WorstP95$_{px}$
Train	297	1.67	1.71	4.30
Valid	134	2.99	3.03	14.71
Test	162	3.33	3.72	24.21

Interpretation. Although **WorstP95** appears numerically larger on validation and test, it reflects the 95th percentile of the single worst joint per frame and thus affects only a small fraction of frames. The low values of mean errors (MPJPE) and high values of PCK@0.05 indicate that the vast majority of predictions are well aligned and suitable for downstream temporal smoothing and phase detection.

With the best fold identified and shown to perform robustly across all splits of the dataset, which together span all six camera views, the fine-tuned model is presented as the reliable backbone for the remainder of the pipeline. The next step is to address the occasional local spikes and jitter through temporal smoothing, ensuring that the right wrist and right ankle trajectories provide clean, cyclic signals suitable for phase detection.

Note. For the downstream stages of the pipeline, four keypoints are critical: the right shoulder (RS), left shoulder (LS), right wrist (RW), and right ankle (RA). However, because the shoulders are the most distinctive points and exhibit stable trajectories with minimal motion relative to the other joints, their precision is already captured within the overall 12-keypoint evaluation. For this reason, only RW and RA are explicitly carried forward into the temporal smoothing stage.

6.2 Temporal Smoothing

To obtain smooth, cyclic peaks and troughs suitable for phase detection, it is important that the underlying joint trajectories do not contain spurious spikes that are not natural to their motion. The fine-tuned model achieved strong overall accuracy. Still, there were occasional spikes and local jitter visible in the trajectories, particularly for the right wrist. Such artefacts can disrupt the detection of peaks and troughs, and hence temporal smoothing is required.

Figure 6.1 shows the right wrist (RW) trajectories across the six Apex cameras. Several sequences show noticeable outliers, most prominently at apex 0, apex 3, and apex 4, while others (e.g. apex 2) remain mostly smooth with only minor local jitter. In contrast, the right ankle (RA) trajectories shown in Figure 6.2 are relatively stable and follow the ground-truth motion closely, with only occasional irregularities (e.g. at apex 5). Although the ankle is less affected, the same smoothing rule is applied to both RW and RA for consistency and to protect against potential irregularities in other scenarios.

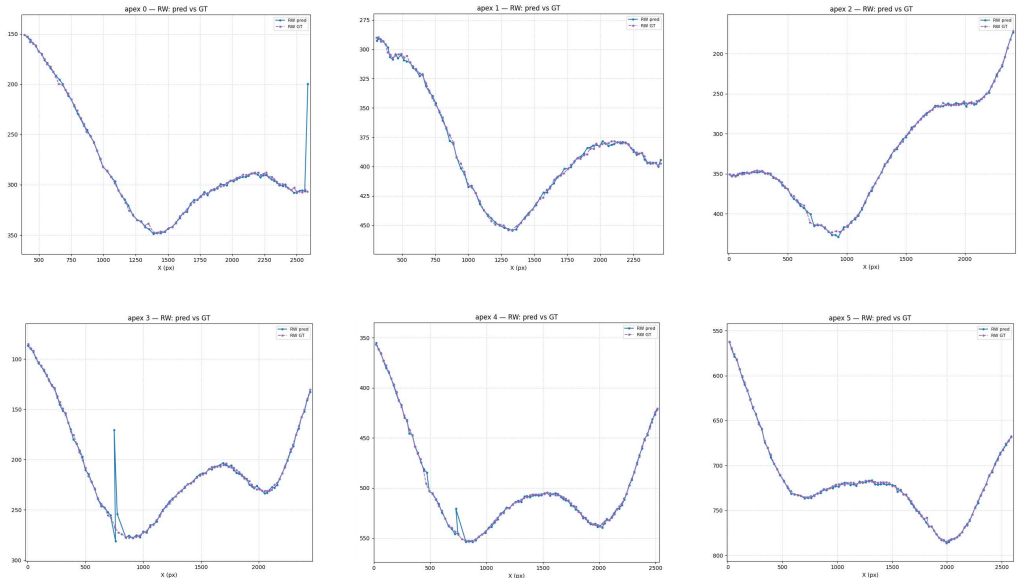

Figure 6.1: Right wrist (RW) trajectories across all six Apex cameras for Skater-1. Blue: model predictions, dashed: ground truth. The x-axis denotes the horizontal image coordinate in pixels, and the y-axis denotes the vertical image coordinate in pixels (origin at top). Problematic spikes are particularly visible at apex 0, apex 3, and apex 4, while other sequences (e.g. apex 2) show only minor jitter.

In the following subsections, we describe the smoothing pipeline. Our approach separates the process into two stages: (i) removing outliers and (ii) filling short gaps. This is contrasted against alternative methods such as Kalman filtering and RANSAC polynomial fitting, which attempt both tasks simultaneously.

6.2.1 Z-score Cleaning

The robust iterative z-score cleaning described in Section 3.3 was applied to the right wrist and the right ankle. This process flagged outliers only in the first pass, only for the right wrist, across 3 cameras, and no new frames were detected in subsequent iterations, causing the process to stop automatically. This indicates that most sequences are stable, with a few clear spikes that can be corrected.

Figure 6.3 highlights three cases for the right wrist (apex 0, apex 3, apex 4) where outliers were flagged. The top row shows the robust z-score distributions of normalised displacements in x or y, with red dashed lines marking the threshold of ± 11. Values outside these thresholds were flagged as outliers. The bottom row shows the corresponding raw vs. cleaned trajectories against the ground truth. The large spikes present in the raw predictions are effectively removed, while the surrounding motion is preserved.

After examining all cameras and both joints, a threshold of ± 11 was found to be a safe and consistent boundary. This value was therefore fixed for all subsequent cleaning, ensuring a standard procedure

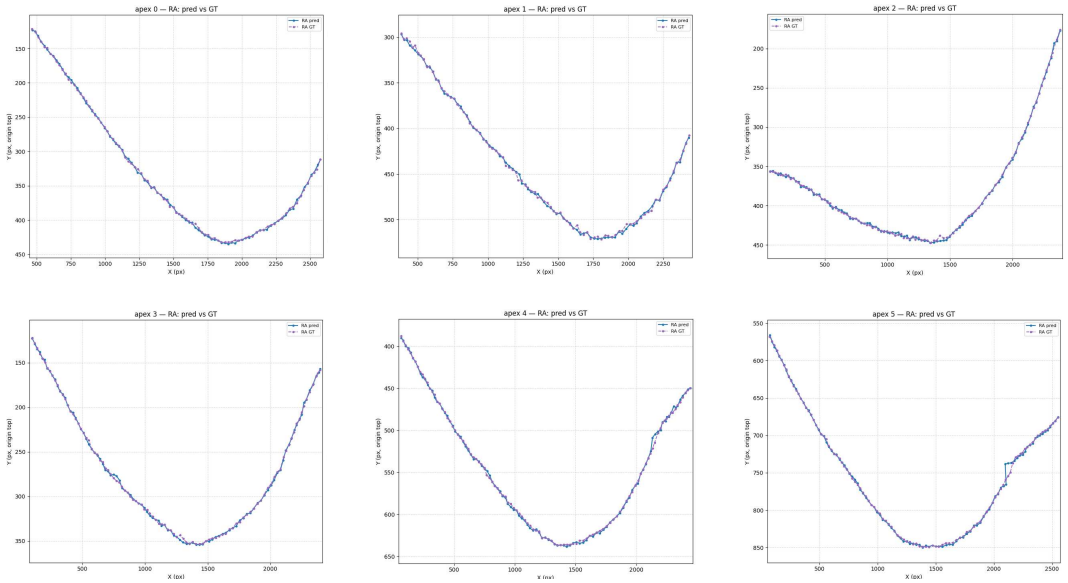

Figure 6.2: Right ankle (RA) trajectories across all six Apex cameras for Skater-1. Blue: model predictions, dashed: ground truth. The x-axis denotes the horizontal image coordinate in pixels, and the y-axis denotes the vertical image coordinate in pixels (origin at top). RA trajectories are generally stable, with only minor local irregularities (e.g. at apex 5).

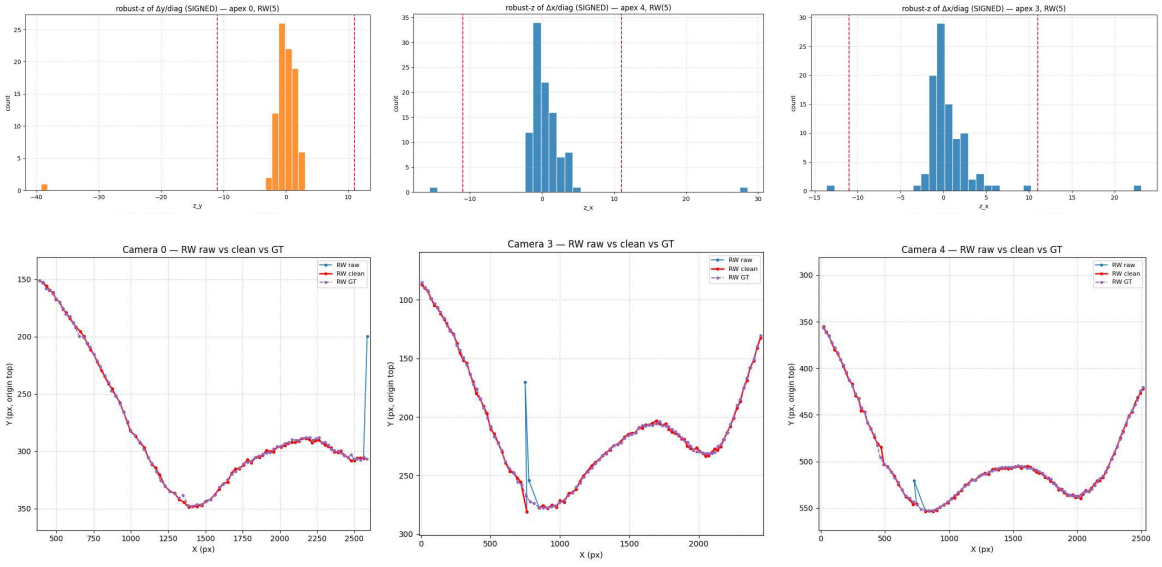

Figure 6.3: Illustration of z-score cleaning for the right wrist in apex 0, apex 3, and apex 4. Top row: robust z-score histograms for Δx or Δy with thresholds ± 11. Bottom row: raw trajectories (blue), cleaned (red), and ground truth (dashed). The x-axis denotes horizontal image coordinates (px), the y-axis vertical coordinates (px, origin at top).

across datasets. Across the remaining Skater-1 sequences (all RA, and RW on apex 1, 2, 5), no frames were flagged by the robust-z test. Hence, the cleaned trajectories remain equal to the raw predictions.

