Université Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Informatique**

Arrêtée ministériel : 25 mai 2016

Présentée par

David PAGNON

Thèse dirigée par **Lionel REVERET**
et codirigéee par **Mathieu DOMALAIN**

préparée au sein du **Laboratoire Jean Kuntzmann**
dans **l'École Doctorale l'École Doctorale Mathématiques, Sciences et technologies de l'Information, Informatique**

"Design, evaluation, and application of a workflow for biomechanically consistent markerless kinematics in sports"

"Conception, évaluation, et application d'une méthode biomécaniquement cohérente de cinématique sans marqueurs en sport"

Thèse soutenue publiquement le **"Date de soutenance"**,
devant le jury composé de :

Président
Laboratoire, Président

Rapporteur
Laboratoire, Rapporteur

Examinateur
Laboratoire, Examinateur

Lionel REVERET
INRIA Grenoble, Directeur de thèse

Mathieu DOMALAIN
Institut Pprime, Co-Encadrant de thèse

Invité
Laboratoire, Invité

"To all of you who care about more important stuff than what follows."

Acknowledgements

Should I start this by declaring that these PhD years have been alternatively depressing and engaging, exhausting and stimulating, infuriating and enthralling? This is trite, and true for everyone working through any kind of commitment, PhD student or not. Covid pandemic or not. Child birth or not. Struggles in close friends' and relatives' lives or not. But there it is. Now that it is stated, let me go straight to my acknowledgements.

Above anyone else, I want to thank my mother. She not only had to deal with the difficult task of raising me and putting up with my constant flow of questions, but also with welcoming the four smaller sisters that came after me. As a widow. With debts to pay off, and very little money coming in. Moving every two years, until we settled in for a small apartment in a neighborhood that some would call a ghetto, although we preferred calling it home. And yet, there was always food on the table. Even better, we had no idea how poor we were, because she literally sacrificed her life for ours, and her passions for our interests. This is quintessential Christlike love. We all had the incredible opportunity of doing at least one physical, and one artistic activity, on top of pursuing university level studies. We also learned how to live happily with very little, which I'm starting to realize is a sort of superpower. Most importantly, she made children that all love each other. Now that I'm a father too, I can measure how high she set the bar, and I can only hope to be half as good as her. I can't award her the Legion of Honor she deserves, but at least here is a little bit of recognition! Thank you from all of us, maman.

I also have a deep thought for my father, who tragically passed away when I was still a little child. He did have to struggle with some issues that would eventually cause his death, but I believe he fought until the very end. He is actually the one who taught me a nice lesson of persistence, surely without even trying. A friend and I were racing up a hill, while my father timed us. I lost. We raced again, I lost again. I tried more, and sure enough, I lost every single race. I went to my dad and complained: "I'm tired papa, can we stop?" "Are you tired, really? Very good, it means that you're on your way to make progress!" I paused, and let it sink in for a few moments. And without a word, I went back running. That's how I learned that getting better goes with accepting to suffer a little. Later on, I also realized that out of any bad experience, be it death, you can take away something positive, something that will help you grow. Against all odds, I even made a first professional carrier in sports. I am very grateful for both my parents: I am who I am, with all my quirks and all that's to be loved or to be hated, thanks to them.

So many more people to thank! I'm just getting started, so you will bear with me for a little while, still. But let's start with the sisters. Esther comes just after me, she married an awesome guy from Congo, and is currently raising two wonderful little girls. She is the closest to what my mom was with us (and still is), making anyone feel home at any time, always on the move, taking care of her family during the day and working at nights, juggling countless tasks and thinking it is all just natural. Then comes Déborah, although she didn't come alone since Joëlla followed 10 minutes later. But believe it or not, she is slightly more than a twin. She has a high sense of justice and a desire to be helpful, which made her switch from the arts history field to the mentally challenging health one, so as to be more true to herself. Joëlla also is incredible. She fights every day her own health issues, could not finish high school but still managed to get a bachelor degree a few years later, and she now is a professional violinist, whose empathy perspires through all her plays. I'm on a roll now, and I don't think you'll be surprised if I tell you that my last sister, Noémie, is decent enough. She also became a professional violinist, she runs every day, and she is currently studying psychology. She also spends a lot of energy mediating arguments between people she loves. A family I'm proud of, not only because of their obvious skills, but because of their virtues.

I want to thank my grandparents, whose house was the ground base for all of my aunts, uncles, and cousins, who met there during each and every vacation. They made us discover the delightful

joy of being cold, wet and exhausted during rainy hikes, to finally end up above a splendid sea of cloud from which would protrude just a few sharp peaks, over which Alpine choughs skillfully maneuvered with their vigorous flight. The grandparents are the true pillars of our extended family. The cycle of life being what it is, they are becoming older and can't hike anymore. I am now very happy to see the whole family striving to take care of them, as much as we have been taken care of. I can sadly not name every single other member of my family, humans or animals, but they are all a crucial part of myself.

I do need to spend some time for the love of my life, Mikaela. We met in Lebanon, she is American, she cares about France as little as I care about the USA, and yet she accepted to come here for me, in the armpit of the stinky old world. She had the courage to take over my mother's difficult job to bear with my incessant questions. She actually has a lot of answers, since the extent of her knowledge is so wide and well-rounded. Mikaela is also an awesome writer who regularly wins writing contests, and a qualified editor who plays a large role in making my productions publishable. She is much more than she believes of herself: exceedingly faithful, remarkably generous, paradoxically very introverted but willing to home all the persons in need she comes across, and unfortunately suffering from how little her power is to make the world a better place. She also comes with a very nice family in law, and of course, she is the mother of my child Cédric! A stunning baby who spends an excessive amount of energy smiling at every one, all day long (aside from sometimes, when he screams his head off.) He might give me a hard time whenever I get started writing my thesis, but he does it in a very cute way. And he always embodies a very good way for us to get away with our shared and legendary absent-mindedness. I'm looking forward to the time I'll be old enough for him to change my own diapers.

Life wouldn't be life without friends, old and new ones, whether I see them several times a week or once every two or three blue moons. Friends of the family, friends from church, friends from parkour, friends from the performing world, friends I have no idea how I got to know them. Not to brag, but they are too numerous to name them all.

Finally, let's remember that this is a PhD thesis that I'm writing, and that there is no thesis without a lab, without supervisors, without fellow PhD students, post-docs, interns, researchers, administrative workers, cleaning operatives, and all who are involved in making work enjoyable (sic.) I want to thank them all. Lionel, my director, who saved me from the happy hell of starving performing arts to give me the chance to throw myself in another highly precariously fun situation. The subject of my thesis could not be better suited to my aspirations: both tightly related to sports, and highly technical. Mathieu, my co-supervisor, who always made himself available, ready to give me quick and valuable feedback at any time, and to come in Grenoble in person despite he lived in the other end of the country. One supervisor is expert in computer vision, the other in biomechanics: the perfect fit for the objectives of my doctorate. Thibault, my faithful and multitasker office colleague, that I often left alone with the sole presence of cold-blooded computer hardware while I worked remotely. Other colleagues from other places such as the INSEP, the LBMC, the Pprime institute, etc. Athletes and coaches, who offered us some of their precious training time, while they knew full well that our research would most likely benefit the next generation, rather than theirs. Thank you all!

To sum it up, I owe this work to my family, my friends, my colleagues, and I'm gullible enough to believe I owe it to God above all. I am happy I have overcome it, not only alone but with all the forenamed people!

On these words, I suppose I can now start with what I'm here for.

Abstract / Résumé

ABSTRACT: DESIGN, EVALUATION, AND APPLICATION OF A WORKFLOW FOR BIOMECHANICALLY CONSISTENT MARKERLESS KINEMATICS IN SPORTS

Marker-based motion capture is hardly compatible with the inherent constraints of sports. As a consequence, numerous markerless methods are being proposed. One of them, Pose2Sim, has been developed and published during this doctoral program. Pose2Sim bridges the most renown programs in their respective fields: OpenPose for 2D pose estimation, and OpenSim for physically consistent 3D kinematics. The robustness of this solution has been evaluated, as well as its accuracy. Pose2Sim has also been tested in more challenging experimental conditions, in spite of which boxing key performance indicators have been correctly evaluated. Finally, BMX start sequences have been analyzed. It has been shown that if the 2D pose estimator was good enough, it would be possible to use Pose2Sim to provide joint {bike+pilot} kinematics to coaches.

RÉSUMÉ : CONCEPTION, ÉVALUATION, ET APPLICATION D'UNE MÉTHODE BIOMÉCANIQUEMENT COHÉRENTE DE CINÉMATIQUE SANS MARQUEURS EN SPORT

Les contraintes inhérentes au mouvement sportif ne permettent généralement pas son analyse à l'aide de marqueurs. En conséquence, des méthodes sans marqueurs sont de plus en plus proposées. L'une d'entre elles, Pose2Sim, a été développée et publiée en open-source dans le cadre de cette thèse. Pose2Sim fait le lien entre les deux programmes les plus utilisés de leurs domaines respectifs : OpenPose pour l'estimation de pose 2D, et OpenSim pour la cinématique 3D physiquement réaliste. La robustesse de cette solution a été évaluée, ainsi que sa précision. Pose2Sim a aussi été mis à l'épreuve de conditions expérimentales plus complexes, malgré lesquelles les indicateurs de performance clé en boxe ont pu être correctement évalués. Enfin, des séquences de départ de BMX ont été analysées. Il a été démontré que si l'estimateur de pose 2D est assez performant, il est possible d'utiliser Pose2Sim pour fournir aux entraîneurs la cinématique de l'ensemble {vélo+pilote}.

KEYWORDS: Markerless motion capture; Sports performance analysis; Kinematics; Computer vision; OpenPose; OpenSim; Python package

Contents

General introduction

Working environment

This doctoral thesis has been undertaken from December 2nd, 2019 to November 30th, 2022, at the LJK (Laboratoire Jean Kuntzmann for applied mathematics and informatics) in Grenoble, France. It was funded by the French National Center for Scientific Research (CNRS), as the result of a collaboration between the PPrime Institute (UPR3346, Poitiers, France), the French Cycling Federation (FFC), and the INSEP (French National Institute of Sport, Expertise and Performance), through the CREPS of Poitiers (Center for Sport Resource, Expertise, and Performance) and TSF Voiron (Springboard for Sports Training). It was later incorporated within the PerfAnalytics project, which aims to boost French sports performance towards the 2024 Olympic Games in Paris, with a particular focus on video analysis.

Context

Currently, the analysis of sports performance is still mostly carried out by subjective visual examination. More objective approaches exist, but they are usually too complex and cumbersome to implement, or too inappropriate for in-situ analysis. Hence, coaches often find them improper for a daily usage. In order to address this issue, other methods are being developed, among which many focus on video analysis. One of the long-term objectives in this regard is to be able to fully quantify the full-body kinematics and dynamics of athletes in context, without interfering with their training or competition, in a clear and timely way, in order for coaches to be able to give them a fast, accurate, objective, and comprehensive feedback on their technique and tactics. To this end, it appeared to us that the most appropriate way would take advantage of the video streams provided by a network of calibrated cameras. Each stream would be processed by some preexisting 2D keypoint detection deep-learning models. These coordinates would then be triangulated, and finally be constrained to a biomechanically consistent full-body skeletal model.

The original goal of this thesis shifted along the program, as it was initially mostly intended for BMX racing performance analysis (bicycle motocross.) BMX racing presented a lot of challenges which needed to be first addressed, such as the large size of the field of view, the direct sunlight, the swiftness of the movements, the occlusions of the pilot by the bike, and the lack of facilities for setting up the capture hardware. As a consequence, we first started with studying walking, running, and cycling tasks indoors, with a virtual and controlled environment map simulating an outdoor scene. We then studied boxing key performance indicators (KPIs) with lightweight consumer-grade hardware, and finally moved outdoors to investigate BMX racing.

Content

The thesis is organized as follows (also see the visual abstracts):
??: Motion capture is traditionally performed with marker-based systems. However, these solutions are hardly compatible with on-field sports analysis, as they involve marker placement on the skin, and a heavy setup. As a consequence, markerless approaches are being investigated.
??: Sports movements don't generally only lie in the sagittal plane, and they often cause body part occlusions. Moreover, results need to be as biomechanically coherent as possible. Hence, one of the most promising prospects for sports motion analysis consists in addressing the problem with several video sources, and then constraining 3D coordinates to a kinematic model. Such research is at the intersection of machine learning for 2D pose estimation, computer vision for 3D reconstruction, and biomechanics for constraining coordinates to anatomically consistent kinematics.
??: Sport scientists would also benefit from using wireless light-weight cameras as well as a user-friendly integrated workflow. Hence, Pose2Sim, an open-source Python package striving to answer these needs, has been proposed and released [**?**]. 2D keypoint coordinates obtained with OpenPose or DeepLabCut are robustly triangulated, and serve as input for a full-body OpenSim inverse kinematics procedure (Figure **??**).
??: Pose2Sim robustness has been evaluated, regarding people entering and exiting the field of view, image quality, calibration errors, and low number of cameras [**?**] (Figure **??**).
??: Its accuracy has also been assessed, and deemed sufficient for walking, running, and cycling analysis [**?**] (Figure **??**).
??: The workflow has then been tested in challenging conditions, such as shadow-boxing. Boxing involves fast, 3 dimensional, and full-body movements. These have been captured with GoPro cameras, post-calibrated on geometric cues, and post-synchronized by correlation of keypoint speeds. We concluded that boxing key performance indicators could be correctly evaluated, and that the protocol, whether it employed a full marker-based system, or a markerless one with research-grade hardware, or a markerless one with consumer-grade hardware, mattered less than the choice of a good 2D pose estimator [**?**] (Figure **??**).
??: Numerous sports disciplines are practiced with special equipment, such as a board in skateboarding, a racket and a ball in tennis, or a bike in BMX racing. The interactions between the athlete and their gear are often important to retrieve. We analyzed BMX start sequences, by using OpenPose for 2D human pose estimation, and a custom trained DeepLabCut model for bike detection. We ran Pose2Sim on the joint pilot+bike 2D estimations, and performed 3D inverse kinematics on a custom OpenSim pilot+bike model. This showed that analyzing simultaneously the athlete and their equipment is possible, which provides additional perspectives for markerless sports motion analysis.

