

3D-BiKE

3D Bicycle Kinematics Estimation
from monocular video

BM51035: BME MSc Thesis

Haye Boonstra



3D-BiKE

3D Bicycle Kinematics Estimation from monocular video

by

Haye Boonstra

5849209

Daily supervisor: B. Gonzalez-Toledo
Main supervisors: J.K. Moore & A. Seth
Project Duration: February, 2026 - July, 2026
Faculty: Faculty of Mechanical Engineering, Delft

Cover: Synthetic bicycle rendering from the 3D-Bike data generator. Annotated by RF-DETR
Style: TU Delft Report Style, with modifications by Daan Zwaneveld

Preface

I want to thanking Benjamin for giving me the chance to do a computer vision based thesis at the bicycle lab. At the start there were some worries regarding the labs ability to support me in this thesis. I took all the feedback to heart and attempted to make the thesis as readable as possible for people without intimate knowledge of computer vision.

The final product turned out better than I expected and I hope the lab will find its capabilities interesting enough to develop further into an actual research grade tool.

I can look back at my thesis quite proudly, it took a lot of self-management skills and dealing with massive amounts of feedback. This experience will surely come in handy once I start writing for my PhD at the UMCG.

*Haye Boonstra
Delft, August 2026*

Statement on AI usage

During the making of this thesis, I made extensive use of AI tooling to help plan, code, and write.

For the planning stage, AI was mostly used to validate ideas, find sources and figuring out if I was on the right track. Most ideas were still my own, but ideas without chance of succeeding were identified and removed from the planning at an early stage. I mainly used ChatGPT's GPT-4o and GPT-5-mini for this stage.

During the coding stage, the Cursor IDE was used with the Pro-plan, giving access to a large amount of usage on Composer 2 and Composer 2.5. These models are specialized for coding tasks and data logistics. I was responsible for the main logic in the code, porting the open source libraries and retargeting to bicycles. AI handled a lot of the data transferring logic between machine learning models and most of the visualizations and table generation from the data acquisition.

The writing stage used a specific ruleset within Cursor that disallowed AI to directly write text in my repository. I used a custom Teacher-Student ruleset, in which AI was always looking over my shoulder during writing and giving direct feedback on the text I wrote. I first made the thesis in bullet points and asked feedback on the contents. I then wrote the thesis in smaller blocks and asked for correctness. Finally, everything was put together using lots of help in structuring the contents to make it as clear as possible. In this stage I used many state-of-the-art models including Claude Opus 4.7/4.8/Fable 5 high thinking models and OpenAI GPT 5.5 medium thinking models.

Summary

Cycling accidents are rising, and most one-sided crashes stem from a loss of control whose cause is rarely understood. The kinematics that would explain such events, such as roll, steering and crank angles in the moments before a crash, are not captured by the, sometimes, available video footage of the crash. This thesis asks whether these kinematics can instead be reconstructed from a single, static, monocular RGB camera.

The core obstacle is data. Unlike human pose estimation, there is no large public dataset of real bicycle motion with three-dimensional ground truth. The work therefore contributes a synthetic data generator that combines MuJoCo rigid-body simulation with photorealistic Blender rendering to produce physically plausible bicycle motion with exact two- and three-dimensional keypoint annotations. Second, it introduces 3D-BiKE, a modular pipeline tool that detects the bicycle with RF-DETR, estimates two-dimensional keypoints with RTMPose, and lifts them to root-relative three-dimensional keypoints with PoseMamba, a neural state-space model, from which roll, steering, and crank angle are extracted. Third, it assesses whether a state-space lifter designed for articulated human motion transfers to the rigid dynamics of a bicycle.

Trained only on synthetic data, the pipeline reaches a mean per-joint position error of 17.6 mm on validation data, better than human pose estimation methods trained on far larger real datasets. The derived roll, steering, and crank angles are recovered to 1.4°, 3.0°, and 5.1° root-mean-square error, substantially improving over the only comparable prior bicycle estimator. These results, though using only synthetic data, establish monocular keypoint lifting trained as a credible foundation for video based cycling safety analysis, with validation on real footage as the essential next step.

Contents

Preface	i
Statement on AI usage	ii
Summary	iii
Nomenclature	vi
1 Introduction	1
1.1 Background and significance	1
1.2 Related work	2
1.3 Research questions and success criteria	2
1.4 Contributions	3
1.5 Thesis outline	4
2 Problem formulation	5
2.1 Kinematic quantities of interest	5
2.2 Semantic keypoints	6
2.2.1 Coordinate frame and root-relative representation	6
2.3 The estimation problem as an inverse problem	7
2.3.1 Depth and scale ambiguity	7
2.3.2 Keypoint error causes angle error	8
2.3.3 The data problem	8
2.4 Summary and scope	8
3 Methods: data generation and learning pipeline	10
3.1 Overview of the pipeline	10
3.2 Synthetic trajectory generation	10
3.3 Rendering and annotations	14
3.4 Dataset generation and structure	14
3.5 Inference and training pipeline	15
3.5.1 Localising the bicycle	16
3.5.2 Estimating two-dimensional keypoints	18
3.5.3 Temporal three-dimensional lifting (PoseMamba)	20
3.6 Kinematics estimation	25
3.6.1 Roll angle estimation	26
3.6.2 Steering angle estimation	27
3.6.3 Crank angle estimation	27
3.6.4 Evaluation metrics for kinematic angles	28
4 Results	29
4.1 Quality of the two-dimensional input	29
4.2 Recovered bicycle kinematics	29
4.2.1 Accuracy of the three angles	29
4.2.2 Consistency across the validation set	30
4.2.3 Agreement and bias	30
4.2.4 Motion frequency recovery	30
4.3 Effect of the training configuration on kinematics accuracy	30
4.4 Keypoint estimation accuracy	33
4.4.1 Detected versus ground-truth two-dimensional input	34
4.5 Qualitative examples	34

5	Discussion	36
5.1	Kinematic estimation	36
5.2	Error origin	37
5.3	PoseMamba variations	38
5.4	Comparison with prior work	39
5.5	Limitations	39
5.6	Future work	40
6	Conclusion	42
	References	43
A	Keypoint definitions and coordinate frames	45
B	Extended quantitative results	47
B.1	Detection and two-dimensional keypoint accuracy	47
B.2	Three-dimensional keypoint diagnostics	47
	B.2.1 Temporal consistency	48
B.3	Kinematics error by model capacity	48
B.4	State-space model interaction maps	48
	B.4.1 Joint–joint spatial relationships	48

Nomenclature

Symbols

Symbol	Definition	Unit
t	Frame index	[–]
j	Keypoint index	[–]
J	Number of semantic keypoints	[–]
T	Temporal window length	[frames]
s	Window stride along a clip	[frames]
r	Root keypoint	[–]
N	Number of frames	[–]
d	feature width	[–]
L	State-space depth	[–]
d_m	Token embedding dimension	[–]
...		
ϕ	Roll angle of the rear frame	[°]
δ	Steering angle of the front assembly	[°]
θ	Crank (pedal) angle	[°]
$\dot{\phi}, \dot{\delta}, \dot{\theta}$	Angular rates of roll, steering, and crank	[°/s]
...		
$\mathbf{X}_{t,j}$	Three-dimensional position of keypoint j in frame t	[m]
$\mathbf{X}'_{t,j}$	Root-relative three-dimensional position	[m]
$\mathbf{u}_{t,j}$	Image-plane location of keypoint j	[px]
$\mathbf{U}$	Bounding-box-normalised two-dimensional input tensor	[–]
I_t	RGB image at frame t	[–]
π	Camera projection onto the image plane	[–]
...		
$h(t)$	Hidden state	[–]
$u(t), y(t)$	Input and output of the state-space model	[–]
A, B, C	State, input, and output matrices	[–]
...		
$\mathcal{L}$	Total training loss	[–]
$\lambda_{3D}, \lambda_N, \lambda_v, \lambda_s$	Weights on MPJPE, N-MPJPE, velocity, and smoothness terms	[–]
$\lambda_\phi, \lambda_\delta, \lambda_{\dot{\phi}}, \lambda_{\dot{\delta}}$	Weights on roll, steering, and their rate losses	[–]
s_t	scale used in N-MPJPE calculation	[–]
$\mathbf{V}_{t,j}$	Inter-frame keypoint velocity	[m]
σ	Standard deviation of added two-dimensional training noise	[–]
...		
r, r_q	Pearson correlation of roll and steer angles	[–]
f	Frequency	[Hz]
$C_{xy}(f)$	Coherence at frequency f	[–]

Introduction

1.1. Background and significance

Cycling is a popular mode of transportation and a healthy activity (Zargarzadeh et al., 2026). However, in the Netherlands the amount of cycling related accidents is on the rise for electric bicycles. According to a report by VeiligheidNL, the number of cycling accidents requiring a visit to the emergency room increased by 100% for electric bicycles in the period 2020-2024 (Krul et al., 2025). The most common type of severe accident, 65% for electric bicycles, is a one-sided accident due to loss of control of the bicycle. No reason is given however why the victims lost control of their bicycle. Understanding why loss of control occurs is crucial for improving safety on bicycles.

Obtaining the bicycle's kinematics during a crash would be ideal for investigating loss of control. Quantities such as roll and steering angle describe how the bicycle–rider system is oriented and controlled in the moments before and during an incident. Together with translational motion, they specify the bicycle's kinematic configuration (Meijaard et al., 2007). That configuration is the input vehicle-dynamics models use to assess whether stability limits were approached, if corrective steering inputs were made, and how the rider interacted with the machine in the pre-crash phase.

In post-incident analysis, this level of detail is rarely available. Police reports and hospital records only document outcomes and injuries, and the perceived cause of the accident by the rider and possible bystanders. Crashes are sometimes recorded though, for example by dashcams, traffic cameras, or smartphones. The footage may be the only remaining trace of the event. Yet, roll and steering cannot be read directly from a single two-dimensional image. Reconstructing these quantities from such recordings therefore requires a method that can infer the three-dimensional configuration of the bicycle from monocular video.

Traditionally, bicycle kinematics are measured in controlled experiments using bicycles fitted with roll and steer sensors, or using optical motion-capture systems that triangulate markers on the frame and rider. Both approaches constrain where and how data can be collected. Roll is particularly difficult to measure in the field because lean angles during normal riding are small, yet roll is central to bicycle stability (Meijaard et al., 2007). Laboratory setups therefore often rely on a ground-referenced frame or sensor cart that measures bicycle roll relative to the road surface. That arrangement is unsuitable for crash analysis: the bicycle may be damaged in the fall or leave the instrumented volume. Motion capture allows the rider and bicycle to move more freely within a capture volume, but still requires a multi-camera installation and a bounded workspace. Neither method yields roll and steering histories for uninstrumented bicycles in uncontrolled, in-the-wild crash footage.

An alternative is to infer the bicycle's three-dimensional configuration directly from monocular video, without instrumenting the bicycle beforehand. In computer vision, much attention has been given to the development of methods capable of estimating the three-dimensional body configuration of humans from a single camera. They are used, for example, in traffic-safety analysis of pedestrians and in clinical gait assessment (Chen et al., 2020). Such methods typically output either anatomical landmarks such

as elbows and knees (Huang et al., 2024; Maji et al., 2022), or kinematic quantities such as joint angles (Bittner et al., 2023). Modern estimators are neural networks trained by matching their predictions to known keypoint locations supplied with each training image or clip (Ionescu et al., 2014; Lin et al., 2014). These datasets are large: COCO alone contains hundreds of thousands of images with two-dimensional person landmarks in everyday scenes, and Human3.6M supplies 3.6 million frames of three-dimensional joint positions. Human pose estimators reach useful kinematic accuracy when that large-scale, high-quality supervision is available. For bicycles, no comparable public resource exists: there is no large dataset of images or monocular video paired with synchronised two and three-dimensional keypoints. Without access to this supervision, the same learning approach cannot be trained or evaluated for bicycle kinematics.

1.2. Related work

Published bicycle vision work has not matched the scale of human pose estimation (Section 1.1), though it does receive some attention. In intelligent transportation research, bicycles are usually treated as vulnerable road users whose global position and orientation must be estimated for behaviour prediction or collision avoidance (Y. Gu & Kamijo, 2014). Six-degree-of-freedom (6D) object-pose methods (Mousavian et al., 2016) would represent a bicycle as a rigid box and estimate its three-dimensional translation and rotation in the camera frame. That formulation is insufficient when handlebar steering or pedal orientation change, because the visible shape of the bicycle and its effective travel direction decouple from a single rigid bounding box. Recent work therefore extends rigid 6D estimation toward articulated bicycle models.

To date, the most direct prior study is the monocular eight-degree-of-freedom (8D) pose estimator of Corral-Soto et al. (2025) for articulated bicycles and cyclists. Their 8D estimation methodology comprises global three-dimensional translation, three body-frame rotation angles, and two articulation angles: steering and pedal rotation, both relative to the bicycle frame. Two parallel prediction networks regress these parameters jointly with 11 keypoints (e.g. wheel centres, handlebars, pedals), using synthetic renderings for three-dimensional supervision and a small set of hand-labelled real images for two-dimensional fine-tuning. On synthetic validation images, global rotation and in-plane translation errors are competitive with similar rigid 6D monocular estimation models, but steering and pedal angles remain difficult: reported mean absolute errors are approximately 25° and 31° respectively. The method operates on single RGB frames, does not model temporal dynamics across a clip, does not target roll angle explicitly, and neither the training data nor the implementation has been released publicly.

Gildea et al. (2024) developed a pipeline that estimates human kinematics during a bicycle crash scenario. They extract rider kinematics from monocular video of cyclist falls and use the reconstructed motion to initialise forward-dynamics human body models for injury analysis. Their pipeline combines deep-learning human pose estimation with a temporally optimised inverse-kinematics refinement, KinePose, that enforces biomechanical joint constraints and temporal consistency on the estimated pose (Gildea et al., 2022). Although the case study concerns a bicycle crash, the bicycle itself is disregarded: only the rider is reconstructed. That omission leaves a gap for loss-of-control analysis, where the bicycle's kinematics are crucial for understanding the cause of the crash. This method does however show that a modular approach utilizing state-of-the-art human pose estimation methods can work well for estimating kinematics. Together with the limitations of existing articulated bicycle pose estimators, prior work therefore still lacks a public video dataset of annotated bicycle three-dimensional keypoints and an evaluation of whether human-pose pipelines can recover bicycle kinematics from monocular video.

1.3. Research questions and success criteria

This thesis addresses the gaps above by recovering local bicycle kinematics from monocular RGB video: roll, steering, and crank angle. A modular approach was taken where root-relative three-dimensional bicycle keypoints are estimated first, and the configuration angles are derived from that skeleton afterwards. Global translation and full body orientation in the scene are deliberately excluded, except for the lean recovered as roll, so that the analysis targets local kinematics rather than full six-degree-of-freedom pose.

Monocular video is a plain sequence of two-dimensional measurements at each instant. A usable three-

dimensional reconstruction must be consistent across frames: neighbouring images should yield compatible keypoint positions, not independent snapshots. Human pose methods therefore analyse entire sequences at once, combining two-dimensional keypoint estimates over time to reconstruct a three-dimensional configuration. A recent class of networks known as neural state-space models carries out this temporal step with a recurrence inspired by dynamical systems: an internal state is propagated forward and updated as new image information arrives, in the same spirit as state estimation. Because bicycle motion is comparatively smooth and constrained by a rigid frame, such a state-based temporal model is a natural candidate for the lifting stage.

The end product of the pipeline is the recovered angles, not the keypoints themselves. However, no established bicycle benchmark defines a pass/fail threshold on roll or steering for this setting, and no directly comparable method reports local roll at all. Success is therefore judged primarily by how well the recovered roll, steering, and crank tracks match simulation ground truth on the validation set of synthetic clips. Mean per-joint position error (MPJPE), the pose estimation standard accuracy metric, is retained as a diagnostic of intermediate keypoint quality and for comparison with the human three-dimensional pose literature. State-of-the-art human three-dimensional pose estimation reaches roughly 35 mm MPJPE on large real datasets (Huang et al., 2024). Approaching that value on the bicycle validation set would indicate a promising intermediate keypoint reconstruction relative to that literature. Approaching or exceeding this value would suggest that the methods transitioned well from the human to the bicycle domain.

On that basis, the central research question is:

Can a temporal state-space keypoint estimation pipeline, trained on synthetic video clips, recover roll, steering, and crank angle from monocular RGB video of a bicycle with accuracy useful for kinematics analysis, when evaluated against simulation ground truth on a held-out validation set?

Success is judged as follows.

1. **Primary outcomes.** Roll, steering, and crank angles derived from the estimated keypoints are compared with simulation ground truth on the validation set using Root-Mean-Square Error (RMSE). No single pass/fail threshold is imposed on the angles, because acceptable error depends on the intended use.
2. **Secondary measure.** MPJPE on the same validation clips. Values near or below the approximate 35 mm indicate good transfer from humans to bicycle pose estimation.

1.4. Contributions

This thesis makes the following contributions:

1. **The 3D-BiKE (3D Bicycle Kinematics Estimation) pipeline.** A modular chain from monocular RGB video to local bicycle kinematics: a bounding-box detector localises the bicycle, a two-dimensional keypoint estimator recovers a fixed set of bicycle landmarks on the bicycle image, and a temporal state-space lifter reconstructs root-relative three-dimensional keypoints over an image sequence. Roll, steering, and crank angle are then obtained by post-processing algorithms on the keypoint set, so the end product is configuration over time rather than keypoints alone. Training and inference both use detected, bounding-box-normalised two-dimensional input so that evaluation matches real-world usage of the pipeline. The model choices and settings are given in Chapter 3.
2. **A synthetic bicycle data generator for supervised training.** A MuJoCo–Blender pipeline produces photoreal images and image sequences of bicycle trajectories with aligned labels: bounding boxes, two- and three-dimensional keypoints, and the corresponding configuration angles. The generator supplies either single images or temporal clips, closing the annotation gap identified in prior work. Because trajectories and labels are produced automatically, the tool makes it practical to generate large training and validation sets without manual annotation.
3. **An evaluation of neural state-space models for estimating bicycle kinematics.** An assessment of whether a state-space lifter designed for human video can estimate keypoints well enough

to be able to extract useful roll, steering, and crank angles on image sequences, with MPJPE retained as a diagnostic of intermediate keypoint quality. Roll, steering, and crank are recovered with three geometric extraction algorithms developed for this work and applied identically to predicted and ground-truth keypoints.

Training and evaluation in this thesis rely on synthetic data. Generalisation to real monocular footage remains open and is discussed as a limitation in Chapter 5. The trained 3D-BiKE pipeline will be made publicly available on the Bicycle Lab GitHub alongside the synthetic dataset generator and the datasets used in the thesis.

1.5. Thesis outline

In this thesis, the development of the 3D-BiKE pipeline from the initial idea and dynamics foundation to the final results is presented in the following chapters. Chapter 2 introduces the dynamics foundation of the thesis and the problem formulation. Chapter 3 presents the models used in the pipeline, the synthetic dataset generation and the inference and training pipeline. Chapter 4 presents the performance metrics of the pipeline and the in depth analysis of the SSM based 3D keypoint estimator. Chapter 5 interprets the results and the limitations of the thesis. Chapter 6 concludes the thesis and presents the future work.

Additional explanations and results are presented in the appendices.

2

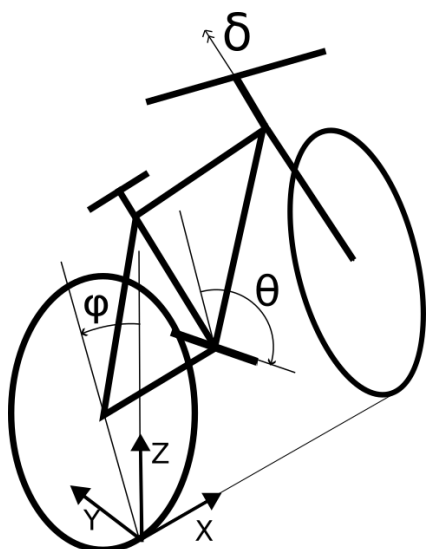
Problem formulation

The introduction motivated why bicycle kinematics are worth recovering from ordinary video. This chapter states what is estimated, from what, and why the estimation is difficult. The aim is a clear description of the problem rather than a description of the solution.

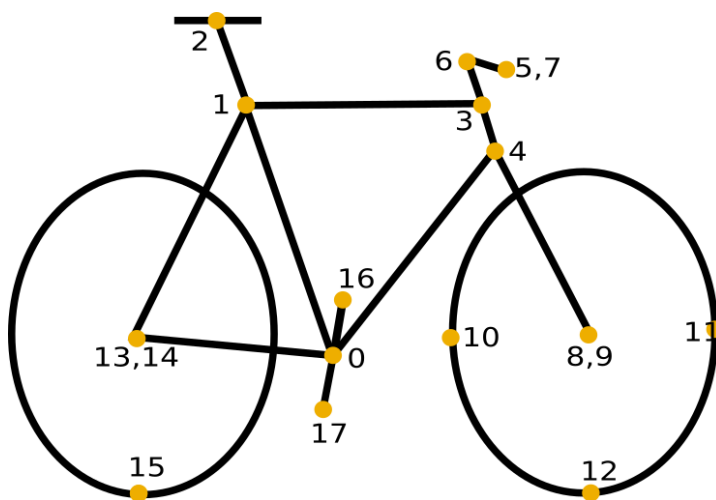
2.1. Kinematic quantities of interest

The objective of this thesis is not to reconstruct the visual appearance of a bicycle, but to estimate its mechanical configuration over time. For vehicle-dynamics applications, the quantities of interest in local frame are a small number of configuration variables that describe how the bicycle is oriented and controlled. Following the work by Meijaard et al. (2007), with the addition of the crank angle, three variables are targeted here (Figure 2.1a):

- the *roll angle* ϕ , the lean of the rear frame about its longitudinal axis relative to the vertical (equivalently the road normal under the flat-road assumption);
- the *steering angle* δ , the rotation of the front assembly about the steering axis relative to the midplane of the rear frame;
- the *pedal (crank) angle* θ , the rotation of the crank about the bottom-bracket axis, with right-pedal-up defined as zero.



(a) Kinematic quantities of interest on the bicycle estimated in the 3D-BIKE pipeline: roll ϕ , steering δ , and crank θ .



(b) Semantic bicycle keypoint set. Left-right pairs that overlap in this view are labelled together.

These configuration variables describe the bicycle as a mechanical system and are more useful for model validation, motion analysis, and comparison with conventional sensor measurements than raw three-dimensional coordinates. The sign conventions and the algorithms used to extract these angles are fixed consistently across dataset generation, training, and evaluation, so that a single definition is used throughout the pipeline.

Crucially, this work does not regress these configuration variables directly from images. Instead it estimates the three-dimensional positions of a fixed set of points on the bicycle, and the angles are computed from that set afterwards. Some human-motion studies regress joint kinematics directly from video without an explicit three-dimensional intermediate representation (Bittner et al., 2023), and the articulated bicycle estimator of Corral-Soto et al. (2025) likewise predicts steering and pedal angles in a single-frame regression. A direct angle regressor with physical constraints on the estimates and their rates can be attractive when the only required outputs are ϕ , δ , and θ . The keypoint intermediate is retained here for being able to report keypoint estimation accuracy in the same terms as the human pose estimation literature. The trade-off is that physical constraints are not enforced at inference: the angles inherit whatever inconsistency remains in the estimated keypoints.

2.2. Semantic keypoints

The interface between the computer-vision pipeline and the kinematic interpretation is a set of semantic keypoints. A semantic keypoint is a labelled point with a fixed physical meaning on the bicycle, for example a rear hub, the bottom bracket, or a wheel ground-contact point. This work uses $J = 18$ such keypoints (Figure 2.1b). Their placement was chosen so that the midplane of the rear frame can be formed reliably from the landmarks, and so that the keypoints used for estimating the steering angle span as large a distance as possible, which theoretically reduces the sensitivity of the derived angles to keypoint noise. Indices in the figure match the full name and description list in Appendix A. In side view, left–right pairs that coincide (handlebar ends, front hubs, rear hubs) share a single marker and are labelled jointly.

Keypoints are used in preference to dense mesh reconstruction or CAD-model fitting for four reasons. First, a sparse keypoint set is compatible with established human pose estimation tooling (COCO Consortium, n.d.; Lin et al., 2014), in which a two-dimensional detector is followed by a three-dimensional lifting model, and this thesis deliberately reuses that existing tooling. Second, sparse keypoint labels can be generated exactly and cheaply from the simulation-and-rendering pipeline, whereas dense correspondence labels cannot. Third, a carefully chosen sparse set is sufficient to constrain the wheel planes, frame orientation, and steering assembly that the roll and steering angles depend on. Fourth, CAD-model fitting requires a parametric model, such as SMPL for humans, that does not exist yet for bicycles.