6.2.2 Alternative Filters: Kalman and RANSAC

Kalman filtering and RANSAC-based polynomial fitting (as defined in Section 3.3) were also evaluated as alternatives to outlier-based z-score cleaning. Both approaches attempt to simultaneously suppress outliers and interpolate across gaps, in contrast to our staged approach (removal first, then interpolation).

Since the right wrist (RW) exhibited the most pronounced outliers, these methods were evaluated in detail on this joint to test whether they could handle the most challenging case. Hyperparameter tuning was performed across cameras, with individual models fit per camera using each parameter combination. The normalised mean per-joint position error (MPJPE_n) for the right wrist was then averaged across all frames of all Apex sequences, resulting in the overall grids reported below. This ensured that the best hyperparameters were selected based on their behaviour across the full dataset, not on a single sequence.

Tables 6.5 and 6.6 summarise the results. For the Kalman filter, we varied the process noise q and measurement noise r, and for RANSAC, we varied the polynomial degree and the inlier threshold. The highlighted cells mark the best-performing settings in each case.

Table 6.5: Grid search for Kalman filter smoothing (RW only, frame-weighted across cameras). Reported values are overall MPJPE (normalised). q denotes process noise variance, r measurement noise variance.

q	$r = 2$	$r = 4$	$r = 9$	$r = 16$	$r = 21$
0.5	0.0022	0.0022	0.0024	0.0026	0.0027
1	**0.0022**	0.0022	0.0023	0.0024	0.0024
2	0.0022	0.0022	0.0022	0.0022	0.0023
4	0.0022	0.0022	0.0022	0.0022	0.0022
6	0.0022	0.0022	0.0022	0.0022	0.0022

Table 6.6: Grid search for polynomial fitting with RANSAC (RW only, frame-weighted across cameras). Reported values are overall MPJPE (normalised). deg is polynomial degree, thr the inlier threshold.

deg	thr=4	thr=6	thr=8	thr=10	thr=12	thr=14	thr=16
4	0.0175	0.0174	0.0177	0.0185	0.0179	0.0188	0.0183
5	0.0092	0.0096	0.0105	0.0122	0.0125	0.0117	0.0086
6	0.0115	0.0088	0.0084	0.0094	0.0069	0.0055	0.0054
7	0.0058	0.0058	0.0056	0.0056	0.0056	**0.0056**	0.0056
8	0.0219	0.0210	0.0183	0.0185	0.0186	0.0180	0.0179

Figure 6.4 shows qualitative examples for RW trajectories at apex 0, 3, and 4, where outliers were most distinctive. The top row compares raw vs. RANSAC-smoothed trajectories, and the bottom row compares raw vs. Kalman-smoothed trajectories, both against ground truth. In both cases, the filters introduce additional artefacts and distort the trajectory shape, while having limited effect on the true spikes.

Overall, despite hyperparameter tuning, Kalman and RANSAC offered little improvement in spike removal and often worsened the signal by introducing new artefacts. We therefore proceed with the outlier-based method described earlier, filling only very short gaps (up to 4 frames) with linear interpolation as described in Section 3.3.

The cleaned trajectories provide stable inputs for multi-camera fusion and phase detection, described next.

6.3 Fusion and Phase Detection

After temporal smoothing, the next stage is to fuse trajectories across cameras and extract the cyclic signal used for phase detection. Following the same procedure described in Section 3.4, the predictions per camera were first reprojected into world coordinates using homography and distortion correction.

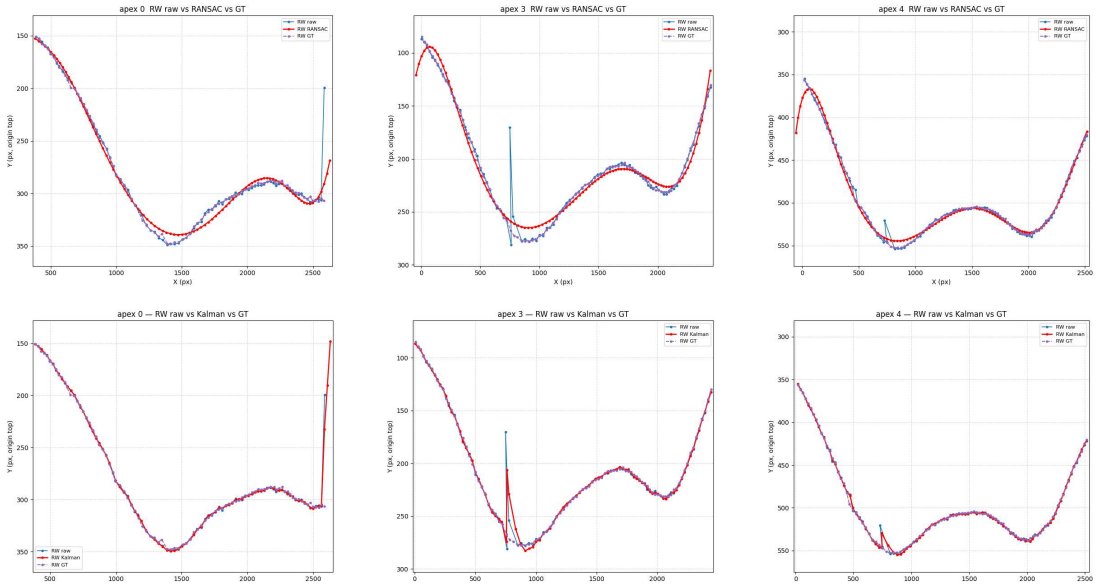

Figure 6.4: Comparison of RANSAC (top row) and Kalman (bottom row) smoothing on RW trajectories for apex 0, 3, and 4. Blue: raw predictions, red: smoothed, dashed: ground truth. The x-axis denotes horizontal pixel coordinate, the y-axis vertical pixel coordinate (origin at top). Both methods oversmooth and introduce distortions, rather than removing true spikes.

The overlapping regions were pruned using the angular position θ, estimated from the midpoint of the shoulders; the fitted circle centre was located at $(-0.39, -1.69)$ in the world coordinates, providing a consistent reference for rotation across cameras. This produces a continuous trajectory covering the entire corner, with a flattened angular range of

$$\theta_{\text{flat}} \in [0.000, 3.283] \text{ rad.}$$

From these fused trajectories, the distance between the right wrist and the right ankle (RW–RA) was calculated and used as a one-dimensional signal for phase detection. As discussed in Section 5, this distance clearly captures the cyclic stroke motion while being relatively robust to occlusion. The RW–RA signal was then processed as in Section 5.2, using Savitzky–Golay smoothing followed by prominence-based detection of local maxima and minima to identify cycle boundaries.

Figure 6.5 shows the resulting RW–RA signal across the stitched trajectory. The detected extrema are overlaid along with the annotated ground-truth phase intervals, demonstrating the alignment between predicted cycles and annotations.

The quantitative accuracy of phase detection is reported in Tables 6.7 and 6.8. At the individual boundary level, the predicted extrema align closely with the annotated midpoints, with a mean angular error of 0.0126 rad (corresponding to about 2.1 frames, given $\Delta\theta/\text{frame} = 0.0059$ rad/frame). Crouched phases are recovered with sub-frame to ≈ 1 frame accuracy, while arm phases show larger deviations of about 3–4 frames.

To place these results in context, Table 6.9 compares the frame-based MAE of the predicted RW–RA signal to the baseline errors obtained directly from ground-truth keypoints (Section 5.2). Both analyses follow the same trend: crouched phases are detected with very high precision (within one frame), while arm phases carry larger errors of around 3–4 frames.

The predicted signal shows slightly higher deviations for the Crouched Forward (+0.3 frames) and Arm Backward (+1.0 frames), but remains closely aligned with the ground-truth baseline. Part of this effect can be attributed to camera geometry, as the transitory frames between cameras often occur at the end of the Arm Backward phase as the skater transitions into Crouched Forward. At these boundaries, the narrow and occasionally occluded field of view makes extrema more difficult to detect consistently, leading to larger deviations in the affected phase timings.

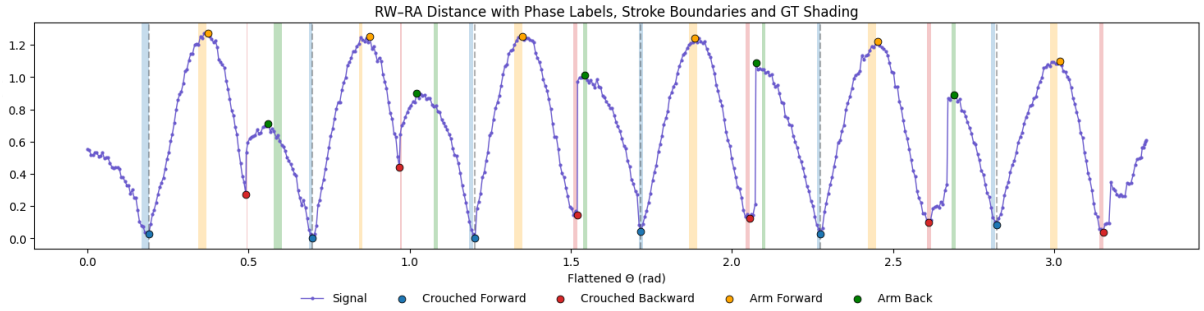

Figure 6.5: RW–RA distance across the fused trajectory for Skater-1, shown as a function of flattened angular position θ_{flat}. Detected extrema are marked, with shaded regions indicating the annotated ground-truth phases (Crouched Forward, Arm Forward, Crouched Backward, Arm Backward). The x-axis denotes θ_{flat} in radians, and the y-axis denotes RW–RA distance in world units.