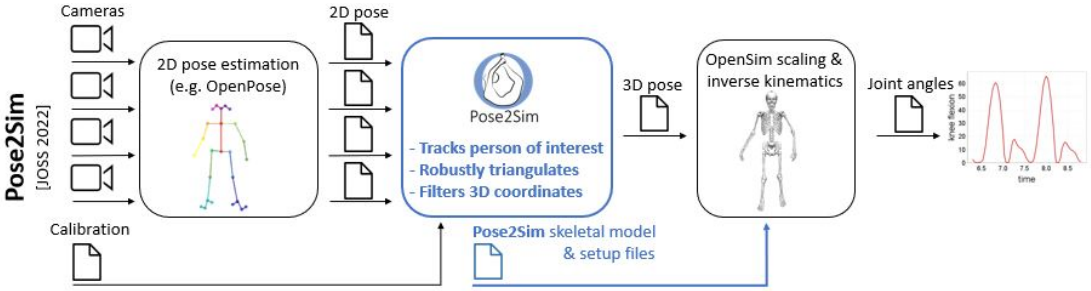
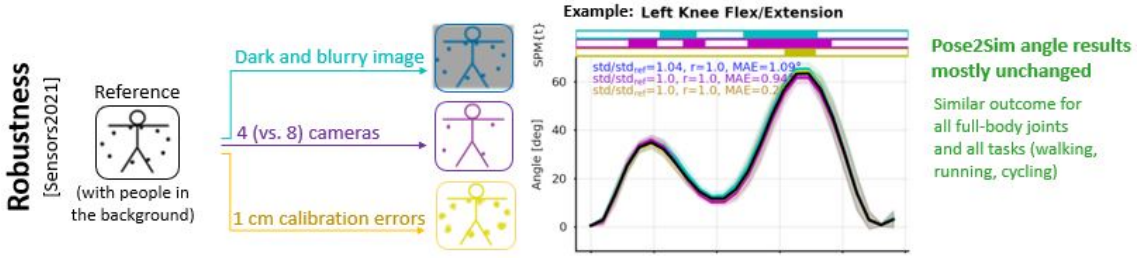
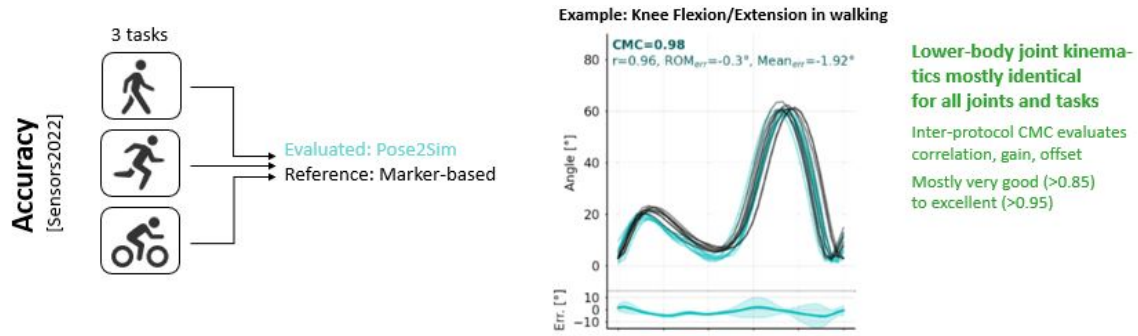


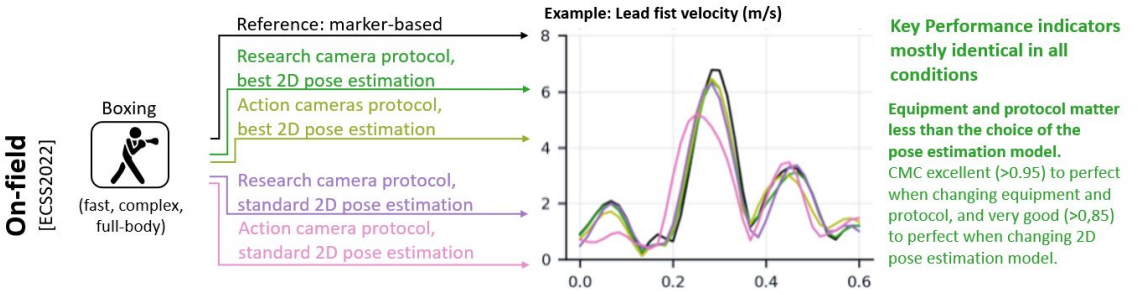
*Figure 1: Visual abstract for the Pose2Sim workflow (**??**) [?].*



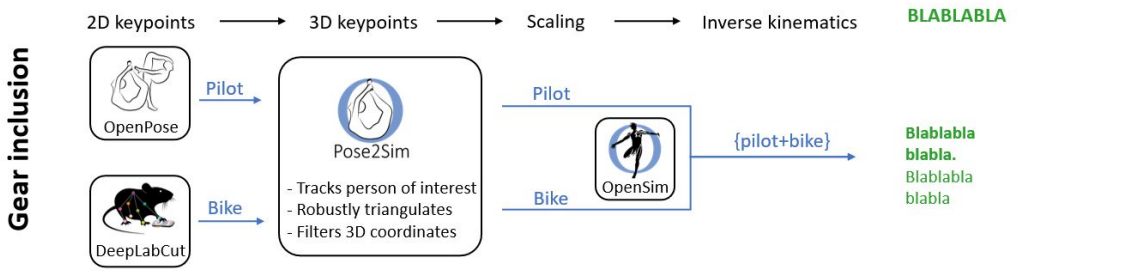
*(a) Visual abstract for Pose2Sim robustness assessment (**??**) [?].*



*(b) Visual abstract for Pose2Sim accuracy assessment (**??**) [?].*



*(c) Visual abstract for the assessment of KPIs in boxing with Pose2Sim (**??**) [?].*
Research camera protocol: Qualisys cameras, calibration with markers, hardware synchronization.
Action camera protocol: GoPro cameras, calibration on boxing ring dimensions, synchronization on 2D keypoint speeds.



*(d) Visual abstract for joint analysis of the athlete with OpenPose, and their equipment with DeepLabCut (**??**).*

Figure 2: Visual abstracts for chapters 4 to 7.

Contributions

This work led to the publication of a few scientific contributions as a first author: 3 peer-reviewed articles, 1 conference paper, and the release of an open-source package. Another conference paper has been published as a secondary author, and a Blender add-on integrated the published package, for 3D character animation. Moreover, another peer-reviewed article has been published as a first author during this period, although it was not related to the program.

[**?**]: David Pagnon, Mathieu Domalain and Lionel Reveret. *Pose2Sim: An End-to-End Workflow for 3D Markerless Sports Kinematics—Part 1: Robustness.* Sensors, vol. 21, no. 19, 2021.

[**?**]: David Pagnon, Mathieu Domalain and Lionel Reveret. *Pose2Sim: An End-to-End Workflow for 3D Markerless Sports Kinematics—Part 2: Accuracy.* Sensors, vol. 22, no. 7, 2022.

[**?**]: David Pagnon, Mathieu Domalain and Lionel Reveret. *Pose2Sim: An open-source Python package for multiview markerless kinematics.* Journal of Open Source Software, vol. 7, no. 77, page 4362, 2022.

[**?**]: David Pagnon, Mathieu Domalain, Thomas Robert, Bhrigu-Kumar Lahkar, Issa Moussa, Guillaume Saulière, Thibault Goyallon and Lionel Reveret. *A 3D markerless protocol with action cameras – Key performance indicators in boxing.* In 2022 Congress of the European College of Sport Science (ECSS), Sevilla (Spain), 2022. Poster.

[**?**]: Bhrigu-Kumar Lahkar, Thibault Goyallon, Anaïs Chaumeil, David Pagnon, Issa Moussa, Andreas Muller, Mathieu Domalain, Lionel Reveret, Raphael Dumas, Thomas Robert. *Assessment of a markerless motion capture system for upper extremity joint kinematics during boxing.* In 17th International Symposium on 3-D Analysis of Human Movement, Tokyo (Japan), 2022. Oral.

[**?**]: Carlos Barreto, CEB Studio. *Mocap MPP2SOS* [Blender add-on], Blender Marker, 2022. https://blendermarket.com/products/mocap-mpp2soss.

[**?**]: David Pagnon, Germain Faity, Galo Maldonado, Yann Daout, Sidney Grosprêtre. *What Makes Parkour Unique? A Narrative Review Across Miscellaneous Academic Fields.* Sports Medicine, vol. 52, page 1029, 2022.

1 State of the art

Motion capture (MoCap) in sports is traditionally performed with marker-based (opto-electronic) systems. However, this is hardly compatible with on-field analysis. As a consequence, alternatives are being investigated, among which those offered by Inertial Measurement Units (IMUs) or dept-field (RGB-D) cameras. Markerless analysis from video sources represents one of the most promising prospects, which has been possible thanks to progress in machine learning. From 2D pose estimation to 3D joint angle determination, this is a new field which opens up new possibilities for motion analysis in a sports context.

This chapter is an up-to-date and more detailed version of the introduction of: "Pose2Sim: An End-to-End Workflow for 3D Markerless Sports Kinematics—Part 1: Robustness" [?].

Contents

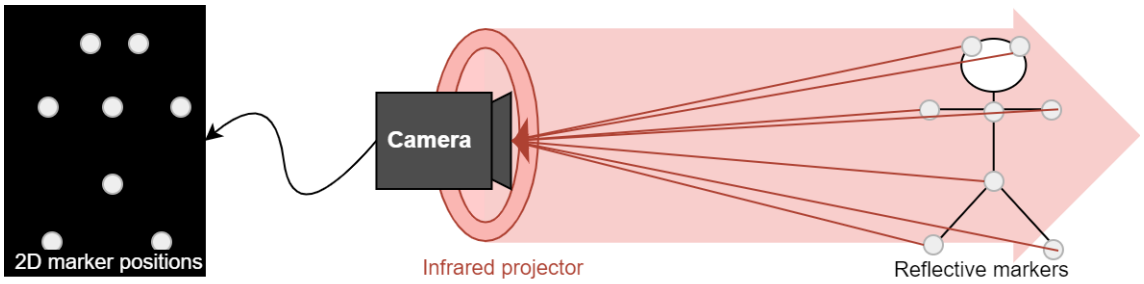
1.1 Overall context of kinematic analysis in sports

1.1.1 General context

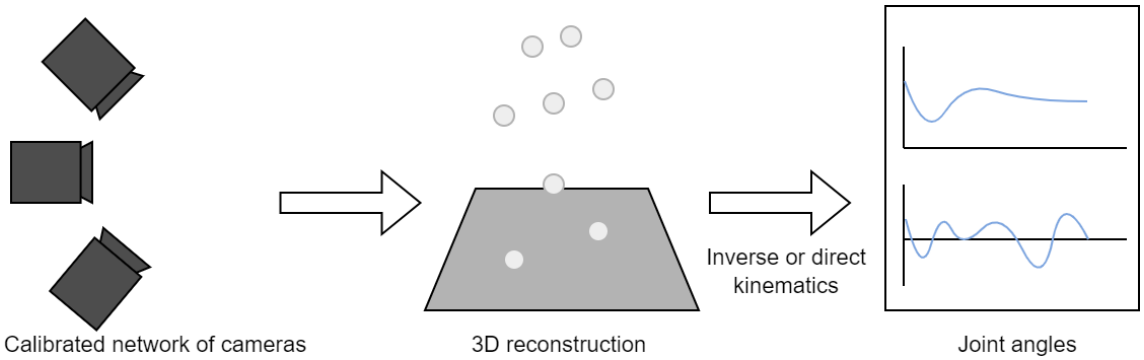
As coaching athletes implies observing and understanding their movements, motion capture (MoCap) is essential in sports motion analysis. It helps to improve movement effectiveness, prevent injuries, predict performances, or reveal different motor skills among athletes. For the last few decades, marker-based systems have been considered the best choice for the analysis of human movement, when regarding the trade-off between ease of use and accuracy. However, these methods have proven to be much more challenging in a sports context than in a laboratory setting, and to be generally inappropriate [**?**, **?**]. As a consequence, other methods have been investigated (see Table **??** and Table **??** at the end of the chapter).

1.1.2 Marker-based systems

Marker-based systems use a network of opto-electronic cameras. Each of these cameras are surrounded by a crown of infrared LEDs, which projects light toward the subject, who is equipped with reflective markers. Ideally, only the light reflected from these markers is captured by the cameras. The camera usually pre-processes the image to make it binary, and only outputs the coordinates of the detected marker (Figure **??**).



(a) An opto-electronic camera is traditionnally surrounded by a crown of infrared LEDs, projecting light toward the subject. The subject wears markers, which reflect light back to the camera. Marker positions are then known in the camera plane.



(b) Once calibrated, a network of these cameras allows for 3D reconstruction of marker positions. Marker coordinates are then used to infer the posture of the subject.

*Figure 1.1: Principles of marker-based motion capture. (Figure **??**) presents the functioning of an opto-electronic camera. (Figure **??**) shows how a network of calibrated motion capture cameras helps obtaining joint angles.*

If calibrated, using a network of these cameras allows for triangulating the 2D coordinates. Calibration involves knowing the cameras' intrinsic properties (such as focal length, optical center, distortion) as well as their extrinsic properties (their position and orientation as regards to the global coordinate system.) See Chapter 2.2 on **??** for more details. The reconstructed 3D marker positions are then used to infer joint kinematics (see Chapter 2.3 on **??** for more explanation about direct and inverse kinematic methods). Most solutions leveraging marker-based analysis are commercial, such as the Vicon Nexus [**?**] or Qualisys Track Manager (QTM) [**?**] softwares.

Yet, reflective marker-based camera systems are complex to set up, are time-consuming, and are very expensive. They also require specific lighting conditions, and involve cumbersome cabling. Moreover, markers may fall off the body of the participant due to sharp accelerations or sweat. They can hinder the natural movement of athletes, which is likely to affect their warm-up, focus, and safety. While the accuracy of landmark location is claimed to be sub-millimetric in marker-based methods [**?**], marker placement is tedious, intrusive, prone to positioning variability from the operator [**?**], and subject to skin movement artifacts, especially on soft tissues. Della Croce et al. found out that inter-operator variations in marker placements range from 13 to 25 mm, which can propagate up to 10° in joint angle prediction [**?**, **?**]. For example, tissue artifacts account for up to a 2.5 cm marker displacement at the thigh, which can cause as much as a 3° error in knee joint angles tissues [**?**, **?**]. Joint positions must be calculated explicitly in marker-based methods, which introduces more variability: these errors range up to 5 cm, which can contribute up to 3° of error in lower limb joint angles [**?**]. As a consequence, marker-based methods cannot be considered as a gold-standard for accurately measuring joint kinematics, unlike bone-anchored pins, Magnetic Resonance Imaging (MRI) [**?**], biplanar videoradiography [**?**, **?**], or 3D ultrasound [**?**]. Nevertheless, since these methods cannot be operated on broad and fast motions such as sports movements, marker-based approaches are still considered as the reference (or "silver-standard") methods, for motion capture.

1.1.3 IMU and RGB-D systems

Consequently, other approaches based on alternative technologies have been investigated over the past years. For instance, wearable Inertial Measurement Units (IMUs) can be placed on an athlete's limbs. IMUs are generally made of an accelerometer, a gyroscope, and a magnetometer. The accelerometer measures the linear acceleration, the gyroscope measures the rotational speed, and the magnetometer measures the orientation of the earth magnetic field. Fusing and integrating these signals allows for the determination of their 3D orientations. The orientation of the athlete's limbs can then be used in combination with a skeletal model to infer their posture (Figure **??**). The most renown IMU-based systems embed both sensors and software, such as APDM [**?**] and Xsens [**?**]. However, OpenSense now allows to use these sensors within the open-source OpenSim software [**?**].