2.2.1. Coordinate frame and root-relative representation

The keypoints are expressed in camera coordinates, which aligns with the intended use case: a static monocular camera observes the scene, and the estimated three-dimensional configuration is naturally expressed relative to that camera. The learning target, however, is root-relative. A single reference keypoint r on the rigid rear frame is subtracted from every joint in each frame,

$$\mathbf{X}'_{t,j} = \mathbf{X}_{t,j} - \mathbf{X}_{t,r}, \quad j = 1, \dots, J, \quad (2.1)$$

where $\mathbf{X}_{t,j} \in \mathbb{R}^3$ is the position of keypoint j in frame t . This removes the global translation of the bicycle in the scene, which is only weakly observable from a single camera, while preserving the relative geometry from which roll and steering are later computed.

The reference keypoint r is chosen to be the bottom bracket. It lies on the rigid rear frame, is near the centre of the machine, and is visible in most views, so the subtraction in Eq. (2.1) is well defined for almost every frame. A ground-contact point is the more common root in bicycle dynamics, where it conveniently places the bicycle on the road. For image-based estimation that choice fails: the contact point is frequently occluded, does not have clear intersections with other frame geometry, and often lies near the noisy edge of the bounding box. The steering assembly is equally unsuitable because it moves relative to the rear frame.

Roll, steering, and crank are obtained from relative vectors and planes formed by the chosen keypoints, not from the absolute position of any point in the world. Steering and crank are intrinsic to the bicycle: they measure relative rotation between assemblies, so the same geometric definitions apply in the camera frame and in the simulation. Roll is different. In the classical bicycle model, ϕ is lean relative to the global vertical vector (Meijaard et al., 2007). Because the raw keypoint output here is expressed in camera coordinates following the OpenCV coordinates (X right, Y down, Z forward along the optical axis) (Bradski, 2000), the roll recovered from them is a local lean relative to the camera, not necessarily the absolute roll stored in MuJoCo. The extraction algorithm and that distinction are detailed in Chapter 3.

2.3. The estimation problem as an inverse problem

The relationship between a bicycle’s configuration and its image can be written as a forward chain. Let $\mathbf{q}_t$ collect the configuration of the bicycle in frame t (the variables of Section 2.1 together with the global pose that is later discarded). The chain proceeds in three stages:

$$\mathbf{q}_t \xrightarrow{g} \mathbf{X}_t \xrightarrow{\pi} \mathbf{u}_t \xrightarrow{R} I_t. \quad (2.2)$$

Here g is the rigid-body forward kinematics that maps a bicycle configuration $\mathbf{q}_t$ to the three-dimensional keypoint positions $\mathbf{X}_t = \{\mathbf{X}_{t,j}\}$. π is the camera projection that maps each three-dimensional keypoint to its image-plane location $\mathbf{u}_{t,j} \in \mathbb{R}^2$; and R is the rendering process that turns the scene into pixels I_t subject to appearance, lighting, and occlusion. Each stage loses information: π discards depth, and R entangles the keypoints with everything else in the scene. Estimating bicycle kinematics from video is the inversion of Eq. (2.2). The pipeline of this thesis performs the inversion in the same staged manner, which can be summarised as

$$I_{1:T} \xrightarrow{f} \hat{\mathbf{X}}_{1:T} \xrightarrow{h} (\hat{\phi}_{1:T}, \hat{\delta}_{1:T}, \hat{\theta}_{1:T}), \quad (2.3)$$

where $I_{1:T}$ is a monocular RGB image sequence, $\hat{\mathbf{X}}_{1:T}$ is the estimated sequence of root-relative three-dimensional semantic keypoints, and $\hat{\phi}_{1:T}, \hat{\delta}_{1:T}, \hat{\theta}_{1:T}$ are the derived configuration variables. The two maps in Eq. (2.3) are fundamentally different in character. The map h is a deterministic extraction: given correct keypoints it returns the angles exactly, by construction. The map f is the learned inverse of the appearance-and-projection part of Eq. (2.2). It must undo a process that is not invertible in the strict sense, and it is therefore the cause of all of the difficulty. The thesis thus reduces to a single hard question: how accurately can f be learned to recover the configuration variables from the estimated keypoints?

Four properties of the problem make the learned inverse f in Eq. (2.3) difficult. The first three are intrinsic to recovering three-dimensional geometry from a single moving camera; the fourth is specific to bicycles and is the practical bottleneck of this thesis.

2.3.1. Depth and scale ambiguity

The projection π in Eq. (2.2) maps three-dimensional points onto a two-dimensional image, and therefore throws away information. The quantity that is lost is depth: how far each point lies from the camera. Because of this, different three-dimensional keypoint configurations $\mathbf{X}$ can project to exactly the same two-dimensional detections $\mathbf{u}$ on the image grid. Geometrically this follows from the pinhole camera model: every point along a ray from the camera centre through a given pixel projects to that same pixel, so moving a keypoint along the viewing direction leaves its image location unchanged. If several keypoints can each slide in depth without altering their projected positions, then several distinct bicycle poses are compatible with one and the same two-dimensional observation. Recovering $\mathbf{X}$ from $\mathbf{u}$ is therefore under-determined from one frame alone, and a useful solution requires prior knowledge about which configurations and motions are plausible.

A second, closely related ambiguity is absolute scale also due to the pinhole camera problem. A small bicycle close to the camera and a large bicycle far away can produce the same image, so the true size of the bicycle cannot be read from a single still frame. The root-relative target of Eq. (2.1) removes the need to recover the absolute position of the bicycle in the camera frame, by expressing all keypoints

relative to the bottom bracket. It does not resolve the underlying ambiguities: the relative depth of each keypoint along the viewing direction, and the overall metric scale of the reconstructed skeleton, still have to be inferred from priors rather than read off the image.

2.3.2. Keypoint error causes angle error

The extraction map h in Eq. (2.3) is exact: given perfect keypoints it returns the true angles by construction. Its weakness is that it has no way to reject error. Any noise in the estimated keypoints passes directly into the angles, and the geometry of the bicycle determines how strongly it is amplified on the way.

For example, roll is derived from the direction of a roughly vertical vector $\mathbf{v} = \mathbf{X}_a - \mathbf{X}_b$ between two keypoints on the rear frame, separated by a distance $L = \|\mathbf{v}\|$, called the baseline. If one endpoint is displaced sideways by a small error ε perpendicular to $\mathbf{v}$, the vector rotates by approximately

$$\Delta\phi \approx \frac{\varepsilon}{L}, \quad (2.4)$$

using the small-angle approximation $\tan \Delta\phi \approx \Delta\phi$. The angle error therefore grows in direct proportion to the keypoint error and in inverse proportion to the baseline: the same 40 mm of keypoint error produces twice the angle error when the baseline is halved. Substituting representative numbers, a keypoint error of 40 mm on a half-metre frame baseline gives $\Delta\phi \approx 0.08$ rad, or about 4.6° .

This sensitivity forms the basis for the accuracy target of the research question. For the derived roll and steering angles to be meaningful at all, the keypoint error budget must stay in the range of a few tens of millimetres.

2.3.3. The data problem

Section 2.3.1 established that f can only succeed by relying on a strong prior over plausible bicycle configurations and their motion. In human pose estimation that prior is learned from large public datasets that pair real images with three-dimensional ground truth from motion capture, together with very large two-dimensional keypoint corpora (Ionescu et al., 2014; Lin et al., 2014). No equivalent resource exists for bicycles.

What is scarce is therefore not “data” in general, and not the ability to render images: it is real monocular video paired with synchronised three-dimensional semantic keypoints and with roll, steering, and crank at the scale needed to train and evaluate a three-dimensional lifter, together with broad coverage of bicycle geometries, riders, and scenes. The conventional response is to collect such a dataset in the field, which is impractical here precisely because capturing bicycle keypoint and kinematics data is incredibly difficult as explained in the introduction (Chapter 1).

The data gap is therefore part of the problem formulation. Because the lifter cannot be trained on real annotated bicycle video at that scale, the prior must be supplied synthetically: motion is simulated, rendered as standard red-green-blue (RGB) video, and exported with exact paired two- and three-dimensional annotations. Generating ground truth on demand closes the annotation gap, but it does not by itself guarantee that the learned prior is realistic compared to physical bicycle behaviour.

The temporal dimension matters here. Because the final estimator uses temporal context to resolve the per-frame ambiguities of Section 2.3.1, the synthetic sequences must contain plausible frame-to-frame evolution of roll, steering, and crank not merely visually varied still images. A dataset of uncorrelated random poses would teach f an unrealistic temporal prior and would sever the link between the estimated keypoints and recoverable bicycle kinematics. The inverse problem therefore imposes a requirement on the generator: the synthetic motion must be physically plausible as well as photoreal. How that requirement is approached in MuJoCo (controller design, fall-over filtering, and the limits of those checks) is described in Chapter 3; full validation of the simulator against real-world bicycle dynamics is outside the present scope and remains a limitation.

2.4. Summary and scope

In short, the thesis inverts Eq. (2.2) through a learned keypoint estimator f and a fixed angle extractor h . Depth and scale are ambiguous from one camera, angle extraction amplifies keypoint error, and the

motion prior has to come from synthetic data. Scope is limited to root-relative local kinematics of one bicycle under a static monocular camera and a flat-road assumption. The next chapter describes the pipeline that implements this inversion.

Methods: data generation and learning pipeline

3.1. Overview of the pipeline

This thesis attempts to solve the inversion of Chapter 2 in two parts: a synthetic generator that supplies labelled training data for training the required models, and an estimation pipeline that utilizes these trained models to recover local bicycle kinematics from monocular RGB video. The estimation side is split into four subproblems:

1. localise the bicycle in the image;
2. estimate two-dimensional semantic keypoints on the focused bicycle;
3. lift those landmarks over a temporal window to root-relative three-dimensional keypoints;
4. recover roll, steering, and crank by geometry from the three-dimensional keypoints.

That decomposition follows the standard process used in human pose estimation: bounding box, then two-dimensional keypoints, then a lift to three dimensions. It is reused because each stage is independently replaceable and evaluable, and because proven tooling already exists for these stages. What is bicycle-specific is the landmark set (Figure 2.1b), the synthetic training motion, the lifter trained on bicycle clips, and the local angle extraction algorithms. Chosen model and their corresponding settings for each stage follow below; Figure 3.1 summarises the overall estimation pipeline graphically.

3.2. Synthetic trajectory generation

To train the models for the estimation pipeline, the data problem needs to supply temporally coherent bicycle motion with exact configuration ground truth (Section 2.3.3). Synthetic trajectories are generated in the MuJoCo rigid-body dynamics simulator (Todorov et al., 2012). MuJoCo is used so that motion is smooth across frames by imposing dynamical constraints rather than a sequence of unrelated poses. The three-dimensional lifter is trained on clips, so unrealistic jumps in position would corrupt the temporal prior it learns. The same physically driven poses are also assumed to help single-image training. By exposing the two-dimensional detector to configurations that arise from continuous motion, the training data distribution also contains more data of situations that are most likely to occur.

A bicycle model was built in the standard model format used by MuJoCo with dimensions and inertial properties of a standard city bicycle (Figure 3.2). In human pose estimation the root joint is often the pelvis, chosen for its central location and robustness under occlusion. In bicycle modelling the rear-wheel contact point is a common road reference, but for image-based estimation that point is frequently occluded and difficult to infer due to being isolated from other well-defined geometry. The bottom bracket is therefore used as the root: it lies near the centre of the rigid rear frame and has clear intersections with the frame tubes.

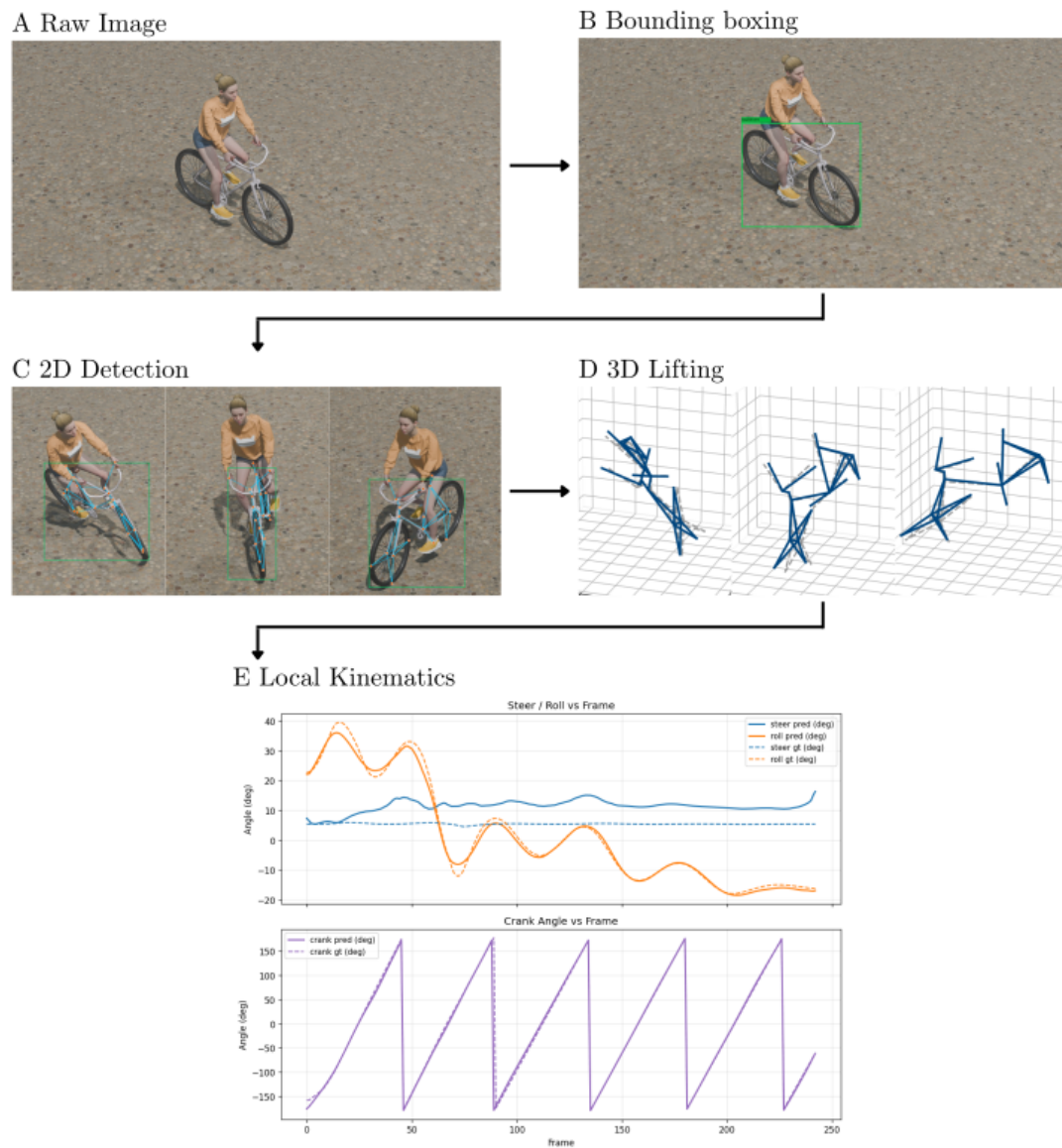


Figure 3.1: The 3D-BiKE pipeline from raw image to bicycle kinematics

The simulated bicycle follows a randomised yaw direction along time built from a library of manoeuvres typical of ordinary riding (Figure 3.3):

- Straight riding
- Smooth turn
- Circle
- Zigzag
- Avoidance
- Lane change
- Figure eight
- Accelerate
- Decelerate
- Coast

Manoeuvres are randomly sampled and concatenated into a complete trajectory (Figure 3.3). The heading change is set by the manoeuvre type, while the time allowed to execute it is randomised, which varies turning rates across clips.

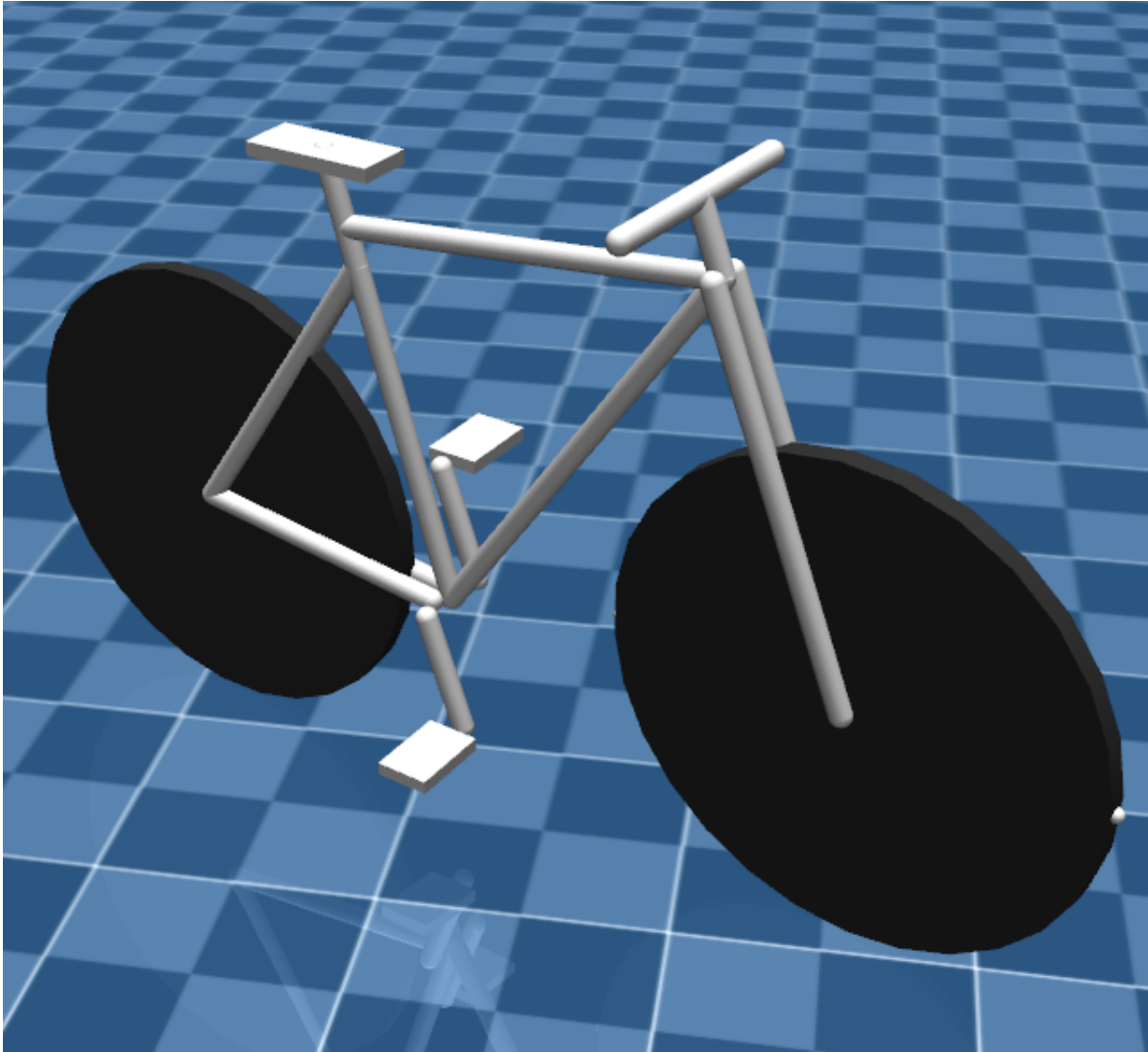
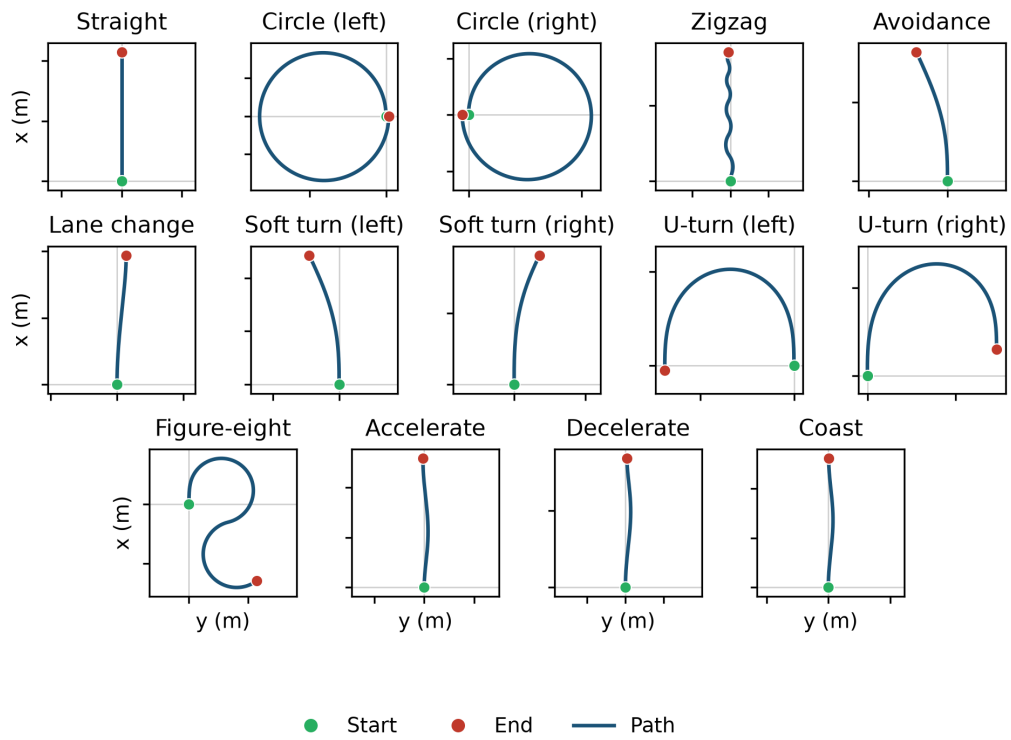
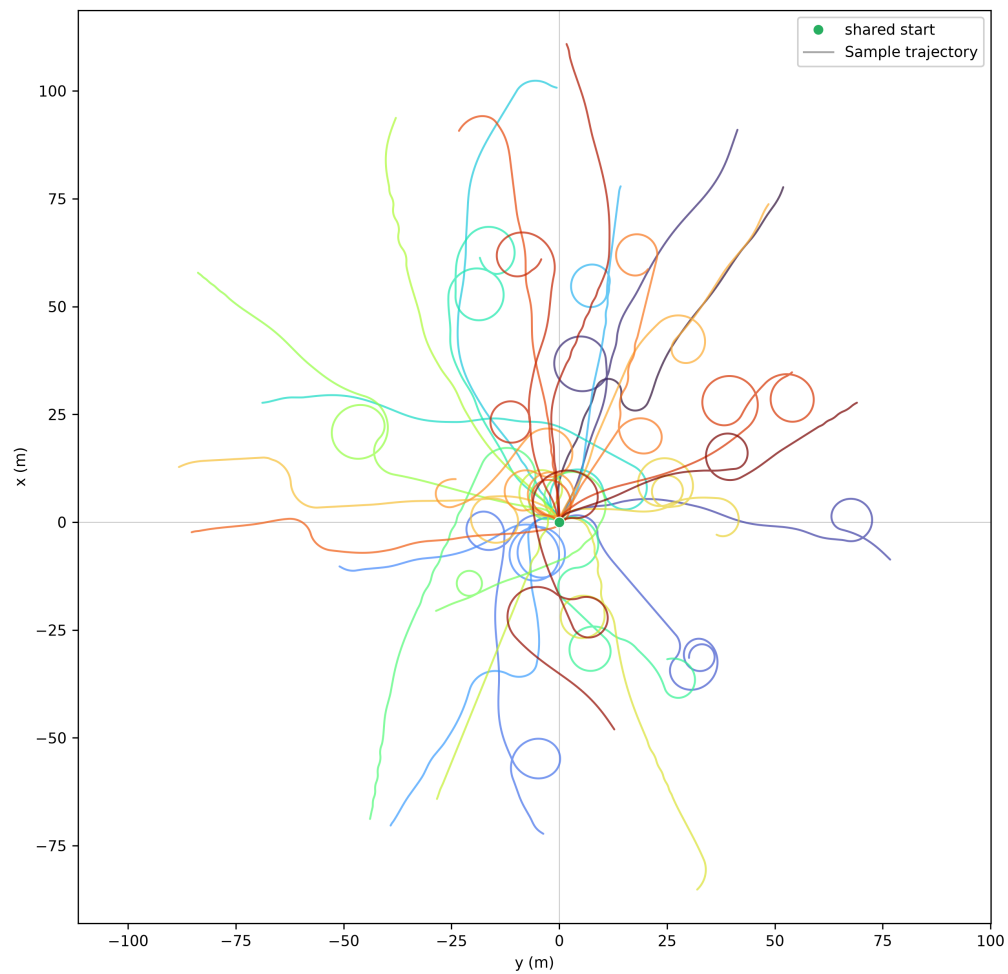


Figure 3.2: The bicycle model in MuJoCo.



(a) Top-down view of each trajectory pattern used to build trajectories.

Randomized full trajectories

(b) Example randomised full trajectories formed by concatenating sampled manoeuvres from a shared start.

Figure 3.3: Trajectory generation: individual manoeuvre patterns (top) and randomised concatenations used for simulation (bottom).