Table 6.7: Phase-level errors between predicted and ground-truth boundaries for Skater-1 across all cameras, using RW–RA distance after Savitzky–Golay smoothing and SciPy peak detection. Start and end keys are given in the form cam-i_*frame*, denoting the frame index within the corresponding camera. Errors are reported as the absolute difference between the predicted angular position and the midpoint of the annotated phase segment.

Phase	**Start Key**	**End Key**	**GT θ (rad)**	**Pred θ (rad)**	**Error (rad)**
Crouched Forward	cam-1_29	cam-1_33	0.179	0.191	0.011
Arm Forward	cam-1_60	cam-1_64	0.357	0.374	0.017
Crouched Backward	cam-1_88	cam-2_1	0.498	0.492	0.006
Arm Backward	cam-2_15	cam-2_19	0.591	0.562	0.029
Crouched Forward	cam-2_34	cam-2_36	0.693	0.699	0.006
Arm Forward	cam-2_61	cam-2_63	0.848	0.876	0.028
Crouched Backward	cam-2_87	cam-2_2	0.976	0.968	0.007
Arm Backward	cam-2_19	cam-2_21	1.081	1.022	0.059
Crouched Forward	cam-2_38	cam-2_40	1.192	1.203	0.011
Arm Forward	cam-2_62	cam-2_66	1.337	1.350	0.012
Crouched Backward	cam-2_93	cam-2_95	1.514	1.520	0.006
Arm Backward	cam-3_11	cam-3_13	1.545	1.545	0.000
Crouched Forward	cam-3_40	cam-3_42	1.718	1.718	0.000
Arm Forward	cam-3_66	cam-3_70	1.879	1.885	0.006
Crouched Backward	cam-3_95	cam-3_97	2.048	2.054	0.006
Arm Backward	cam-4_15	cam-4_17	2.098	2.074	0.024
Crouched Forward	cam-4_43	cam-4_45	2.269	2.275	0.006
Arm Forward	cam-4_69	cam-4_73	2.434	2.453	0.018
Crouched Backward	cam-4_99	cam-4_101	2.610	2.610	0.000
Arm Backward	cam-5_29	cam-5_31	2.688	2.688	0.000
Crouched Forward	cam-5_49	cam-5_51	2.810	2.822	0.012
Arm Forward	cam-5_79	cam-5_83	2.998	3.016	0.018
Crouched Backward	cam-5_105	cam-5_107	3.146	3.151	0.006

Table 6.8: Phase-wise mean absolute error (MAE) between predicted and annotated boundaries for Skater-1, based on RW–RA predictions. Errors are reported both in angular units (radians) and in frame units, with the global average error per frame also shown.

Phase	**MAE (rad)**	**MAE (frames)**
Crouched Forward	0.0077	1.32
Crouched Backward	0.0051	0.88
Arm Forward	0.0167	2.85
Arm Backward	0.0224	3.83

Table 6.9: Comparison of phase-wise mean absolute error (MAE) in frames between ground-truth signal and RW–RA predicted signal.

Phase	Ground Truth (frames)	Predicted (frames)
Crouched Forward	1.03	1.32
Crouched Backward	0.82	0.88
Arm Forward	2.85	2.85
Arm Backward	2.82	3.83

6.4 Generalisation to Skater-2 and Skater-3

While the fine-tuned model demonstrated strong performance on Skater-1, its ability to generalise was assessed by applying the Regime A model (trained exclusively on Skater-1) to Skater-2 and Skater-3. These datasets share the same camera topology but differ in appearance and recording conditions. Tables 6.10 and 6.11 summarise the detection-level and keypoint-level metrics, respectively.

Table 6.10: Detection-level behaviour of the Skater-1-only model on Skater-2 and Skater-3. Columns report the number of images, missed detections, detection rate, and mean IoU with ground-truth boxes.

Dataset (Camera)	#Images	Not Detected	Detection Rate	Mean IoU
Skater-3 (cam-3)	55	0	100.00%	0.856
Skater-2 (cam-1)	104	8	92.30%	0.911
Skater-2 (cam-2)	110	7	93.60%	0.915
Skater-2 (cam-3)	110	2	98.20%	0.917
Skater-2 (cam-4)	89	0	100.00%	0.846
Skater-2 (cam-5)	80	0	100.00%	0.916
Skater-2 (cam-6)	87	0	100.00%	0.884

Table 6.11: Keypoint-level behaviour of the Skater-1-only model on Skater-2 and Skater-3. MPJPE_n is normalised by the per-frame main-skater bounding-box diagonal; K2 refers to the two focus joints (RW, RA).

Dataset (Camera)	MPJPE_n (K12)	MPJPE_n (K2)	PCK@0.05 (K12)	PCK@0.10 (K2)	WorstP95_n
Skater-3 (cam-3)	0.093	0.074	0.679	0.864	0.586
Skater-2 (cam-1)	0.090	0.130	0.617	0.630	0.483
Skater-2 (cam-2)	0.109	0.145	0.526	0.549	0.446
Skater-2 (cam-3)	0.097	0.133	0.573	0.542	0.456
Skater-2 (cam-4)	0.244	0.442	0.594	0.628	7.058
Skater-2 (cam-5)	0.097	0.130	0.581	0.575	0.486
Skater-2 (cam-6)	0.124	0.160	0.533	0.651	0.457

The results in Tables 6.10 and 6.11 demonstrate that the Skater-1 only model does not generalise to Skater-2 and Skater-3. Compared to Skater-1 performance, the normalised keypoint error (MPJPE_n) increases by a factor of 6–10×, and PCK@0.05 falls into the 0.52–0.68 band.

In particular, Skater-2 cam-4 is highly brittle: MPJPE_n(K2) reaches 0.442 and WorstP95_n increases to 7.06, indicating the presence of severe outliers. This behaviour is visible in the raw vs. GT plots of Skater-2's right ankle and right wrist (Figures 6.6 and 6.7), where the predictions show long vertical spikes, flat stretches, and discontinuous jumps. Such errors are systemic rather than occasional and cannot be removed through iterative blanking and interpolation.

Skater-3 (cam-3) represents the "least bad" case, but still falls far short of Skater-1 quality. The right ankle is relatively stable, but the right wrist exhibits large gaps and discontinuities. Even when applying the outlier cleaning pipeline (Figure 6.8), the RW trajectory remains broken, losing the natural curvature required for a robust RW–RA distance signal, while the RA trajectory is only partially corrected.

These results confirm that cleaning and filling cannot smooth the predictions for Skater-2 and Skater-3. The raw vs. GT comparisons (Figures 6.8) reveal that these are not isolated outliers, but dense repeated mispredictions that systematically distort the trajectories. Hence, Skater-2 and Skater-3 signals are not

suitable for downstream smoothing and phase detection. This highlights a fundamental limitation of Regime A and motivates the design of Regime B, where pooled training across multiple skaters is used to improve robustness and generalisation.

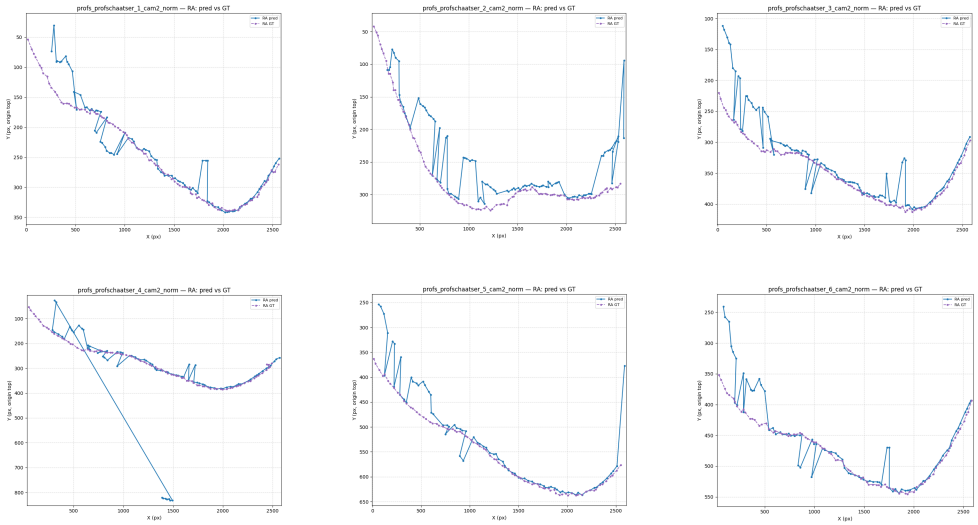

Figure 6.6: Skater-2 right ankle predictions vs ground truth. x-axis: horizontal pixels, y-axis: vertical pixels (origin at top). Long excursions and spikes reflect systemic prediction errors.

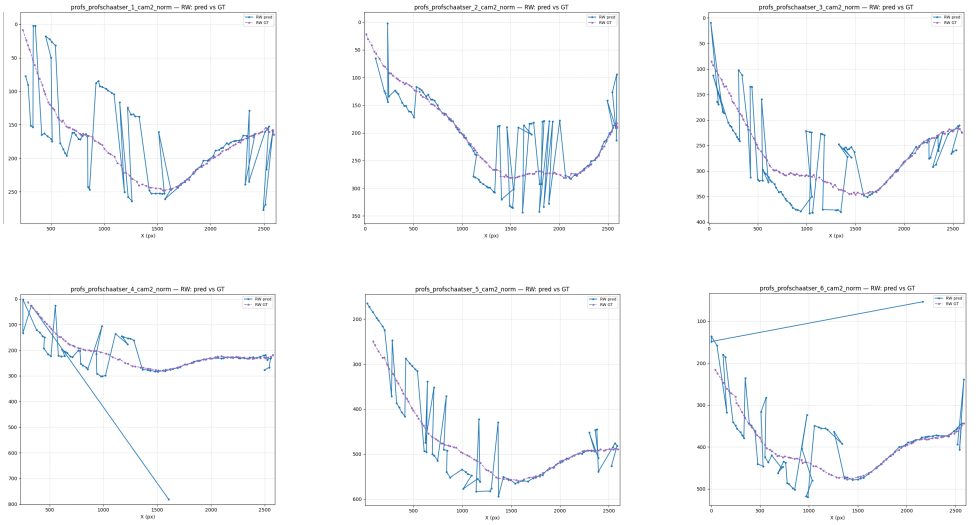

Figure 6.7: Skater-2 right wrist predictions vs ground truth. Systematic spikes and discontinuities cannot be corrected by cleaning.