IMUs offer the advantages of getting away from all camera-related issues. They are less expensive, they do not involve any complex setup and calibration, the field of view is larger, data do not take much storage space, they are not sensitive to self- and gear-occlusions, they can be operated outside of a controlled environment, and they can work in real-time [**?**, **?**]. However, they are not fully sensorless as they require external equipment to wear. They also have the drawback of involving the implication of a trained operator, and are sensitive to ferromagnetic disturbances. Above all, they are exposed to drift over time and need to be calibrated every few minutes. Joint angle accuracy is relatively good in the flexion/extension plane, but less so in other rotational planes where errors are greater than 5° for most motions [**?**, **?**]. Moreover, they are not suitable for joint positions assessment, since these are obtained through multiple integrations of the original signal [**?**].

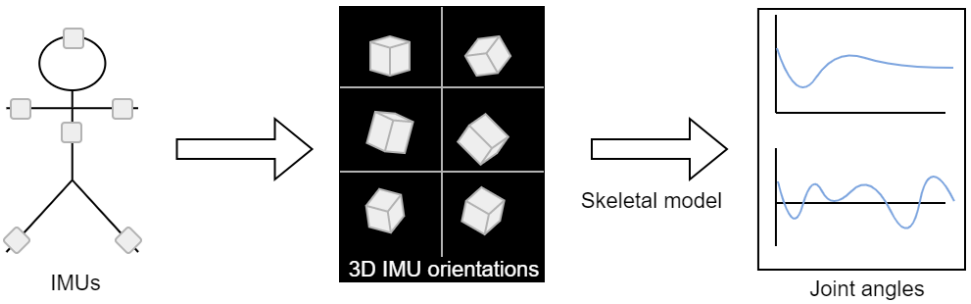


Figure 1.2: IMUs are placed on the subject's limbs. The orientation of the limbs is then used to infer the posture of the subject.

Another approach involves depth-field cameras (RGB-D), usually with Kinect devices [**?**]. These cameras were released for the video game industry, which allowed to greatly lower prices without hampering performance. Kinect also provided a Software Developer Kit (SDK), which was quickly adopted by the research community. Older models projected infrared *structured* light (i.e., a pattern) onto the scene. The relative deformation of the pattern reflected from the scene was then used to estimate depth. Newer models project infrared *modulated* light onto the scene. The time of flight of the light reflected from the scene is then used to estimate depth. Results are commonly considered to be 2.5D, since only the depth of the front facing plane of view is measured. Gait analysis results are natively poor, but after an optimization by a neural network, [**?**] manage to get root-mean-square errors under 7° for knee flexion/extension angle at the most challenging part of the gait cycle, although 3D joint angle errors usually stay under 2-3°. However, it may not perform as well on other motions on which the neural network has not been trained. A network of a few RGB-D cameras can give access to full 3D [**?**, **?**, **?**]. Nevertheless, these cameras hardly function in direct sunlight nor at a distance over 5 meters, and they work at lower frame rates (generally under 30 Hz) [**?**, **?**].

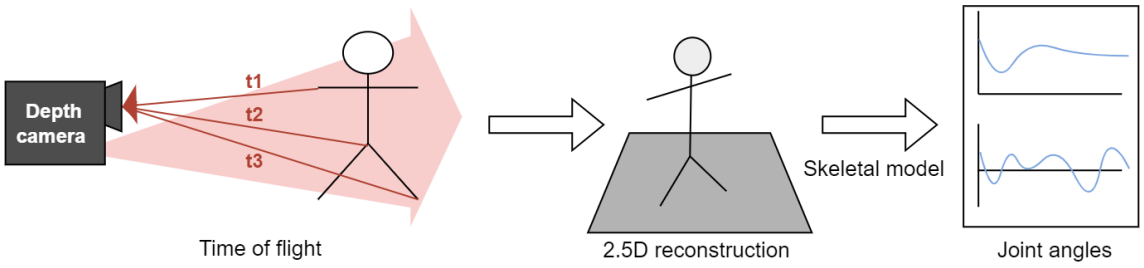


Figure 1.3: A depth-field camera (RGB-D) projects infrared modulated light onto the subject's body. The time it takes for the light to be reflected to the camera sensor (time of flight) depends on distance, and gives access to the depth of the scene. Older RGB-D cameras use structured light rather than time of flight calculations to infer depth.

1.1.4 Video-based markerless approaches

A recent breakthrough has come from computer vision, and the advent of 2D pose estimation from image sources, which quickly became more efficient and accurate. The explosion of deep-learning based methods from camera videos, for which the research has skyrocketed around 2016 [**?**], is related to the increase in storage capacities and huge improvements in GPU computing. A search on the ScienceDirect database for "deep learning 3D human pose estimation" produced fewer than 100 papers per year until 2015, and the number is now reaching over 1,000, fitting an exponential curve (Figure **??**).

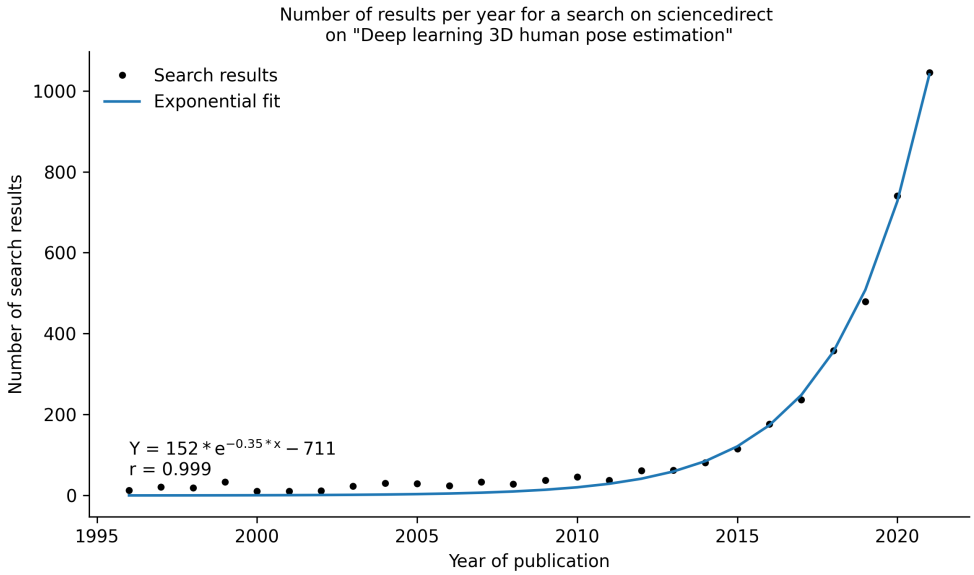


Figure 1.4: The search for "deep learning 3D human pose estimation" (dots) fits an exponential curve (line). The search produced less than 100 results until 2015, and is now well over a 1,000 per year.

It has rekindled interest from the biomechanics community towards image-based motion analysis, which is where it all started with the invention of chronophotography in the 19th century by Marey in France, and Muybridge in the USA [**?**]. Currently, two approaches coexist in human and animal motion analysis: the first one mostly focuses on joint positions, and is lead by the computer vision and the deep-learning communities; while the second one is interested in joint angles, such as the biomechanics community uses to obtain physically coherent kinematics individualized to each subject. One of the main current challenges is to bridge the gap between these two worlds, and to take advantage of deep-learning technologies for kinematic analysis [**?**, **?**].

1.2 2D markerless analysis

1.2.1 2D pose estimation

The most well-known off-the-shelf 2D human pose estimation solutions are OpenPose [**?**] (Figure **??**), and to a lesser extent AlphaPose [**?**]. To our knowledge, they are also the only multi-person 2D pose estimation solutions that provide foot keypoints, which are essential for most sports motion analysis. While both show similar accuracy, AlphaPose is faster when few people are in the scene, and more accurate when people are small on the image [**?**]. However, OpenPose has the advantage of being a bottom-up approach, whose computational cost does not increase with the number of persons detected [**?**]. A bottom-up approach first detects all available joint keypoints, and then associates them to the right persons; while a top-down approach first detects bounding boxes around each person, and then finds joint keypoints inside of them. OpenPose is also more widespread (25,000 stars on the GitHub repository, vs. 6,000 for AlphaPose): as a consequence, it has been evaluated more thoroughly and represents a common ground for comparing further kinematic research.

Other approaches have shown even better results on evaluation datasets (see review [**?**]), but they are generally slower and not as widespread. The technology, however, is still maturing and some light-weight systems such as BlazePose [**?**], UULPN [**?**], or YOLOv7 [**?**] are being proposed, which can operate in real time on a mobile phone; however, they either support single-

person detection only, are not accurate enough for quantitative motion analysis [**?**], or haven't been embraced by the community yet. Some work has also been done on temporal consistency across frames with OpenPifPaf, which makes the system much faster, and helps it perform better on low-resolution regime or with occlusions such as in crowds [**?**].

Two other 2D pose estimation toolboxes are DeepLabCut [**?**, **?**] and SLEAP [**?**], which were initially intended for markerless animal pose estimation. They have the advantage that they can be custom trained for the detection of any human or not human keypoint with a relatively small dataset.

All the tools presented in this section are open-source. See Chapter 2.1.3 on **??** for more technical details on their architecture.

*Figure 1.5: 2D pose estimation by OpenPose [**?**].*

1.2.2 2D kinematics from 2D pose estimation

Some authors bridge 2D pose estimation to more biomechanically inspired variables, such as in gait kinematics analysis. Kidzinski et al. present Mobile Gaitlab, a toolbox for quantifying gait pathology parameters, that runs in a Google Colab or in a web application [**?**]. Stenum et al. evaluate gait kinematics calculated from OpenPose input concurrently with a marker-based method. Mean absolute error of hip, knee and ankle sagittal angles were 4.0°, 5.6° and 7.4° [**?**]. Liao et al. have not released their code, but they use OpenPose outputs to train a model invariant to view [**?**]. Viswakumar et al. perform direct calculation of the knee angle from an average phone camera processed by OpenPose [**?**]. They show that OpenPose is robust to challenging clothing such as large Indian pants, as well as to extreme lighting conditions. Other sports activities have been investigated, such as lower body kinematics of vertical jump [**?**] or underwater running [**?**]. Both works train their own model with DeepLabCut. Serrancoli et al. fuse OpenPose and force sensors to retrieve joint dynamics in a pedaling task [**?**].

Although it doesn't specifically use deep-learning approaches, another noteworthy tool for 2D sports movement analysis is Kinovea [**?**, **?**]. It allows for manually labeling keypoints on a frame, and tracking them in time in order to obtain point trajectories, speeds, or angle data. These can then be plotted or exported as .csv files. Other features, such as the ability to import and export videos, to add various annotations, or to take into account distortion and perspective effects, make it a versatile and widespread tool for sports analysis. Dartfish provides similar features along with other services, but it is not free nor open-source [**?**]. However, both these approaches do not provide 3D analysis, and they can break down if the tracked points are not contrasted enough.

1.3 3D markerless analysis

1.3.1 3D pose estimation from a single video

There are a lot of different approaches for 3D human pose markerless estimation, and listing them all is beyond our scope (see review [**?**]). Some directly lift 3D from a single 2D camera (see review [**?**]). Shape approaches such as SPIN [**?**] or VIBE [**?**] fit an SMPL 3D mesh model [**?**] to OpenPose 2D keypoints. Mediapipe GHUM works similarly, although it comes with its own 2D keypoint estimation (BlazePose) and 3D mesh model [**?**]. These approaches are designed by the computer vision community, and are not suited for accurate biomechanics as is: they are mostly applicable for retargeting a human motion to fictional character. Since they are monocular, the 3D pose is subject to indeterminates: the model is often leaning forward, and the lower body is not always facing the same side as the upper body. And since they consist in performing a regression from just a few 2D keypoints, the corpulence of the mesh is only a function of the segment sizes, and describes a generic human body. Other approaches such as STRAPS [**?**] offer a more accurate body shape, by using adjusting SMPL pose and shape parameters both from 2D keypoint estimations, and from 2D silhouette segmentations. [**?**] proposes another approach. They records the 3D shape of a speed climber in a studio equipped with 68 video cameras, and then animate it to follow 2 calibrated drone views by optimizing its manifold parameters. This allows for tracking the center of mass and for detecting hand contacts with holds, without the use of machine learning.

Some other methods focus on the estimation of 3D keypoints, rather than on fitting a 3D shape model. One of them estimates the positions of a set of keypoints around the joint, instead of determining only the joint center keypoint. This way, axial rotation along the limb is solved [**?**]. As for XNect, it primarily focuses on real time [**?**]. A few approaches even strive to estimate 3D dynamics and contact forces from a 2D video input [**?**,**?**,**?**]. Some incorporate kinematic priors into their neural networks in order to take advantage of human knowledge [**?**]. Surprisingly, this does not seem to be done in multi-view approaches. Rempe et al. solve occlusions from a 2D input [**?**], but this remains a probabilistic guess that may be unsuccessful in case of unconventional positions of hidden limbs, whereas using more cameras would have given more trustworthy results. More than a model, MMPose is a toolkit for 2D and 3D pose, as well as shape estimation, which supports the implementation of different datasets, different architectures, and different feature extractors [**?**]. The resulting models can be more accurate, but also much slower than the conventional ones.

1.3.2 3D pose estimation from multiple videos

The earliest 3D pose estimation approaches were addressing the issue under the perspective of shape reconstruction. They were not based on deep-learning, and calculated 3D shapes by visual hull reconstruction from 2D silhouette detection [**?**, **?**]. However, they required specific lighting, clothing, and background conditions. Nowadays, solutions like Mask R-CNN [**?**] or Detectron2 [**?**] are able to effectively detect human silhouettes, without any constraints on the environment nor on clothing. This makes it possible to reconstruct 3D human shapes in a sports context [**?**]. Other shape methods such as EasyMocap [**?**] triangulate OpenPose 2D keypoint estimations, and use a regressor to fit an SMPL model on it. It solves the issues previously reported in the monocular case, but the morphology of the SMPL mesh is still only based on segment length, and does not account for any variability in human corpulence. Interestingly, there are currently no approaches that use both keypoint and silhouette information, despite the fact that it could remain accurate with fewer cameras, or provide more useful information such as the estimation of body segment inertial parameters. These methods, however, would be particularly computationally intensive.

Some research attempts to solve 3D pose estimation from a network of uncalibrated cameras, i.e., cameras whose extrinsic parameters (translation and rotation with respect to the coordinate system), intrinsic parameters (focal length, pixel size, etc.), and distortion coefficients are not known (See **??** on **??** for more details.) It either uses 2D pose estimations of each view as visual cues to calibrate on [**?**, **?**, **?**], or an adversarial network that predicts views of other cameras, compares them to real views, and adjusts its calibration accordingly [**?**]. Dong et al. recover 3D human motion from unsynchronized and uncalibrated videos of a repeatable movement found on internet videos (such as a tennis serve performed by a celebrity) [**?**]. Using uncalibrated videos is still a very experimental trend, that would require more research before being used in biomechanics.

As a consequence, we focus on methods which infer 3D pose by triangulating 2D pose estimations, from a network of multiple calibrated and synchronized cameras. They require a more complex setup, but they allow for more accurate 3D reconstruction, based on geometry rather than on statistics. They are also more reliable than monocular methods in a wider variety of contexts, in particular when occlusions are encountered. The classical evaluation metric is the MPJPE (Mean Per Joint Position Error), which is the average Euclidian distance between the estimated joint coordinate and its ground truth. DLTdv8 provides a user-friendly GUI written on Matlab for semi-automatically tracking 2D points, which can be enhanced with a simple deep-learning approach [**?**, **?**]. These points can then be triangulated by Direct Linear Transform (DLT, see section **??**) to obtain 3D coordinates.