Tracking of the yaw trajectory uses two PID controllers: a forward-velocity controller that applies torque to the rear wheel to hold the speed set by the trajectory, and a heading controller that applies a roll torque to the frame to lean the bicycle into a turn. A steering-angle controller was considered but rejected as it was too complex for this use case, because it would require a difficult to tune two-loop controller. The goal of this controller is to allow smooth trajectory generation, not completely accurate bicycle control. The currently implemented, simpler, heading controller has the side benefit that it can apply a restoring torque when roll grows too large, which improves stability during aggressive manoeuvres. Pedal motion is not independently controlled: a rotational coupling between the rear wheel and the crank, with a randomised gear ratio, drives the crank angle.

Stable motion is ensured by two closed-loop controllers for the speed and heading instead and an algorithm assertion that discards any trial whose roll exceeds a fall-over threshold. Heading is actuated by a roll torque applied to the bottom bracket rather than by counter-steer as on a real bicycle, the rider is not part of the physics, and crank cadence is tied to wheel speed. A full comparison of the simulated motion to real-world bicycle dynamics is outside the present scope (Section 2.3.3).

Trajectories are simulated at 200 physics steps per second and record the following generalised coordinates: a seven-dimensional floating-base pose (Cartesian XYZ position and orientation quaternion) plus rear- and front-wheel rotation, pedal angle, and steering angle. The output generalized coordinates are resampled to 60 steps per second using linear interpolation for use in Blender.

3.3. Rendering and annotations

MuJoCo trajectories are transferred onto four different bicycle models in Blender, with each time step stored as a keyframe. Linear keyframe interpolation is used so that Blender does not alter the physics-driven motion. Six Blender-ready riders with varied appearance and gender obtained from Mixamo (Adobe, 2026) are attached to the bicycle and animated using inverse kinematics. The riders were not part of the simulation but were added after the simulation to add appearance diversity and occlusion of the keypoints on the bicycle.

Scene appearance is varied with publicly available High Dynamic Range Image backgrounds. These backgrounds contain 360 degree photos of real-world locations with accurate lighting information for accurate shadows in the Blender rendering. Lighting strength is randomised to mimic different times of day, and fog with randomised density is added to produce lower-contrast images. This visual diversity is aimed mainly at the two-dimensional keypoint detector, which must generalise across scenes and lighting.

Two camera modes are used because the two training tasks impose different geometric constraints. In fixed mode, used for static images for the two-dimensional detector, the camera is placed at a random point on a sphere of radius 8–14 m around the bicycle at one instant of the trajectory. Elevation is drawn from a Gaussian distribution about the horizontal plane with standard deviation 20° , the polar angle is uniform from 0 to 360° , and the camera is aimed at the middle of the handlebars. A ray-cast occlusion check rejects blocked views and resamples until the bicycle is visible. That placement gives useful depth and scale variation for the detector, but a fixed camera that must cover a long path sits far away and the bicycle often leaves the camera view, impractical for high-quality long clips. Tracking mode solves this: the camera is again placed on a sphere around the bicycle but then follows the same path, keeping the subject relatively large and centered for temporal windowing of the three-dimensional lifter. Because the lifter consumes bounding-box-normalised two-dimensional coordinates, absolute depth is less critical at that stage than for the two-dimensional detector.

Rendered frames are exported as PNG images at 1920×1080 pixels. For each frame the pipeline stores semantic keypoints in image coordinates and in OpenCV camera-space three dimensions, together with ground-truth bounding boxes so the same generator can feed detector training and the later lifter dataset.

3.4. Dataset generation and structure

The renders above are organised into two separate datasets, one for each learning stage.

Static-image dataset This set contains 10 000 images rendered with the fixed-camera protocol above, labelled with bounding boxes and two-dimensional keypoints, and split 8000/1000/1000 for training/validation/test. Assets include four bicycle frames, manually modified and rigged in Blender to match the MuJoCo degrees of freedom, and six unique riders. Scene appearance, weather, lighting, frame, and rider vary as in the previous section.

Clip dataset This set preserves sequence structure: 43 unique tracking-camera clips of 2189 frames at 60 fps (94 101 unique frames), each with randomised trajectories. Clips store two-dimensional keypoints, confidences, bounding boxes, and root-relative three-dimensional keypoint targets. The lifter does not consume a full 2189-frame clip at once. Instead, each clip is cut into shorter windows of T consecutive frames; T is the temporal context length of the model (default $T = 243$). Windows are advanced along the clip by a “data” stride s : the next window starts s frames after the previous one. For training, $s \approx T/3$ (default $s = 81$), so consecutive windows overlap by about two thirds of their length and the same motion is seen in several slightly shifted sequences. For the validation and test parts of the dataset the stride is set to $s = T$, so windows do not overlap and each frame is scored once. Table 3.2 lists how many windows result for each (T, s) configuration discussed later.

The default lifter training set uses detected two-dimensional inputs: bounding boxes from RF-DETR and keypoints from RTMPose, normalised by the detected box so that training matches real world use of the pipeline. A parallel set with ground-truth projected two-dimensional keypoints is kept only to isolate how much three-dimensional error comes from the lifter itself when upstream detection noise is removed.

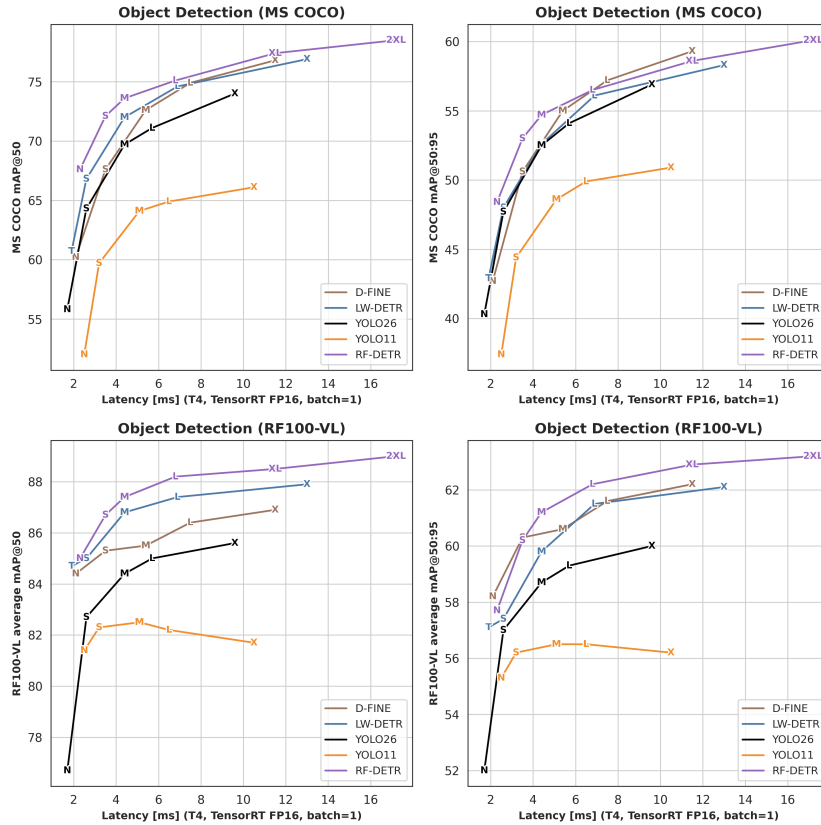


Figure 3.4: RF-DETR performance on the COCO dataset (Robinson et al., 2025).

3.5. Inference and training pipeline

The following subsections cover the first three estimation pipeline subproblems: localise the bicycle, estimate two-dimensional landmarks, then lift them over time. Each subsection states the subproblem, the method class, and the chosen model before going into architecture and settings. Landmark

definitions are those of Figure 2.1b and Appendix A.

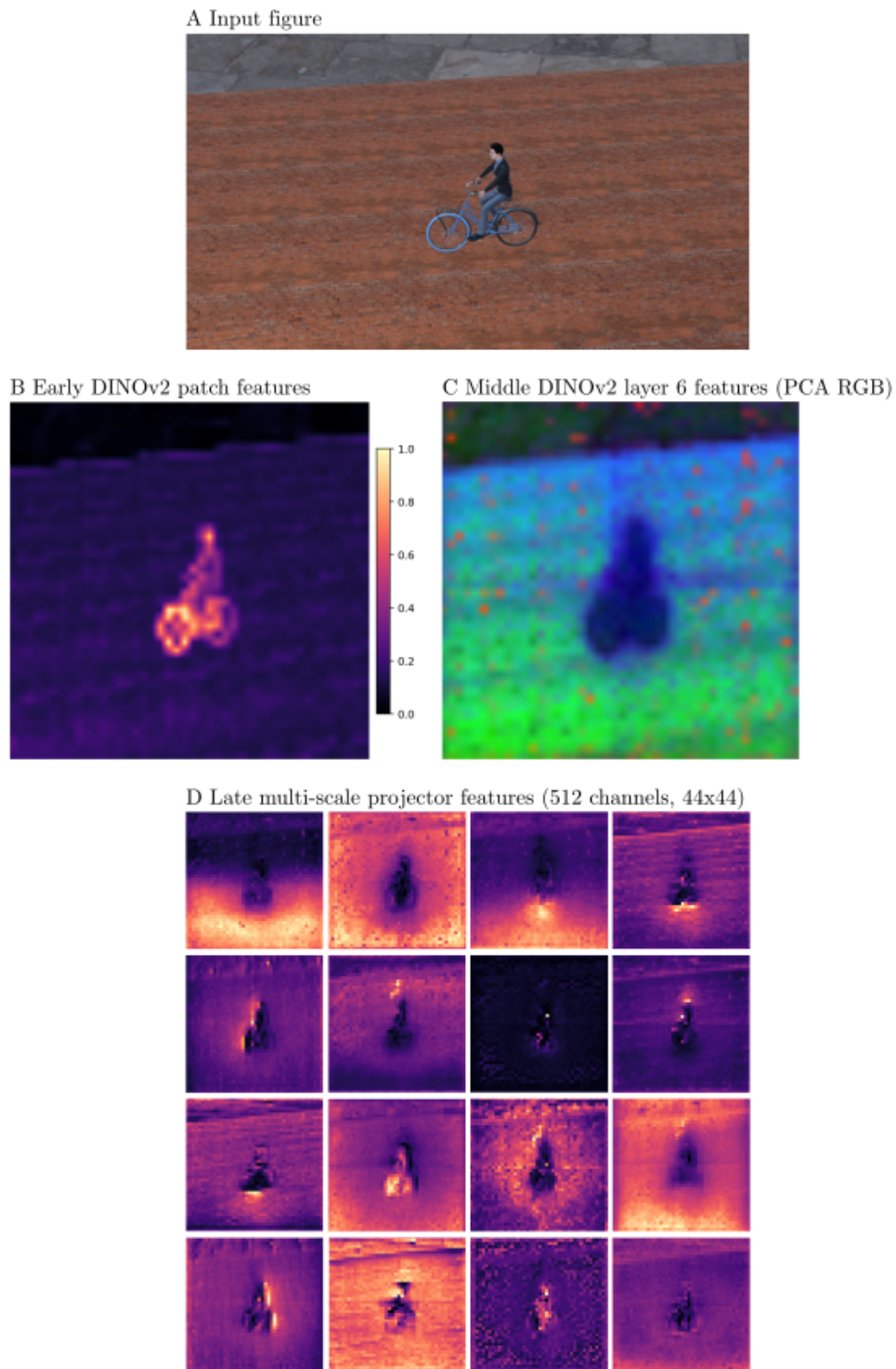


Figure 3.5: First stages of RF-DETR processing an input image.

3.5.1. Localising the bicycle

The first subproblem is to find a rectangular image region that contains the bicycle so that later stages process a crop: a cut-out of that region rather than the full frame. Cropping removes background clutter and presents the bicycle at a more consistent scale to the two-dimensional keypoint estimator, which

both improves accuracy and reduces computation time.

Localisation is a form of generic object detection, a well known and established field. Training a detector from scratch is unnecessary here: large public benchmarks such as COCO already include a bicycle class (Lin et al., 2014), and the later stages only require a box that reliably contain the bicycle, as slight inaccuracies in the box prediction are resolved by noise rejection algorithms within those models. Therefore, an off-the-shelf detector rather than collecting a custom detection training set was used.



Figure 3.6: The result of RF-DETR detecting a bicycle with 91% confidence.

RF-DETR 2x-large was chosen because it combines competitive COCO detection accuracy with real-time inference (Robinson et al., 2025) (Figure 3.4), and because the bicycle class is already present in that training set so no custom detector training was required. This performance is obtained by using the state-of-the-art DINOv2 image backbone (Oquab et al., 2023). The RF-DETR family also offers smaller variants with the same interface, which keeps the later stages unchanged if latency must be traded against accuracy.

DETR-style detectors replace the classical convolutional neural network (CNN) pipeline with a transformer. A CNN slides a fixed-size filter (kernel) across the image; that local window struggles when objects appear at very different scales. A transformer instead splits the image into a grid of patches, treats each patch as a token (a small vector summary of that patch), and uses self-attention so that every token can exchange information with every other, a useful property when the bicycle occupies only part of a cluttered scene. The cost is that transformers typically need large pretraining datasets and have higher inference compute cost than small CNNs.

Figures 3.5 and 3.6 illustrate how RF-DETR builds a bicycle detection from an input image. The model first tokenises the image into patches. Early DINOv2 features still show local structure, and the bicycle outline remains visible (Figure 3.5). Deeper layers produce more abstract embeddings (high-dimensional vectors per patch). Those embeddings cannot be plotted directly, so principal component analysis (PCA) is used to map them to three channels displayed as red, green, and blue: similar colours indicate similar learned representations, which makes it easier to see how the model groups bicycle, rider, and background. Multi-scale features are then combined by a projector; different channels emphasise different parts of the scene (background, wheels, shadows).

In the final stage, a set of learned object queries searches for instances in the image. The query with the highest bicycle class score attends to bicycle-related regions. A detection head turns that attention into a bounding box and confidence score (Figure 3.6). The box is stored as the pixel coordinates of

two opposite corners. It is passed to the two-dimensional keypoint estimator as the crop, and it is also used later to normalise keypoint coordinates for the three-dimensional lifter.



Figure 3.7: RTMPose-L input and output for a single frame: the bicycle image (left) and the predicted two-dimensional keypoints with the bicycle skeleton (right).

3.5.2. Estimating two-dimensional keypoints

The second stage localises the $J = 18$ semantic bicycle keypoints (Figure 2.1b) inside the RF-DETR crop. These two-dimensional keypoints are the measurements from which the three-dimensional lifter must resolve depth, so inaccurate localisation of these points at this stage directly affect the later ability to estimate roll, steering, and crank angles.

A top-down two-dimensional pose estimator is a natural choice once a bounding box is available as the network directly detects keypoint coordinates on the cropped image. A different approach than bottom-up methods such as YOLOPose that detect all keypoints in the full frame and grouping them into their respective groups. Crowded multi-person grouping is not the main difficulty here as clips usually contain one bicycle, so the priority is accurate localisation of a fixed keypoint set. RTMPose (Jiang et al., 2023), from OpenMMLab, was chosen for that role. It offers a favourable speed–accuracy trade-off on COCO-style benchmarks, and is easily modified to the 18-joint bicycle keypoint set instead of the standard 17-joint COCO human skeleton. The authors already show fine-tuning to larger keypoint sets such as COCO-WholeBody. That makes it reasonable to replace the human output with the bicycle skeleton and fine-tune on the static synthetic images described above. Fine-tuning pre-trained models preserves useful low- and mid-level visual features while adapting the later stages to wheel centres, frame points, handlebars, and pedals. The RTMPose-L variant is used with an input resolution of 256×320 pixels. The wider aspect ratio fits bicycles better than the taller crops common for standing humans.

RTMPose follows a standard convolutional pose layout: a backbone builds an internal image representation, intermediate modules sharpen it into a high-contrast feature map (often visualised as edge- or heatmap-like responses), and a head turns that map into keypoint coordinates (Figure 3.7). The backbone used here is CSPNeXt, a convolutional network family that gave the best COCO results in the RTMPose paper (Jiang et al., 2023). Its output is passed through a gated attention update (GAU) block, a compact attention mechanism related to transformer self-attention, but cheaper. This GAU emphasises image regions that matter for the landmarks. The head then follows the SimCC formulation (Li et al., 2021): instead of regressing (x, y) directly or predicting a full two-dimensional heatmap per joint, it classifies each keypoint into separate horizontal and vertical coordinate bins. Bin classification keeps the head small while still allowing sub-pixel localisation. Performance matters because this stage runs on every frame.

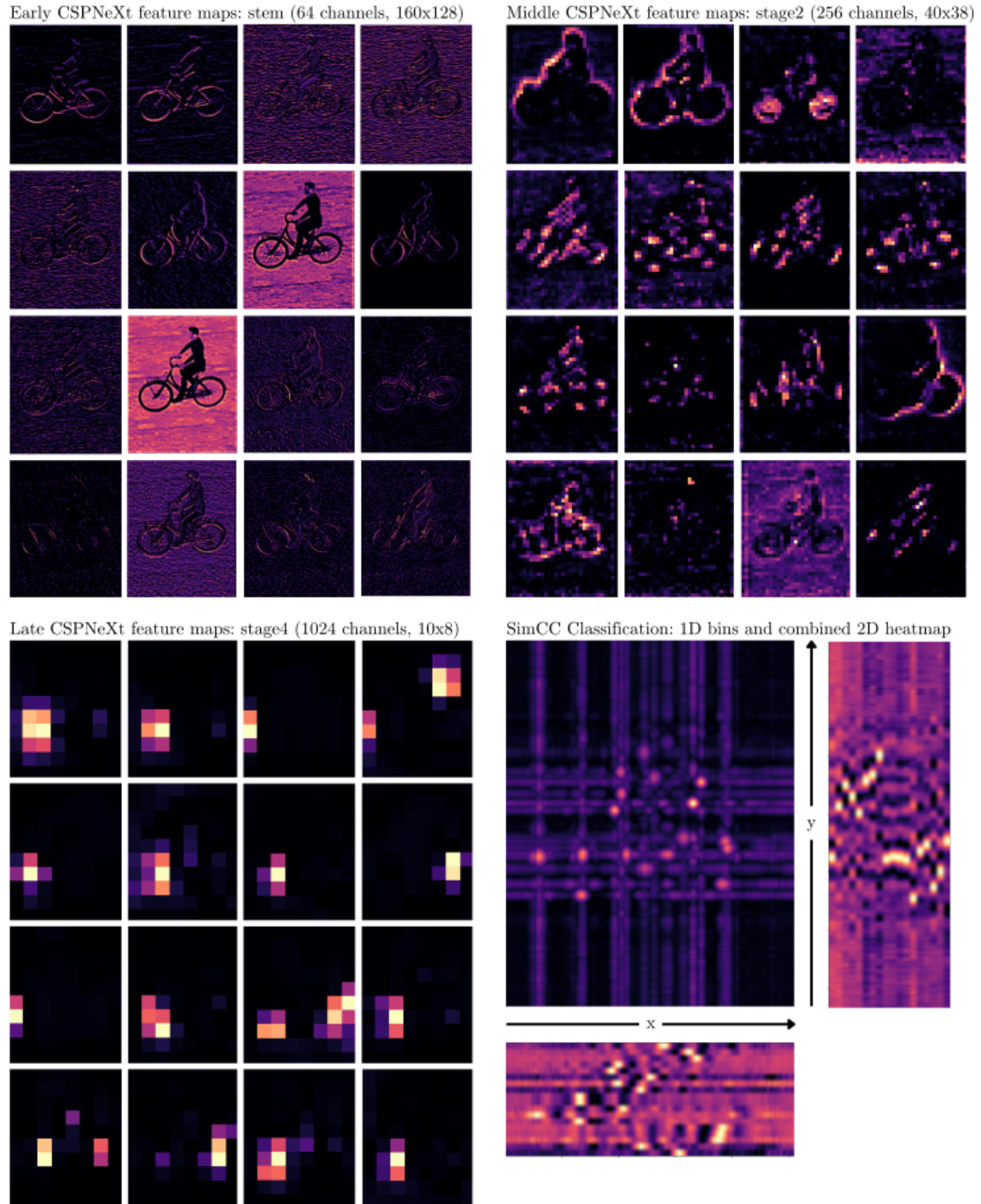


Figure 3.8: Feature representations inside RTMPose-L for the crop in Figure 3.7. Top-left: early CSPNeXt stem features (160×128). Top-right: mid-depth features (40×32). Bottom-left: final backbone features (10×8). Bottom-right: SimCC 2D keypoint estimation

Figure 3.8 shows how the representation evolves for the same crop. Early “stem” features (160×128) still look like the input: bicycle and rider outlines remain clear. At mid depth (40×32 , 256 channels) individual channels begin to respond to parts such as wheels or the rider rather than to edges everywhere. The final backbone stage (10×8 , 1024 channels) is a coarse grid of abstract activations, with only a few locations strongly lit. The SimCC head reads that grid; the bottom-right panel shows the horizontal and vertical bin distributions for all 18 keypoints and a reconstructed heatmap per joint. Narrow bright peaks correspond to confident localisations; wider or faint peaks mark uncertain points, for example

Table 3.1: default RTMPose-L settings used for fine-tuning.

Parameter	Value
Framework	MMPose TopdownPoseEstimator
Backbone	CSPNeXt-P5-L (SiLU, channel attention, SyncBN)
Checkpoint	cspnext-l_udp-aic-coco_210e-256x192
Head	RTMCCHead + GAU ($d=256$, $s=128$, expand 2)
Label encoding	SimCC (split ratio 2.0; sigmas 4.9, 5.66)
Loss function	KLDiscretLoss
Input resolution	256×320
Keypoints	$J=18$
Optimizer	AdamW (AMP, dynamic loss scaling)
Base learning rate	5×10^{-4}
Weight decay	0.05
Warmup	1000 iterations (linear)
LR schedule	constant to epoch 125, then cosine to $0.05 \times$ base
Batch size	64 per GPU ($2 \times$ RTX A5000)
Total epochs	250 (stage 1: 200; stage 2: 50)
Stage 1 augmentation	flip, shift, scale 1.0–1.2, rot $\pm 45^\circ$, HSV
Stage 2 augmentation	flip, scale 1.0–1.2, rot $\pm 30^\circ$, HSV (no shift)

handlebar landmarks occluded by the rider’s hands.

Training protocol for RTMPose

RTMPose-L was fine-tuned on the static synthetic bicycle images using the MMPose TopdownPoseEstimator module. Each training sample is a crop from a bounding box as in the RF-DETR $\rightarrow$ RTMPose inference path. Table 3.1 lists the fixed settings. Batch size is the number of input images that update the weights together and was optimized to fill 24 GB of VRAM per GPU. The learning rate, warmup, and cosine schedule control how large each weight update is over training. SimCC sigmas set how sharply the coordinate-bin training targets are peaked around the ground-truth pixel. Higher sigmas provide more stable output but are generally less precise. Because the output changes from 17 human joints to 18 bicycle joints, training uses two stages: 200 epochs with stronger augmentation, then 50 epochs with milder augmentation to settle into the, hopefully, global optimum.

3.5.3. Temporal three-dimensional lifting (PoseMamba)

The third stage recovers a root-relative three-dimensional keypoint trajectory $\hat{\mathbf{X}}_{1:T}$ from the bbox-normalised RTMPose detections. That lift must resolve the monocular depth ambiguity of Section 2.3.1.

For each clip, the model receives bbox-normalised two-dimensional keypoints from RTMPose. Confidence scores are not passed to the lifter, so each frame is only a set of normalised image-plane coordinates. The input tensor is

$$\mathbf{U} \in \mathbb{R}^{T \times J \times 2}, \quad (3.1)$$

where T is the temporal window length, $J = 18$ is the number of bicycle keypoints, and $\mathbf{U}_{t,j}$ stores the normalised coordinates of keypoint j in frame t , consistent with $\mathbf{u}_{t,j}$ in Chapter 2. The output is the corresponding root-relative three-dimensional sequence in camera coordinates, where each position is expressed relative to the bottom-bracket root of Section 2.2.1.

$$\hat{\mathbf{X}} \in \mathbb{R}^{T \times J \times 3}. \quad (3.2)$$

Temporal modelling

Monocular depth alone is severely under-determined from a single frame. Pavllo et al. (2018) showed that giving a lifter temporal context improves accuracy and, more importantly, temporal consistency. Temporal consistency is that keypoint estimates between frames should not jump when the bicycle moves smoothly. That consistency is measured by mean per joint velocity error (MPJVE), sometimes described as jitter when the reconstruction looks noisy. Because the bicycle and rider have substantial mass and inertia, roll, steer, and keypoint positions change smoothly over short intervals, so large

frame-to-frame jumps are dynamically implausible. Not only did Pavlo et al. (2018) show that temporal information reduces jitter, it also improves the model's ability to solve the depth ambiguity problem explained in Section 2.3.1. Together, the depth ambiguity and that smoothness motivate a lifter that uses a temporal window and estimates all frames in the window jointly.