6.5 Interim Conclusion (Regime A)

Regime A validates the pipeline design on Skater-1. The fine-tuned model substantially outperformed the pre-trained baseline in bounding-box detection and keypoint prediction. Iterative z-score cleaning with interpolation provided stable trajectories while preserving biomechanical plausibility, and multi-camera fusion with phase detection reproduced ground-truth cycle boundaries with high fidelity (errors within one frame for crouched phases and 3–4 frames for arm phases).

At the same time, Regime A is inherently limited. The model does not generalise to Skater-2 and Skater-3, and its reliance on the specific homographies of Skater-1 restricts its applicability for phase detection beyond that dataset. These observations underline that Regime A serves primarily as a proof of concept. The next chapter therefore introduces Regime B, where the model is trained on pooled data from multiple skaters to improve generalisation and robustness.

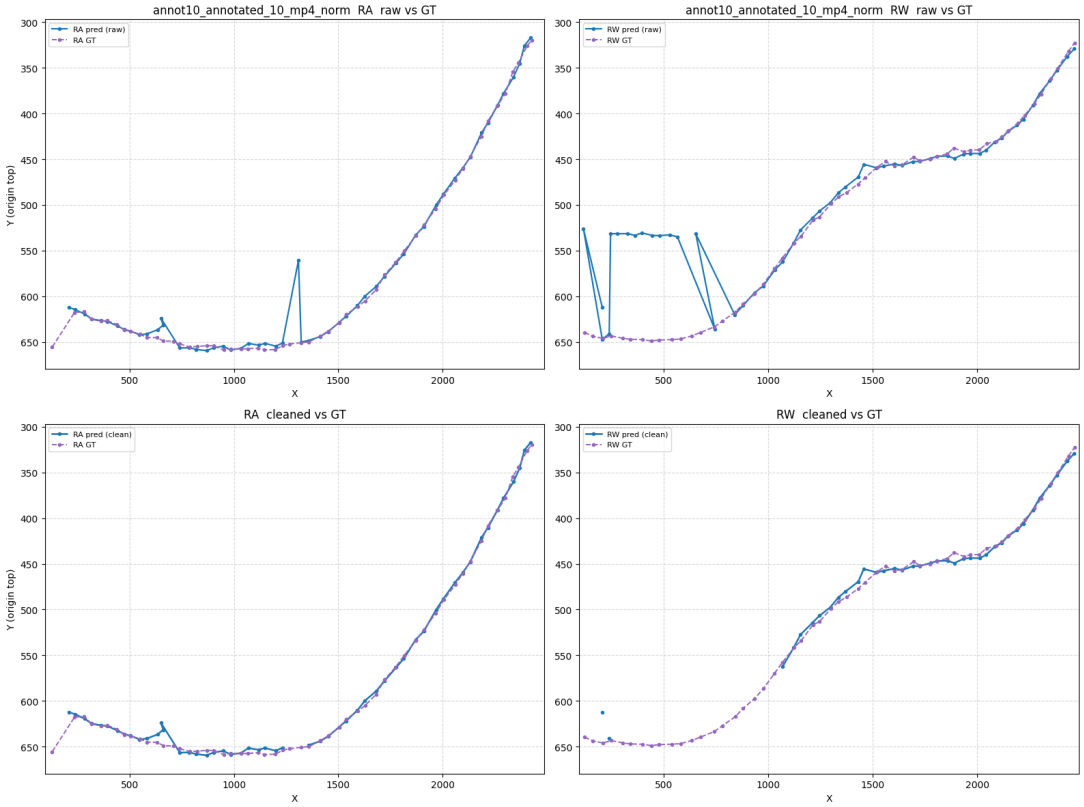

Figure 6.8: Attempted outlier removal on Skater-3 right ankle and right wrist. RA trajectory is partially corrected, but RW remains broken, losing the natural curvature required for phase detection.

Chapter 7

Regime B : Pipeline with Multi-Skater Model

Regime B addresses the limitation observed in Section 6.4 by training a single model on the combination of data from all available skaters. The goal is to improve robustness to appearance, illumination, and recording differences while keeping the pipeline identical in structure to Regime A wherever possible. The model predictions are evaluated for Skater-1, Skater-2, and Skater-3, and temporal smoothing is applied to all three. Fusion and phase detection are performed only for Skater-1, since valid homographies are available only for this dataset (Skater-2 recordings are cropped, invalidating the Skater-1 homographies, and Skater-3 has single camera coverage that prevents stitching).

7.1 Model Predictions

7.1.1 Validation and Fold Selection

The multi-skater dataset defined in Table 4.3 was split into training, validation, and test partitions. Similar to Regime A, a five-fold cross-validation procedure was applied on the training set, with the validation set used to monitor performance. Table 7.1 reports the operational metrics on the validation split across all five folds. The metrics follow the definitions in Section 3.5, and include normalised mean per-joint position error (MPJPE_n) over all 12 joints and over the two focus joints (RW, RA), percentage of correct keypoints (PCK) at thresholds of 0.05 and 0.10, and the worst-joint 95th percentile error (WorstP95_n).

Table 7.1: Validation performance across the five folds for the multi-skater dataset (Regime B). Metrics include detection coverage, MPJPE normalised (all joints and RW/RA), PCK at 5% and 10%, and worst-joint 95th percentile error.

Fold	Total	Not Detected	IoU $\geq$ 0.5	Rate|Det	MPJPE_n (K12)	MPJPE_n (K2)	PCK@0.05 (K12)	Worst-P95_n
0	307	0	307	1.000	0.012	0.013	0.992	0.063
1	307	0	307	1.000	0.012	0.011	0.993	0.061
2	307	0	307	1.000	0.012	0.012	0.992	0.068
3	307	0	307	1.000	0.012	0.012	0.991	0.059
4	307	0	307	1.000	0.012	0.012	0.992	0.059

All folds achieved nearly identical performance, with MPJPE_n around 0.012, PCK@0.05 close to 0.99, and WorstP95_n between 0.059 and 0.068.

Given the stability of the results across folds, Fold 1 was selected as the final model for downstream evaluation, since it has achieved the lowest MPJPE_n and highest PCK values while also maintaining a relatively low WorstP95_n. This choice ensures that subsequent test-set results reflect the strongest-performing fine-tuned model while remaining representative of the overall cross-validation.

7.1.2 Bounding Box Detection: Pre-trained vs Fine-tuned

Reliable detection of the main skater (according to Algorithm 1) across all frames is a prerequisite for downstream stages such as temporal smoothing and phase detection. Table 4.3 summarised the dataset partitions used in this regime, and Table 7.2 compares bounding box detection performance of the pre-trained Detectron2 model against the fine-tuned model (Fold 1, chosen from Section 7.1.1) across the train, validation, and test splits. The metrics reported include the number of undetected frames, detection coverage, and mean intersection-over-union (IoU) with ground-truth bounding boxes.

Table 7.2: Bounding box detection performance of the pre-trained and fine-tuned models across train, validation, and test splits of the pooled multi-skater dataset (Regime B). Fine-tuned model consistently achieves near-perfect detection and high IoU, while the pre-trained model fails to generalise, particularly on darker Skater-2 images.

Model	Split	#Images	Not Detected	Detection Rate	Mean IoU
Fine-tuned	Train	1825	0	100.0%	0.969
	Valid	307	0	100.0%	0.961
	Test	307	1	99.3%	0.958
Pre-trained	Train	1825	1753	3.5%	0.785
	Valid	307	295	3.3%	0.743
	Test	307	299	1.6%	0.650

The results highlight that the pre-trained model is unreliable on the pooled dataset, particularly on Skater-2 recordings, where darker illumination leads to frequent missed detections. Across the train, validation, and test splits, the pre-trained model missed the vast majority of frames, with detection coverage as low as 1–3% and mean IoU values well below acceptable levels (0.65–0.78). In contrast, the fine-tuned model consistently achieved perfect detection on the training and validation sets, and 99.3% detection coverage on the test set with mean IoU above 0.95. This demonstrates that fine-tuning on the pooled dataset not only restores reliable detection coverage but also improves localisation accuracy, providing a stable basis for subsequent smoothing and phase detection stages.

7.1.3 Best Fold's Evaluation Across Data Splits

After selecting the best-performing fold from the validation results, its performance was evaluated across the training, validation, and test splits of the pooled dataset (v16). Tables 7.3 and 7.4 report the results using normalised metrics and pixel-space metrics, respectively. The metrics follow the same definitions as in Section 3.5, allowing direct comparison with Regime A (Section 6.1).

Table 7.3: Performance of the best fold across the v16 splits, reported with normalised metrics. Metrics include detection coverage, mean IoU, MPJPE for all 12 keypoints and for RW/RA, PCK at 5% and 10%, and the worst-joint 95th percentile error (all normalised by bounding box diagonal).

Split	#Images	Not Detected	Det. Coverage	Mean IoU	MPJPE_n (K12)	MPJPE_n (K2)	PCK@0.05	PCK@0.10	WorstP95_n
train	1825	0	100.0%	0.969	0.006	0.006	1.000	1.000	0.016
valid	307	0	100.0%	0.961	0.012	0.011	0.993	1.000	0.061
test	307	1	99.3%	0.958	0.012	0.011	0.992	0.998	0.059

Table 7.4: Performance of the best fold across the v16 splits, reported in pixel-space metrics. Metrics include MPJPE for all 12 keypoints and for RW/RA (in pixels), and the worst-joint 95th percentile error in pixels.

Split	#Images	MPJPE_{px} (K12)	MPJPE_{px} (K2)	WorstP95_{px}
train	1825	1.33	1.29	3.44
valid	307	2.51	2.41	12.11
test	307	2.53	2.40	13.10

Across all splits, the fine-tuned Regime B model maintains strong performance. On the training set, errors are nearly negligible, with mean per-joint errors below 0.01 (normalised) and sub-pixel accuracy

in absolute terms. The validation and test sets show slightly larger errors, with MPJPE$_{px}$ around 2.5 pixels and WorstP95 reaching 12–13 pixels. As in Regime A, these high worst-joint values reach a very small fraction of frames, while the majority of predictions remain well aligned, as confirmed by PCK values exceeding 0.99. Overall, the results demonstrate that the pooled multi-skater training improves robustness while retaining the precision needed for downstream smoothing and phase detection.