However, when human beings are concerned, most 3D pose estimation methods take OpenPose as an input for triangulation, and more specifically the body_25 model. Aside from ours, a number of tools have been made available for triangulating OpenPose, although usually not peer-reviewed: the experimental OpenPose 3D reconstruction module [**?**], the FreeMoCap Python and Blender toolbox [**?**], or the pose3d Matlab toolbox [**?**], and the EasyMocap pipeline [**?**]. Labuguen et al. evaluate 3D joint positions of a pop dancer with a simple Direct Linear Transform triangulation (DLT [**?**, **?**]) from 4 cameras [**?**]. Apart from the upper body for which error goes up to almost 700 mm, the average joint position error is about 100 mm. Nakano et al. examine three motor tasks (walking, countermovement jumping, and ball throwing), captured with 5 cameras and triangulated with the same methods, with a subsequent Butterworth filter [**?**]. 47% of the errors are under 20 mm, 80% under 30 mm, and 10% are above 40 mm. The largest errors are mostly caused by OpenPose wrongly tracking a joint, for example by swapping the left and the right limb, that causes large errors up to 700 mm. This may be fixed either by using a better 2D pose estimator, or by using more cameras to reduce the impact of an error on a camera, or else by considering the temporal continuity in movement. Needham et al. use 9 cameras and find that ankle MPJPEs are within the margin of error of marker-based technologies (1–15 mm), whereas knee and hip MPJPEs are greater (30–50 mm). These errors are systematic and likely due to "ground-truth" images being mislabeled in the training dataset [**?**]. They also run the comparison with AlphaPose and with DeepLabCut. While AlphaPose's results are similar to OpenPose's; DeepLabCut errors are substantially higher.

Slembrouck at al. go a step further and tackle the issue of limb swapping and of multiple person detection [**?**]. In case of multiple person detection, one needs to make sure they associate the person detected on one camera to the same person detected on other ones. Slembrouck et al. manage to associate persons across cameras by examining all the available triangulations for the neck and mid-hip joints: the persons are the same when the distance between the triangulated point and the line defined by the detected 2D point and the camera center is below a certain threshold. They only focus on lower limb. Their first trial features a person running while being filmed by seven cameras, whereas their second one involves a person doing stationary movements such as squats while filmed by 3 cameras. After filtering, the average positional error in the first case is about 40 mm, and it is roughly 30 mm in the second case (less than 20 mm for the ankle joint). Other authors deal with the multiperson issue in a slightly different way [**?**, **?**, **?**]. In average, if the detected persons are correctly associated and the limbs don't swap, the average joint position

error for an OpenPose triangulation is mostly below 40 mm.

Some triangulation methods not based on OpenPose reach even better results on benchmarks, although it comes at the cost of either requiring heavy computations, or of being out of reach for non-expert in deep-learning and computer vision. The classic approach reduces the joint detection heatmap to its maximum probability, and then to triangulate these scalar 2D positions. Instead of this, the main state-of-the art methods directly perform a volumetric triangulation of the whole heatmaps, and only then take the maximum probability as a 3D joint center estimate. By working this way, they keep all the information available for as long as possible. They manage to lower their MPJPE to about 20 mm [**?**, **?**].

1.3.3 3D kinematics from 3D pose estimation

Numerous studies have focused on the accuracy of 3D joint center estimation, but far fewer have examined joint angles [**?**]. Yet, when it comes to the biomechanical analysis of human motion, it is often more useful to obtain joint angles. Joint angles allow for better comparison among trials and individuals, and they represent the first step for other analysis such as musculoskeletal analysis and inverse dynamics. They are also more intuitive to grasp for coaches and athletes, who would rather need to learn about the extension of their elbow, rather than about the respective position of their wrist, elbow and shoulder joint centers.

3D joint estimation was initially tackled with shape approaches, by visual-hull reconstruction. [**?**] reports joint errors generally above 10° for the lower-body, using 4 calibrated and synchronized cameras. Some commercial solutions have been released such as Simi Shape [**?**] or The Captury [**?**]. No peer-reviewed article has been published for validating the first one, but results have been shown to bear high variability [**?**]. Concerning the second one, flexion/extension joint angles were not deemed to be equivalent to marker-based ones, with errors between 4 and 20° [**?**], although a study conducted by the company seemed to find results closer to marker-based ones [**?**].

Most other research is based on the triangulation of 2D keypoints. It started out kinematic analysis with spatio-temporal parameters. For example, Zago et al. evaluate gait parameters computed by triangulating 2 videos processed by OpenPose, and notice that straight gait direction, longer distance from subject to camera, and higher resolution make a big difference in accuracy [**?**]. [**?**] finds that most gait parameters are very accurately measured by the Theia3D markerless system, used with 8 cameras.

D'Antonio at al. perform a simple triangulation of the OpenPose output of two cameras, and compute direct flexion-extension angles for the lower limb [**?**]. They compare their results to IMU ones, and point out that errors are higher for running than for walking, and are also rather inconsistent: Range of Motion (ROM) errors can reach up to 14°, although they can get down to 2 to 7° if the two cameras are set laterally rather than in the back of the subject. Wade et al. calculate planar hip and knee angles, and compare results from OpenPose, AlphaPose, and DeepLabCut processing the input of 9 cameras [**?**]. They deem the method accurate enough for assessing step length and velocity, but not for joint angle analysis. AniPose, a Python open-source framework, broadens the perspective to the kinematics of any human or animal with a DeepLabCut input, instead of OpenPose. They offer custom temporal filters, as well as spatial constraints on limb lengths [**?**]. To our knowledge, it has only been concurrently validated for index finger angles in the sagittal plane, resulting in a root-mean-square error of 7.5° [**?**].

The previous studies calculated single degree of freedom angles between 3 joint centers. However, the human skeleton is complex and not only made of pin joints: aside from the flexion/extension rotation axis, the abduction/adduction axis and the internal/external axis are typically also engaged; and some joints also involves some translation, such as the shoulder girdle. In this case, either several markers per segment are needed, or a solid kinematic model with accurate joint definitions. So far, little work has been done towards obtaining 3D angles from multiple views [**?**]. Aside from our solution (see Chapter 3 on Pose2Sim), two main others are worth men-

tioning. Theia3D is a commercial software application for human gait markerless kinematics. It estimates the positions of a set of keypoints around the joint, and then uses a multi-body optimization approach to solve inverse kinematics. A study conducted by the company notices an offset in hip and ankle angles between their markerless system and the reference marker-based one, likely due to different skeletal models. Once this offset is removed, the root-mean-square error (RMSE) in lower limb roughly ranges between 2 and 8° for flexion/extension and abduction/adduction angles, and up to 11.6° for internal/external rotation [**?**]. [**?**] focuses on the upper-body in boxing, and finds good to excellent agreement on segment velocities, and reports RMSE up to 10° in flexion/extension angles, and up to 23° on internal/external rotations. Although the GUI is user-friendly can be integrated in the Vicon or Qualisys softwares, Theia3D is neither open-source nor customizable. On the other side, OpenCap [**?**] has been released very recently, about two years after our own solution. It offers a user-friendly web application working with low-cost hardware. It predicts the coordinates of 43 anatomical markers from 20 triangulated keypoints, and imports them in OpenSim, which is an open-source biomechanical 3D analysis software using a multi-body optimization approach to solve inverse kinematics [**?**, **?**]. OpenCap then performs classic inverse kinematics with numerous inferred markers and a skeletal model. However, OpenCap has not yet been peer-reviewed, it is limited to single person analysis, it only works natively with Apple products, and the data sent to the cloud for calculation are not treated in a GDPR compliant way (General Data Protection Regulation), which makes its use prohibited by research centers in Europe.

Pose estimation from videos can also be fused with the information provided by other sensors, such as IMUs [**?**, **?**]. This enables solving occlusions in videos, and compensation of the drift consecutive to the integration of accelerations and rotation speeds in IMUs. For example, [**?**] fuse OpenPose results from a single monocular video and two IMU outputs, and solve kinematics of the upper body in OpenSim in order to examine the effects of fatigue on boxing. Results are promising, but this cannot be considered as fully markerless. Fusing the depth map of a single RGB-D camera with its image processed by OpenPose has also been investigated [**?**], although 3D coordinate errors were close to 10 cm.

1.4 Statement of need

According to Atha [**?**], an ideal motion analysis system involves the collection of accurate information, the elimination of interference with natural movement, and the minimization of capture and analysis times. Yet, even though a marker-based system gives relatively accurate results, it requires placing markers on the naked body, which is invasive, time-consuming, and can hinder focus and natural movement. It is also hard to set up outdoors or in context, strenuous to analyze, and very costly. And there are other potential pitfalls to investing in technology, despite its advantages: the information gathered can be unhelpful, or inaccurate, or not easily interpretable, or simply not implementable in the context of sports [**?**]. Moreover, the specificities of biomechanics differ between the fields of medicine and sports, and so systems developed for one cannot be directly applied to the other. As a consequence, in the overwhelming majority of cases, coaches solely use subjective visual observation to assess an athlete's movement patterns and to compare performances.

The emergence of markerless kinematic analysis opens up new possibilities. It does not involve any constraints on the athlete, who can wear their usual clothing, nor on the environment, which can be complex and subject to unpredictable lighting. Nevertheless, sports motions are usually fast, atypical, and occluded by the use of equipment. This can be solved for the most part by using multiple view points. Although this requires delicate synchronization and calibration, and high computational capacities, standard hardware is now sufficient for this usage. In addition to these

robustness requirements, the accuracy and speed of such system would determine its adoption by the community. In sports, 3D kinematics are more relevant than 2D sagittal plane kinematics; and full-body analysis (including the upper limb) is often desired. Hence, constraining markerless coordinates to an individually scaled, and biomechanically realistic model is important. However, as coaches and athletes usually need a mere feedback rather than a definitive diagnosis, they don't need as thorough of an accuracy as physicians. Competition conditions are often fast-paced and congested, which makes it relevant to use a quick, flexible, and straightforward setup, for example with wireless light-weight cameras. Ideally, results should be given in real time, and they should be more visual than time series graphs.

The objective of this thesis is to participate in building a bridge between the communities of computer vision and biomechanics, by providing a simple and open-source pipeline connecting the two aforementioned state-of-the-art tools: OpenPose and OpenSim. Robustness and accuracy will be assessed, and concrete applications in elite sports context will be discussed.

Sensor type	**Mono/Multi camera**	**2D/3D**	**Pros and Cons**
Opto-electronic (marker-based)	Multi	3D	+ Standard + Good ease-of-use/accuracy trade-off + Numerous commercial solutions - Not open-source - Not suitable in sports contexts
IMU	N/A	3D	+ Good angle accuracy - Angle drift & poor position analysis - Not sensorless, can be cumbersome
RGB-D	Mono	2.5D	+ Markerless - Generally poor accuracy - Frame-rate $\leq$ 30 Hz - Needs distance $\leq$ 5 m and no direct sunlight
	Multi	3D	+ Full 3D markerless + Better accuracy - Same as above re. frame-rate, distance, and light
RGB video	Mono	2D	+ Very robust in all contexts + Cheap and easy to set up - Only 2D - Not very accurate
		3D	+ Full 3D with one single RGB camera - Probabilistic guess when occlusions: accuracy $\searrow$ - Slow
	Multi uncalibrated	3D	+ Removes difficult step of calibration - Not accurate enough yet
	Multi calibrated	3D	+ Solves occlusions + Robust - Systematic offsets due to labelling errors - Calibration and synchronization are not trivial
	Multi calibrated with kin. constraints	3D	+ Compensates offsets + Constrains limb lengths and joint angles - Inaccuratein some joint angles
Sensor fusion with RGB video	N/A	3D	• With IMUs: More accurate, but not markerless • With one RGB-D camera (Depth + OpenPose on RGB): still inaccurate

Table 1.1: Pros and cons in state-of-the-art approaches for human motion analysis. The multi-person prospect is not addressed, as it can be available with all approaches, but it is not always. IMU: Inertial Measurement Unit. N/A: Not Applicable. kin.: kinematic. RGB-D: red-green-blue-depth.

Type	Name	License	Tasks	Notes
Marker-based, 3D	Vicon Nexus [?]	Commercial	3D kinematics	Reference method
	Qualisys QTM [?]	Commercial	3D kinematics	Reference method
IMU, 3D	Xsens [?]	Commercial	3D kinematics	Sensors & software
	APDM [?]	Commercial	3D kinematics	Sensors & software
	OpenSense [?]	Open-source	3D kinematics	Uses Xsens or APDM sensors
RGB-D, 2.5D	Kinect [?]	Open-source	2.5D pose & shape	Kinect 1 & 2 discontinued, Azure Kinect is different
Mono RGB, 2D	Dartfish [?]	Commercial	2D kinematics	Semi-automatic approach but very widespread
	Kinovea [?]	Open-source	2D kinematics	Semi-automatic approach but very widespread
	Mobile Gaitlab [?]	Open-source	2D kinematics	Only gait parameters
	OpenPose [?]	Open-source	2D pose	Bottom-up: better in crowds
	AlphaPose [?]	Open-source	2D pose	Top-down: better at low resolution
	DeepLabCut [?]	Open-source	2D pose	Trainable any on custom dataset
	SLeap [?]	Open-source	2D pose	Trainable on any custom dataset
	OpenPifPaf [?]	Open-source	2D pose	Temporal consistency, no feet
	Mask R-CNN [?]	Open-source	2D pose & silhouette	Accurate but slow, no feet
	YOLOv7 [?]	Open-source	2D pose & silhouette	Fast, no feet
Mono RGB, 2D & 3D	Mediapipe BlazePose GHUM [?]	Open-source	2D & 3D pose	Single person, fast
	MMPose [?]	Open-source	2D & 3D pose, 3D shape	Allows for choosing dataset and architecture
Mono RGB, 3D	VIBE [?]	Open-source	3D shape	Successor of SPIN. Fits 3D SMPL model on 2D OpenPose keypoints
	STRAPS [?]	Open-source	3D shape	Fits not only from pose, but aksi from silhouette for better morphology
Multi RGB, 3D	Theia3D [?]	Commercial	3D kinematics	Patent-protected, rather accurate
	The Captury [?]	Commercial	3D kinematics	Patent-protected, not accurate enough for biomechanics
	Simi Shape [?]	Commercial	3D kinematics	Not peer-reviewed
	Pose2Sim [?]	Open-source	3D kinematics	3D pose constrained to a skeletal model, rather accurate
	OpenCap [?]	Open-source	3D kinematics & kinetics	Markerset inferences from OpenPose keypoints. Not GDPR compliant
	Anipose [?]	Open-source	3D pose	Usable with any DeepLabCut model
	DLTdv8 [?]	Open-source	3D pose	Semi-automatic approach, in Matlab (not free)
	EasyMocap [?]	Open-source	3D pose, 3D shape	Inaccurate morphology
	FreeMocCap [?]	Open-source	3D pose	Simple OpenPose triangulation
	pose3d [?]	Open-source	3D pose	Simple OpenPose triangulation, in Matlab (not free)
	OpenPose 3D module [?]	Open-source	3D pose	Not maintained since 2017

Table 1.2: A number of approaches has been proposed in the literature. Fewer of them have been released, and are available for further kinematic analysis. IMU: Inertial Measurement Unit. RGB: image sensor (red-green-blue). RGB-D: image and depth sensor.