Several architectures can supply that context. TCNs, as in VideoPose3D (Pavlo et al., 2018), convolve along time with dilated kernels so distant frames still influence the centre frame. They are efficient, but interactions stay inside fixed local windows rather than also influencing parts outside of this window. Spatial relations among keypoints within a frame are also not modelled jointly with time unless the architecture is extended, for example with topology graphs as in TBDU-Net (Cheng et al., 2022). Transformer video lifters apply self-attention over frames and joints, which captures long-range dependence more flexibly, but attention scales quadratically with T and usually needs large training sets. Neural state-space models keep linear cost in T while still supporting long-range dependencies, which is why they are used here.

Neural state-space models

State-space models originate in control theory: an internal state summarises the past and evolves under applied inputs. In continuous time, a linear time-invariant (LTI) model maps an input $u(t) \in \mathbb{R}^L$ to an output $y(t) \in \mathbb{R}^L$ through a state $h(t) \in \mathbb{C}^N$:

$$\dot{h}(t) = Ah(t) + Bu(t), \quad (3.3)$$

$$y(t) = Ch(t) + Du(t), \quad (3.4)$$

where $A \in \mathbb{C}^{N \times N}$ and $B, C \in \mathbb{C}^N$ are the system matrices. Some formulations keep the direct feedthrough $Du(t)$, but the Mamba block diagram omits it, and it is not used further here (A. Gu & Dao, 2023). For a bicycle, a physical state might include roll, steering, and their derivatives. Eq. (3.4) would then describe how that mechanical state responds to inputs and how observations are formed.

A neural state-space model does not use such a physical state explicitly. It keeps the same recurrence structure, but h is a learned latent vector with no direct physical meaning, encoding whatever helps three-dimensional lifting. At time step i the input u_i is a feature derived from the two-dimensional keypoints, not a control torque or a measured roll angle. The matrices A , B , and C are learned from data rather than identified from a dynamical model.

Video clips are discrete, so Eq. (3.4) must be discretised frame by frame. Following Gu and Dao (A. Gu & Dao, 2023), the solution over an interval $[t_a, t_b]$ can be written

$$h(t_b) = e^{A(t_b - t_a)} \left(h(t_a) + \int_{t_a}^{t_b} B(\tau) u(\tau) e^{-A(\tau - t_a)} d\tau \right). \quad (3.5)$$

Sampling with step size Δ_i between frames t_i and t_{i+1} gives

$$h_b = e^{A(\sum_{i=a}^{b-1} \Delta_i)} h_a + \sum_{i=a}^{b-1} B_i u_i e^{-A(\sum_{j=a}^i \Delta_j)} \Delta_i, \quad (3.6)$$

$$y_b = Ch_b \quad (3.7)$$

which is zero-order hold (ZOH) discretisation (A. Gu & Dao, 2023). For a single step ($b = a + 1$) this reduces to

$$h_{a+1} = \bar{A}_a h_a + \bar{B}_a u_a, \quad \bar{A}_a = e^{A\Delta_a}, \quad (3.8)$$

so the hidden state advances by a matrix multiplication at each frame.

Equations (3.4) and (3.8) describe an LTI system: A , B , and C do not depend on the input sequence. That is appropriate in vehicle dynamics when h is a physical state and the matrices come from a model grounded in experiments. In a neural SSM, h is latent and u_i is a visual feature. A single fixed linear recurrence must then approximate a nonlinear, viewpoint-dependent map from two-dimensional detections to three-dimensional keypoints over the whole clip. This means that the same update happens whether landmarks are visible, occluded, or noisy. Mamba (A. Gu & Dao, 2023) addresses this by making B , C , and the step size Δ input-dependent, yielding a selective/neural state-space model

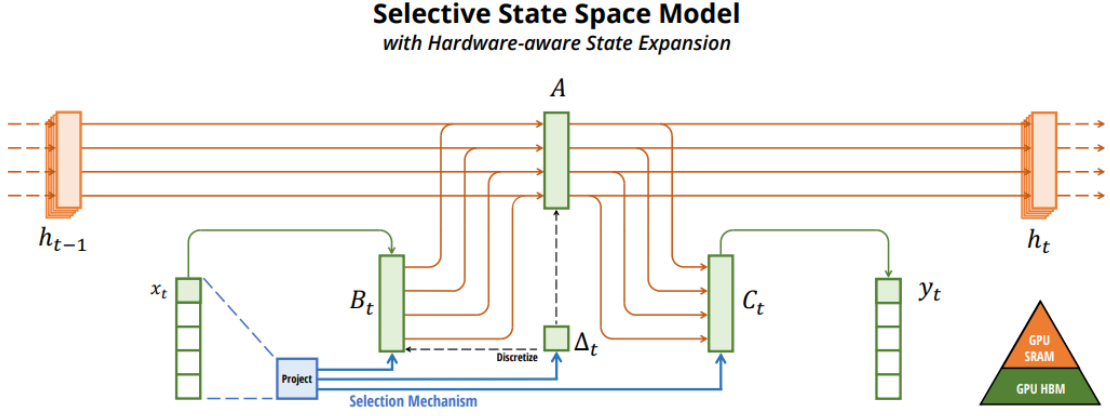


Figure 3.9: Mamba selective state-space mechanism (A. Gu & Dao, 2023).

(Figure 3.9). The hidden state still follows the discrete form of Eq. (3.7), but the effective transition matrices change from frame to frame with the two-dimensional input. The model can emphasise informative frames and down-weight noisy parts of the input while keeping linear cost in T , unlike full self-attention of transformer architectures. The recurrence can be evaluated on a GPU with parallel associative scans rather than a sequential loop over frames (A. Gu & Dao, 2023), further enhancing inference speed.

PoseMamba architecture

PoseMamba is used as the Mamba-based lifter because it reports strong results on Human3.6M and MPI-INF-3DHP, is written in standard PyTorch (industry standard machine learning library), and keeps the loss weights configurable (Huang et al., 2024). The encoder is reused with $J = 18$ bicycle key-points (Appendix A) instead of $J = 17$ human joints. The linear embedding, positional embeddings, and regression head are resized accordingly. Figure 3.10 summarises the data flow. Each keypoint coordinate pair in $\mathbf{U}$ is projected to a d_m -dimensional token, giving $\mathbf{P} \in \mathbb{R}^{T \times J \times d_m}$. Learned spatial and temporal positional embeddings are added so the model knows which landmark and which frame each token represents.

The embedded sequence passes through a stack of bidirectional global–local spatio-temporal Mamba blocks (Huang et al., 2024). Each block first applies layer normalisation: each token’s feature vector is rescaled so the following layers see a normalized input amplitude, independent of how large the activations were before the normalization. Layer normalization improves the stability of the hidden state in recurrent networks (Ba et al., 2016). A depthwise convolution then mixes neighbouring tokens over a short range, followed by a neural state-space model scan. Each of these steps uses a feedback layer that maps the output back onto the input layer. A second layer normalisation and a small multilayer perceptron follow to assist in the generalizability of the neural state-space model, again with a feedback path, producing $\mathbf{Z}_l \in \mathbb{R}^{T \times J \times d_m}$. Blocks are stacked and the number of blocks depends on model capacity. Positional embeddings are added only at the input.

Inside the SSM stage, tokens are read along four scan paths: forward and backward spatial scans across the J keypoints in each frame, and forward and backward temporal scans along the T frames for each keypoint. Spatial scanning splits into a global path (fixed skeleton order) and a local path that reorders keypoints along shorter chains so strongly coupled neighbours are scanned consecutively (Huang et al., 2024). Path outputs are merged so each token sees past and future context in the clip and nearby joints on the skeleton. The local spatial scan order is not predefined in this thesis like the PoseMamba authors did, so the model is allowed to learn this structure itself.

After the final encoder block, a linear head maps $\mathbf{Z}$ to $\hat{\mathbf{X}} \in \mathbb{R}^{T \times J \times 3}$. Training compares that prediction directly to the synthetic ground-truth keypoints. The remainder of this section describes training and evaluation on the bicycle clips.

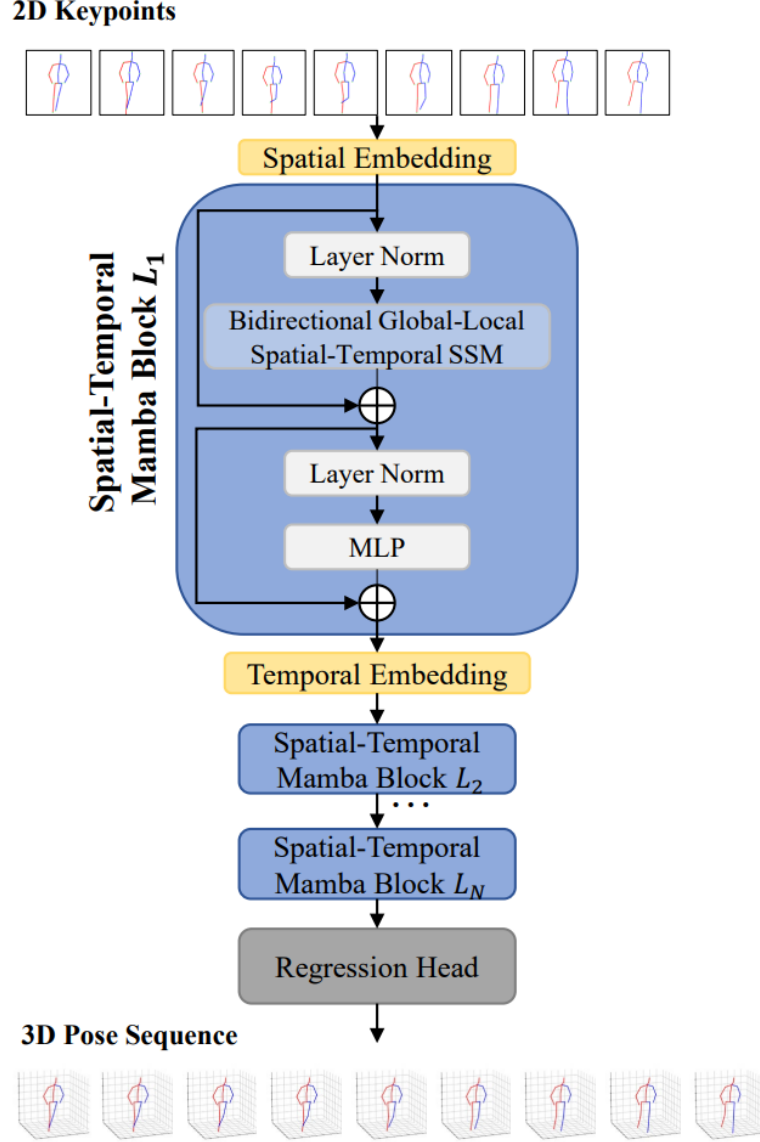


Figure 3.10: PoseMamba architecture for human pose estimation (Huang et al., 2024), reused here with $J = 18$ bicycle keypoints.

Training protocol and metrics for PoseMamba

Unlike RTMPose, which is trained through MMPose, PoseMamba is trained with a standard PyTorch training setup (Paszke et al., 2019). Learning rate, batch size, augmentations, and especially the loss weights are therefore easy to change. The published default weights were tuned for humans on Human3.6M and MPI-INF-3DHP, they might not be optimal for bicycles. Several loss combinations are therefore trained to see which terms matter for this specific setting.

The public PoseMamba objective combines four terms (Eq. (3.9)).

$$\mathcal{L} = \lambda_{3D}\mathcal{L}_{3D} + \lambda_N\mathcal{L}_{N-3D} + \lambda_v\mathcal{L}_{3Dv} + \lambda_s\mathcal{L}_{Smooth} \quad (3.9)$$

The leading term $\mathcal{L}_{3D}$ is mean per joint position error (MPJPE), also called Protocol 1 error in the human pose literature: the mean Euclidean distance between predicted and ground-truth three-dimensional keypoints (Eq. (3.10)).

$$\mathcal{L}_{3D} = \frac{1}{TJ} \sum_{t=1}^T \sum_{j=1}^J \left\| \hat{\mathbf{x}}_{t,j} - \mathbf{x}_{t,j} \right\|_2 \quad (3.10)$$

The second term is normalised MPJPE (N-MPJPE). A per-frame scale s_t is chosen to best align the predicted skeleton to the ground truth in a least-squares sense (Eq. (3.11)), then MPJPE is computed after that rescaling (Eq. (3.12)). This emphasises skeleton shape over absolute size.

$$s_t = \frac{\sum_{j=1}^J \hat{\mathbf{X}}_{t,j} \cdot \mathbf{X}_{t,j}}{\sum_{j=1}^J \|\hat{\mathbf{X}}_{t,j}\|_2^2} \quad (3.11)$$

$$\mathcal{L}_{N-MPJPE} = \frac{1}{TJ} \sum_{t=1}^T \sum_{j=1}^J \|s_t \hat{\mathbf{X}}_{t,j} - \mathbf{X}_{t,j}\|_2 \quad (3.12)$$

The third term $\mathcal{L}_{3Dv}$ penalises errors in inter-frame keypoint velocity relative to the ground truth (Eqs. (3.13)–(3.14)).

$$\hat{\mathbf{V}}_{t,j} = \hat{\mathbf{X}}_{t+1,j} - \hat{\mathbf{X}}_{t,j}, \quad \mathbf{V}_{t,j} = \mathbf{X}_{t+1,j} - \mathbf{X}_{t,j} \quad (3.13)$$

$$\mathcal{L}_m = \frac{1}{(T-1)J} \sum_{t=1}^{T-1} \sum_{j=1}^J \|\hat{\mathbf{V}}_{t,j} - \mathbf{V}_{t,j}\|_2 \quad (3.14)$$

The fourth term is unsupervised: it penalises large predicted frame-to-frame jumps without using ground-truth velocity, acting as a low-pass filter on the output. (Eq. (3.15)). The optional per-keypoint weights w_j are all set to 1 in this work.

$$\mathcal{L}_t = \frac{1}{(T-1)J} \sum_{t=1}^{T-1} \sum_{j=1}^J w_j \|\hat{\mathbf{X}}_{t+1,j} - \hat{\mathbf{X}}_{t,j}\|_2^2 \quad (3.15)$$

Supervised roll and steering losses, and their frame-to-frame rates, were added on top of this objective to test whether direct angle supervision helps the lifter for bicycle kinematics. They enter as extra weighted terms (Eq. (3.16)).

$$\mathcal{L} = \lambda_{3D} \mathcal{L}_{3D} + \lambda_N \mathcal{L}_{N-3D} + \lambda_v \mathcal{L}_{3Dv} + \lambda_s \mathcal{L}_{Smooth} + \lambda_\phi \mathcal{L}_\phi + \lambda_\delta \mathcal{L}_\delta + \lambda_{\phi,v} \mathcal{L}_{\phi,v} + \lambda_{\delta,v} \mathcal{L}_{\delta,v}. \quad (3.16)$$

Because roll and steering are periodic, angular differences are wrapped onto $(-\pi, \pi]$ before the absolute value is taken:

$$\text{wrap}(\alpha) = \text{atan2}(\sin \alpha, \cos \alpha). \quad (3.17)$$

This maps any signed difference onto the shortest path on the circle, so an error of $2\pi - \varepsilon$ is counted as $-\varepsilon$ rather than nearly a full turn.

Angle errors use an L1 norm after wrapping (Eq. (3.18)).

$$\mathcal{L}_\phi = \frac{1}{T} \sum_{t=1}^T \left| \text{wrap}(\hat{\phi}_t - \phi_t) \right|, \quad \mathcal{L}_\delta = \frac{1}{T} \sum_{t=1}^T \left| \text{wrap}(\hat{\delta}_t - \delta_t) \right| \quad (3.18)$$

A similar wrapped L1 loss is applied to frame-to-frame angular rates (Eq. (3.19)).

$$\begin{aligned} \mathcal{L}_{\phi,v} &= \frac{1}{T-1} \sum_{t=1}^{T-1} \left| \text{wrap}(\text{wrap}(\hat{\phi}_{t+1} - \hat{\phi}_t) - \text{wrap}(\phi_{t+1} - \phi_t)) \right| \\ \mathcal{L}_{\delta,v} &= \frac{1}{T-1} \sum_{t=1}^{T-1} \left| \text{wrap}(\text{wrap}(\hat{\delta}_{t+1} - \hat{\delta}_t) - \text{wrap}(\delta_{t+1} - \delta_t)) \right|. \end{aligned} \quad (3.19)$$

Training uses the sliding windows described earlier in the dataset section: windows of length T with training stride $s \approx T/3$, and non-overlapping windows ($s = T$) on the validation and test parts of the dataset. Table 3.2 lists the resulting sample counts for each (T, s) configuration (94 101 unique frames across 43 clips).

Table 3.2: Number of windowed samples used for each PoseMamba temporal window configuration.

Window model	T	Stride (train/val-test)	2D input	Train	Val	Test
PoseMamba $f_{27}s_9$	27	9 / 27	detected 2D	8484	1060	1060
PoseMamba $f_{81}s_{27}$	81	27 / 81	detected 2D	2782	347	347
PoseMamba $f_{121}s_{40}$	121	40 / 121	detected 2D	1832	228	228
PoseMamba $f_{162}s_{54}$	162	54 / 162	detected 2D	1338	167	167
PoseMamba $f_{243}s_{81}$	243	81 / 243	detected 2D	880	110	110
PoseMamba $f_{243}s_{81}$	243	81 / 243	GT-projected 2D	880	110	110

Settings shared by all runs are listed in Table 3.3. Batch size is the number of windows updating the parameters together and was optimized to fill 24 GB VRAM of the GPU. Batch size was 4 windows for all experiments except the extra-large model which could only fit 2. Learning rate and the per-epoch decay 0.99 of the learning rate. Weight decay is a similar slowdown setting and was set to an ℓ_2 penalty on the weights. Flip randomly mirrors the two-dimensional input and the corresponding three-dimensional target for left–right augmentation. Input masking, which would hide joints or frames during training, is left disabled.

Table 3.3: Default training settings for the PoseMamba bicycle experiments.

Parameter	Value
Epochs	120
Checkpoint frequency	10
Batch size	4
Learning rate	2×10^{-4}
Weight decay	0.01
LR decay	0.99
Dropout	0.0
Mamba fusion	true
MLP ratio	2
Root-relative coordinates	true
2D input source	detected 2D
Confidence usage (no_conf)	true (confidences unused)
2D flip augmentation	true
2D noise sigma	0.0
Input masking	disabled

What varies across experiments is summarised in Table 3.4. Rows named `capacity_*` change feature width d and SSM depth L (Small, Base, Large, X-large, matching the original PoseMamba sizes). Rows `window_t*` change the temporal window T and stride s (still $T/3$ stride), trading shorter temporal context for more training samples. Rows `loss_*` and `dyn_*` change which loss weights are active. Rows `aug_*` change flip and additive two-dimensional noise σ . Loss-weight and augmentation trials use the Base capacity (`capacity_b`) to limit training time and remain comparable to the original PoseMamba paper. Window-length experiments use the same Base backbone so temporal context can be compared fairly.

For three-dimensional pose estimation accuracy, the main reported score is usually MPJPE in millimetres on the validation set. Because the research question is focused on bicycle kinematics, the lifted keypoints are also converted to roll, steering, and crank and compared with the simulation ground truth. These angle errors are treated as the primary product metrics, with MPJPE as quantification of performance against the human pose estimation variant of PoseMamba. The epoch with the lowest MPJPE was saved as the best checkpoint.

3.6. Kinematics estimation

With the three-dimensional keypoints estimated using the neural networks the last step is to calculate the configuration variables ϕ_t , δ_t , and θ_t defined in Section 2.1. Unlike the previous stages, this is a deterministic calculation on the estimated bicycle keypoints, not a learned model. No dynamical

Table 3.4: PoseMamba parameters varied across experiment configurations. Columns d and L denote feature dimension and SSM depth. λ_s , λ_v , and λ_d are the N-MPJPE, velocity, and smoothing weights. λ_ϕ , $\lambda_{\dot{\phi}}$, λ_δ , and $\lambda_{\dot{\delta}}$ are the roll, roll-rate, steer, and steer-rate losses.

Run	T	s	d	L	λ_{3d}	λ_s	λ_v	λ_d	λ_ϕ	$\lambda_{\dot{\phi}}$	λ_δ	$\lambda_{\dot{\delta}}$	Flip	σ
capacity_s	243	81	64	10	1.0	0.5	20.0	0.5	0	0	0	0	T	0.0
capacity_b	243	81	128	10	1.0	0.5	20.0	0.5	0	0	0	0	T	0.0
capacity_l	243	81	128	20	1.0	0.5	20.0	0.5	0	0	0	0	T	0.0
capacity_x	243	81	256	20	1.0	0.5	20.0	0.5	0	0	0	0	T	0.0
window_t27	27	9	128	10	1.0	0.5	20.0	0.5	0	0	0	0	T	0.0
window_t81	81	27	128	10	1.0	0.5	20.0	0.5	0	0	0	0	T	0.0
window_t121	121	40	128	10	1.0	0.5	20.0	0.5	0	0	0	0	T	0.0
window_t162	162	54	128	10	1.0	0.5	20.0	0.5	0	0	0	0	T	0.0
loss_no_velocity	243	81	128	10	1.0	0.5	0.0	0.5	0	0	0	0	T	0.0
loss_no_nmpjpe	243	81	128	10	1.0	0.0	20.0	0.5	0	0	0	0	T	0.0
loss_no_diff	243	81	128	10	1.0	0.5	20.0	0.0	0	0	0	0	T	0.0
dyn_steer	243	81	128	10	1.0	0.5	20.0	0.5	0	0	1.0	0	T	0.0
dyn_roll	243	81	128	10	1.0	0.5	20.0	0.5	1.0	0	0	0	T	0.0
dyn_steer_roll	243	81	128	10	1.0	0.5	20.0	0.5	1.0	0	1.0	0	T	0.0
dyn_steer_roll_vel	243	81	128	10	1.0	0.5	20.0	0.5	1.0	1.0	1.0	1.0	T	0.0
aug_no_flip	243	81	128	10	1.0	0.5	20.0	0.5	0	0	0	0	F	0.0
aug_noise2d	243	81	128	10	1.0	0.5	20.0	0.5	0	0	0	0	T	0.02

constraints or stiffness optimizations are applied: the angles follow from the keypoints, so keypoint error passes through to ϕ , δ , and θ . The same calculation is applied on predicted and ground-truth keypoints. Comparing those two angle series therefore measures how well the lifter reconstructed the skeleton, and removes the global-local mismatch between MuJoCo and these algorithms.

3.6.1. Roll angle estimation

The pipeline outputs keypoints in OpenCV camera coordinates (X right, Y down, Z forward), so the keypoints first need to be converted to relative standard cartesian coordinates. Rear-frame keypoints with the addition of the middle handlebar are used to form the midplane of the bicycle (Figure 3.11a): bottom bracket, seat stay, saddle, lower and upper head tube, handlebar middle, and rear-wheel ground contact. Front-wheel and handlebar end keypoints are excluded because they leave the sagittal plane when the fork is steered. Plane-fitting is straight-forward if the keypoints lie exactly on the midplane as in the case of the ground truth data. However, noisy predicted keypoints lie slightly off this plane and therefore require a more complex approach. A suitable approach was found in the form of principal component analysis and was implemented as following: Given the K selected points in frame t , a plane is fit by principal component analysis: the covariance of centered positions is formed, and the eigenvector $\mathbf{n}_t$ with smallest eigenvalue is taken as the plane normal. That the smallest eigenvalue is the normal on the midplane can be assumed due to the large magnitude of the forward and up vectors compared to the sideways displacement. This method also negates sideways noise in the roll-angle estimate. The sign of $\mathbf{n}_t$ is fixed by aligning it with the rear-hub lateral direction (left hub to right hub) so left and right stay consistent across frames.

With the midplane normal available as $\mathbf{n}_t = (n_x, n_y, n_z)^\top$, the roll angle follows from how far this normal has tilted out of the horizontal plane. When the bicycle is upright the normal lies in the camera-horizontal plane spanned by X and Z , so its vertical component is zero and roll is zero. Lean rotates $\mathbf{n}_t$ out of that plane; the signed angle of that tilt is

$$\phi_t = \text{atan2}\left(-n_y, \sqrt{n_x^2 + n_z^2 + \varepsilon}\right). \quad (3.20)$$

The minus sign on n_y is needed because OpenCV takes Y positive downward, so “up” in the scene is opposite to that axis. The resulting ϕ_t is therefore lean relative to the camera, not absolute road roll as

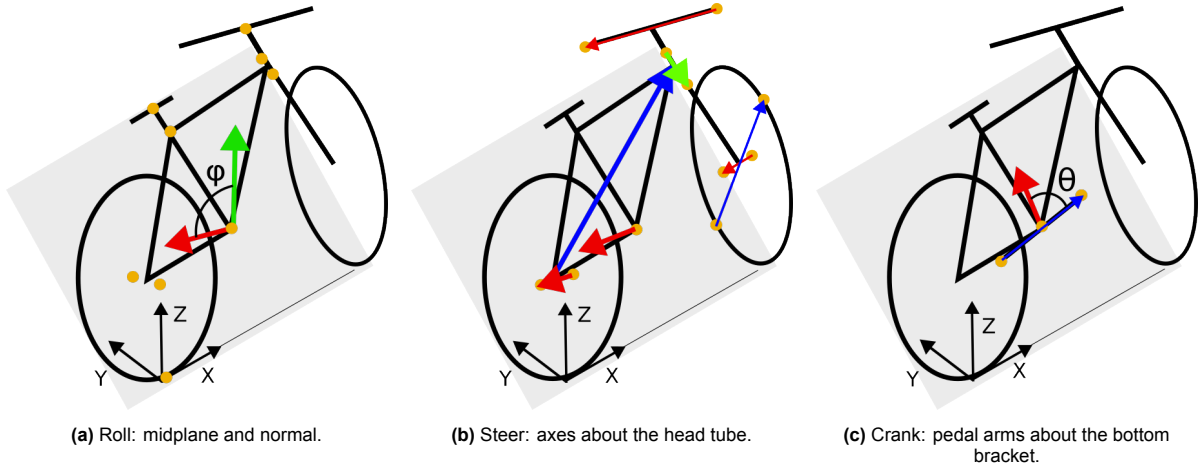


Figure 3.11: Angle estimation algorithms for roll, steering, and crank from the three-dimensional keypoints. Orange markers are bicycle keypoints used in the estimation. Coloured axes show the planes and directions used in each calculation grouped by color.

stored on the MuJoCo floating base. As noted above, scoring uses the same extractor on predicted and ground-truth keypoints so both angles live in that camera-relative frame.