Having established the baseline accuracy of the best fold across all splits, we now turn to the next stage of the pipeline: applying temporal smoothing to the predicted trajectories.

7.2 Temporal Smoothing

We follow the same two-stage policy described in Section 3.3 and validated for Regime A in Section 6.2: (i) iterative robust-z outlier removal on frame-to-frame, diagonal-normalised steps; followed by (ii) linear interpolation for gaps up to four frames. For methodological consistency, this procedure is applied uniformly to both right-wrist (RW) and right-ankle (RA) trajectories across Skater-1, Skater-2, and Skater-3.

7.2.1 Outlier detection procedure

We follow the same robust-z based method described in Section 3.3, where frame-to-frame steps are normalised by the bounding-box diagonal, outliers are flagged using the median and MAD with a fixed threshold ($k = 11$), contiguous runs are blanked, and only short gaps (up to four frames) are subsequently filled by linear interpolation.

Skater-1 (raw vs GT)

Under Regime B, both RA and RW trajectories closely track the ground truth across all six cameras; no points are flagged by the robust-z test, so cleaning is a no-op. Figure 7.1 shows RA (raw predictions vs. GT), and Figure 7.2 shows RW. Axes denote pixel coordinates (x: horizontal; y: vertical, origin at top).

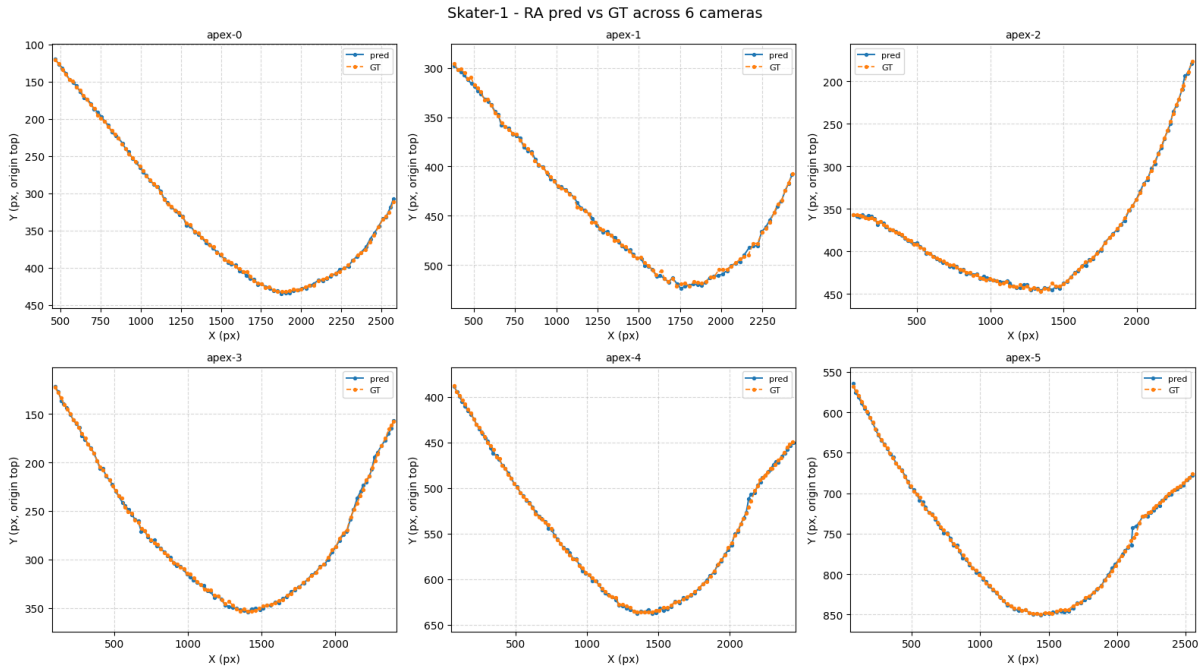

Figure 7.1: Skater-1, RA raw predictions (solid) vs. ground truth (dashed) across six cameras. No outliers are flagged; the cleaning stage leaves trajectories unchanged.

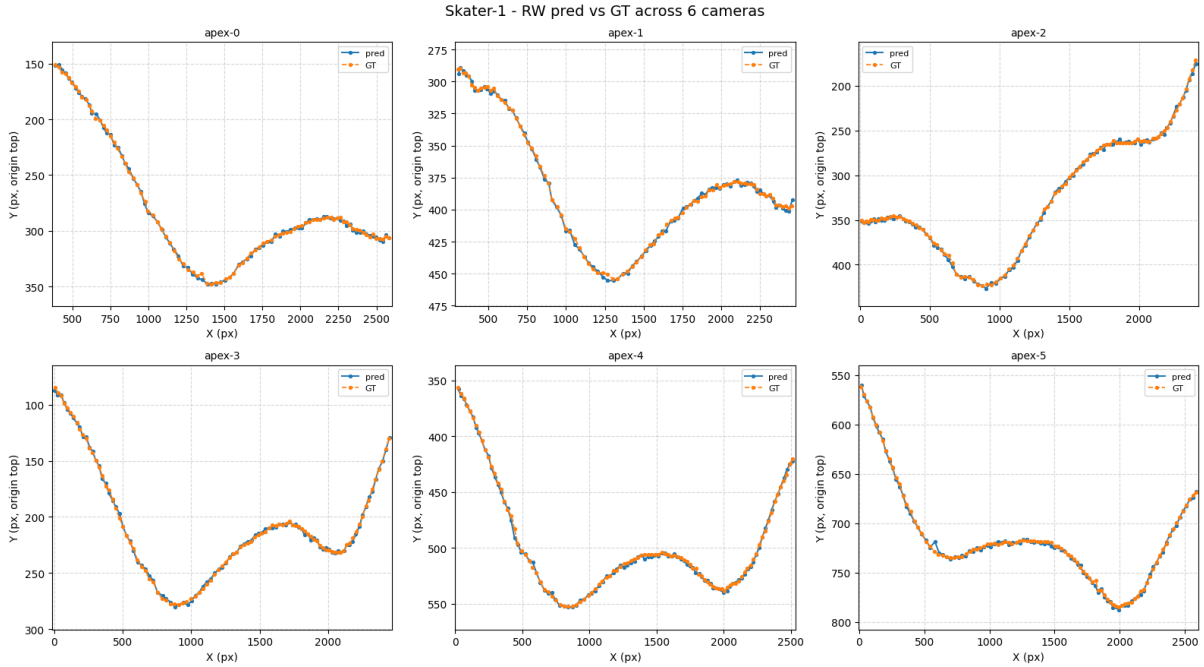

Figure 7.2: Skater-1, RW raw predictions vs. ground truth across six cameras. No flags; cleaned = raw.

Skater-2 (raw vs. GT, and the one anomaly)

Across six cameras, RA and RW are also well aligned with ground truth in the raw predictions (Figures 7.3 and 7.4). The only severe anomaly appears for **RA on camera 2**: a short, high-magnitude excursion in the vertical direction that breaches the robust-z threshold. Figure 7.5 shows the complete diagnostics and fix: the top panel plots the signed robust-z distributions for the diagonal-normalised steps, $\Delta x/d$ and $\Delta y/d$, with dashed bands at $\pm k$ (here $k = 11$). The same outlier is flagged in both components. The *bottom-left* panel overlays the raw trajectory with the flagged segment, and the *bottom-right* panel shows the cleaned result after blanking, which restores a smooth and physically plausible curve without distorting the turn geometry. All other Skater-2 sequences contain no flagged points and therefore pass unchanged.

Skater-3 (raw vs GT)

For Skater3, both joints behave well in the raw output, and no outliers are detected. Figure 7.6 plots RA and RW versus ground truth together. Since nothing is flagged, the cleaning stage is a no-op for all Skater-3 sequences.

Alternatives (Kalman/RANSAC) with updated grids

As in Section 6.2, we contrasted the outlier removal and fill the tiny gaps strategy, with Kalman filtering and RANSAC-based polynomial fitting. Hyperparameters were tuned on RW using averaged normalised MPJPE (over all visible frames across evaluation sequences). The Kalman grid (Table 7.5) varied the process noise q and the measurement noise r; the best entry was q=14, r=2 (0.0084). The RANSAC grid (Table 7.6) varied polynomial degree and inlier threshold; the best entry was deg=7, thr=16 px (0.0265). Despite tuning, both filters tended to (i) remove legitimate curvature near extrema (oversmoothing) and (ii) leave true spikes undercorrected or introduce new artefacts. Figure 7.7 illustrates these effects on a Skater-3 RA sequence, corroborating that the robust-z cleaner, with minimal, local edits only where flags occur, remains the most reliable and least invasive choice under Regime B.