2 Theoretical framework

Obtaining coherent 3D kinematics from a network of calibrated video cameras involves understanding a certain theoretical framework. First, keypoints must be recognized in images. This is mostly achieved with machine learning models. Then, all the 2D features detected for each camera need to be reconstructed in the 3D space. Finally, these coordinates must be constrained to an anatomically consistent model, in order to obtain coherent 3D joint kinematics.

Contents

2.1 Introduction

Sports movements don't generally only lie in the sagittal plane, and they often cause body part occlusions. Moreover, although the need is not as strong as for clinical applications, it is important for results to be as biomechanically coherent as possible. Hence, one of the most promising prospects for sports movement analysis consists in addressing the problem with several video sources, and then constraining 3D coordinates to a kinematic model. Such research is at the intersection of machine learning for 2D pose estimation, computer vision for 3D reconstruction from a network of calibrated videos, and biomechanics for constraining 3D point coordinates to an anatomically consistent model, in order to obtain reliable kinematics.

2.2 2D pose detection

2.2.1 Why machine learning?

As a first step, achieving motion analysis from a network of cameras involves detecting features in images. These features can be whole human beings, joint centers, body landmarks, sports gear such as tennis balls, climbing holds, or much more.

Two broad approaches can be implemented: the first one consists in using dedicated algorithms for each task. The gist of it is to understand the task well enough to build an appropriate solution: this is a knowledge-driven approach. Among other techniques, corner and contour detection, color thresholding, affine transformation, template matching, watershed segmentation, can be used. For example, if one wants to differentiate two boxers wearing respectively a blue and a red shirt, they can filter them by color. If one needs to identify on which portion of a speed climbing wall an athlete is, they can match the template of each holds on the whole image. OpenCV [**?**] provides convenient tools for this purpose, in C++ and Python languages. This approach is often fast, but also quite complicated to implement, and neither flexible nor robust. If there are other red or blue patches in the boxing scene, if the boxer wears green or if the light is poor, this will not work anymore. Likewise for holds, if the sun casts a large shadow which changes its apparent shape, or if holds are seen from a different perspective. Likewise, semi-automatic approaches such as Kinovea [**?**], which tracks manually annotated points, do not generalize well to challenging contexts.

The second approach takes advantage of machine learning algorithms, which constitute an entirely different paradigm. The idea is to show the machine enough examples for it to "understand" by itself its underlying attributes, so that it manages to detect and label automatically new images: this is a data-driven approach. It can be used for both aforementioned tasks, in a much more flexible way: if one wants the system to recognize boxing gloves or holds in challenging conditions, they simply have to include such examples while training the model. The machine learning approach is also suitable for other tasks, such as whole-image classification (e.g., determining whether this is a boxing or a BMX scene), object detection (e.g., localization of a bike and of a person with a bounding box), background extraction [**?**], semantic and instance segmentation (e.g., extracting the shape of the bike and of the person) [**?**], or keypoint detection (e.g., localization of human joint centers and keypoints on a bike [**?**]) (Figure **??**). By 2015, data-driven methods definitely took over knowledge-driven ones in vision analysis problems, and by extension in sports motion analysis from videos (Figure **??**).

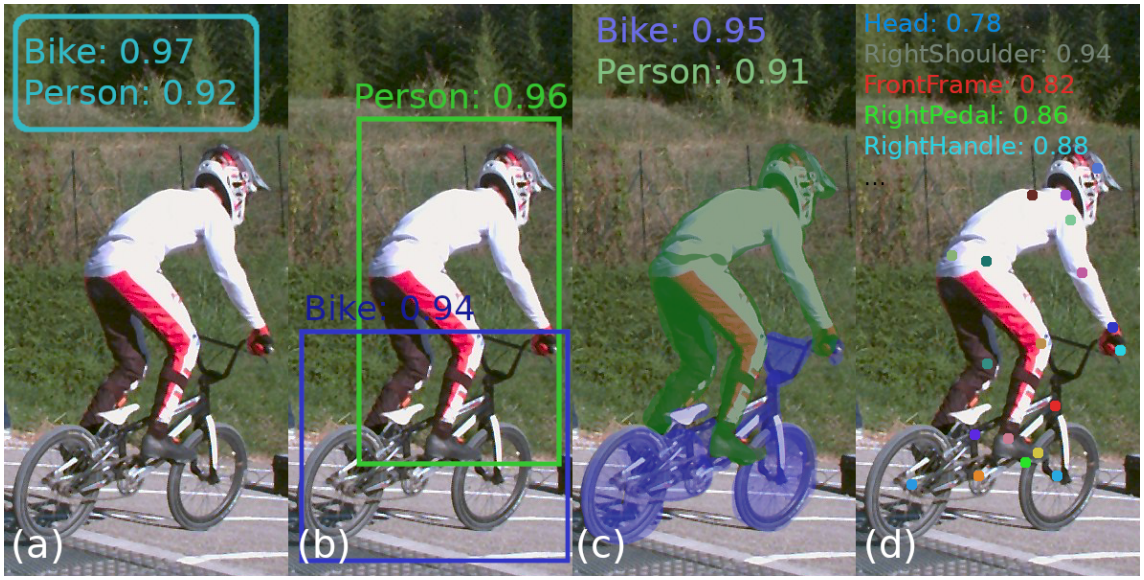


Figure 2.1: Mock examples of different types of image analysis. (a) Whole image classification, (b) Object detection and localization, (c) Instance segmentation and shape extraction, (d) Keypoint detection.

2.2.2 Machine learning timeline and principles

Machine learning is a subset of artificial intelligence (AI.) As such, one can trace its origin back to the discovery of the natural neuron at the end of the 19th century, by Nobel Prize Ramón y Cajal [**?**], followed half a century later by the first model of an artificial neuron [**?**]. A natural neuron is a simple learning unit, which collects the nervous influx sent by other neurons to its dendrites, and sends an action potential when the total influx weighted and summed in the soma overcomes a threshold value. This potential is then transmitted through the axon to the next neuron as a new influx. Similarly, an artificial neuron receives output vectors from previous neurons, weighs and sums them with a summation function, and transfers the resulting output vector to the next neurons if it reaches a certain threshold determined by an activation function (Figure **??**a-b).

The perceptron, invented in 1956 [**?**], represents the first practical application of an artificial neuron. It acts as a binary classifier which predicts class 1 if the neuron is fired, and class 0 otherwise. It automatically adjusts its weights by learning from previously labeled example data (see Algorithm **??** and Figure **??**b). It could be used, for example, to predict whether an athlete is going to be "good" or not, given his force-velocity results on an ergometer test (see step-by-step Example 1 and Figure **??**), and given enough example data. Needing previously labeled data makes it is a supervised classifier – we will not discuss unsupervised methods here. Of course, this example is oversimplified. Being good or not as a sport is a complex and multifactorial outcome, and two variables can't sum it up. However, the perceptron can take more than two variables as inputs (for example, force, velocity, and endurance), and it can also be generalized to multiclass classification with more than two outputs (for example, to differentiate between strong, explosive, and resistant type of athletes.)

Nevertheless, it often takes a lot of iterations over good quality training data for the perceptron to converge. Moreover, it does converge if and only if the data are linearly separable, i.e., if they can be separated with a straight line [**?**] (see Figure **??**). Some fundamental problems such as the XOR gate can't be solved with a basic single layer Artificial Neural Network (ANN) [**?**]. This constituted one of the early setbacks for AI. Then, the high computational cost of these approaches, combined with the complexity of common-sense problems, hampered the trust in learning methods. Indeed, vision and language problems require enormous amounts of data, and

can't be solved with a simple dictionary (for example, "the spirit is willing but the flesh is weak" becomes "the vodka is good but the meat is rotten" when translated back and forth from English to Russian.) Overinflated promises and expectations, followed by disappointment in academia and industries, led to cuts in funding, and eventually loss of skills in the 1970s: this is referred to as the first AI winter.

The AI field survived by focusing on specific problems, called expert systems. In the early 1980s, a new rise was triggered by massive funding such as the Japanese Fifth Generation Computer project, aiming to build a supercomputer that could solve any problem. Shortly after, multilayer neural networks were made possible with the (re)discovery of backpropagation [**?**], or more rigorously of weight adjustment thanks to the backpropagation of error gradient, from the last layer to the first one. As it is not the central subject of this thesis, the algorithm and early references will not be detailed here, but the interested reader can refer to [**?**]. This allowed for solving non-linearly separable problems, and for tackling real world issues (Figure **??**c.). [**?**] proved that one single intermediate layer is enough to solve any given classification problem, granted that this layer contains enough neurons (although sometimes too many to make it possible in practice.) On the other hand, kernel tricks were also rediscovered [**?**, **?**], and made non-neural networks such as support vector machines (SVMs) [**?**] able to treat non-linearly separable data with much less training data, more optimally, and on a clearer mathematical ground (Figure **??**). However, again, unrealistic expectations were confronted with unplanned technical difficulties both on expert systems and on general intelligence projects. This led to a second AI winter in the 1990s.

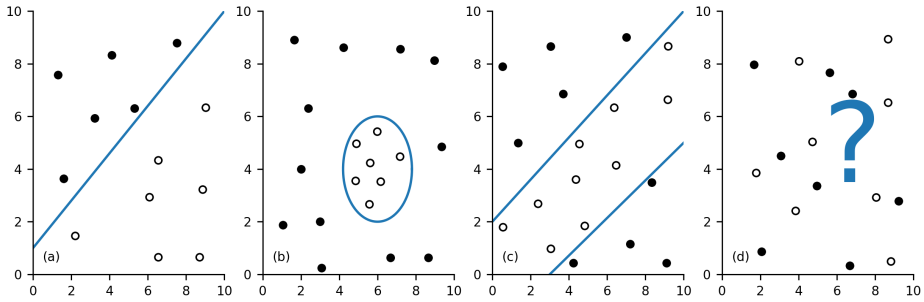


Figure 2.2: Single layer artificial neural networks such as the perceptron can only classify linearly separable data. (a) is linearly separable. (b) is not linearly separable. However, data are contained in an ellipse. The equation of an ellipse is of the form $a \times x^2 + b \times y^2 = 1$, so if we transform the feature variables into $X = x^2$ and $Y = y^2$, the data become linearly separable. (c) is equivalent to a fundamental XOR gate, and is not linearly separable, which was part of the reasons for the first AI winter. It can either be solved by combining several layers of artificial neurons, or by complex kernel tricks which map the data from the original space into a higher dimensional space where they become linearly separable. (d) is possibly not separable at all. AI: Artificial Intelligence. XOR: Exclusive OR.

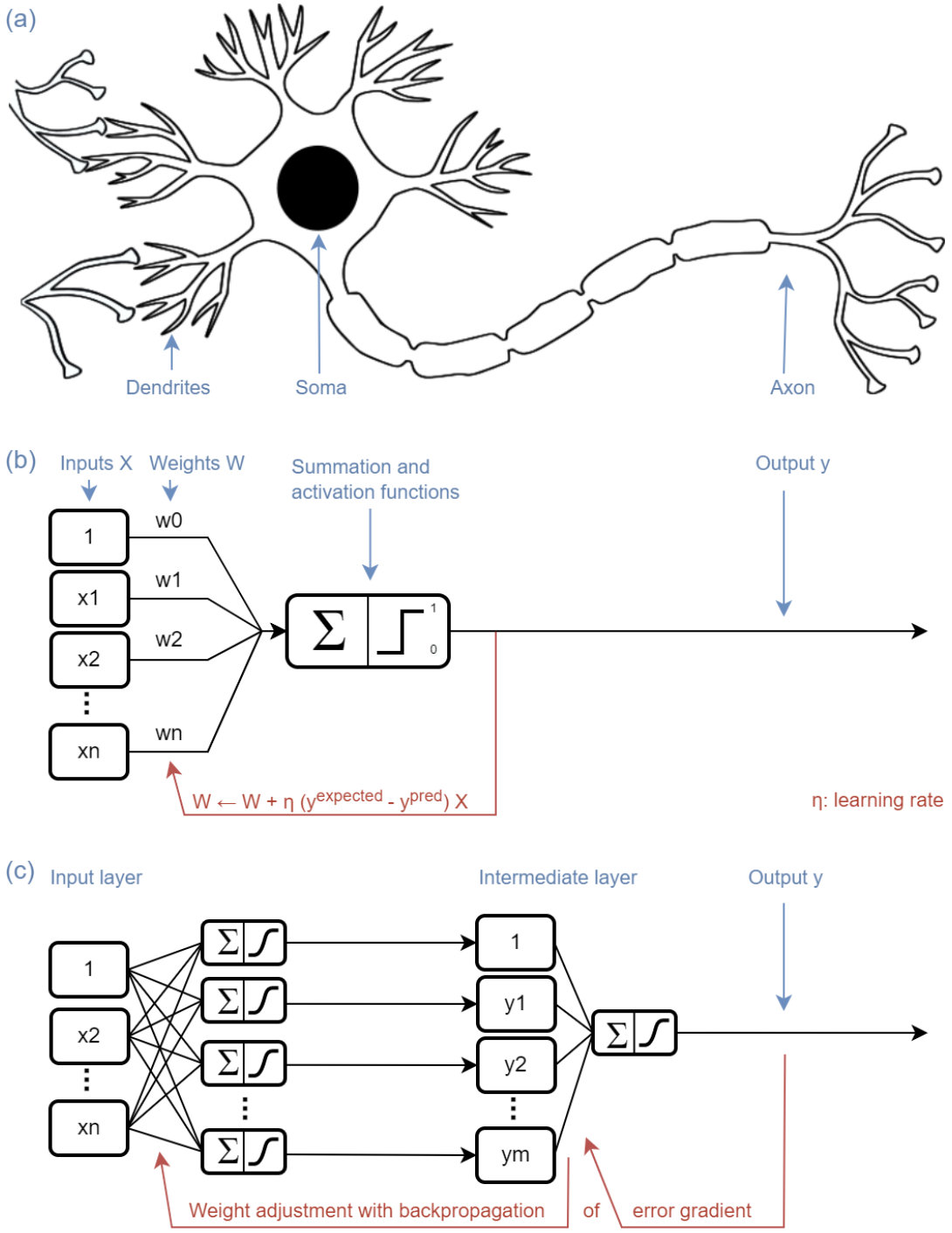


Figure 2.3: The artificial neuron (b) has been modeled after the natural neuron (a). Inputs and weights act as the total nervous influx firing the dendrites. The collected values are summed, and a signal is activated if a threshold is overcome, as the soma does in a natural neuron. The output signal is conveyed the axon in a natural neuron. (b) In the case of a perceptron, the neuron adjusts its weights to minimize the error between the predicted and the expected output. It can be used as a classifier, which outputs class 1 or class 0 depending on the inputs. (c) A dense (fully connected) neural network with one intermediate layer and backpropagation can solve any non-linearly separable classification.