3.6.2. Steering angle estimation

Steering angle δ_t is the rotation of the front assembly about the steering axis relative to the rear frame (Meijaard et al., 2007) (Figure 3.11b). The steering axis is approximated by the head-tube direction unit vector

$$\mathbf{e}_t = \frac{\mathbf{X}_{t,\text{UHT}} - \mathbf{X}_{t,\text{LHT}}}{\|\mathbf{X}_{t,\text{UHT}} - \mathbf{X}_{t,\text{LHT}}\|}, \quad (3.21)$$

where UHT and LHT denote the upper and lower head-tube keypoints. Directions on the rear frame and on the fork are compared only after projection into the plane perpendicular to $\mathbf{e}_t$. This keeps the estimate independent of fork geometry, a ground-plane angle would include a non-linear angle caused by the rake of the fork. For two directions $\mathbf{a}$ and $\mathbf{b}$, the signed rotation about $\mathbf{e}_t$ is

$$\angle_{\mathbf{e}}(\mathbf{a}, \mathbf{b}) = \text{atan2}((\mathbf{a} \times \mathbf{b}) \cdot \mathbf{e}, \mathbf{a} \cdot \mathbf{b}), \quad (3.22)$$

where $\mathbf{a}$ and $\mathbf{b}$ have first been projected perpendicular to $\mathbf{e}_t$. Using a plane fitting similar to the roll angle estimate caused an unstable solution, so a weighted average of multiple different vectors located on different parts of the frame was used instead.

Three vectors were chosen as seen in figure 3.11b: rear-hub lateral versus front-hub lateral, rear-hub lateral versus handlebar lateral, and frame-forward versus front-wheel forward. Same color vectors indicate they were used to calculate their specific term of the angle calculation. The frame-forward direction runs from the rear-hub centre to the head-tube centre; the wheel direction connects the front and back wheel markers. Each vector yields a steer angle $\delta_{t,i}$, weighted by the product of the projected vector lengths so geometry size is neglected. Because steer passes through zero, the estimates are combined with a weighted circular mean,

$$\delta_t = \text{atan2}\left(\sum_i w_i \sin \delta_{t,i}, \sum_i w_i \cos \delta_{t,i}\right). \quad (3.23)$$

Handlebar and wheel markers receive slightly lower weights than the hub axle because they had less reliable positional estimations. Steer is intrinsic to the bicycle, so keypoint-derived δ_t should agree with the MuJoCo steering joint.

3.6.3. Crank angle estimation

Crank angle θ_t is the rotation of the left crank about the bottom-bracket axis, with right pedal up defined as zero (Section 2.1, Figure 3.11c). The crank axis is taken as the sagittal-plane normal $\mathbf{n}_t$ from the

same calculation as the roll extraction. A forward reference $\mathbf{f}_t$ is formed by projecting the bottom bracket to central head tube vector into the plane perpendicular to $\mathbf{n}_t$ and normalising. An in-plane “up” direction is $\mathbf{y}_t = \mathbf{n}_t \times \mathbf{f}_t$. The crank-arm is the left pedal to right pedal vector, likewise projected into the crank plane. Decomposing that arm onto $(\mathbf{f}_t, \mathbf{y}_t)$ gives

$$\theta_t = \text{atan2}(-\mathbf{a}_t \cdot \mathbf{f}_t, \mathbf{a}_t \cdot \mathbf{y}_t), \quad (3.24)$$

where $\mathbf{a}_t$ is the projected crank-arm vector. The minus sign fixes the mismatch between this method and the output degree of freedom in the MuJoCo model.

3.6.4. Evaluation metrics for kinematic angles

Recovered angles are scored on the validation part of the dataset. Each validation window yields time series $\hat{\phi}_t$, $\hat{\delta}_t$, and $\hat{\theta}_t$ from predicted keypoints and reference series from ground-truth keypoints through the same algorithm h . All metrics are based on frames across windows their respective model’s window size. Periodic angles use the same wrapping as in the training losses (Eq. (3.17)).

For each angle $q \in \{\phi, \delta, \theta\}$, root mean square error (RMSE) and mean absolute error (MAE) against ground-truth across the entire window quantify general performance:

$$\text{RMSE}_q = \sqrt{\frac{1}{N} \sum_t \text{wrap}(\hat{q}_t - q_t)^2}, \quad \text{MAE}_q = \frac{1}{N} \sum_t |\text{wrap}(\hat{q}_t - q_t)|, \quad (3.25)$$

where N is the total number of frames scored and q_t denotes the reference from $h(\mathbf{X}_{\text{GT}})$. Mean signed error is the average difference also known as bias. It exposes systematic offsets such as a constant crank phase shift. Pearson correlation coefficient r_q measures linear association between predicted and reference series and gives an early indication of phase agreement. Angular rate RMSE on $\dot{\phi}$, $\dot{\delta}$, and $\dot{\theta}$ are reported to expose velocity estimate errors. Mean per joint position error (MPJPE) is the computer vision standard way of scoring pose estimation methods and shows the raw performance of the pipeline. It is used as a diagnostic metric for the lifter itself (Eq. (3.10)).

Alongside the table of performance are four visual summaries to allow better judgement of performance of the pipeline as some fine details are hidden in the averages of the table’s numbers. All figures are made using the base capacity model unless stated otherwise. To show the final result of the pipeline, a cherry picked window was chosen that highlights some key characteristics. A time-series plot of the three kinematic quantities is shown here with the prediction versus the ground truth. To complement this graph, an average RMSE across time of all windows in the validation set is shown with a one standard deviation area around it. All unique results of the clip are also shown as thin lines to make outliers visible.

To check whether the recovered angles follow the ground truth at different frequencies of motion, a coherence $C_{xy}(f)$ is computed between the predicted and ground truth series for roll, steering, and crank. For each validation clip the data is treated as time series at the simulation rate of 60 Hz, and $C_{xy}(f)$ is estimated with Welch’s method set to the length of the window. Clip-wise curves are then averaged to a mean coherence spectrum over the validation set. C_{xy} lies between 0 and 1: values near 1 at a certain frequency mean that predicted and ground-truth motion are well represented by the model at that time scale. Lower values mean the predicted series does not share that frequency content with the ground truth, so lean or steer changes at those rates are poorly recovered. The resulting plot therefore shows, for example, that slow lean and steer motion is tracked more consistently than faster maneuvers, which RMSE alone cannot separate. Bland–Altman plots show agreement by plotting the difference between predicted and reference against their mean at each frame. They are useful for spotting constant phase dependent bias, but they do not show whether a trajectory follows the ground truth over time.

4

Results

Evaluation of the models uses 110 validation windows of $T = 243$ frames unless otherwise stated. The primary configuration is `capacity_b` as seen in Table 3.4. On that model, roll RMSE is 1.4° , steer RMSE 3.0° , and crank RMSE 5.1° . Mean per-joint position error (MPJPE) was lower than the original PoseMamba model on a human benchmark, going from 35.5 to 17.6 mm respectively. Supplementary tables detection performance and figures showing RF-DETR and RTMPose are in Appendix B.

4.1. Quality of the two-dimensional input

The three-dimensional lifting models were trained on two-dimensional keypoint sequences from RF-DETR and RTMPose, not on raw images. Training on detected keypoints, including their noise, is necessary for teaching the correct dynamics to the lifter. Those sequences come from the tracking-camera dataset (Section 3), so the two-dimensional sequences are guaranteed to have a bicycle in it. On all input frames used for training ($N=94,041$), the bicycle was always detected, predicted boxes overlapped ground truth by 0.66 on average, and mean keypoint error was 14.8 px. Handlebar landmarks remained the hardest group (28.6 px versus 9.3 px on the rear frame). Static-image results for the first two stages, including overlap histograms and per-scene breakdowns, are in Appendix B.1.

4.2. Recovered bicycle kinematics

The end product of the pipeline is a time series of roll, steering, and crank angle for each validation window. The following subsections summarise accuracy, consistency across windows, agreement with ground truth, and frequency content of the recovered motion.

4.2.1. Accuracy of the three angles

Table 4.1: Kinematics accuracy for `capacity_b` on the validation set.

Quantity	RMSE	Bias	Pearson r	Rate RMSE
Roll ϕ	1.4°	-0.1°	0.995	6.2 deg/s
Steer δ	3.0°	0.0°	0.805	15.2 deg/s
Crank θ	5.1°	0.2°	0.999	48.6 deg/s

Table 4.1 lists metrics for the Base model on the full validation set. Roll, steer and crank angles are effectively unbiased in their estimate, with the crank showing the highest bias at only 0.2 degrees. Pearson r is high for roll and crank, but lower for steering angle (0.81). Angular-rate RMSE was highest for the crank angle and lowest for the roll angle.

Figure 4.2 shows predicted and ground-truth roll and steer for one cherry picked validation window. This figure was chosen as it shows different modes of riding and does not contain outlying predictions. Steer prediction tracks the ground-truth through turns and straight sections except for the start where

the prediction is opposite from the ground-truth. Roll prediction follows the ground-truth well in shape but has a clear negative bias in the first half of this example. Crank predictions follow the ground-truth accurately across the entire clip.

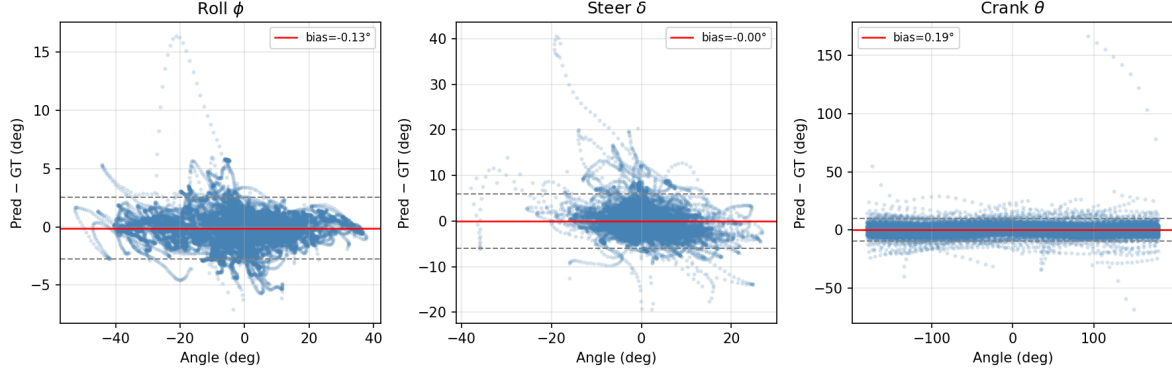


Figure 4.1: Bland-Altman plots for roll, steering, and crank on the validation set (`capacity_b`). Red lines: bias. dashed grey: 95% confidence interval

4.2.2. Consistency across the validation set

Figure 4.3 pools RMSE of the angle estimates across all validation windows normalized over the $T = 243$ window length. Thin curves show individual windows while the thick line is the mean and the shaded band is $\pm 1\sigma$.

Roll error stays near 1° for most of the window. Steer error is higher at the start (4° – 5°) but settles near 2° in the middle section where PoseMamba has access to past and future data. Crank error shows the same edge effect: it is largest at window boundaries and lowest in the centre.

4.2.3. Agreement and bias

Figure 4.1 plots predicted minus ground-truth angle against ground-truth magnitude for every frame. Roll and steer both have a bias near zero (-0.1° and 0.0° respectively). For roll this holds across the whole range as the slope against reference angle is -0.01 . This can be verified in the figure as the error stays within 0.5° across all magnitudes. Steering behaves differently. Although its bias is 0.0° , the slope against the reference angle is -0.16 . The error runs from 3.2° for reference angles below -20° , through 0.3° near zero, to -2.8° above 20° . Crank bias is 0.2° and flat across the whole pedal phase, with most frames within roughly $\pm 10^\circ$. Occasional outliers reach larger values. These likely originate from the start and end of the window.

4.2.4. Motion frequency recovery

Magnitude-squared coherence $C_{xy}(f)$ compares predicted and ground-truth angle series in the frequency domain (Section 3.6.4). Figure 4.4 shows mean coherence across validation windows for four model capacities.

For roll and steer, coherence is almost 1 below roughly 1.5 Hz, then settles near 0.8–0.9 at higher frequencies. Crank coherence stays high across the plotted band for Base, Large, and Extra-large; Small is slightly lower.

4.3. Effect of the training configuration on kinematics accuracy

Table 4.2 summarises all variation experiments grouped by the factor changed. Bold entries highlight the best score achieved. The default $T = 243$ window appears as `capacity_b` in the Capacity block and as the reference in the Window block. Full performance metrics that quantify raw lifter performance are in Appendix Table B.10.

Capacity. Four model sizes were trained with identical data and loss settings (Table 3.4). Roll, steer, and crank RMSE improve from Small to Large: $1.7^\circ/4.4^\circ/13.7^\circ$ for `capacity_s` against $1.1^\circ/2.3^\circ/3.9^\circ$ for

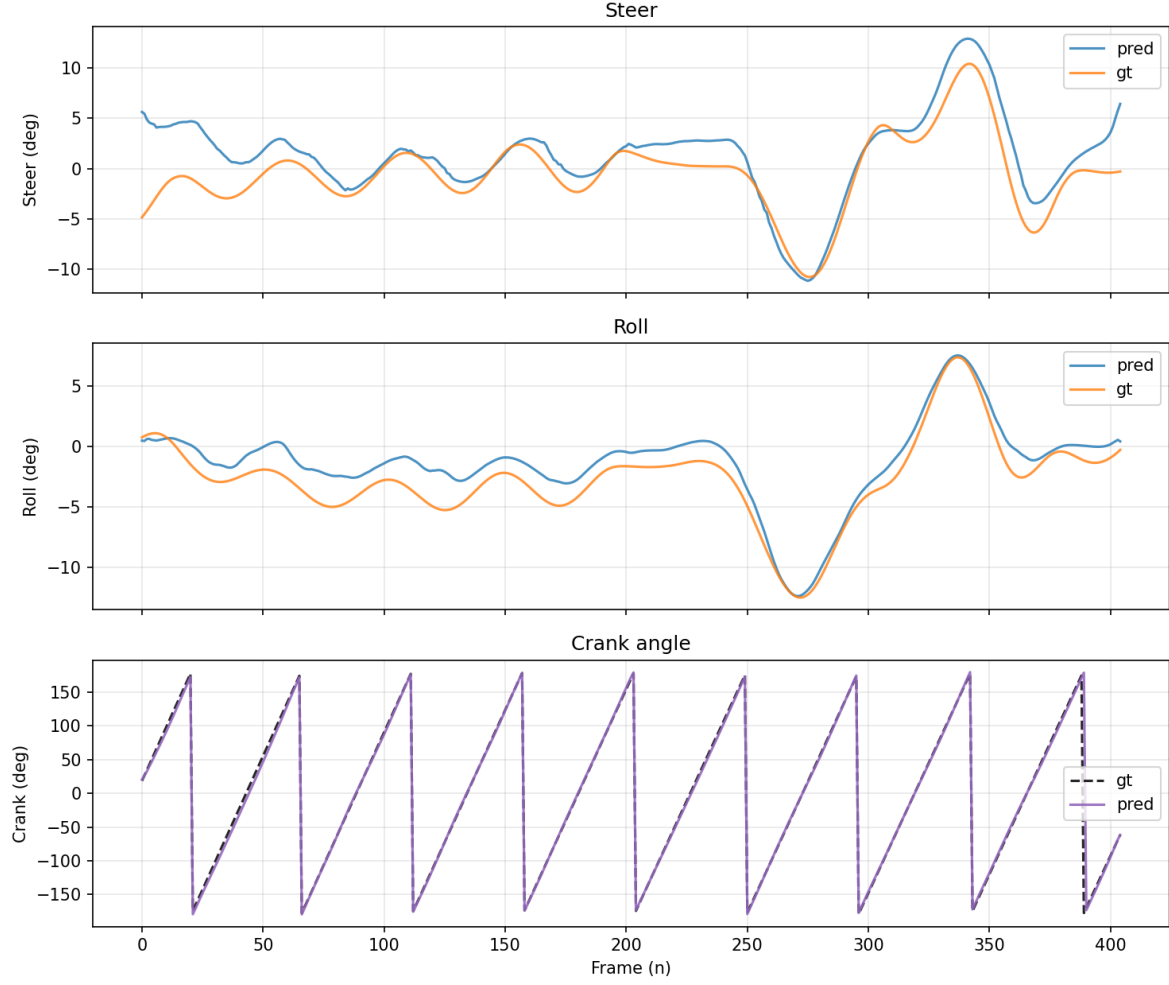


Figure 4.2: Example kinematics estimation: predicted versus ground-truth steer (top), roll (middle), and crank (bottom) for capacity_b.

Table 4.2: Bicycle kinematics metrics of different model variants. RMSE values are reported in degrees or degrees/second. Bold values are the best in each group block.

Experiment	MPJPE (mm)	ϕ RMSE	ϕ bias	ϕ r	δ RMSE	δ bias	δ r	θ RMSE	θ bias	θ r	$\dot{\phi}$ RMSE	$\dot{\delta}$ RMSE	$\dot{\theta}$ RMSE
<i>Capacity</i>													
capacity_s	25.00	1.69	0.03	0.994	4.37	-0.35	0.748	13.69	1.43	0.995	7.84	28.08	75.69
capacity_b	17.57	1.35	-0.13	0.995	3.03	-0.00	0.805	5.07	0.19	0.999	6.21	15.17	48.60
capacity_l	15.48	1.09	0.26	0.998	2.30	0.25	0.915	3.88	-0.42	1.000	5.27	12.74	36.31
capacity_x	19.34	1.35	0.01	0.996	2.55	0.33	0.901	5.75	0.09	1.000	6.04	9.33	38.97
<i>Loss</i>													
loss_no_diff	17.48	1.13	0.13	0.997	3.02	-0.56	0.822	7.39	0.57	0.999	5.99	16.16	47.11
loss_no_nmpipe	17.50	1.25	0.13	0.997	2.59	-0.17	0.878	7.77	-0.50	0.999	5.63	13.54	49.03
loss_no_velocity	17.59	1.40	-0.45	0.996	3.55	-0.61	0.775	9.77	1.05	0.996	11.98	69.53	116.61
<i>Augmentation</i>													
aug_no_flip	15.16	1.17	-0.51	0.997	2.38	0.09	0.897	6.76	0.16	0.999	5.85	16.11	59.18
aug_noise2d	18.47	1.64	0.15	0.993	3.00	-0.43	0.866	10.12	-1.82	1.000	6.86	16.39	50.73
<i>Window</i>													
window_t27	14.68	1.13	-0.08	0.998	1.73	-0.43	0.969	8.31	-0.48	0.998	6.78	14.80	152.43
window_t81	14.32	1.22	-0.03	0.994	2.15	-0.00	0.943	3.27	0.40	0.996	8.31	19.51	49.44
window_t121	15.53	1.34	0.18	0.997	1.87	-0.18	0.950	3.82	0.53	0.989	5.78	12.05	28.19
window_t162	18.89	1.63	-0.28	0.994	2.52	0.03	0.902	4.17	-1.60	1.000	7.11	14.14	29.76
<i>Dynamics</i>													
dyn_roll	20.64	1.07	0.09	0.998	4.08	-0.31	0.736	8.74	-1.44	0.996	6.21	22.70	94.03

capacity_1. Extra-large breaks this trend: all three angle RMSEs are higher than Large (1.3°, 2.6°, and 5.8°), although its steer-rate RMSE of 9.3 deg/s is the lowest of the four. Steer correlation follows

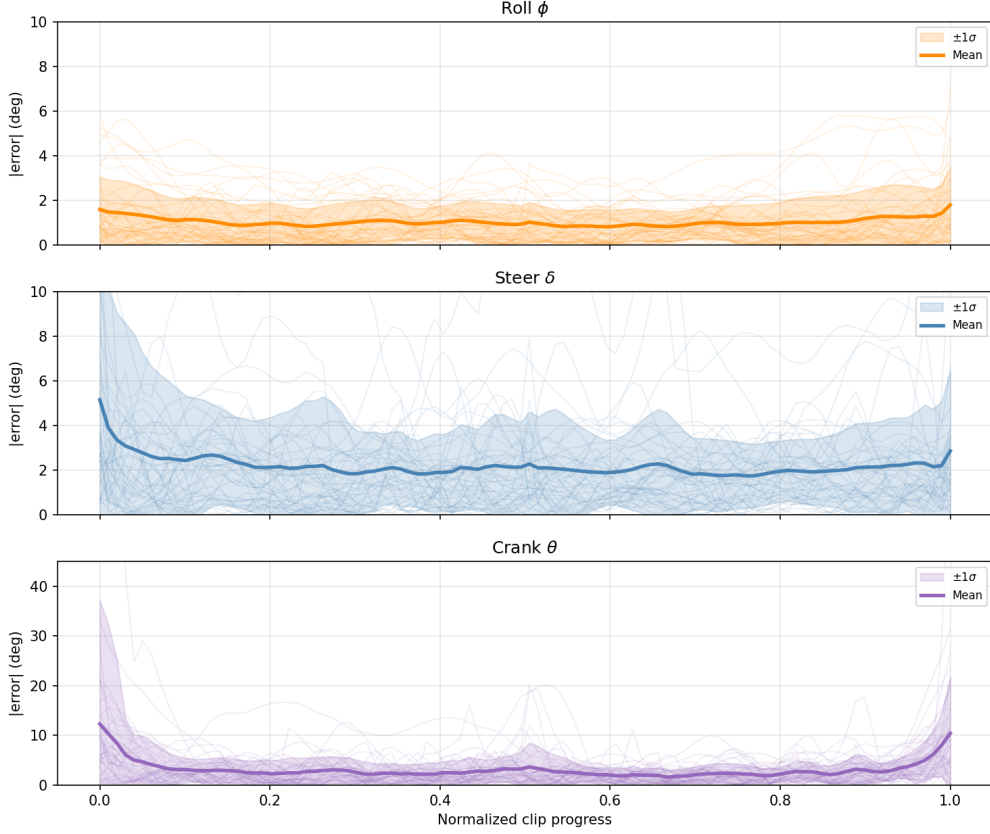


Figure 4.3: Absolute angle error versus normalised clip progress for `capacity_b` (110 validation windows).

the same ordering as steer RMSE, rising from 0.748 for Small to 0.915 for Large. Bias stays below 0.5° for roll and steer at every capacity; the largest crank bias in this block is 1.4° for Small. The angle ordering matches the MPJPE ordering, with the exception of steer, where Extra-large scores between Base and Large while its MPJPE is above both. Latency and compute figures for the four sizes are reported in Appendix Table B.8.

Temporal window. The window experiments retrain the Base configuration with $T \in \{27, 81, 121, 162\}$ frames, the $T = 243$ default is the `capacity_b` row. Steer RMSE is lowest for the shortest windows, at 1.7° for $T = 27$ and 1.9° for $T = 121$, against 3.0° for the default. Steer correlation is above 0.9 for every window, compared with 0.805 at $T = 243$. Crank behaves differently: RMSE is 8.3° at $T = 27$, drops to 3.3° at $T = 81$, and stays below 4.2° for the longer windows. The crank-rate RMSE at $T = 27$ is 152.4 deg/s , the highest value in the table, while $T = 121$ reaches the lowest at 28.2 deg/s . Roll is the least sensitive of the three angles, varying between 1.1° and 1.6° RMSE across all window lengths.

Loss terms. Three runs each remove one term from the default training objective (Section 3): the unsupervised frame-difference term (`loss_no_diff`), the normalised position term (`loss_no_nmpjpe`), and the supervised velocity term (`loss_no_velocity`). Removing the frame-difference or normalised position term leaves roll and steer within 0.3° of the default, while crank RMSE rises from 5.1° to 7.4° and 7.8° respectively. Removing the velocity term raises all three angle RMSEs (1.4° , 3.6° , and 9.8°) and produces the largest rate errors of the three loss runs: roll-rate RMSE rises from 6.2 deg/s to 12.0 deg/s , steer-rate RMSE from 15.2 deg/s to 69.5 deg/s , and crank-rate RMSE from 48.6 deg/s to 116.6 deg/s .