Uniform treatment, minimal modification

To keep the pipeline consistent across videos, we run the same robust-z outlier detection combined with the short-gap interpolation policy on all sequences. Since Regime B predictions are already stable,

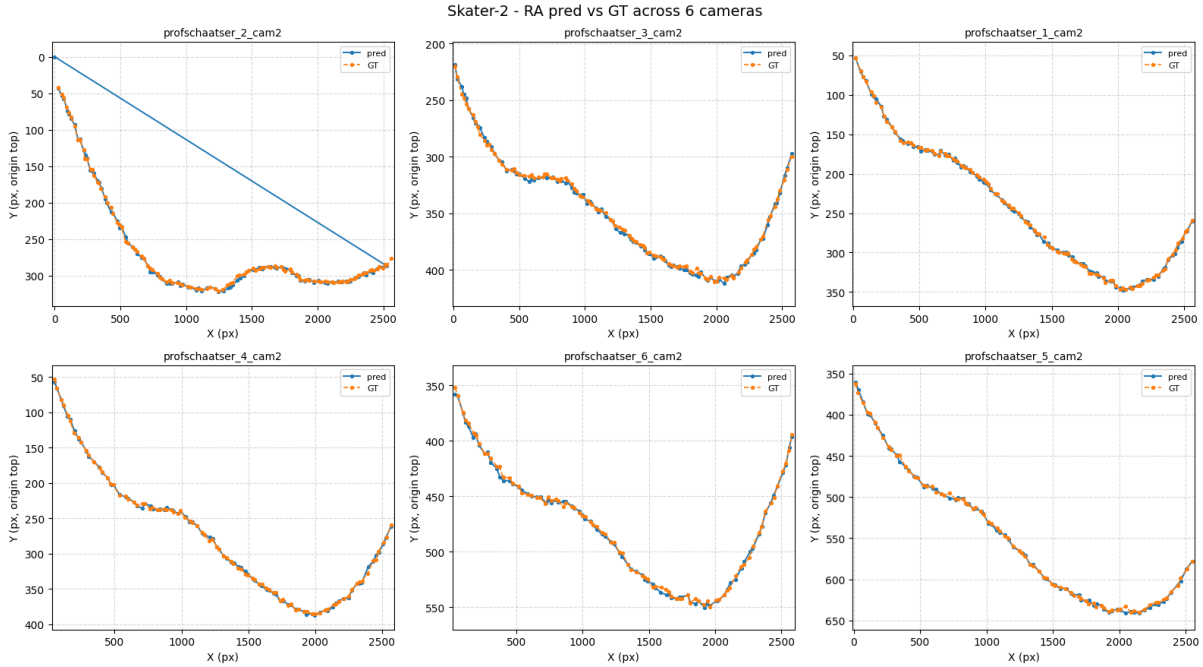

Figure 7.3: Skater-2, RA raw predictions vs. ground truth across six cameras. All cameras are stable except cam 2 (see Fig. 7.5).

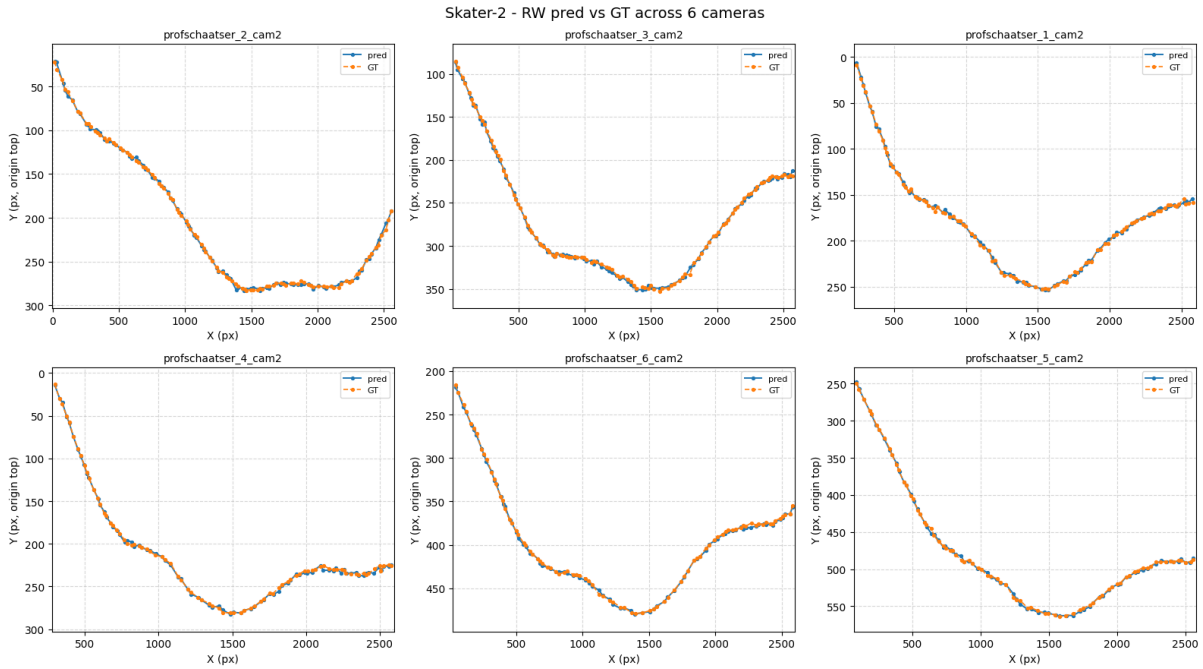

Figure 7.4: Skater-2, RW raw predictions vs. ground truth across six cameras. No points are flagged; cleaned = raw.

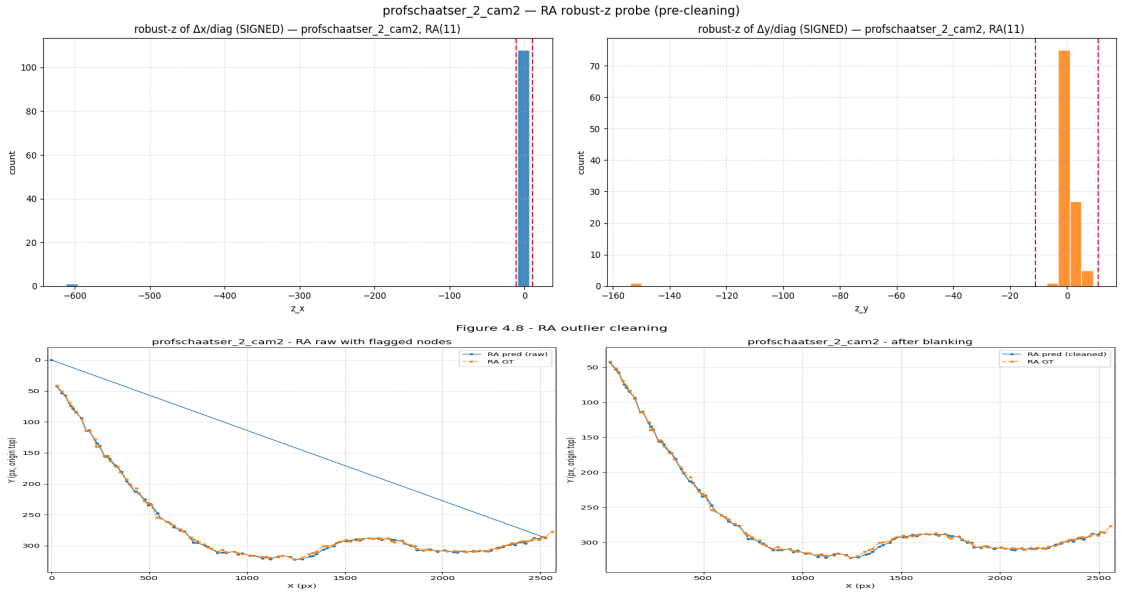

Figure 7.5: Skater-2, RA, camera 2. Robust-z cleaning and effect on the trajectory. *Top:* signed robust-z histograms for $\Delta x/d$ (left) and $\Delta y/d$ (right); dashed lines show the $\pm k$ bands ($k = 11$). *Bottom-left:* raw trajectory with the flagged segment. *Bottom-right:* cleaned trajectory after blanking recovering a smooth, GT-consistent curve.

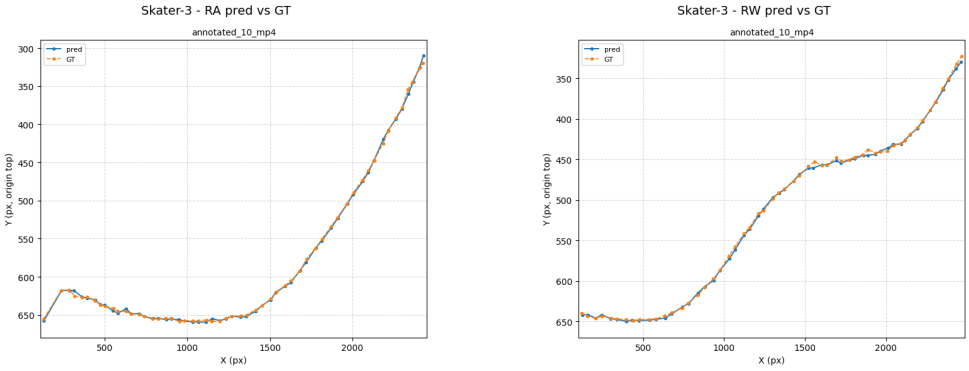

Figure 7.6: Skater-3, RA and RW raw predictions vs. ground truth across the available cameras. No points are flagged; cleaned = raw.

Table 7.5: Kalman grid (RW, v17). Entries are averaged normalised MPJPE over evaluation sequences; lower is better. Best value in **bold**.

q	r=2	r=4	r=9	r=16	r=21
10	0.0084	0.0085	0.0089	0.0095	0.0098
12	0.0084	0.0085	0.0088	0.0093	0.0096
14	**0.0084**	0.0084	0.0087	0.0092	0.0094
16	0.0084	0.0084	0.0087	0.0091	0.0093
18	0.0084	0.0084	0.0086	0.0090	0.0092

Table 7.6: RANSAC grid (RW, v17). Entries are averaged normalised MPJPE; lower is better. Best value in **bold**.

deg	thr=12	thr=14	thr=16	thr=18	thr=20	thr=22
5	0.0442	0.0417	0.0384	0.0374	0.0370	0.0369
6	0.0299	0.0283	0.0276	0.0275	0.0275	0.0276
7	0.0268	0.0265	**0.0265**	0.0266	0.0266	0.0266
8	0.0659	0.0621	0.0615	0.0611	0.0611	0.0605
9	0.1050	0.1088	0.1066	0.1031	0.1005	0.1008

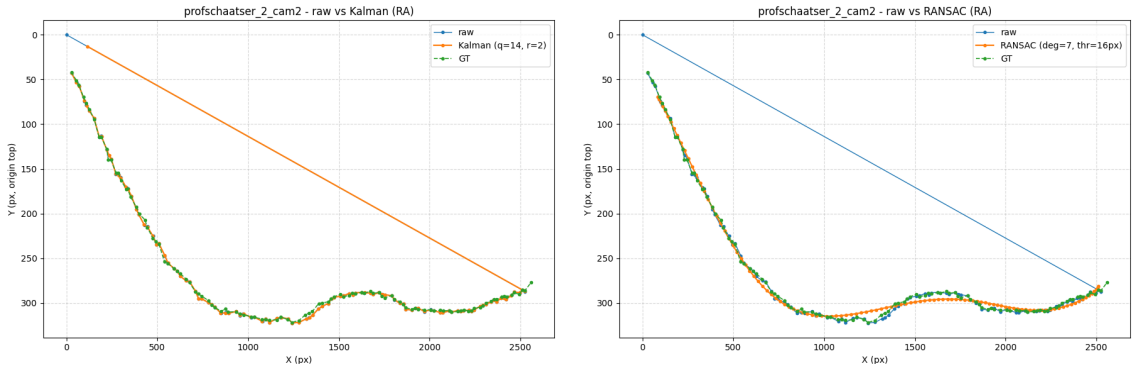

Figure 7.7: Kalman and RANSAC on a Skater-3 RA sequence: both alternatives oversmooth and distort trajectories relative to the robust-z approach, consistent with Section 6.2.

the policy acts as a no-op everywhere except the few frames flagged in Skater-2 (RA, cam 2), where it produces a clear and local improvement.