Algorithm 1 Perceptron

Let $\overrightarrow{X^0}$ be the input vector of a first instance of variables $(1, x_1^0, \cdots x_M^0)$, $\overrightarrow{W^0}$ the corresponding weights randomly initialized $(w_0^0, w_1^0, \cdots w_M^0)$ with w_0^0 a bias, and $y^{0,pred}$ the output predicted binary class.

1: The summation function is computed:

$$\overrightarrow{W^0} \cdot \overrightarrow{X^0} = \sum_{m \in [0,M]} w_m^0 x_m^0 \tag{2.1}$$

2: This result is processed by an activation function, which is a threshold in the case of the perceptron. It determines whether the neuron will be fired or not, i.e., whether one or the other class will be predicted. $y^{0,pred} = 1$ corresponds to one class, and $y^{0,pred} = 0$ to the other.

$$y^{0,pred} = \begin{cases} 1 & \text{if } \overrightarrow{W^0} \cdot \overrightarrow{X^0} > \theta, \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

3: This prediction $y^{0,pred}$ is compared to the actual class $y^{0,expected}$.

$$\varepsilon^0 = y^{0,expected} - y^{0,pred} \tag{2.3}$$

4: Then weights are updated:

$$\overrightarrow{W^1} = \overrightarrow{W^0} + \eta \; \varepsilon^0 \; \overrightarrow{X^0} \tag{2.4}$$

with η the learning rate $\in [0,1]$. Note that if the class is correctly predicted, then $\varepsilon^0 = 0$ and weights are not adjusted.

5: The algorithm is repeated with another example $\overrightarrow{X^1}$, and so on until it has gone through the whole batch of the training set. If weights still need to be updated, one can go over it again, for a determined number of epochs or until the average error is under a given value. Then the perceptron is considered trained, and ready to correctly predict a class y with the retained weights.

Example 1 Athlete classification with a perceptron

N.B. The code for running this example is available on the thesis repository https://github.com/davidpagnon/These_David_Pagnon/blob/main/Thesis/Chap2/perceptron.py.

Let's consider force-velocity test results as an input
$\overrightarrow{X} = (1, velocity\ (m/s), force\ (hN))$,
and the classification of an athlete as "skilled" or "unskilled" as an output $y = 1$ *or* 0.
A batch of training data, i.e., example data the perceptron will learn from, could be:
$\left\{(\overrightarrow{X^i}, y^{i,expected})\right\}_{i \in [0,4]} = \left\{\left((1,1,5),1\right), \left((1,2,3),0\right), \left((1,7,1),1\right), \left((1,4,1),0\right), \left((1,5,4),1\right)\right\}$.
Let's randomly initialize weights at $\overrightarrow{W^0} = (-9,1,3)$, take a threshold θ=0.1, and a learning rate η = 0.3.

The first instance of the training set gives:
$\overrightarrow{W^0} \cdot \overrightarrow{X^0} = \sum_{m \in [0,2]} w_m^0 x_m^0 = -9 \times 1 + 1 \times 1 + 3 \times 5 = 7$.
Now $\overrightarrow{W^0} \cdot \overrightarrow{X^0} = 7 > \theta = 0.1$, so $y^{0,pred} = 1$.

$y^{0,expected} = 1 = y^{0,pred}$, so the prediction is true and weights don't need to be updated.
As a consequence, $\overrightarrow{W^1} = \overrightarrow{W^0} = (-9,1,3)$.

The second instance gives $\overrightarrow{W^1} \cdot \overrightarrow{X^1} = (-9,1,3) \cdot (1,2,3) = 2 > \theta = 0.1$, so $y^{1,pred} = 1$.
But $y^{1,expected} = 0 \neq y^{1,pred} = 1$, so weights need to be updated.
The error is $\varepsilon^1 = y^{1,expected} - y^{1,pred} = 0 - 1 = -1$.
As a consequence, $\overrightarrow{W^2} = \overrightarrow{W^1} + \eta \ \ \varepsilon^1 \ \ \overrightarrow{X^1} = (-9,1,3) + 0.1 \times (-1) \times (1,2,3) = (-9.3, 0.4, 2.1)$.

Third instance: $\overrightarrow{W^2} \cdot \overrightarrow{X^2} = (-9.3, 0.4, 2.1) \cdot (1,7,1) = 3 - 4.4 < 0.1$, so $y^{2,pred} = 0$.
$y^{2,expected} = 1 \neq y^{2,pred} = 0$, so weights need to be updated.
$\varepsilon^2 = y^{2,expected} - y^{2,pred} = 1$.
$\overrightarrow{W^3} = \overrightarrow{W^2} + \eta \ \varepsilon^2 \ \overrightarrow{X^2} = (-9.3, 0.4, 2.1) + 0.1 \times 1 \times (1,7,1) = (-9, 2.5, 2.4)$.

Fourth instance: $\overrightarrow{W^3} \cdot \overrightarrow{X^3} = (-9, 2.5, 2.4) \cdot (1,4,1) = 3.4 > 0.1$, so $y^{3,pred} = 1$.
$y^{3,expected} = 0 \neq y^{3,pred} = 1$, so weights need to be updated.
$\varepsilon^3 = y^{3,expected} - y^{3,pred} = -1$.
$\overrightarrow{W^4} = \overrightarrow{W^3} + \eta \ \varepsilon^3 \ \overrightarrow{X^3} = (-9, 2.5, 2.4) + 0.1 \times (-1) \times (1,4,1) = (-9.3, 1.3, 2.1)$.

Fifth instance: $\overrightarrow{W^4} \cdot \overrightarrow{X^4} = (-9.3, 1.3, 2.1) \cdot (1,5,4) = 17.6 > 8$, so $y^{4,pred} = 1$.
$y^{4,expected} = 1 = y^{4,pred} = 1$, so weights don't need to be updated.
$\overrightarrow{W^5} = \overrightarrow{W^4} = (-9.3, 1.3, 2.1)$(*Figure* **??**).

Next instances: Once we have gone over the batch of training data, if the average error is below a given value, we can assume that the perceptron is trained. If not, we can use the next batch to pursue training. If it still didn't converge after all batches, we can iterate over all training data again, for a given number of times. If results are still not satisfying, either the data are not linearly separable, or the training sample is not large enough or of good enough quality. In our case, it seems like our example data have allowed us to correctly separate skilled and unskilled athletes based on their force and velocity test results (Figure **??**).

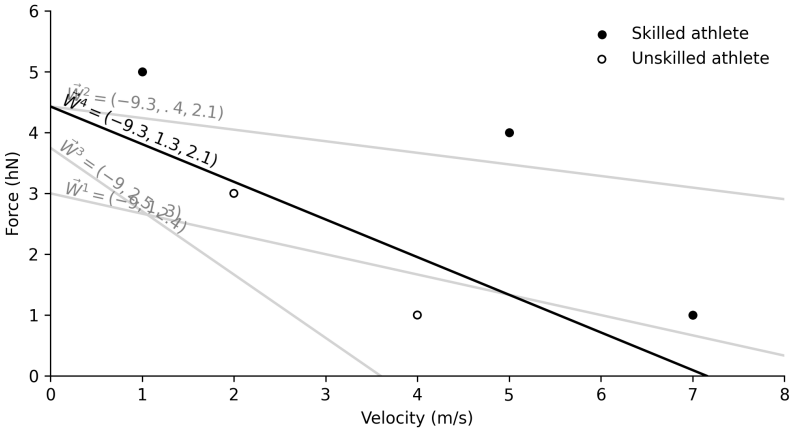


Figure 2.4: Classification of athletes as "skilled" (black dot) or "unskilled" (circle) according to their Force-Velocity results. Weights are adjusted (grey lines), until the perceptron classifies athletes correctly (black line.)

From the end of the 1990s, there has been no theoretical breakthrough in AI, but larger databases have become available with the advent of the Internet, and greater computational power has become accessible, especially thanks to groundbreaking progress in Graphics Processing Units (GPUs), which made heavy parallel computing available to the wider audience. As a consequence, more layers could be used in neural networks, which progressively set off the onset of deep learning. Finally, complex "common-sense" problems, such as natural language processing or image recognition, could be treated with some success [**?**].

One particular type of deep learning algorithms is the convolutional neural network (CNN), which is particularly suited for image recognition. It was first used for classifying handwritten and low-resolution digits [**?**], and then applied to more complex images as greater computing resources became available [**?**]. Nowadays, CNNs have sometimes surpassed humans at image classification [**?**, **?**]. A convolution layer consists in a series of filters that slide across the image, each of them outputting a result close to 0 or to 1, depending on how well it can be overlaid on each image area. In the same way as with a simple artificial neuron, each of these filters can be seen as a weight vector $\overrightarrow{W}$, and each image area as an input vector $\overrightarrow{X}$. The filters of the first convolution layer are simple patterns such as lines, but then they become circles and corners, until the last layers, when they have developed into complex features corresponding to whole object parts. Once a filter has covered the whole image, it forms a feature map, which will then be downsampled by a pooling layer in order to save computing resources. All the feature maps produced by each filter are processed by a determined number of other convolution layers, and then flattened into a 1D vector. This 1D vector is processed by a few dense layers (dense layers are fully connected, i.e., all outputs are produced by a weighted sum of each input), and lastly a softmax layer computes a probability for the image to correspond to each available class. If the CNN is correctly trained, the class with highest probability corresponds to the correct one: for example, if the image displays a BMX start, the probability for the bike class will be the highest (Figure **??**).

However, results will not be good until a lot of iterations are done on a lot of data. Indeed, filters at each layer are randomly initialized, and then refined with backpropagation in order to predict all classes as best as possible. One of the risks is overfitting, i.e., to excessively adapt to the training data and to fail to generalize to new data. This is dealt with by cross-validation, i.e., the separation between training and test data, by regularization methods such as radomization, batch normalization and dropout, and by data augmentation, e.g., image rotations, crops, color distortion, noise addition, etc. [**?**, **?**] (Figure **??**). An enormous amount of data is also needed to correctly train the CNN, which makes it complicated when unusual classes need to be recognized (for example, a climbing hold, a BMX starting gate, a medial malleolus on the ankle, etc.) Fortunately, one can consider that a CNN trained on a massive dataset, such as ImageNet and its 14 million annotated images [**?**], has learned to recognize most features that can be found in any image. One can take the learned filters of its convolutional layers as is, use them as a feature extractor (sometimes called backbone), and just fine-tune the last dense layers to recognize new classes. It will be much less computationally expensive to train, and will need much fewer data: about a hundred images, instead of thousands. This is called transfer learning [**?**].

Now, classification of a whole image is not sufficient in sports motion analysis. One needs to detect where an object or a person is, and ideally to localize more precise features such as joint centers so as to estimate the person's pose.

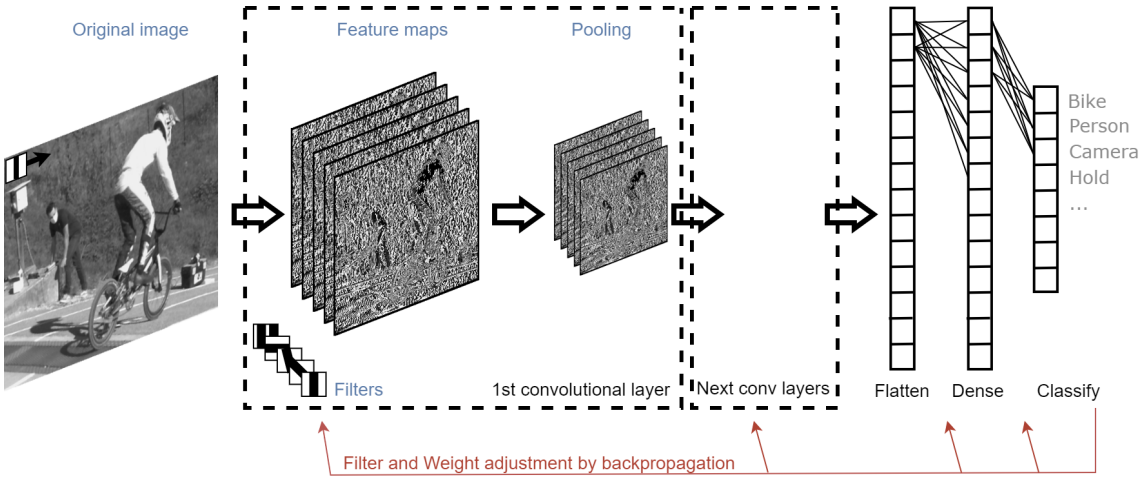


Figure 2.5: A simplified convolutional neural network (CNN.) A convolutional layer consists in a series of filters running across the input image, and producing feature maps, which are then downsampled by pooling. Filters become more and more elaborated along layers, and produce feature maps which look like whole object parts. Filters and weights are randomly initialized at first, and then are adjusted by backpropagation. After the convolutional layers, the feature maps are flattened to produce a 1D vector, which is then processed by dense layers, and finally a softmax layer computes a probability for the image to correspond to each available class.

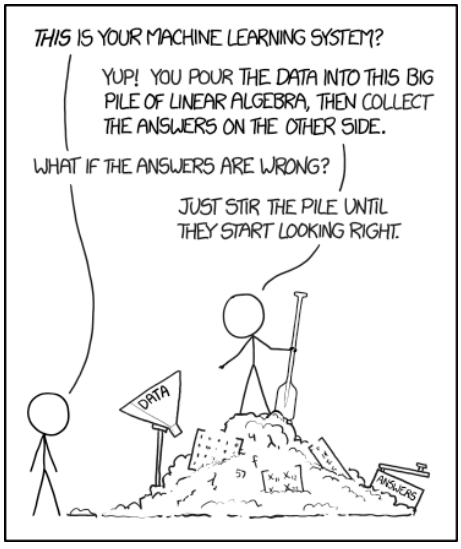


Figure 2.6: The amazingly rigourous world of machine learning. XKCD .

2.2.3 Machine learning for 2D pose detection

Older methods for object detection used to run a sliding and pyramidal window across the image, and then to apply a non-neural classifier on each window, such as an SVM on carefully handcrafted histogram of oriented gradients descriptors (HOG) [**?**]. They then had to be followed by non-maximum suppression, in order to select one bounding box over many overlapping ones. As the classifier is run on each window iteration like if they were independent images, these methods were very computationally intensive, and in the same time not very robust nor accurate.

More modern approaches are based on CNNs, and as such, they involve a preliminary step: extracting the last layer of a pre-trained neural network such as ImageNet, in order to make it able to classify the objects of interest. One of the precursors, R-CNN (Regions with CNN features) [**?**], first looks for a lesser amount of regions of interest (ROIs) by selective search, instead of with a sliding window. Selective search is an algorithm which segments image based on pixel intensities, without any learning involved [**?**]. Then three learning models are used: one CNN for extracting features from each ROI, an SVM for classifying each ROI, and a regression model for adjusting bounding boxes. It takes about 45 seconds to process a single image on benchmarks. Fast R-CNN [**?**] uses one single network for all steps, and switches the first two: it first extracts features from the whole image, and only then uses selective search to find ROIs on the resulting feature map, and finally classifies the ROIs and tightens the bounding boxes. It is much faster and takes about 2 seconds per image. A last incrementation on this basis is Faster R-CNN [**?**], which works similarly to the latter, but finds ROIs with a neural network instead of with selective search, which is very time-consuming. This allows for predicting an "objectness" score on each ROI, and for fitting the bounding boxes directly, and thus on avoiding the last regression step. It is even faster, and takes about 0.2 seconds per image. YOLO (standing for You Only Look Once) [**?**] proposes another approach, and does not separate the steps of finding ROIs with classification. It divides the image into regions, and predicts both classes and bounding boxes for each region. For example, if there is a shoulder in a region, it will predict a "person" class, and a larger box in which this person is likely to fit. YOLO takes about 0.02 seconds per image (45 fps), and is thus able to run real time. However, it is not as accurate as the previous methods, especially on smaller objects. This being said, new versions are very frequently released (although not by the same authors), and the current YOLOv7 [**?**] is both faster and more accurate than all previous approaches as it entirely reviews the whole network architecture to deal with all observed bottlenecks.