Data augmentation. Two runs change the augmentation of the two-dimensional input during training: `aug_no_flip` disables the default horizontal flipping, and `aug_noise2d` adds Gaussian noise with $\sigma = 0.02$ to the normalised keypoints. Without flipping, roll and steer RMSE drop to 1.2° and 2.4° , crank

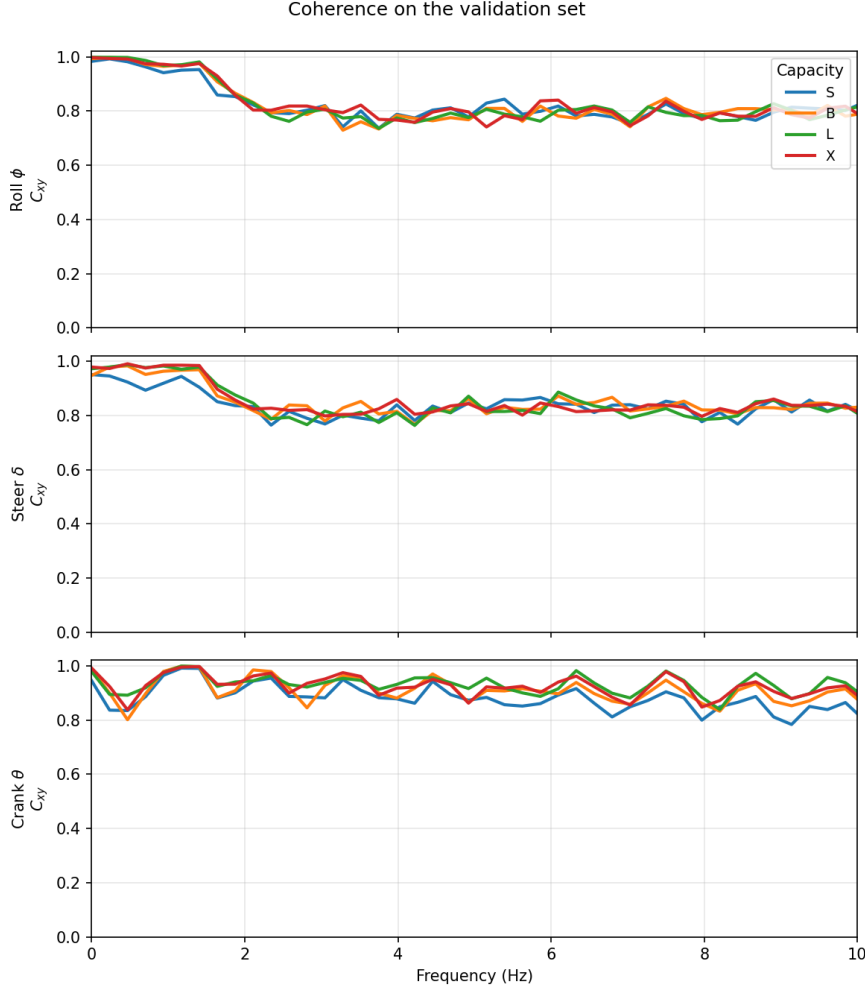


Figure 4.4: Mean magnitude-squared coherence between predicted and ground-truth angles on the validation set (capacities S/B/L/X).

RMSE rises from 5.1° to 6.8° , and MPJPE decreases from 17.6 mm to 15.2 mm . With added noise, roll RMSE rises to 1.6° , steer stays at 3.0° , and crank reaches 10.1° with the largest crank bias in the table (-1.8°).

Roll supervision. The `dyn_roll` run adds a supervised roll-angle term to the default objective (Section 3). Roll RMSE decreases from 1.4° to 1.1° , the lowest roll error in the table together with `capacity_1`. The other outputs move in the opposite direction: steer RMSE rises to 4.1° with correlation dropping to 0.736, crank RMSE rises to 8.7° , and MPJPE increases to 20.6 mm . Runs that added steer supervision did not converge to stable models; those rows are omitted from Table 4.2 and listed in Appendix Table B.10.

4.4. Keypoint estimation accuracy

The three-dimensional lifter turns the normalised two-dimensional keypoints into three-dimensional positions in the camera frame, measured relative to the bottom bracket. The bottom bracket itself therefore has zero error by definition. For the Base model, mean MPJPE is 17.6 mm : rear-frame landmarks are the most accurate (13.2 mm) and the handlebar group the least (21.8 mm) (Appendix Figure B.6). This follows the same pattern as in the two-dimensional errors of Section 4.1. 17.6 mm is double the accuracy of the original PoseMamba result on human benchmarks where it had a MPJPE of 35.5 mm .

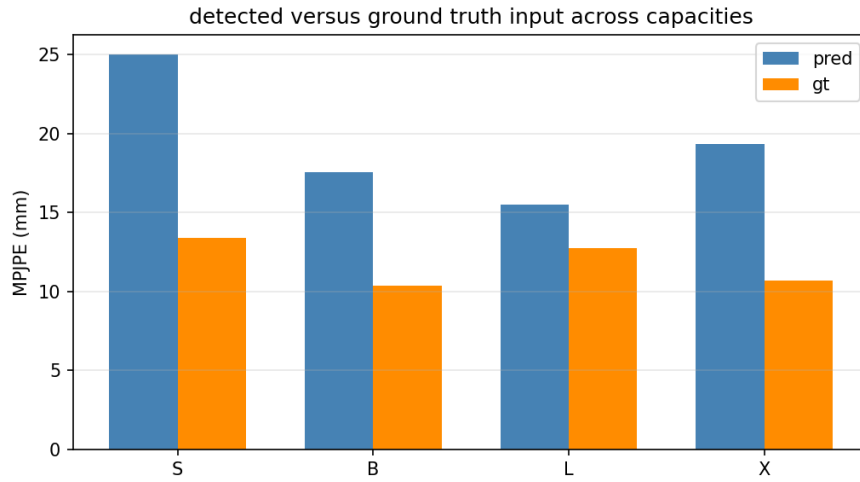


Figure 4.5: Mean MPJPE for detected versus ground-truth two-dimensional training at each capacity.

4.4.1. Detected versus ground-truth two-dimensional input

To measure how much error the first two stages add, the same four model sizes were retrained on ground-truth two-dimensional keypoints taken directly from Blender, skipping the detection stages (PoseMamba_f243s81_gt). Figure 4.5 compares the mean MPJPE of both training variants at each size. With ground-truth input, MPJPE is lower at every size: Base drops from 17.6 mm to 10.3 mm and Large from 15.5 mm to 12.8 mm. The difference between the two variants is largest for Small (11.6 mm) and smallest for Large (2.7 mm). For the Base model, about 41% of the total error disappears when the detection stages are bypassed.

4.5. Qualitative examples

Figure 4.6 shows three frames from different scenes. In the first two rows, the two-dimensional keypoints and three-dimensional reconstruction agree closely with ground truth. In the third row, the frame is reconstructed well but the handlebar segment is offset from the dashed ground truth line, consistent with the higher handlebar errors in Sections 4.1 and 4.4.

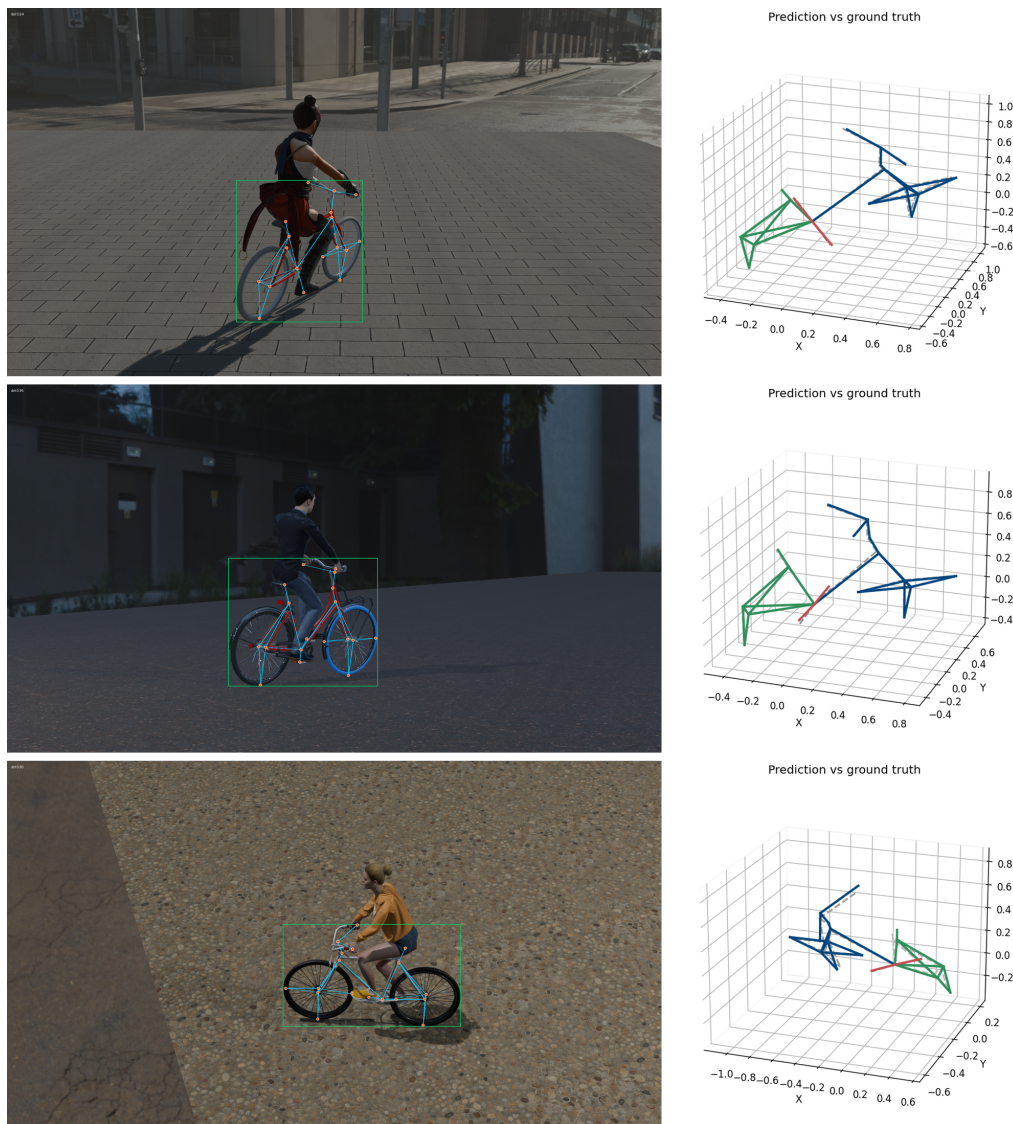


Figure 4.6: Three validation examples: input image with detected box and two-dimensional keypoints (left), lifted three-dimensional skeleton (opaque) versus ground truth (dashed) (right). Predictions appear distorted due to scaling.

5

Discussion

The research question asked whether a temporal state-space keypoint pipeline, trained only on synthetic clips, can recover roll, steering, and crank angle from monocular video with accuracy useful for kinematics analysis. On the validation set the base configuration recovers roll with an RMSE of 1.4° , steering with 3.0° , and crank with 5.1° . For roll and crank that is well inside the range a dynamics study would need, steering angle is more dependent on what type of study is done as it is the weakest of the three. The answer to the research question is however positive: the pipeline recovers the roll and pedalling motion reliably, and the steering angles with enough fidelity to follow the general sense of direction but does not expose small corrections.

The MPJPE metric shows good transfer from humans to bicycles of the models used in the pipeline. MPJPE reached half the value the original PoseMamba paper achieved on human benchmarks (17.6 mm versus 35.5 mm). It matters here only because the angle extractor cannot reject error: keypoint noise passes straight into the angles at a rate set by the baseline it is measured over (Section 2.3.2).

5.1. Kinematic estimation

As mentioned in the Methods, the three angles are not predicted by the network. They are computed from the reconstructed keypoints by the algorithms of Section 3.6. The results therefore not only show the predicted angles, but also the capability of the algorithms to deal with the noise in the keypoint predictions.

Roll is the most accurately predicted, and it's likely due to how it is calculated. The midplane is fitted by principal component analysis over seven keypoints, so the estimate is an average over several of the best-reconstructed points and the sideways noise on any individual keypoint is largely cancelled. A good indicator that the lateral noise is rejected, is the very low bias the roll estimate has. Steering is roughly twice as inaccurate, and while still usable, it fails on both counts that help roll. Its rotation axis comes from the two head-tube keypoints alone, which are very close together and that Eq. (2.4) shows amplifies error. Also, the directions compared against it involve the handlebar landmarks, which were identified as the noisiest points in the pipeline. The lower correlation of the steering angle predictions ($r = 0.805$ against 0.995 for roll) is consistent with this weaker set of keypoints.

The steering error is also not uniform across its range. The bias of 0.0° hides a steering-angle dependent error: predicted steering angle is estimating worse angles as the actual steering angle rises in magnitude. This might be caused by the training data distribution having straight riding and small turns included more than harder turning manoeuvres. This also explains the lower correlation seen in the steering angle predictions. The consequence of this slope is that higher steering angles are understated by the models. Roll shows no such effect, with a slope $16\times$ smaller than the steering angle predictions. The angle magnitudes of the roll in the training data is higher than the steering which could explain the improved performance in the Bland–Altman plots.

Crank behaves differently from the other two. Its correlation with the reference is almost perfect and

its bias is 0.2° , so the pedalling frequency is tracked well and without a systematic offset. Its RMSE of 5.1° is nonetheless the largest of the three, which has to do with the small keypoint set used for estimating this angle. It faces a similar issue to the steering angle prediction where it feeds a lot of noise from keypoint prediction into the angle calculation. The rate error amplifies this jitter inherent to differentiating a noisy signal, with crank at 48.6 deg/s against 6.2 deg/s for roll.

The time series curves highlight a major problem with the estimation in Figure 4.2. Steering error starts at 4° to 5° and settles near 2° once the model has past and future frames to draw on. Similarly, the crank angle estimates are worse near the borders. This is a direct consequence of how the lifter processes the sequence. Frames at the edge of a window have context on one side only, so the model cannot make full use of its bidirectional architecture there. Roll shows different behaviour which does not show this worse prediction at the start, but it has an offset in the first half of the window. This could also be an effect of the principal component analysis averaging out the noise in the keypoints on the edges. These errors can be overcome by discarding the first and last section of predictions, but this would limit the workable range of the estimator for later purposes.

Interestingly, all three angles do show good agreement in the coherence plots below roughly 1.5 Hz and settle to between 0.8 and 0.9 above it. Slow lean and steering changes are therefore reproduced accurately, but faster content is recovered less completely. For analyzing traffic accidents this is likely not a problem as the motions are usually quite large. This does however present a problem for one-sided accidents where the frequencies become higher such as a “wobble” which have a frequency of 6 Hz or higher (Plöchl et al., 2012).

5.2. Error origin

The error present in the kinematics is directly tied to the performance of each individual stage of the pipeline.

The first stage of bounding-boxing the bicycle looks poor according to the metrics. RF-DETR finds the bicycle in 98% of static images, yet the mean overlap with the ground-truth box is only 0.46 and no predicted box exceeds 0.75 . The combination of a near-perfect detection rate and no well-fitting boxes points to a mismatch between the COCO box convention the detector was trained on and the convention used in the Blender annotations, rather than to a detector that fails to locate bicycles. This disagreement was likely not a problem for the 2D detector as it uses generous margins that extend the box boundaries. As long as the bicycle is within the bounding box, RTMPose should be able to do its 2D keypoint estimation. Though, a box that is too loose will likely degrade the accuracy of the estimations. Similarly, the bounding boxes not completely matching is not an issue as the PoseMamba training data was made using only the RF-DETR detections. No ground-truth boxes were used for this stage, except for the ground-truth retrained models to isolate PoseMamba performance. Missed detections would be detrimental to the performance, but 98% is more than adequate for properly training PoseMamba, and could have also introduced noise required for training dynamics properly.

The two-dimensional keypoint estimation stage is where the pipeline loses accuracy that it never recovers. Overall localisation is good, but the steering group is by far the worst at 47.1 px against 36.6 px overall. This has to do with the visibility of the handlebar keypoints as seen in the visibility breakdown. Occluded keypoints are localised more than three times less accurately than visible ones (115.4 px against 36.2 px). The lateral handlebar keypoints are always occluded, as the rider’s hands grip them during cycling. Therefore these points almost always have to be inferred rather than observed. This makes the points inherently ambiguous and difficult to locate. Another characteristic that makes detection difficult is that the handlebar keypoints are defined as the middle of the grip, and the grips vary in length and thickness across the four bicycle models.

The inaccuracies are also present in the 3D lifting output. Rear-frame keypoints are reconstructed to roughly 13 mm MPJPE while the handlebar group reaches 22 mm . PoseMamba therefore cannot filter out the noise on these points, and it cannot infer their positions from the other parts of the frame either. Because steering is derived from these keypoints, the error is highly visible in the steering angle estimation compared to roll derived from the rear-frame.

Retraining on ground-truth two-dimensional input isolates PoseMamba from the earlier detection stages.

For the base model, about 41 % of the total keypoint error disappears when the detection stages are bypassed, with MPJPE falling from 17.6 mm to 10.3 mm. The gain correlates strongly with capacity: the small model improves by 11.6 mm while the large model gains only 2.7 mm. Small capacities are therefore less capable of rejecting input noise, whereas larger models seem more capable of that. This result shows that smaller models are capable of accurate predictions, if the earlier detection stages perform well. If the detection stages cannot be improved more, a larger model can better handle the input noise.

Lifting model capacity trades accuracy against computation, where more computationally heavy models provide higher accuracy. The exception is the extra-large model which has worse performance on the detected input on all three angles than the large model, despite having twice the feature width. Overfitting is the most likely explanation. Further evidence of overfitting is found in the ground-truth retrained model that does perform better than the large model. Without the noise in the input data, the extra-large model can likely completely fit the motion distribution inside of the parameters. Hardware limits forced the extra-large model to train with a batch size of two instead of four, and a high-capacity model on a small training set with a small batch size is known for memorizing rather than generalising. Large gives the best angles and is likely the best candidate for implementing into the final version of the pipeline.

5.3. PoseMamba variations

The neural state-space lifter was chosen because its recurrence propagates an internal state forward in a way that is similar to physical state-space models, which was hypothesized to suit the motion of a bicycle well. The output of the lifter is good, but it's difficult to say if the state-space structure is the reason for this performance as some experiments show opposite effects.

The window size experiments show the reverse of what the original PoseMamba paper reports. Shorter windows gave better steering angles, with $T = 27$ reaching 1.7° against 3.0° at the default $T = 243$. Crank behaves differently, where it performed worst at $T = 27$ and best at $T = 81$ but longer windows also do not show further improvement. For the human case longer context normally helps estimate keypoints more accurately, so this result is strange. The most likely cause of this effect has to do with the dataset rather than bicycle motion. Only 43 longer unique trajectories were available. The windows for each model were generated from these 43 trajectories, which means the shorter window models had substantially more window samples available for training. The solution to fixing this difference could be as simple as generating an equal number of windows for each model. Generating infinite amounts of data is theoretically possible, but the variation within this data would maintain the same distribution with just four unique bicycle frames. Therefore to properly generate the required amount of data, more variation in the training data should be present.

The loss experiments show that temporal stability across frames is supplied by supervision rather than by the state-space model architecture. Removing the supervised velocity term let the angle RMSE remain similar, but multiplied the rate errors three to four times, with steering rate rising from 15.2 deg/s to 69.5 deg/s and crank rate from 48.6 deg/s to 116.6 deg/s. Removing the unsupervised smoothing term or the normalised position term left the rate errors within a few percent of the default. If the state-space recurrence truly ensured the frame-to-frame continuity that the mass and inertia of a real bicycle, removing a supervised velocity term should not have this large of an effect. These experiments show that the model is capable of reconstructing pose well, but that the frame-to-frame stability is learned by a loss function and not implicit to the architecture.

The flip augmentation behaves opposite to the convention it was inherited from, and this is worth stating plainly because it is a case where a human pose default is actively harmful for bicycles. Disabling horizontal flipping improved roll, steering, and MPJPE together. Mirroring a human body produces another valid human pose at no cost, which is why the augmentation is standard. Mirroring a bicycle scene produces configurations that are geometrically valid but rare in practice, so the lifter spends capacity modelling a pose distribution it will never encounter. Adding Gaussian noise to the two-dimensional input points the same way: the detected input already carries realistic noise, and injecting more degrades the fit rather than regularising it. Crank is the exception: it degrades in both runs, including the one where roll, steering, and MPJPE all improve. The pedal landmarks apparently benefit from

augmentation variety in a way the frame landmarks do not, which is consistent with them being the only keypoints whose position is genuinely periodic rather than fixed to the frame.

The direct angle-supervision experiments failed, but there is information to be learned. Adding a roll loss did reduce roll RMSE from 1.4° to 1.1° , but MPJPE rose to 20.6 mm and both steering and crank estimates got worse. The network did this by simply putting more emphasis on the rear-frame keypoints to improve the roll, but sacrificed front wheel and handlebar accuracy to do so. Supervising steering directly caused the training to not properly converge to a stable solution. This was likely caused by the steering loss term causing a highly non-convex and rugged cost landscape, on which the optimizer could not find an optimum. The current implementation of the loss function simply combines multiple keypoints into a single group, a worse way than optimizing for each keypoint individually. This does not show that physics-informed supervision cannot improve kinematics estimation, only that implementing these losses on the intermediate keypoint representation does not work.

5.4. Comparison with prior work

Comparing the keypoint accuracy of this pipeline to the original PoseMamba results on humans shows great promise. The 17.6 mm against PoseMamba's 35.5 mm indicates that the 2D to 3D lifting approach is a good choice for bicycles. Direct comparison is however difficult due to the different domains. The original PoseMamba was trained on huge real datasets such as Human3.6M, whereas this work uses only synthetic data in a relatively small quantity. Also, the current model trained for bicycles was never evaluated on real data, which could show that the limited frame quantity hinders real performance.

A more meaningful comparison is available against the monocular estimator of Corral-Soto et al. (2025), which targets the same object, two of the same angles and is trained solely on synthetic data. They achieved roughly 25° and 31° angle error on the steering and crank angle respectively. The present pipeline reaches 2.1° and 3.0° which are an order of magnitude better, although roll is not directly comparable. There are two major design decisions that could explain this difference in performance. The first, and probably the most important, is the use of temporal context. Corral-Soto et al. (2025) used only single frames, while PoseMamba leverages the temporal context to solve the depth ambiguity problem. The second is the direct regression to angles compared to solving for the keypoints and then calculating the angles. The two methods are evaluated on different datasets, so these differences are just informative, not to be taken as absolute.

However, the single-shot alternative should not be dismissed based on these two methods alone. A neural state-space model can have additional output network layers that predict angles directly. These can have physical constraints built into its output, which can be a method for introducing the physics-informed loss function properly.

5.5. Limitations

The biggest limitation of this work is that training and evaluation are both entirely synthetic made from the same 43 unique trajectories. No real monocular bicycle footage with three-dimensional ground truth was used, so the good performance shown on synthetic data cannot be directly seen as evidence for good performance on real footage. The difference between real and synthetic footage is not the same for each stage. The bounding box detector and 2D keypoint estimator see a difference between rendered Blender scenes and real camera footage. The lifter is affected by differences in real world vs simulated riding, and is subject to more unique bicycle frames in both scale and shape. Every quantitative claim in this thesis should be read with that in mind but this distinction lets future work on the pipeline focus on solving smaller problems. The unique bicycle frame problem can, for example, be solved by having a randomized bicycle frame generator.

It is important to address what the data generator does and does not solve, since generating an infinite amount of labelled data on demand appears to solve the data scarcity problem entirely. What is missing is a real-world distribution of bicycle geometries, riders, and scenes, not more images. The generator fixes the annotation part of that gap perfectly without manual annotation error, but the real-world distribution part only as far as the assets and controllers allow. Without solving the realism part of the synthetic data, generating more data will not improve the models performance on real data.

That said, the dataset is likely too small to train the models properly. The lifter training data was only 94 101 frames while the original PoseMamba was trained on Human3.6M containing 3.6 million frames in addition to other datasets. This limits confidence in the finer details of the variation experiments. The window-length and augmentation findings in particular may depend on large differences in training dataset size used. Also, it is the most likely explanation for the extra-large model's failure to improve on the large model, as was seen in the original paper.