7.3 Fusion & Phase Detection (Skater-1)

Multi-camera fusion and phase detection are performed only for Skater-1. Skater-2 recordings are cropped differently across videos, invalidating the Skater-1 homographies, and Skater-3 has single-camera coverage that prevents stitching. The procedure mirrors Section 5 and Section 6.3. The per-camera RW/RA keypoints are first cleaned (robust-z followed by short-gap interpolation, performed in the previous section), reprojected via homographies to a common rink frame, and converted to a one-dimensional RW–RA distance signal. The signal is then smoothed with a low-order Savitzky–Golay filter, and cycle boundaries are found with a standard peaks and troughs finder (with the same settings as in Section 5).

7.3.1 RW–RA signal (Regime B predictions)

Figure 7.8 shows the fused RW–RA signal with annotated phase labels and stroke boundaries. The flattened angular domain spans $\Theta_{\text{flat}} \in [0.000, 3.279]$ rad, with the fitted circle centre located at $(-0.38, -1.66)$ in world coordinates.

7.3.2 Phase-level errors

Table 7.7 lists the per-boundary angular errors against the midpoint of the corresponding annotated phase segment. Using the global conversion $\Delta\Theta/\text{frame} = 0.0059$ rad/frame, the mean absolute angular error (MAE) over all boundaries corresponds to ≈ 2.16 frames.

Phase-wise MAE (grouped)

Table 7.8 aggregates the errors by phase type. With the global average of $\Delta\Theta/\text{frame} = 0.0059$ rad/frame, crouched phases remain near or below one frame on average, "Arm Forward" improves vs. Regime A (Section 6.3), and "Arm Backward" is still the hardest segment.

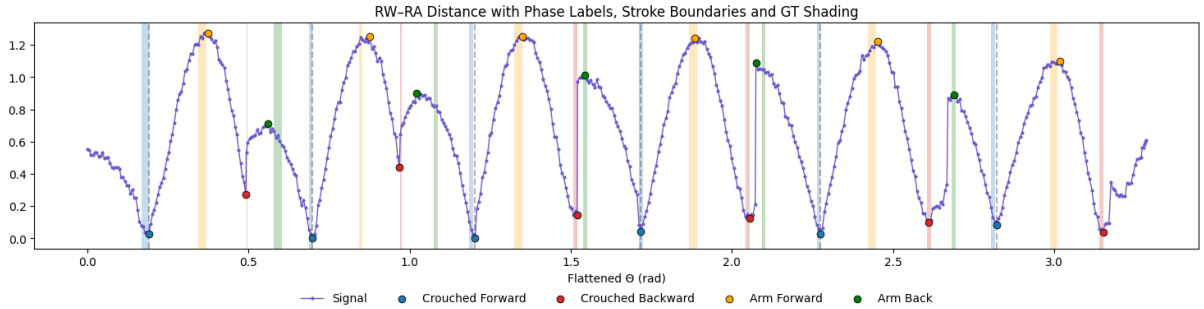

Figure 7.8: Skater-1 (Regime B): fused RW–RA signal with phase labels and ground-truth segment shading. Horizontal axis: flattened angle Θ_{flat} (rad); vertical axis: RW–RA distance (world units). Dot marks detected peaks and troughs.

Table 7.7: Phase-level errors between Regime B predictions and ground-truth for Skater-1 across all cameras. Start and end keys follow `apex-`*`i`*`_`*`frame`* notation. Errors are $|\Theta_{\text{pred}} - \Theta_{\text{GT-mid}}|$.

Phase	**Start Key**	**End Key**	**GT θ (rad)**	**Pred θ (rad)**	**Error (rad)**
Crouched Forward	apex-0_29	apex-0_33	0.180217	0.186111	0.005894
Arm Forward	apex-0_60	apex-0_64	0.357430	0.369542	0.012112
Crouched Backward	apex-0_88	apex-1_1	0.498526	0.493060	0.005465
Arm Backward	apex-1_15	apex-1_19	0.591492	0.544876	0.046616
Crouched Forward	apex-1_34	apex-1_36	0.694414	0.700121	0.005707
Arm Forward	apex-1_61	apex-1_63	0.849382	0.843689	0.005694
Arm Backward	apex-2_19	apex-2_21	1.082568	1.023534	0.059034
Crouched Forward	apex-2_38	apex-2_40	1.192887	1.198579	0.005691
Arm Forward	apex-2_62	apex-2_66	1.339054	1.351073	0.012018
Crouched Backward	apex-2_93	apex-2_95	1.516448	1.510742	0.005705
Arm Backward	apex-3_11	apex-3_13	1.546542	1.534923	0.011619
Crouched Forward	apex-3_40	apex-3_42	1.719794	1.719549	0.000245
Arm Forward	apex-3_66	apex-3_70	1.881453	1.905580	0.024128
Crouched Backward	apex-3_95	apex-3_97	2.050299	2.050699	0.000401
Arm Backward	apex-4_15	apex-4_17	2.106538	2.119318	0.012780
Crouched Forward	apex-4_43	apex-4_45	2.271662	2.283611	0.011949
Arm Forward	apex-4_69	apex-4_73	2.436846	2.461445	0.024600
Crouched Backward	apex-4_99	apex-4_101	2.613177	2.613463	0.000286
Arm Backward	apex-5_29	apex-5_31	2.697353	2.703031	0.005678
Crouched Forward	apex-5_49	apex-5_51	2.813086	2.818984	0.005898
Arm Forward	apex-5_79	apex-5_83	3.001112	2.989291	0.011821
Crouched Backward	apex-5_105	apex-5_107	3.148425	3.142462	0.005963

Table 7.8: Phase-wise MAE for Skater-1 under Regime B. Angular MAE and the equivalent in frames are reported using $\Delta\Theta/\text{frame} = 0.005887$ rad/frame.

Phase	**MAE (rad)**	**MAE (frames)**
Crouched Forward	0.00590	1.002
Crouched Backward	0.00356	0.605
Arm Forward	0.01506	2.558
Arm Backward	0.02715	4.611

Table 7.9: Frame-wise MAE by phase: GT-derived signal (baseline), Regime A predictions, and Regime B predictions. Best (lowest) values are shown in bold.

Phase	GT baseline (frames)	Regime A (frames)	Regime B (frames)
Crouched Forward	1.026	1.323	**1.002**
Crouched Backward	0.818	0.876	**0.605**
Arm Forward	2.851	2.851	**2.558**
Arm Backward	**2.815**	3.834	4.611

Interpretation. The Regime B RW–RA signal reproduces the cyclic segmentation with sub-frame accuracy on crouched phases and improved alignment on *Arm Forward* compared to Regime A. *Arm Backward* remains the most error-prone, with deviations of about 4–5 frames. Two factors explain this pattern: (i) the RW–RA waveform is flatter during arm phases, making peak timing less distinct, and (ii) camera switches often coincide with the backward sweep into *Crouched Forward*, reducing visibility and narrowing the field of view.

Comparison with the GT baseline and Regime A. Table 7.9 contrasts frame-wise MAE across the GT-derived signal (Section 5), Regime A (Section 6.3), and Regime B. The trend is consistent across all three: crouched phases are captured most precisely, arm phases are more ambiguous, and Regime B narrows the error on *Arm Forward* relative to Regime A. Small shifts between GT and model-based signals reflect both annotation variability and the model's ability to regularise timings toward biomechanically plausible extrema. The remaining gap on *Arm Backward* is attributed to the limited visibility at camera transitions.

Finally, note that the GT baseline itself is built from human frame-level keypoint annotations. Small, unbiased annotation noise is expected, and the multi-skater model (Regime B) can occasionally shift peak positions toward biomechanically plausible timings. This explains why, in Table 7.9, Regime B achieves slightly lower MAE than the GT baseline for the crouched phases and *Arm Forward*. The remaining gap on *Arm Backward* aligns with the visibility limitations and the intrinsic flatness of that segment in the RW–RA waveform.

7.4 Interim Conclusion (Regime B)

Regime B demonstrates that pooled multi-skater fine-tuning restores reliable detection and pose accuracy across athletes while keeping the pipeline identical to Regime A. Robust-z cleaning is typically a no-op (with a single local fix for Skater–2, cam 2, RA), and multi-camera fusion with phase detection remains applicable only to Skater–1 due to available homographies. Using the fused RW–RA signal, phase timing achieves a mean absolute angular error of $\approx 0.0127\,\mathrm{rad}$ (≈ 2.16 frames), with *Crouched Forward/Backward* near or below one frame on average and *Arm Forward/Backward* harder (about 2.6 and 4.6 frames, respectively). Overall, Regime B provides a robust configuration carried forward to the final analysis and conclusions.

Chapter 8

Conclusion

8.1 Summary of Findings

This study demonstrated an end-to-end camera-to-phase pipeline for long-track speed skating using ceiling-mounted Apex camera recordings. The pipeline integrates: (i) Mask R-CNN keypoint detection instantiated with a ResNeXt-101 FPN backbone (§3.1), (ii) deterministic main-skater selection (§3.2), (iii) robust temporal refinement (§3.3), (iv) homography-based multi-camera fusion and angular parameterisation (§3.4), and (v) phase detection from a *one-dimensional* cyclic signal (§3.4.3).