But again, in order to perform joint kinematics, one cannot just detect whole objects: precise keypoints need to be localized. Mask R-CNN [**?**], and the more recent Detectron2 [**?**], still predict the bounding boxes and their class like Faster R-CNN does, but they also add a small overhead in parallel, which predicts the shapes of masks overlaying the object in a pixel-to-pixel manner. Keypoints can be seen as a very small mask, and Mask R-CNN can also detect them in order to predict human pose estimation. In the next paragraph, only multi-person pose estimation models will be considered. Datasets, evaluation metrics, and comparison of results won't be detailed: see [**?**] for a comprehensive overview.

Two main approaches for multi-person 2D pose estimation coexist. The "top-down" one first detects bounding boxes around persons, and then finds keypoints inside each box. In the area of object detection methods, they are analogous to region-proposed methods such as the R-CNN suite, which propose ROIs and then find and classify objects. Conversely, the "bottom-up" approach first finds keypoints, and then groups them into persons. They are analogous to the single-shot object detection methods such as the YOLO suite, which first find small details, and then predict full-size objects. These approaches are nowadays almost as fast as the top-down ones, however their inference time does not increase with the amount of persons detected.

Mask R-CNN belongs to the first kind, as well as AlphaPose [**?**], which mostly differentiates from the latter by using a network predicting higher quality bounding boxes from inaccurate ones, in order to facilitate the task of the joint regressor. On the opposite, DeepCut and DeeperCut [**?**, **?**], as well as DeepLabCut [**?**, **?**] upon which it is built, are bottom-up approaches. They find a large number of keypoint candidates, label them as hand, head, etc., and then select the best candidates and separate them into persons. Since they calculate every possible association between keypoints, this is very slow. OpenPose [**?**] uses a network which jointly predicts keypoint locations, and the connections between them (i.e., it also predicts limbs, which define a skeleton), and is much faster while still being accurate. OpenPifPaf [**?**] adds to it both temporal consistency across frames, and an intensity map for each keypoint instead of punctual locations (i.e., a further keypoint will have a lower intensity). This allows for better accuracy in low-resolution regime and in occluded images.

YOLOv7 supports keypoint detection by integrating YOLO-Pose [**?**], and claims to be faster and more accurate than all other state-of-the-art methods. It brings together top-down and bottom-up approaches, and uses a single network predicting both bounding boxes and their corresponding poses. SLEAP [**?**], which is built for training animal pose estimation models, implements both top-down and bottom-up approaches. In this context, top-down approaches are slightly more accurate, and considerably faster as long as few animals are in the scene.

Figure 2.7: The body_25b OpenPose model is more accurate than the default body_25 one. As an example, the left knee is slightly misplaced on the default model. Keypoint definition and order also differ between both models.

Like all previously presented methods, OpenPose has been trained on the COCO dataset [**?**]. However, OpenPose body_25 standard model also provides foot keypoints, which are often primordial in sports motion analysis. To do so, 6 more keypoints have been labeled for the feet on the COCO dataset before training. OpenPose also supports the single-network whole-body pose estimation network body_135 [**?**], which has been trained in the same time on COCO+foot, MPII [**?**], and on Total Capture [**?**] in order to provide hand, face, feet, and body keypoints in one single network. The body_135 model is slow and requires high capacity hardware, however a submodel of it is body_25b, which provides body and foot keypoints as body_25 does, and in addition decreases the number of false positives without hampering speed. Its keypoint definition differs slightly to the default model's (Figure **??**): it adds the MPII head and neck keypoints, and removes the artificially created neck and middle hip points of the body_25 model (which are simply the middle point of the shoulders and the hips). In a similar way, AlphaPose provides full-body models, either trained on the Halpe dataset [**?**], or on the COCO-WholeBody one [**?**]. Note that BlazePose [**?**], trained on the GHUM dataset [**?**], also provides hand and feet keypoints, but since it is a single-person pose estimation model, the architecture is different and will not be addressed here. Indeed, this is rarely suitable in sports conditions, where people are usually present in the background.

2.3 3D reconstruction

Once the pose of an athlete is correctly detected, the next step is to obtain their 3D pose. While some approaches strive to infer 3D pose from a monocular video source, they are generally not considered sufficiently accurate, especially when body parts are occluded. It is, then, important to use several cameras, and to fuse their 2D pose estimation results to obtain more reliable 3D coordinates.

2.3.1 Pinhole camera model

History

If it passes through a small enough hole, the light emitted from a point can be seen as a fine ray rather than a large beam. Then, when all the rays from the object go through, they do not overlap, and actually recreate an inverted image of it, instead of a blurry spot of light. If the hole gives entrance to a dark enough room, and if a light-colored sheet is positioned close enough to the hole, one can observe the projection of a dim, but distinct image of the object. This is the principle of the camera obscura, or pinhole camera, which might have been discovered as early as in the paleolithic era, as cave paintings seem to suggest. The concept was later used as a drawing aid in the 17^{th} century by some artists, probably such as Vermeer or Canaletto [**?**] (Figure **??**). In order to let more light in and to avoid diffraction issues, they increased the apperture of the hole and inserted in a convex lens, at the expense of a shorter depth of field and of some image distortion (Figure **??**). Then, if instead of a sheet, a light-sensitive film is placed, the image is progressively printed, after enough light has struck it. This is how photography was invented in the 19^{th} century, by the French inventors Niépce, and then Daguerre [**?**].

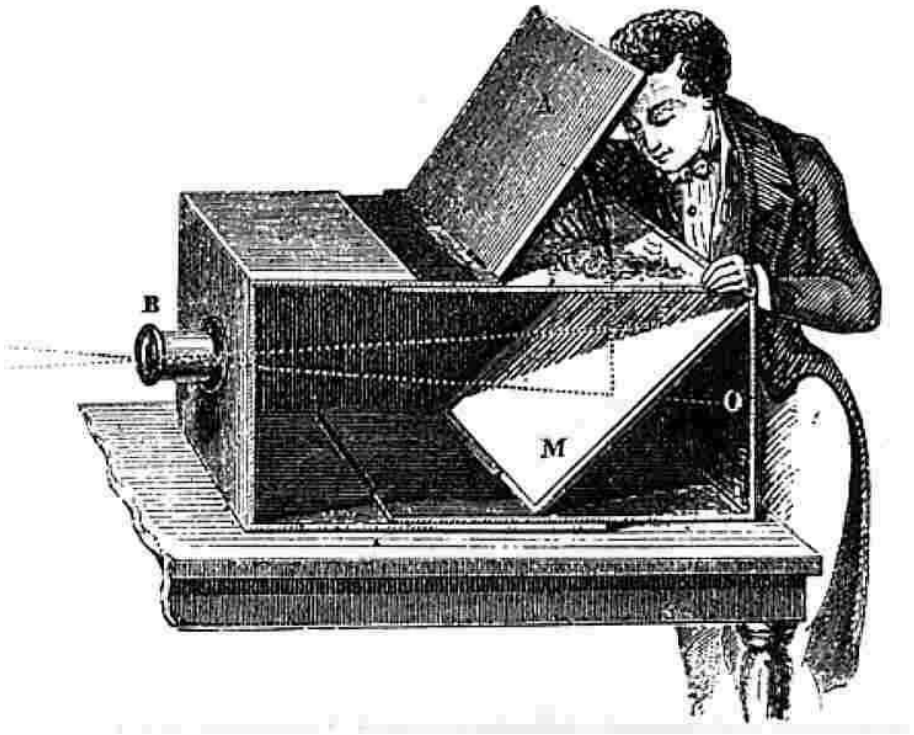


Figure 2.8: A camera obscura used as a drawing aid, which uses a lens rather than a pinhole (unknown author).

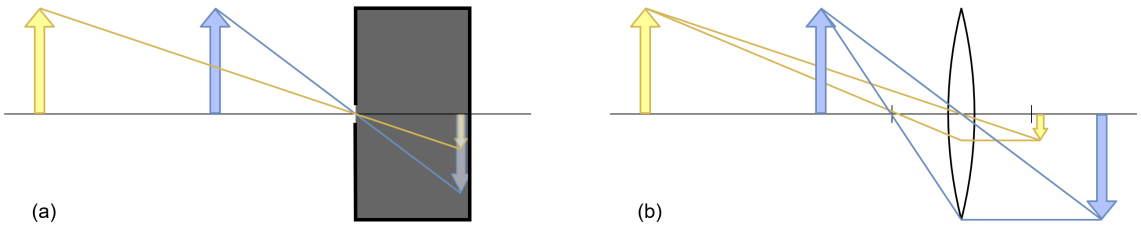


Figure 2.9: (a) A pinhole camera does not need to be focused, because in theory, only one single ray per object point will pass through the apperture–which incidentally makes the image dimmer. (b) On the contrary, more light rays can pass through a lens, which makes the projected image clearer–but incidentally, it introduces a finite depth of field, as the image is focused on a different plane depending on the object distance to the lens.

Thin lens

When an object of size X_O (O standing for object) is projected from a distance Z_O through a thin convex lens of focal length f, it is in focus at a specific distance z. In this case, let the image size be x. Assuming that the lens is thin, and that the object is close to the optical axis(i.e., that paraxial conditions are satisfied), Thales' theorem gives (Figure **??**):

$$\begin{cases} \frac{x}{X_O} = \frac{z}{Z_O} \text{ (dashed triangles)}, \\ \frac{x}{X_O} = \frac{z-f}{f} \text{ (dotted triangles)}, \end{cases} \tag{2.5}$$

Which leads to the thin lens formula, discovered by Descartes in the 17^{th} century:

$$\frac{1}{Z_O} = \frac{1}{f} - \frac{1}{z} \tag{2.6}$$

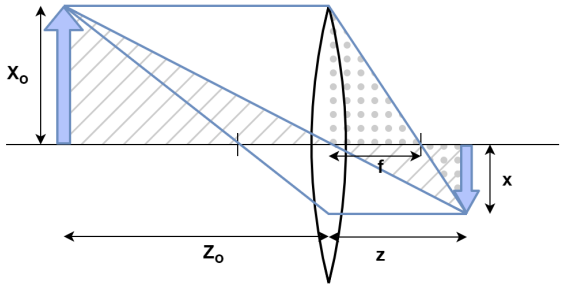


Figure 2.10: Thales' theorem leads to the thin lens equation. X_O is the object of size, and Z_O its distance to the lens; x is the image size, and z its distance to the lens; and$f$ is the focal length f.

Basic pinhole model

According to the thin lens formula, in the specific case where the object is at a distance Z_O much larger than the focal length f, one can approximate that the image is into focus at a distance z equal to the focal length.

$$z = f \tag{2.7}$$

This is generally true in motion capture, for which the camera lens is usually focused to infinity, i.e., it looks at objects at least 100 times as far as the focal length. In this case, the Thales theorem gives the relation:

$$x = \frac{f \times X_O}{Z_O} \tag{2.8}$$

This model is called the basic pinhole camera model [**?**,**?**,**?**]. This is strictly speaking improper, since a pinhole camera does not have any focal length. For the sake of simplification and since it does make a difference in practice, the image plane is usually represented upside-up and on the same side as the object (Figure **??**).

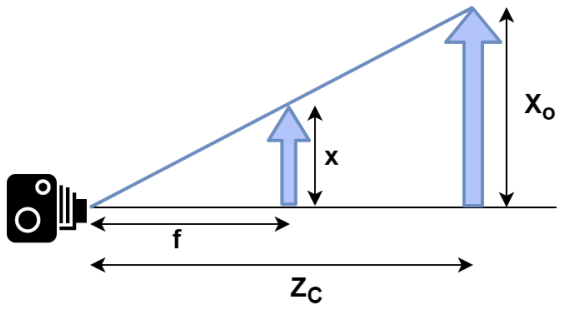


Figure 2.11: The simplified pinhole camera model.

3D pinhole model in camera coordinate system

In 3 dimensions, the object can be represented in the camera coordinate system (X_C, Y_C, Z_C), with Y_C pointing downwards, and Z_C facing toward the object (Figure **??**). The relation becomes:

$$\begin{cases} x = \frac{f \times X_C}{Z_C}, \\ y = \frac{f \times Y_C}{Z_C}, \end{cases} \tag{2.9}$$

Which can be written as:

$$Z_C \times \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f & 0 \\ 0 & f \end{pmatrix} \begin{pmatrix} X_C \\ Y_C \end{pmatrix} \tag{2.10}$$

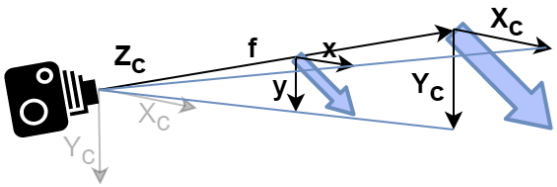


Figure 2.12: The pinhole camera model in 3D.