The physics of the simulation were not validated against real bicycle motion. Trajectories are generated in a multibody physics simulator, with the model using realistic bicycle parameters. But, no formal test of physical plausibility was performed other than filtering out trajectories that fell over. Also, the rider is added for visual occlusion only and is not part of the physics as attempts to include them resulted in unstable models, so the riders mass and inertia are neglected. Crank angle is also directly coupled to rear-wheel rotation, similar to a fixed-gear bicycle. Therefore, coasting with no crank movement is not included in the dataset. Heading is controlled by a direct roll-angle controller rather than by countersteering, so the initiation of a turn does not follow the characteristic countersteering motion pattern. The reason for using the simulator was to produce a realistic temporal prior for the models to learn. The unrealistic parts of the simulator therefore directly transfer into the models learned and affect all result, not only the data generation part.

The rider was available as a source of information but was not directly used by the models. Hand and foot positions, for example, are strongly coupled to steering and crank angle. Using a secondary, readily available human pose estimator could have been used to improve estimates of the, currently noisy, handlebar keypoints. Improving these estimates would benefit steering angle estimation greatly. This was not explored to limit the scope of this thesis to the bicycle itself, but it is a missed opportunity to not have included an integration of the human into the estimation.

Root-relative estimation was, however, a deliberate scope limiting measure that greatly affects the usability of the pipeline for research. Global translation and orientation are not recovered, and roll is expressed relative to the camera rather than to the road. Detailed crash analyses will require global pose alongside local kinematics as without global motion, dynamics are missing crucial information.

5.6. Future work

The biggest priority is to validate the pipeline on real footage and integrate real data into the training datasets. A large real dataset remains impractical. A more realistic approach is to obtain a small set of clips with ground truth from an optical motion-capture system with the same keypoint set used in the synthetic data generator. Such a set would resolve many of the questions posed by the limitations section. It could be used either from the start of training, or used to fine-tune the currently trained models. Integration of a small amount of real data into a synthetic dataset was shown to significantly improve the generalisation of the trained models over only synthetic data (Tremblay et al., 2018)

More work should be put into improving the first two stages as their error directly transfers to the lifter and kinematics estimation stages. Priority should be given to the handlebars specifically as they are clearly the most difficult part to estimate. Several ways exist to do this from simple solutions such as extra training weight to those keypoints, to integrating a parallel human pose estimator specifically for these keypoints. Another simple fix is to align the generators bounding boxing algorithm with the COCO convention. This makes sure the data supplied to the 2D detector matches what the developers of this model intended. The dataset should also grow in both quantity and variety. Particularly in the number of variety in the bicycle frames should be increased, with additional shapes and sizes, perhaps even with a random bicycle generator. For the lifter it is important to increase the realism of the manoeuvres and also include actual crash scenarios that the current stability filter discards.

The lifter could be improved by implementing a thought-out local scanning strategy as was implemented in the original PoseMamba paper for humans. The current model leaves it up to the model to learn the spatial relationships. The window-length variation experiments should be redone once the dataset is larger and includes real data, since the current result is most likely a data-quantity artefact. Boundary degradation can be addressed by using the model in a sliding window mode, similar to a moving average filter. This method would make sure the model always has access to the maximum amount of temporal context, though it would completely discard the boundaries of the measurement. A further

option is to give the lifter access to the detection uncertainty that the current input discards. Similarly, the output heatmap of the 2D detector can be directly used by the lifter, something shown increasing lifting performance by Zhao et al. (2023).

The kinematics output can also be directly regressed in the model, instead of being calculated after the fact. Bittner et al. (2023) showed an increase in performance by doing this. A physics based optimizer could be added in conjunction with this direct regressor to further enhance kinematics estimation accuracy. As the human is a major part of the crash investigation, the work by Gildea et al. (2024) could be combined with the current pipeline to estimate the entire system together. During a crash, the bicycle will likely detach itself from the human, so a separate 6D estimator for the bicycle would still be needed.

6

Conclusion

This thesis set out to make a pipeline tool capable of estimating bicycle kinematics from monocular video to allow investigation into bicycle crashes. The research question driving the development of this tool asked whether a neural state-space keypoint estimator, trained only on synthetic clips, could recover roll, steering, and crank angle from such video with accuracy useful for kinematics analysis.

The answer to the research question is positive, with certain limitations kept in mind. Evaluation of the base model configuration reaches an RMSE of 1.4° for roll, 3.0° for steering, and 5.1° for crank. Roll and pedalling are recovered reliably enough for kinematic analysis. Steering captures the overall direction of the rider's input but has a steering angle dependent error and understates the larger angles, so how useful it is depends on the study. All three angles agree closely with the reference below roughly 1.5 Hz, which covers the manoeuvring range but not the higher-frequency behaviour involved in a wobble. Mean per-joint position error, is 17.6 mm for the base configuration and 15.5 mm for the larger one, both outperforming the original model made for human pose estimation trained on large real datasets. Against the only comparable bicycle method, the steering and crank errors are an order of magnitude lower, which supports the idea of providing more context to the model rather than estimating the angles on single frames.

The experiments also show that all the separate components of the pipeline are important for estimating the final kinematics output. The two-dimensional stage is responsible for a lot of error, mostly coming from difficulty localizing the handlebar keypoints. The neural state-space lifter is unable to filter out these inaccuracies in its input. Bypassing the detection stages removes about 41 % of the keypoint error. The state-space lifter reconstructs the pose well, but its temporal stability is a learned feature instead of being part of the recurrence itself, so the similarity between these models and actual state-space models should not be overstated. Attempts to supervise the kinematics directly during training either traded keypoint accuracy for one angle or failed to converge, so for angle supervision, a separate regression head should be used.

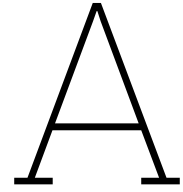
All results were obtained from models trained and evaluated on synthetic data without evaluation on real data. Therefore the results should be interpreted as indicative rather than absolute. Validation against real world data, such as a small set of clips recorded with optical motion capture, is the required next step. Real world data should also be added to the current synthetic dataset, if only in a small amount.

All in all, the currently presented pipeline tool shows great promise as a way to obtain the kinematics of a bicycle from simple monocular video, but requires more work to be used as a tool to gather research grade data.

References

- Adobe. (2026). *Mixamo*. Retrieved August 14, 2026, from <https://www.mixamo.com/>
- Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. <https://doi.org/10.48550/ARXIV.1607.06450>
- Bittner, M., Yang, W.-T., Zhang, X., Seth, A., van Gemert, J., & van der Helm, F. C. T. (2023). Towards single camera human 3d-kinematics. *Sensors 2023*. <https://doi.org/10.48550/ARXIV.2301.05435>
- Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.
- Chen, Y., Tian, Y., & He, M. (2020). Monocular human pose estimation: A survey of deep learning-based methods. <https://doi.org/10.48550/ARXIV.2006.01423>
- Cheng, Y., Wang, B., & Tan, R. T. (2022). Dual networks based 3d multi-person pose estimation from monocular video. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*. <https://doi.org/10.48550/ARXIV.2205.00748>
- COCO Consortium. (n.d.). *COCO Keypoint Detection Task*. Retrieved July 15, 2026, from <https://cocodataset.org/#keypoints-eval>
- Corral-Soto, E. R., Liu, Y., Ren, Y., Bai, D., & Liu, B. (2025). Monocular visual 8d pose estimation for articulated bicycles and cyclists [Huawei Noah's Ark Lab].
- Gildea, K., Hall, D., Cherry, C. R., & Simms, C. (2024). Forward dynamics computational modelling of a cyclist fall with the inclusion of protective response using deep learning-based human pose estimation. *Journal of Biomechanics*, 163, 111959. <https://doi.org/10.1016/j.jbiomech.2024.111959>
- Gildea, K., Mercadal-Baudart, C., Blythman, R., Smolic, A., & Simms, C. (2022). Kinepose: A temporally optimized inverse kinematics technique for 6dof human pose estimation with biomechanical constraints. *24th Irish Machine Vision and Image Processing Conference*, 96–112. <https://doi.org/10.56541/qtuv2945>
- Gu, A., & Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. <https://doi.org/10.48550/ARXIV.2312.00752>
- Gu, Y., & Kamijo, S. (2014). Recognition and pose estimation of urban road users from on-board camera for collision avoidance. *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 1266–1273. <https://doi.org/10.1109/itsc.2014.6957861>
- Huang, Y., Liu, J., Xian, K., & Qiu, R. C. (2024). Posemamba: Monocular 3d human pose estimation with bidirectional global-local spatio-temporal state space model. <https://doi.org/10.48550/ARXIV.2408.03540>
- Ionescu, C., Papava, D., Olaru, V., & Sminchisescu, C. (2014). Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7), 1325–1339. <https://doi.org/10.1109/TPAMI.2013.248>
- Jiang, T., Lu, P., Zhang, L., Ma, N., Han, R., Lyu, C., Li, Y., & Chen, K. (2023). Rtmpose: Real-time multi-person pose estimation based on mmpose.
- Krul, I., Veldhuis, S., & Bregman, A. (2025, August). *Ongevallen met (niet-)elektrische fietsen: Actuele inzichten en trends: Risicogroepen, oorzaken en gevolgen op basis van seh-bezoeken 2020–2024* (No. 1051). VeiligheidNL. Retrieved August 14, 2026, from https://www.veiligheid.nl/sites/default/files/2025-08/def_rapport_ongevallen_met_niet-elektrische_fietsen_2024_13_aug_2025.pdf
- Li, Y., Yang, S., Liu, P., Zhang, S., Wang, Y., Wang, Z., Yang, W., & Xia, S.-T. (2021). Simcc: A simple coordinate classification perspective for human pose estimation. <https://doi.org/10.48550/ARXIV.2107.03332>
- Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *Proceedings of the European Conference on Computer Vision (ECCV)*, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

- Maji, D., Nagori, S., Mathew, M., & Poddar, D. (2022). Yolo-pose: Enhancing yolo for multi person pose estimation using object keypoint similarity loss. <https://doi.org/10.48550/ARXIV.2204.06806>
- Meijaard, J., Papadopoulos, J., & Schwab, A. (2007). Linearized dynamics equations for the balance and steer of a bicycle: A benchmark and review. *Royal Society of London. Proceedings A. Mathematical, Physical and Engineering Sciences*, 463, 1955–1982.
- Mousavian, A., Anguelov, D., Flynn, J., & Kosecka, J. (2016). 3d bounding box estimation using deep learning and geometry. <https://doi.org/10.48550/ARXIV.1612.00496>
- Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., ... Bojanowski, P. (2023). Dinov2: Learning robust visual features without supervision. <https://doi.org/10.48550/ARXIV.2304.07193>
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. <https://doi.org/10.48550/ARXIV.1912.01703>
- Pavlo, D., Feichtenhofer, C., Grangier, D., & Auli, M. (2018). 3d human pose estimation in video with temporal convolutions and semi-supervised training. <https://doi.org/10.48550/ARXIV.1811.11742>
- Plöchl, M., Edelmann, J., Angrosch, B., & Ott, C. (2012). On the wobble mode of a bicycle. *Vehicle System Dynamics*, 50(3), 415–429. <https://doi.org/10.1080/00423114.2011.594164>
- Robinson, I., Robicheaux, P., Popov, M., Ramanan, D., & Peri, N. (2025). Rf-detr: Neural architecture search for real-time detection transformers. <https://doi.org/10.48550/ARXIV.2511.09554>
- Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. 2012 *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. <https://doi.org/10.1109/iros.2012.6386109>
- Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., & Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. <https://doi.org/10.48550/ARXIV.1804.06516>
- Zargarzadeh, M., Ribeiro, A. S. N., Santos, A. M. C., & Rodrigues, R. N. (2026). Health benefits of cycling as a form of active travel: A pilot empirical study. *International Journal of Environmental Research and Public Health*, 23(1), 79. <https://doi.org/10.3390/ijerph23010079>
- Zhao, Q., Zheng, C., Liu, M., & Chen, C. (2023). A single 2d pose with context is worth hundreds for 3d human pose estimation. <https://doi.org/10.48550/ARXIV.2311.03312>



Keypoint definitions and coordinate frames

Table A.1: Bicycle semantic keypoint definitions ($J = 18$).

Index	Name	Description
0	bottom bracket	Centre of the bottom-bracket axle
1	seat stay	Attachment of the seat stays on the seat tube.
2	saddle	Centre of the saddle.
3	upper head tube	Upper end of the steering axis (handlebar stem height on the fork).
4	lower head tube	Lower end of the head tube (junction with the top and down tubes).
5	handlebar left	Left end of the handlebar, middle of the grip.
6	handlebar middle	Mid-point of the handlebar, upper part of the stem.
7	handlebar right	Right end of the handlebar, middle of the grip.
8	front hub left	Left end of the front-wheel axle.
9	front hub right	Right end of the front-wheel axle.
10	front wheel back	Rear-most rim point of the front wheel.
11	front wheel front	Forward-most rim point of the front wheel.
12	front wheel ground	Ground-contact point of the front wheel.
13	rear hub left	Left end of the rear-wheel axle.
14	rear hub right	Right end of the rear-wheel axle.
15	rear wheel ground	Ground-contact point of the rear wheel.
16	left pedal	Centre of the left pedal.
17	right pedal	Centre of the right pedal.

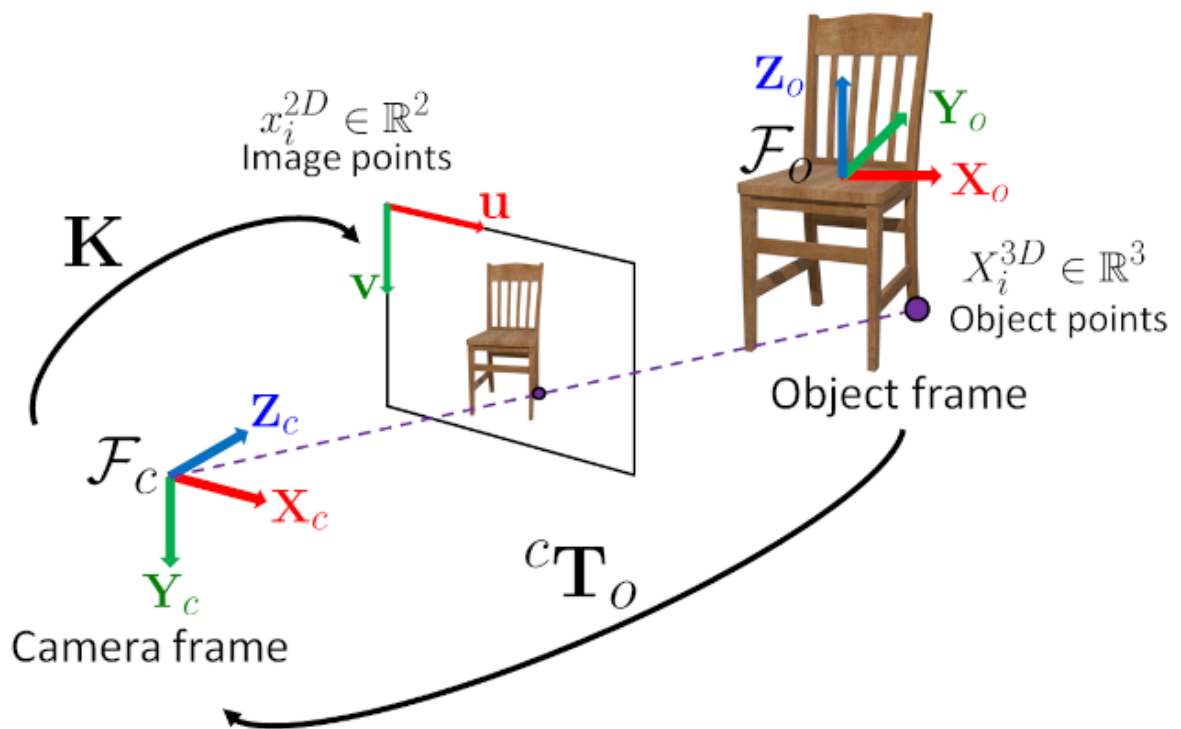


Figure A.1: Opencv pinhole model based coordinate system (Bradski, 2000)

B

Extended quantitative results

This appendix contains supplementary tables and figures referenced from Chapter 4. Headline kinematics findings are stated in the main chapter. These results show the performance of the entire pipeline that did not fit in the main text.

B.1. Detection and two-dimensional keypoint accuracy

The first two stages of the pipeline were evaluated on the static-image validation set from the synthetic bicycle pose dataset (1000 images, fixed camera, 80/10/10 partition). RF-DETR produces a bounding box per image, and RTMPose-L estimates the 18 bicycle keypoints inside that crop.

On the static validation set, RF-DETR detected the bicycle in 98% of images (Table B.1). Mean intersection-over-union (IoU) was 0.46 and only 47% of detections reached $\text{IoU} \geq 0.5$, so predicted boxes were generally loose rather than tightly fitted. At $\text{IoU} 0.5$, recall was 0.96 but precision was 0.47 (F1 0.63): the bicycle was usually found, but the crop was often oversized or shifted. RTMPose on those crops achieved a mean pixel error of 36.6 px, normalised mean error 0.10, and a fraction of 0.91 of keypoints within 10% of the box diagonal (PCK@0.1; Table B.2). Errors were highest on steering landmarks (47.1 px mean) and on occluded keypoints (115.4 px versus 36.2 px visible).

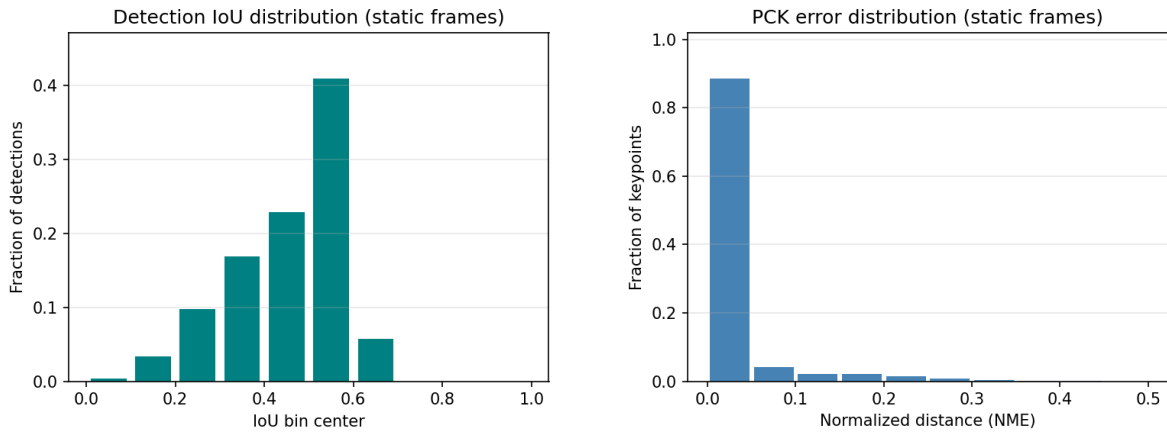


Figure B.1: RF-DETR IoU distribution (left) and RTMPose PCK curve (right) on the static-image validation set.

B.2. Three-dimensional keypoint diagnostics

Four PoseMamba capacities were compared at fixed $T = 243$ with detected two-dimensional input. Capacity is controlled by feature width (`dim_feat`) and state-space depth: Small (64, 10), Base (128, 10), Large (128, 20), and Extra-large (256, 20).

Table B.1: RF-DETR bicycle detection on the static-image validation set (bicycle pose dataset, $N=999$ frames).

Category	Metric	Value
<i>Coverage</i>		
	Detection rate	0.98
<i>Overlap</i>		
	Mean IoU	0.46
	Median IoU	0.49
	Share with IoU ≥ 0.5	0.47
	Share with IoU ≥ 0.75	0.00
<i>At IoU = 0.5</i>		
	Recall	0.96
	Precision	0.47
	F1	0.63
<i>Box geometry</i>		
	Mean centre error	26.7 px
	Mean scale error (area ratio)	0.54
	Mean aspect-ratio error	0.17

Mean MPJPE was 25.0 mm (S), 17.6 mm (B), 15.5 mm (L), and 19.3 mm (X). Large gives the best accuracy, but Extra-large is worse than Large despite having twice the feature width.

B.2.1. Temporal consistency

MPJPE measures per-frame position error but not whether temporal context is used as intended. Mean per-joint velocity error (MPJVE) and mean per-joint acceleration error (MPJAE) assess temporal smoothing at 60 fps (Section 3). Temporal errors follow the same trend as MPJPE: Small is least smooth (MPJVE 89.8 mm/s, MPJAE 3050 mm/s²), Base improves to 54.6 mm/s and 1870 mm/s², and Large is best at 45.5 mm/s and 1588 mm/s² (Table B.7). For the Base model, MPJPE in five equal temporal segments along each clip stayed between 16.9 mm and 18.5 mm, so the $T = 243$ window does not introduce drift toward clip boundaries.

Table B.10 lists all variation runs, including model capacity and ground-truth two-dimensional input baselines omitted from the main-chapter kinematics scorecard.

B.3. Kinematics error by model capacity

Figures B.7–B.9 repeat the absolute error versus clip-progress plot of Figure 4.3 for Small, Large, and Extra-large capacities.

B.4. State-space model interaction maps

B.4.1. Joint--joint spatial relationships

PoseMamba exposes the spatial relationships learned inside the state-space blocks as a spatial relationship map. Figure B.10 shows the map for the Large-capacity model, averaged over all blocks on the validation set. Each cell is the magnitude of correlation between two keypoint channels after spatial scanning; the diagonal is masked so off-diagonal structure is visible.

Four groups stand out in the map. The three handlebar landmarks (handlebar_left, handlebar_middle, handlebar_right) form a tight block with pairwise correlation above 0.95. The upper-frame landmarks (upper_head_tube, lower_head_tube, saddle, and seat_stay) form a second block with similarly high correlation. front_hub_left falls within this block; front_hub_right does not, with correlation near 0.79 to the head-tube and handlebar rows. The three front-wheel rim landmarks (front_wheel_back, front_wheel_front, and front_wheel_ground) form a third cluster, separate from the front hubs. The rear landmarks rear_hub_left, rear_hub_right, and rear_wheel_ground form the fourth group. left_pedal and right_pedal correlate strongly with this block and with the front-wheel rim landmarks.

Table B.2: RTMPose-L 2D keypoint accuracy on RF-DETR crops from the static-image validation set ($N=17,982$ keypoint instances).

Category	Metric	Value
<i>Overall accuracy</i>		
	Mean pixel error	36.6 px
	Mean NME	0.10
	Mean confidence	0.70
<i>PCK (bbox-normalised)</i>		
	PCK@0.1	0.91
	PCK@0.2	0.95
	PCK AUC	0.92
<i>Mean pixel error by skeleton group</i>		
	Frame	31.8 px
	Wheels	33.1 px
	Steering	47.1 px
<i>Visibility</i>		
	Visible keypoints	36.2 px
	Occluded keypoints	115.4 px

bottom_bracket shows the lowest values on the map, about 0.80–0.83 against most other landmarks.

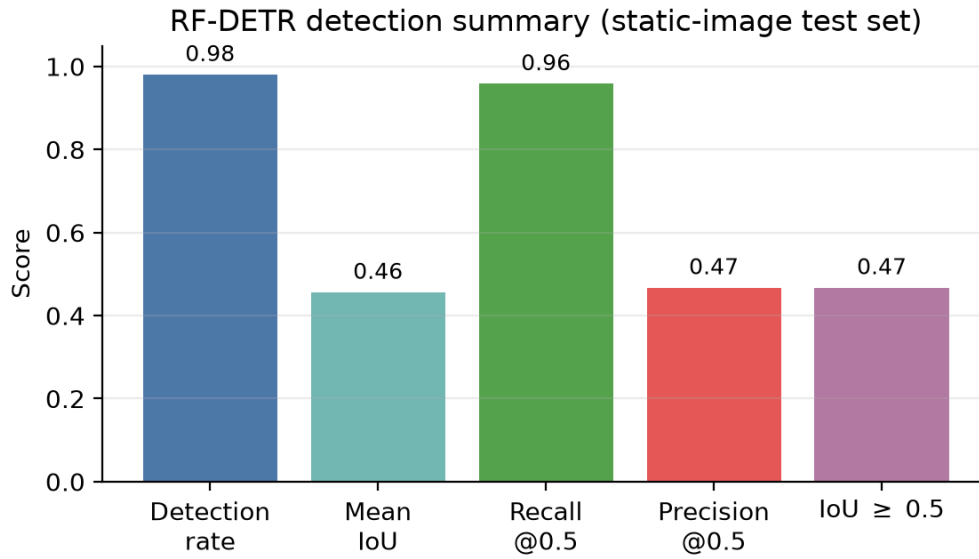


Figure B.2: RF-DETR summary on the static-image validation set: detection rate, mean IoU, recall and precision at IoU 0.5, and share of detections with $\text{IoU} \geq 0.5$.