Two training regimes were evaluated. **Regime A** (Skater-1 only) validated the pipeline design and reproduced the ground truth phase timings with high fidelity for Skater-1 (crouched phases within ≈1 frame; arm phases ≈3–4 frames; §6.3). However, this single-skater model did not generalise to Skater-2 and Skater-3 (§6.4). **Regime B** (pooled multi-skater training) restored robust detection and keypoint accuracy across athletes while retaining reliable phase timing for Skater-1 (§7.1.3, §7.3). Across regimes, the **RW–RA** distance emerged as the most stable canonical signal for phase timing (§5).

8.2 Key Contributions

This work establishes the feasibility of extracting interpretable stroke phases from Apex camera recordings and contributes methodological advances that form the foundation for future extensions. A complete pipeline was operationalised for overhead corner views, covering skater and keypoint detection through to phase timing. Within this pipeline, a deterministic geometry based logic was developed to isolate the target athlete reliably, even in the presence of distractors and partial entries or exits (§3.2). Temporal refinement was addressed through a staged policy that first removes spikes via robust-z (MAD) on diagonal-normalised steps and then fills only short gaps by linear interpolation, in contrast to Kalman and RANSAC filters, which were shown to oversmooth and distort trajectories (§3.3, §6.2, §7.2). A homography-driven stitching scheme with angular parametrisation was introduced to fuse multi-camera trajectories, trimming overlaps by monotonic angle and yielding a turn-aligned representation (§3.4). In selecting a phase signal, the Right Wrist–Right Ankle (RW–RA) distance was identified as a compact and interpretable one-dimensional waveform for peak–trough timing, robust across both ground-truth annotations and model predictions (§5, §6.3, §7.3). Finally, pooled multi-skater training (Regime B) was shown to deliver a clear generalisation gain over Regime A without altering the downstream pipeline (§7.1.2, §7.1.3).

8.3 Significance and Impact

The results establish that routine **Apex** video can yield frame-level stroke phases using an interpretable, low-dimensional signal derived from a small set of joints. Despite operating as a proof-of-concept with constrained datasets, the pipeline shows: (i) *scalability*, as it can run on standard toolchains and moderate GPUs; (ii) *actionability*, as it produces cycle boundaries that map directly to coaching concepts (e.g., forward/backward arm reach, crouched transitions); and (iii) *extensibility*, as each stage is modular, enabling later substitution with stronger detectors, trackers, or sequence models.

The approach also unlocks downstream analytics once phases are available, such as: *frames (or time) per stroke cycle*, per-phase durations and ratios, cadence (cycles), extremum amplitudes (peak to trough) and asymmetry indices. These summaries can anchor longitudinal monitoring and corner entry and exit comparisons in applied settings.

8.4 Limitations and Future Work

The present study has demonstrated that ceiling-mounted Apex video can be transformed into interpretable stroke-phase information, but it has also revealed areas that invite further development. The dataset covered three skaters at a single rink, introducing some exposure variability but offering limited diversity overall. Valid homographies were available only for Skater-1, which restricted multi-camera fusion and phase detection to that dataset. Expanding recordings to larger cohorts and more varied conditions would strengthen the robustness of the approach and support generalisable evaluation.

The proof of concept also focused on a single high-capacity detector, Mask R-CNN with a ResNeXt-101 FPN backbone [6, 22, 13]. While this choice provided strong baselines, future work should explore alternative model families such as HRNet/MMPose [18, 2], YOLO-NAS Pose [1], RTMPose [7], or DWPose [23], which may balance accuracy and efficiency differently. Because the pipeline is modular, such models can be plugged in with minimal adaptation, enabling systematic benchmarking once richer datasets are available.

Ambiguity remains most evident in the arm phases, where the RW-RA signal exhibits broader and flatter extrema, and camera handoffs often coincide with reduced visibility. These conditions create boundary uncertainty in both annotation and prediction. Addressing this will require either more sophisticated temporal modelling, or the inclusion of additional complementary signals (e.g., joint angles or vertical displacements) to stabilise phase timings.

Together, these considerations do not diminish the significance of the present results. Instead, they underlined that the pipeline provides a solid foundation: one that demonstrates feasibility under constrained conditions and is ready to be extended with more data, a wider range of models, and richer datasets to achieve broader generalisability.

8.5 Applications

Beyond demonstrating technical feasibility, the study opens up possibilities for practical applications in speed-skating analytics. Once stroke phases are automatically extracted, they can be summarised into coach-facing features that extend far beyond qualitative video review. Examples include frames (or time) per cycle, per-phase durations and ratios, stroke cadence, amplitude of arm and leg extensions, and cycle-to-cycle variability.

These features can be surfaced through a lightweight dashboard that visualises key metrics across training sessions. Such a tool would allow coaches to compare cycles across turns, monitor changes over time, and identify asymmetries or inefficiencies with minimal manual effort. By grounding the metrics in biomechanically interpretable phases, the outputs are readily actionable and can complement existing performance monitoring systems.

To illustrate, Tables 8.1 and 8.2 present example outputs derived from the signal constructed using the *ground-truth annotations of Skater-1*. These values are reported in *frames*, assuming a recording frame rate of 90 FPS (0.011 s per frame).

Table 8.1: Global stroke-level summary for Skater-1 (ground-truth signal, assumes 90 FPS). "Frames per Stroke" reports the average number of frames per cycle, "Cadence" is the number of strokes per minute, and "Signal Range" indicates the mean amplitude of the cyclic signal.

Number of Strokes	**Frames per Stroke (mean)**	**Cadence (strokes/min)**	**Signal Range (mean)**
5	90.8	59.5	1.24

These simple summaries already reveal useful insights for coaching. The global view (Table 8.1) shows that Skater-1 maintained an average cycle length of around 91 frames (just over one second), corresponding to a cadence of about 59 strokes per minute. The "Signal Range" represents the mean amplitude

Table 8.2: Phase-level summary for Skater-1 (ground-truth signal, assumes 90 FPS). "Frames (mean/std)" indicate the average and variability in the interval length between successive phase timings, "Fraction of Stroke" reports each interval's share of the full cycle, and "Seconds (mean)" converts the mean duration to time assuming 90 FPS.

Interval	**Frames (mean)**	**Frames (std)**	**Fraction of Stroke (mean)**	**Seconds (mean)**
CF → AF	30.0	2.74	0.33	0.33
AF → CB	25.0	5.43	0.28	0.28
CB → AB	10.0	4.30	0.11	0.11
AB → CF	28.8	4.66	0.32	0.32

of the RW–RA distance within a stroke cycle, that is, the difference between its maximum and minimum values. For coaches, this provides a quantitative baseline for stroke frequency and the size of joint excursions that can be tracked across training sessions.

The phase-level view (Table 8.2) summarises the intervals between successive phase timings. On average, the transition from *Crouched Forward* → *Arm Forward* lasts about 30 frames (0.33 s), while *Arm Forward* → *Crouched Backward* spans about 25 frames (0.28 s). The shortest interval is *Crouched Backward* → *Arm Backward*, around 10 frames (0.11 s), and the return from *Arm Backward* → *Crouched Forward* is about 29 frames (0.32 s). Together, the arm-related intervals ($CF \rightarrow AF$ and $AB \rightarrow CF$) occupy roughly two-thirds of the cycle, while the crouched propulsion intervals make up the remaining third. For a coach, this breakdown highlights whether adequate time is being devoted to the powerful crouched phases, or whether arm swing phases dominate, and also points to variability (e.g., higher standard deviation in AF→CB) that could indicate inconsistency in technique.

Although only a proof-of-concept pipeline has been established, the modular design ensures that additional analytics can be integrated as datasets grow and models improve. The ability to move seamlessly from raw Apex recordings to cycle-level summaries demonstrates how computer vision can augment traditional speed-skating coaching practices, providing a preview of the more comprehensive analytics platforms that could emerge in the near future.

Bibliography

[1] Deci AI. Yolo-nas: Neural architecture search for real-time object detection and pose estimation, 2023.

[2] MMPose Contributors. Mmpose: Openmmlab pose estimation toolbox and benchmark, 2020.

[3] Michael de Rooij. Video-based tracking of a speed skater in the corner. Master's thesis, Leiden University, 2024.

[4] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24 (6):381–395, 1981.

[5] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004. ISBN 9780521540513.

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969. IEEE, 2017.

[7] Tao Jiang, Peng Lu, Li Zhang, Ningsheng Ma, Rui Han, Chengqi Lyu, Yining Li, and Kai Chen. RTMPose: Real-time multi-person pose estimation based on MMPose. *CoRR*, abs/2303.07399, 2023.

[8] Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[9] Yuya Kimura and Toshiharu Yokozawa. Skating techniques of ladies' world-class long-distance speed skaters to shorten curved-section time during the official 3,000 m race. *Frontiers in Sports and Active Living*, 6, 2024.

[10] Arno J. Knobbe, Jac Orie, Nico Hofman, Benjamin van der Burgh, and Ricardo Cachucho. Sports analytics for professional speed skating. *Data Mining and Knowledge Discovery*, 31(6):1872–1902, 2017.

[11] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013.

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[13] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017*, pages 936–944. IEEE Computer Society, 2017.

[14] Zimeng Liu, Meilin Ding, Masen Zhang, Bing Yu, and Hui Liu. Effects of technique asymmetry on 500 m speed skating performance. *Bioengineering*, 11(9):899, 2024.

[15] Banoth Thulasya Naik, Mohammad Farukh Hashmi, and Neeraj Dhanraj Bokde. A comprehensive review of computer vision in sports: Open issues, future trends and research directions. *Applied Sciences*, 12(9):4429, 2022.

[16] Yeong-Je Park, Ji-Yeon Moon, and Eui Chul Lee. Automatic stroke measurement method in speed skating: Analysis of the first 100 m after the start. *Electronics*, 12(22):4651, 2023.

[17] Abraham Savitzky and Marcel J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[18] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5693–5703. IEEE, 2019.

[19] Alexander Toshev and Christian Szegedy. DeepPose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660. IEEE, 2014.

[20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, 2020.

[21] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2, 2019.

[22] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5987–5995. IEEE Computer Society, 2017.

[23] Zhendong Yang, Ailing Zeng, Chun Yuan, and Yu Li. Effective whole-body pose estimation with two-stages distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 4212–4222. IEEE, 2023.

[24] Qihang Zhu, Chunxin Yang, Peng Ke, Han Yang, and Ping Hong. A ground reaction force model of speed skating based on non-contact measurement system. *iScience*, 27(1), 2024.