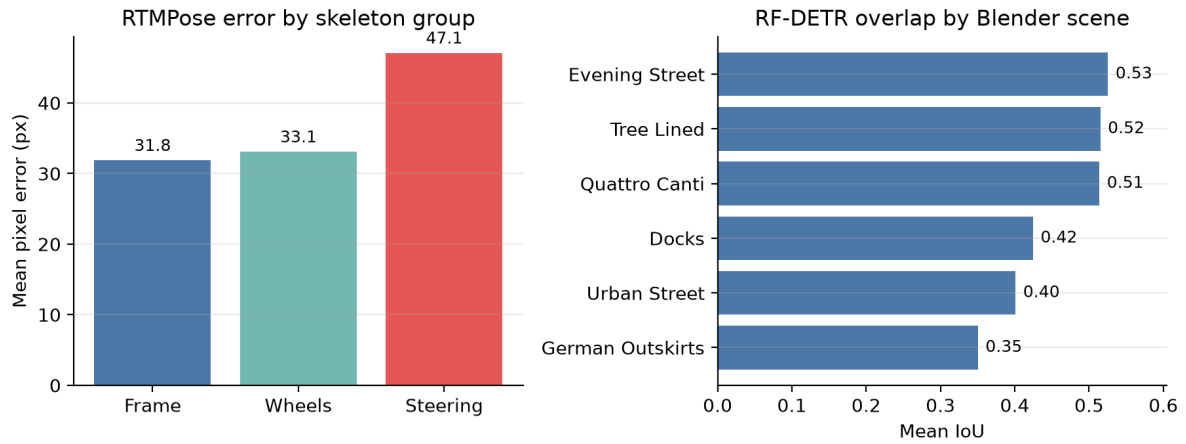


Figure B.3: RTMPose mean pixel error by skeleton group (left) and RF-DETR mean IoU by Blender scene (right) on the static-image validation set.

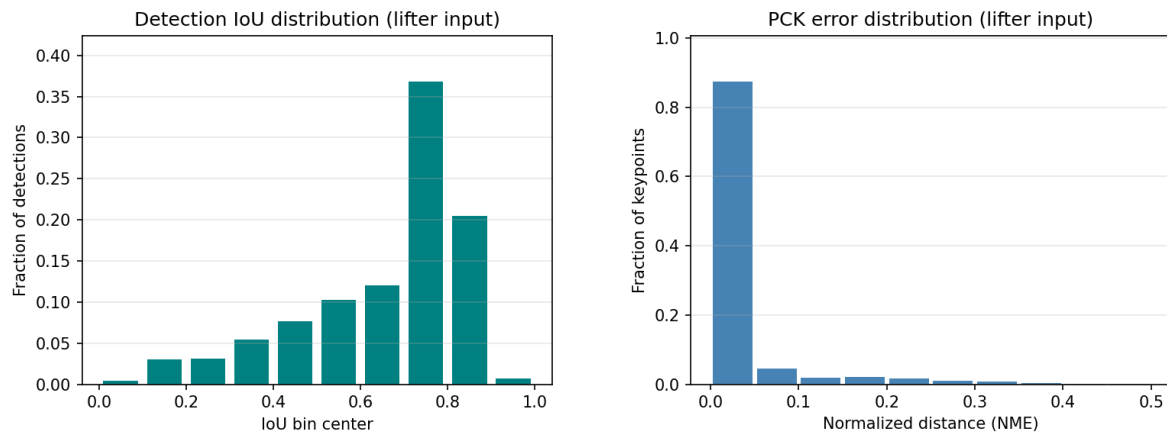


Figure B.4: RF-DETR IoU distribution (left) and RTMPose PCK curve (right) on all frames from the clip dataset used as PoseMamba input.

Table B.3: RF-DETR bicycle detection on all frames from the temporal clip dataset used as PoseMamba input ($N=94,041$ frames).

Category	Metric	Value
<i>Coverage</i>		
	Detection rate	1.00
<i>Overlap</i>		
	Mean IoU	0.66
	Median IoU	0.74
	Share with IoU ≥ 0.5	0.80
	Share with IoU ≥ 0.75	0.45
<i>At IoU = 0.5</i>		
	Recall	1.00
	Precision	0.80
	F1	0.89
<i>Box geometry</i>		
	Mean centre error	22.8 px
	Mean scale error (area ratio)	0.34
	Mean aspect-ratio error	0.18

Table B.4: RTMPose-L 2D keypoint accuracy on RF-DETR crops from the temporal clip dataset ($N=1,692,738$ keypoint instances).

Category	Metric	Value
<i>Overall accuracy</i>		
	Mean pixel error	14.8 px
	Mean NME	0.03
	Mean confidence	0.36
<i>PCK (bbox-normalised)</i>		
	PCK@0.1	0.92
	PCK@0.2	0.96
	PCK AUC	0.93
<i>Mean pixel error by skeleton group</i>		
	Frame	9.3 px
	Wheels	9.6 px
	Steering	28.6 px

Table B.5: PoseMamba 3D lifting results across model capacities using detected two-dimensional input ($T = 243$, validation set).

Model	MPJPE (mm)	NMPJPE (mm)	MPJVE (mm/s)	MPJAE (mm/s ²)	Roll RMSE (deg)	Steer RMSE (deg)	Crank RMSE (deg)
S	25.00	23.94	89.8	3050	1.69	4.37	13.69
B	17.57	17.21	54.6	1870	1.35	3.03	5.07
L	15.48	14.99	45.5	1588	1.09	2.30	3.88
X	19.34	19.14	48.2	1520	1.35	2.55	5.75

Table B.6: Capacity comparison between detected input and ground truth input, specifically trained models

Model	MPJPE detected (mm)	MPJPE GT (mm)	Difference (mm)
S	25.00	13.37	11.63
B	17.57	10.35	7.22
L	15.48	12.76	2.72
X	19.34	10.67	8.67

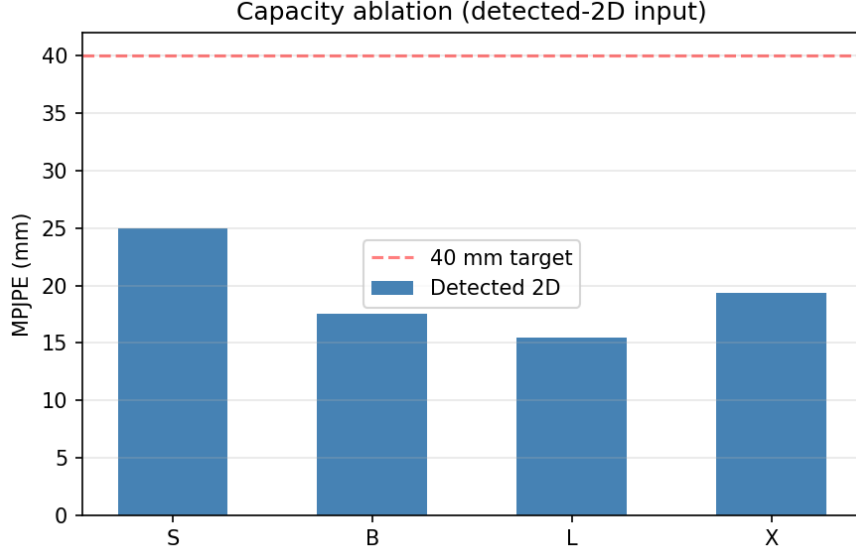


Figure B.5: MPJPE versus model capacity (detected two-dimensional input, $T = 243$).

Table B.7: Temporal consistency across PoseMamba capacities (detected 2D input, $T = 243$, validation set). MPJVE and MPJAE at 60 fps.

Model	MPJPE (mm)	MPJVE (mm/s)	MPJAE (mm/s ²)
S	25.00	89.8	3050
B	17.57	54.6	1870
L	15.48	45.5	1588
X	19.34	48.2	1520

Table B.8: Accuracy and compute across PoseMamba capacities

Model	MPJPE (mm)	MPJVE (mm/s)	Params (M)	MAC (G)	MAC/frame (M)	Latency (ms)
S	25.00	89.8	1.47	5.62	23.1	40.0
B	17.57	54.6	4.74	19.12	78.7	77.5
L	15.48	45.5	9.45	38.23	157.3	165.6
X	19.34	48.2	33.32	116.37	478.9	334.6

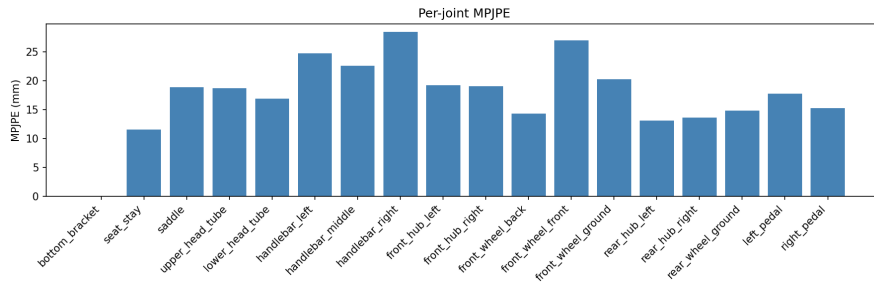


Figure B.6: Per-joint MPJPE for the primary PoseMamba configuration (capacity_b, detected two-dimensional input) on the validation set.

Table B.9: Kinematic recovery across PoseMamba capacities using detected two-dimensional input ($T = 243$, validation set).

Model	Roll RMSE (deg)	Steer RMSE (deg)	Crank RMSE (deg)
S	1.69	4.37	13.69
B	1.35	3.03	5.07
L	1.09	2.30	3.88
X	1.35	2.55	5.75

Table B.10: Full PoseMamba variation experiment matrix. MPJVE and MPJAE are mean per-joint velocity and acceleration errors on root-relative 3D at 60 fps.

Experiment	MPJPE (mm)	NMPJPE (mm)	MPJVE (mm/s)	MPJAE (mm/s ²)	Roll (deg)	Steer (deg)	Crank (deg)
<i>Model capacity ($T=243$)</i>							
capacity s	25.00	23.94	89.8	3050	1.69	4.37	13.69
capacity s GT	13.37	12.63	39.1	1656	0.81	3.09	2.24
capacity b	17.57	17.21	54.6	1870	1.35	3.03	5.07
capacity b GT	10.35	9.67	23.6	1035	0.67	2.28	1.60
capacity l	15.48	14.99	45.5	1588	1.09	2.30	3.88
capacity l GT	12.76	12.12	22.4	922	0.74	2.54	2.22
capacity x	19.34	19.14	48.2	1520	1.35	2.55	5.75
capacity x GT	10.67	10.44	19.5	736	0.78	2.34	1.37
<i>Temporal window length (capacity b)</i>							
window t27	14.68	13.92	66.3	4365	1.13	1.73	8.31
window t81	14.32	13.60	53.3	2411	1.22	2.15	3.27
window t121	15.53	14.93	48.8	2016	1.34	1.87	3.82
window t162	18.89	17.69	56.3	2058	1.63	2.52	4.17
<i>Loss function components (capacity b)</i>							
loss no diff	17.48	16.95	56.7	1930	1.13	3.02	7.39
loss no nmpjpe	17.50	16.96	52.9	1775	1.25	2.59	7.77
loss no velocity	17.59	17.26	151.1	10292	1.40	3.55	9.77
<i>Data augmentation (capacity b)</i>							
aug no flip	15.16	14.41	49.9	1801	1.17	2.38	6.76
aug noise2d	18.47	17.45	63.7	1912	1.64	3.00	10.12
<i>Dynamics supervised losses (capacity b)</i>							
dyn roll	20.64	19.17	69.4	2499	1.07	4.08	8.74
dyn steer	349.60	346.60	373.2	14247	19.34	60.90	102.25
dyn steer roll	57.04	53.69	139.1	4773	8.70	31.38	52.29
dyn steer roll vel	348.79	346.06	378.5	14288	13.59	61.32	98.70

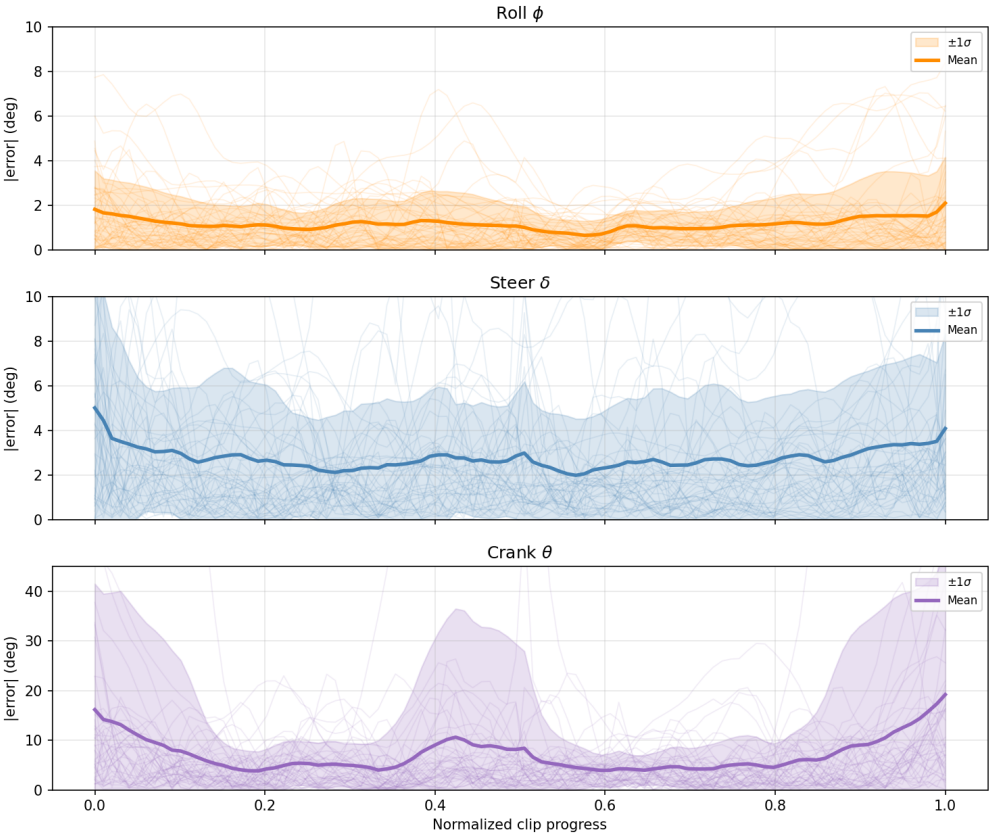


Figure B.7: Absolute angle error versus normalised clip progress (capacity_s).

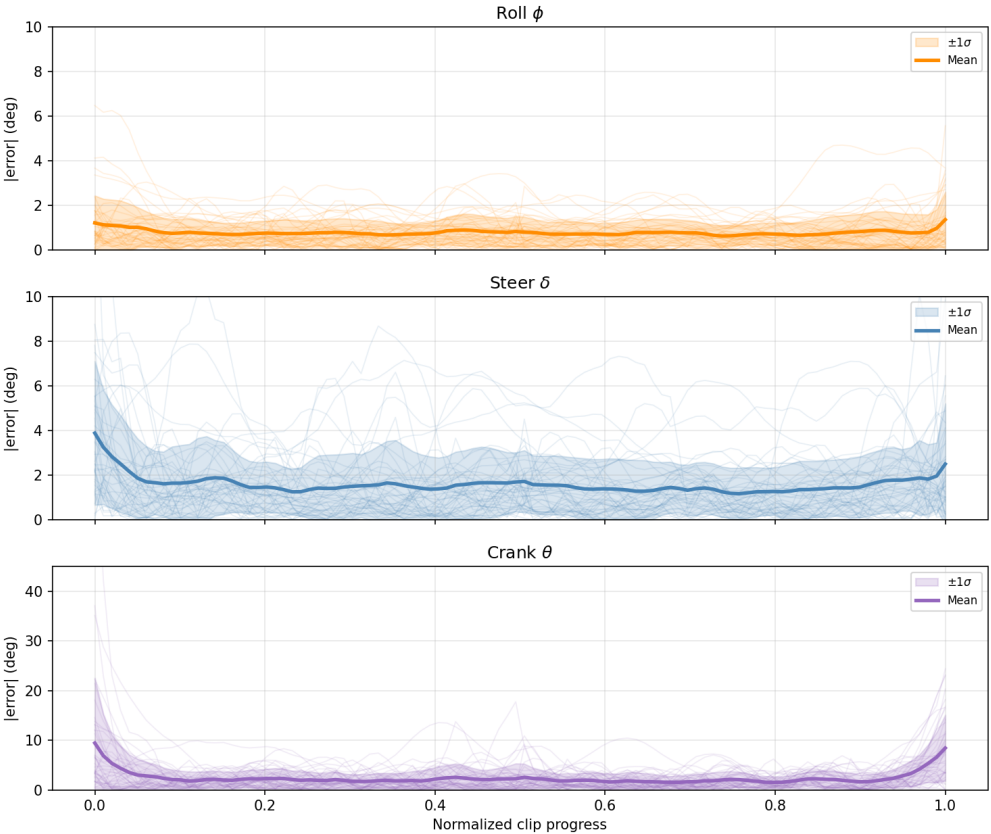


Figure B.8: Absolute angle error versus normalised clip progress (capacity_1).

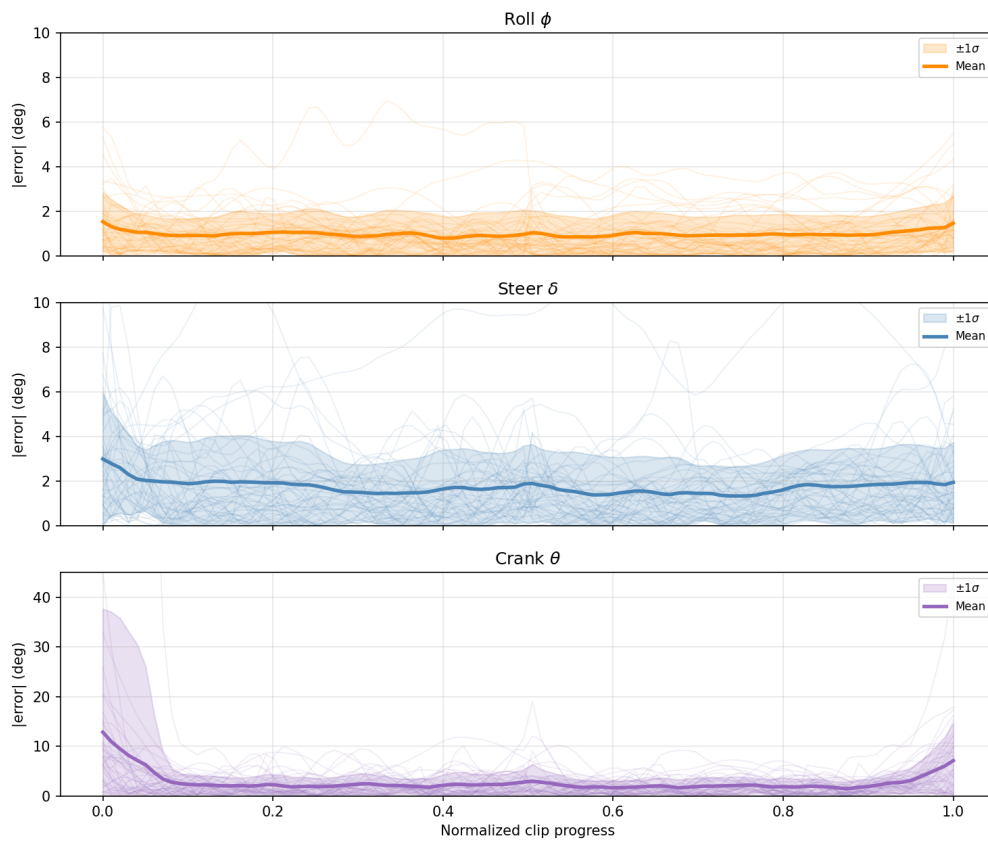


Figure B.9: Absolute angle error versus normalised clip progress (capacity_x).

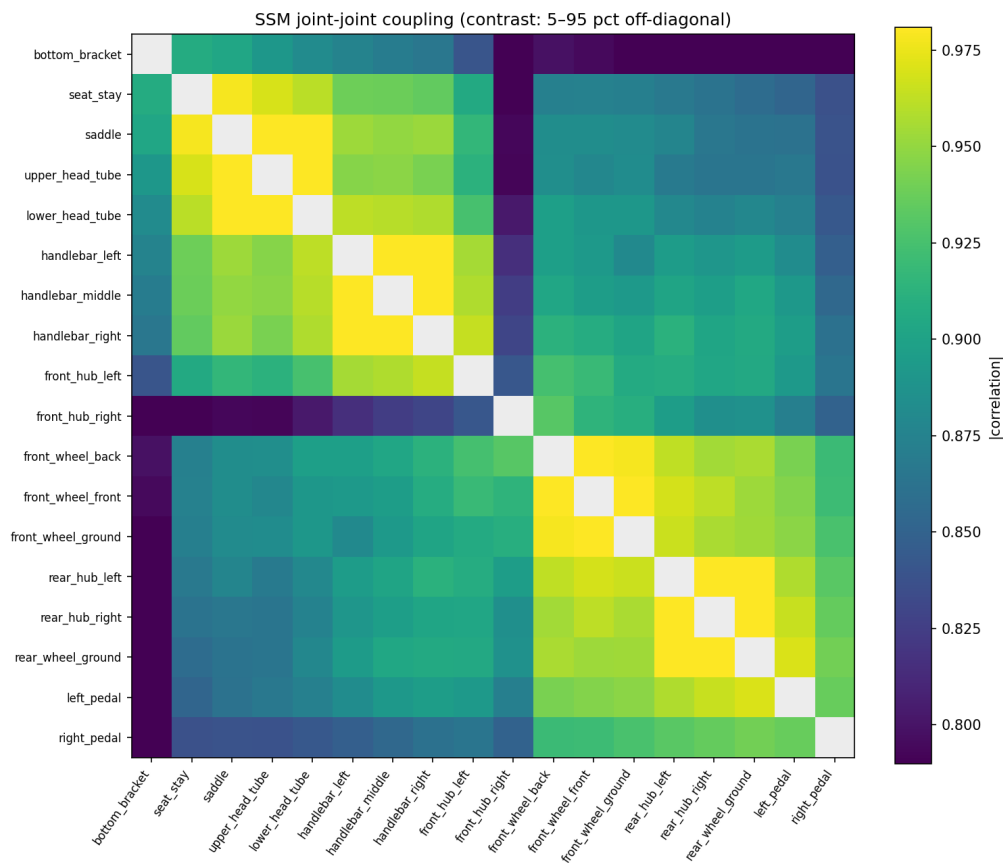


Figure B.10: State-space spatial relationship map for capacity_1 (residual mode), averaged over blocks. The diagonal is masked.

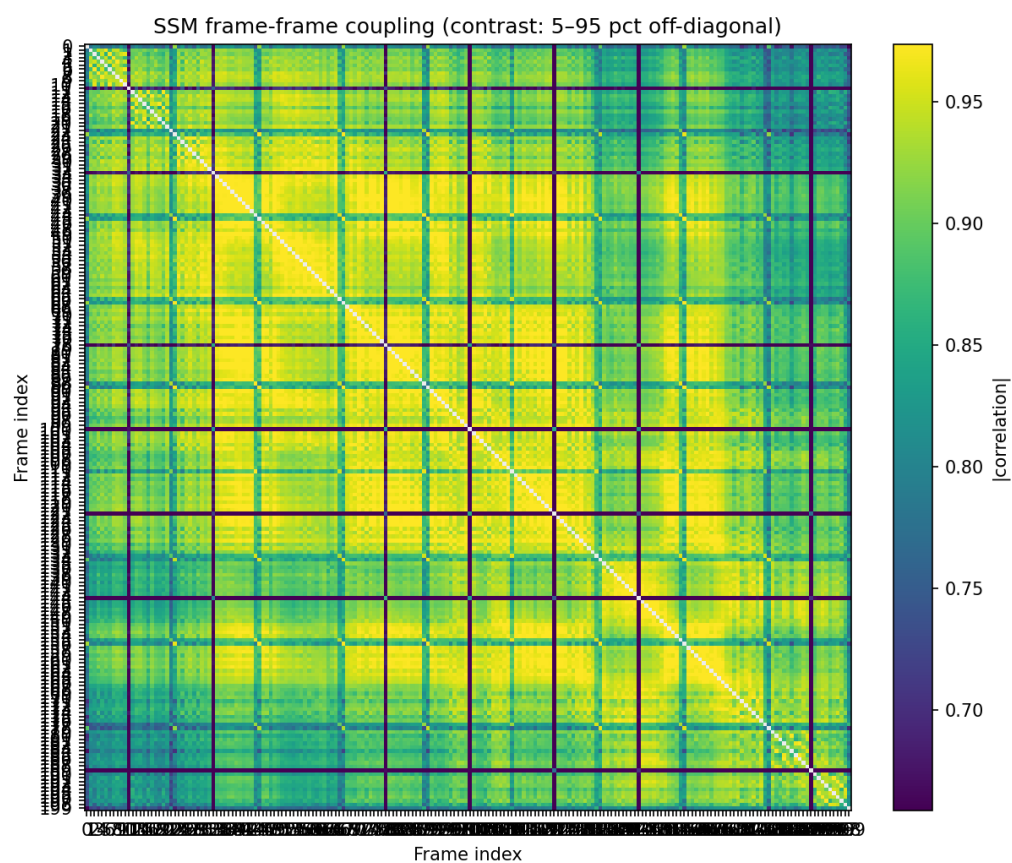


Figure B.11: State-space temporal relationship map for capacity_1, averaged over blocks.