3D pinhole model with pixel image coordinates

The origin of the image coordinates is not usually set on the optical axis (i.e., at the center of the sensor, also called the principal point), but typically at its upper left (Figure **??**). With c_u and c_v the horizontal and vertical coordinates of the principal point, the coordinates in the image frame of reference can be written as:

$$\begin{cases} u = x + c_u = \frac{f \times X_C}{Z_C} + c_u \\ v = y + c_v = \frac{f \times Y_C}{Z_C} + c_v \end{cases} \tag{2.11}$$

The relation is thus:

$$Z_C \times \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix} \tag{2.12}$$

Moreover, instead of expressing the image coordinates and the focal length in meters, it can be convenient to express them in pixels. Let p_u, p_v be the pixel dimensions, then $u_p = u/p_u$ and $v_p = v/p_v$. The relation becomes:

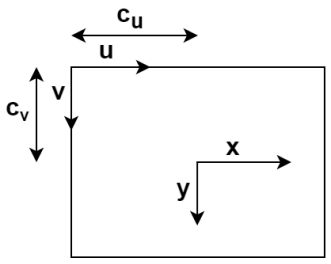


Figure 2.13: Image coordinates in pixel vs. sensor coordinates in meters.

$$
\begin{aligned}
Z_C \times \begin{pmatrix} u_p \\ v_p \\ 1 \end{pmatrix} &= \begin{pmatrix} f/p_u & 0 & c_u/p_u \\ 0 & f/p_v & c_v/p_v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix} \\
&= \begin{pmatrix} f_{pu} & 0 & c_{pu} \\ 0 & f_{pv} & c_{pv} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix}
\end{aligned}
\tag{2.13}
$$

3D pinhole model in world coordinate system

When several cameras are used in the same time, for instance for 3D reconstruction, it is important that they share a common frame of reference. Hence, the camera coordinate system needs to be expressed in the "world" system accordingly. Assuming that the camera is translated along a vector $T_{3\times1}$ and rotated according to a matrix $R_{3\times3}$, the coordinates in the camera frame of reference can be expressed in the world frame of reference as:

$$
\begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix} = R_{3\times3} \times \begin{pmatrix} X_W \\ Y_W \\ Z_W \end{pmatrix} + T_{3\times1} \tag{2.14}
$$

This formalism does not represent a linear system connecting the world and the camera coordinates, which can make calculations complicated. Instead, one can add one more dimension to the system and use the homogeneous coordinates, introduced by Möbius in 1927 [**?**], and which constitutes one of the basis of projective geometry.

$$
\begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix} = \begin{pmatrix} & & \\ & R_{3\times3} & T_{3\times1} \\ & & \end{pmatrix} \begin{pmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{pmatrix} \tag{2.15}
$$

Finally, the full pinhole model becomes:

$$Z_C \times \begin{pmatrix} u_p \\ v_p \\ 1 \end{pmatrix}_{3\times1} = \begin{pmatrix} f_{pu} & 0 & c_{pu} \\ 0 & f_{pv} & c_{pv} \\ 0 & 0 & 1 \end{pmatrix}_{3\times3} \begin{pmatrix} & \mathbf{R}_{3\times3} & & \mathbf{T}_{3\times1} \\ & & & \end{pmatrix}_{3\times4} \begin{pmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{pmatrix}_{4\times1} \tag{2.16}$$

Or more concisely, with $\overrightarrow{q_p}$ the 2D image coordinates, $\overrightarrow{Q_w}$ the 3D world coordinates, **K** the intrinsic matrix, **H** the homogeneous matrix of extrinsic parameters, and **P** the projection matrix:

$$\begin{aligned} Z_C \, \overrightarrow{q_p} &= \mathbf{K}\,\mathbf{H}\,\overrightarrow{Q_w} \\ &= \mathbf{P}\,\overrightarrow{Q_w} \end{aligned} \tag{2.17}$$

To sum it up, this system allows for a transformation between the coordinates of an object in meters in the world reference frame, and its coordinates in pixel on the image. It is a linear system, thus not computationally intensive to solve. The intrinsic matrix **K** is constant for a given camera, while the homogeneous (or extrinsic) matrix **H** depends on the camera position and orientation. The projection matrix **P** is the product of the intrinsic and extrinsic matrices. Z_C corresponds to the distance between the camera origin and its sensor, and is considered as an arbitrary scaling factor.

Skew factor and distortions

Some corrections can be brought to this model. First, the pixel sides may not be perfectly perpendicular. In this case, a skew parameter γ has to be added to the intrinsic matrix, and the relation becomes:

$$Z_O \times \begin{pmatrix} u_p \\ v_p \\ 1 \end{pmatrix} = \begin{pmatrix} f_{pu} & \gamma & c_{pu} \\ 0 & f_{pv} & c_{pv} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_O \\ Y_O \\ Z_O \end{pmatrix} \tag{2.18}$$

However, this is extremely rare in practice, and in the overwhelming majority of cases, γ can safely be set to zero [**?**].

Second, the use of a lens instead of a pinhole not only introduce a finite depth of field, but also distortions. These are mostly radial, caused by the curvature of the lens, especially if it has a wide angle. Radial distortions are particularly visible when using a wide angle lens, in the form of a "barrel" effect. Straight lines are then curved near the edges of the image (Figure **??**). Some tangential distortion can also be observed, in case the sensor is not perfectly perpendicular to the optical axis.

These can be corrected by adding radial and tangential terms to x and y, the image coordinates as regards to its center [**?**]:

$$\begin{cases} \begin{aligned} x' &= x + \underbrace{x(k_1r^2 + k_2r^4 + k_3r^6)}_{\Delta x_{radial}} + \underbrace{2p_1xy + p_2(r^2 + 2x^2)}_{\Delta x_{tangential}} \\ y' &= y + \underbrace{y(k_1r^2 + k_2r^4 + k_3r^6)}_{\Delta y_{radial}} + \underbrace{2p_1xy + p_2(r^2 + 2y^2)}_{\Delta y_{tangential}} \end{aligned} \end{cases} \tag{2.19}$$

Where k_1, k_2, k_3 are the radial distortion coefficients, and p_1 and p_2 the tangential distortion ones. $r^2 = x^2 + y^2$ is the distance from the center of the sensor. This is called the DR3DT2 model, also called the Brown-Conrady distortion model [**?**, **?**]. Note that it is possible to introduce more coefficients for an even more accurate model, and that in the case of an ultra wide-angle or fisheye lens, the Kannala-Brandt model [**?**] is mode appropriate.

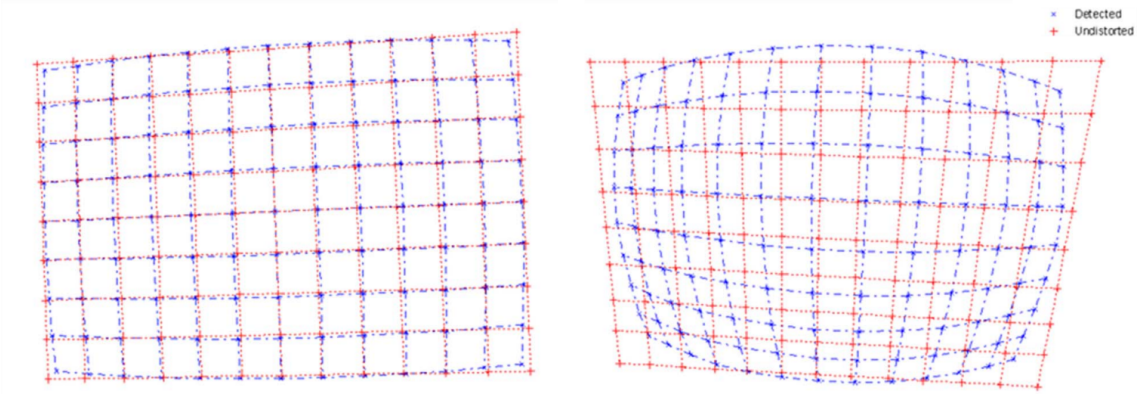


*Figure 2.14: Lens distortions are mostly radial (left image, with "barrel" effect), and sometimes tangential (right image). Image from [**?**].*

2.3.2 Calibration

The calibration problem

Camera calibration, also known as resectioning, consists in determining both intrinsic and extrinsic parameters of the camera. Extrinsic parameters are especially important to obtain for 3D reconstruction, which involves knowing the position of each camera. Intrinsic parameters can be determined at any time, however extrinsic parameters depend on the camera positions at the moment of capture, which involves that it is complicated to retrieve them afterwards – although we proposed a method to address this issue in **??**.

The intrinsic matrix is composed of 4 unknowns f_{pu}, f_{pv}, c_{pu}, and c_{pv}, while the extrinsic matrix has 3 parameters of translation, and 9 of rotation. However, the given representation of the rotation matrix is not minimal, and can be reduced to 3 parameters, for example with the Euler-Rodrigues formula [**?**]. Assuming no skew and no distortion, this amounts for a total of 10 unknowns. These can be determined by matching a number of points of known position in the world reference frame, to their corresponding coordinates in the image.

Direct Linear Transform (DLT) of calibration equations

Considering P_1, P_2, P_3 the rows of the projection matrix P, the relation between image points and their 3D coordinates (see Equation **??**) can be written as [**?**] :

$$
\begin{aligned}
Z_C \times \begin{pmatrix} u_p \\ v_p \\ 1 \end{pmatrix}_{3\times1} &= \begin{pmatrix} & \mathbf{P} & \end{pmatrix}_{3\times4} \begin{pmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{pmatrix}_{4\times1} \\
&= \begin{pmatrix} \overrightarrow{P_1} & \text{———} \\ \overrightarrow{P_2} & \text{———} \\ \overrightarrow{P_3} & \text{———} \end{pmatrix}_{3\times4} \begin{pmatrix} | \\ \overrightarrow{Q_W} \\ | \end{pmatrix}_{4\times1}
\end{aligned}
\tag{2.20}
$$

Or equivalently:

$$\begin{cases} Z_C\, u_p = \overrightarrow{P_1} \cdot \overrightarrow{Q_W} = \overrightarrow{Q_W^T} \cdot \overrightarrow{P_1^T} \\ Z_C\, v_p = \overrightarrow{P_2} \cdot \overrightarrow{Q_W} = \overrightarrow{Q_W^T} \cdot \overrightarrow{P_2^T} \\ Z_C \quad = \overrightarrow{P_3} \cdot \overrightarrow{Q_W} = \overrightarrow{Q_W^T} \cdot \overrightarrow{P_3^T} \end{cases} \tag{2.21}$$

We can reduce the system to two equations, and eliminate the scale factor Z_C:

$$\begin{cases} \overrightarrow{Q_W^T} \cdot \overrightarrow{P_1^T} - u_p\, \overrightarrow{Q_W^T} \cdot \overrightarrow{P_3^T} = 0 \\ \overrightarrow{Q_W^T} \cdot \overrightarrow{P_2^T} - v_p\, \overrightarrow{Q_W^T} \cdot \overrightarrow{P_3^T} = 0 \end{cases} \tag{2.22}$$

Which can be written as:

$$\begin{pmatrix} \overrightarrow{Q_W^T} & \overrightarrow{0^T} & -u_p\, \overrightarrow{Q_W^T} \\ \overrightarrow{0^T} & \overrightarrow{Q_W^T} & -v_p\, \overrightarrow{Q_W^T} \end{pmatrix}_{2\times 12} \begin{pmatrix} \overrightarrow{P_1^T} \\ \overrightarrow{P_2^T} \\ \overrightarrow{P_3^T} \end{pmatrix}_{12\times 1} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}_{2\times 1} \tag{2.23}$$

This operation, called Direct Linear Transform (DLT) [**?**], eliminates the Z_C scaling factor, and rewrites the system in a form that allows for a resolution with linear methods. The system is composed of two equations, for 10 unknowns. Hence, it can be solved analytically with 5 corresponding image and 3D points, assuming that they are perfectly measured. This is never the case, and an approximate numerical resolution with more points is preferred. A good rule of thumb is to measure 5 times as many points as needed, and thus to measure 25 points at least. This is classically done by using a checkerboard pattern.

Camera calibration from one single image of a checkerboard, with DLT method

Assuming that the checkerboard is used to set the origin of the world frame, the Z_W coordinates are set to 0, and X_W and Y_W coordinates of each corner can be inferred from the dimensions of the squares. As a result, 3D coordinates of corners are known. Their 2D coordinates on the image can also be automatically detected with a subpixel accuracy, for example with the OpenCV function findChessboardCorners() [**?**], which first detects edges [**?**], then straight lines, and then intersects the lines to obtain corner locations. Hence, 2D corners coordinates are also known. Considering M measures of matching 3D to image points, the system **??** has now $2M$ equations, for 10 unknowns still:

$$\begin{pmatrix} \overrightarrow{Q_{W1}^T} & \overrightarrow{0} & -u_{p1}\, \overrightarrow{Q_{W1}^T} \\ \overrightarrow{0} & \overrightarrow{Q_{W1}^T} & -v_{p1}\, \overrightarrow{Q_{W1}^T} \\ \vdots & \vdots & \vdots \\ \overrightarrow{Q_{WM}^T} & \overrightarrow{0} & -u_{pN}\, \overrightarrow{Q_{WM}^T} \\ \overrightarrow{0} & \overrightarrow{Q_{WM}^T} & -v_{pN}\, \overrightarrow{Q_{WM}^T} \end{pmatrix}_{2M\times 12} \begin{pmatrix} \overrightarrow{P_1^T} \\ \overrightarrow{P_2^T} \\ \overrightarrow{P_3^T} \end{pmatrix}_{12\times 1} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}_{2M\times 1} \tag{2.24}$$

This system is now overdetermined. As it can be written in the form $\mathbf{A}\overrightarrow{X} = \overrightarrow{0}$, a linear-eigen pseudo-solution can be found [**?**] by using Singular Value Decomposition (SVD) [**?**] (See Algorithm **??**). In order to fully determine $\mathbf{P}$, it first needs to be normalized, then the least-square solution needs to be found, and then it can be denormalized.

Algorithm 2 Linear-eigen pseudo-solution of $\mathbf{A}\overrightarrow{X} = \overrightarrow{0}$

Let $\mathbf{A}$ be a rectangular matrix of size $2M \times N$, composed of real or complex coefficients, and $\overrightarrow{X}$ be a vector of size N. The objective is to solve

$$\mathbf{A}\overrightarrow{X} = \overrightarrow{0} \tag{2.25}$$

If $2M > N$, then the system is overdetermined, but a least-square pseudo-solution can be found.

1: $\mathbf{A}$ can be factorized by Singular Value Decomposition (SVD):

$$A = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^T \tag{2.26}$$

with $\mathbf{U}$ an orthonormal basis of size $2M \times 2M$, $\mathbf{S}$ the rectangular diagonal matrix of $\mathbf{A}$ of size $2M \times N$, filled with its singular values $\sigma_1, \ldots, \sigma_{2M}$, and $\mathbf{V}$ an orthonormal basis of size $N \times N$. $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{S}$ can be efficiently computed by the Python function numpy.linalg.svd().

2: $\overrightarrow{X}$ can be expressed as a linear combination of basis vectors:

$$\overrightarrow{X} = \mathbf{V}\overrightarrow{\alpha}, \tag{2.27}$$

with $\overrightarrow{\alpha}$ an undetermined vector of size N.

3: Minimizing $(\mathbf{A}\overrightarrow{X})^2$ also minimizes $\mathbf{A}\overrightarrow{X}$.

$$\begin{aligned}(\mathbf{A}\overrightarrow{X})^2 &= (\mathbf{A}\overrightarrow{X})^T(\mathbf{A}\overrightarrow{X}) \\ &= (\overrightarrow{\alpha}^T\mathbf{V}^T\,\mathbf{V}\mathbf{S}\mathbf{U}^T)(\mathbf{U}\mathbf{S}\mathbf{V}^T\,\mathbf{V}\overrightarrow{\alpha}) \\ &= \overrightarrow{\alpha}^T\mathbf{S}\overrightarrow{\alpha} \\ &= \sum_{i\in[1,N]} \alpha_i^2\sigma_i^2\end{aligned} \tag{2.28}$$

which is minimum when all α factors are set to zero, except for the factor of the smallest singular value σ.

4: Assuming that $\sigma_{min} = \sigma_N$, then $(\mathbf{A}\overrightarrow{X})^2$ is minimum when all α_i are null, except for α_N.

$$(\mathbf{A}\overrightarrow{X})^2_{min} = \alpha_N\sigma_N \tag{2.29}$$

and from equation **??**:

$$\overrightarrow{X} = \mathbf{V}\overrightarrow{\alpha} = \begin{pmatrix} V_{11} & \ldots & V_{1N} \\ \vdots & & \vdots \\ V_{N1} & \ldots & V_{NN} \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \alpha_N \end{pmatrix} = \alpha_N \begin{pmatrix} V_{1N} \\ \vdots \\ V_{NN} \end{pmatrix} \tag{2.30}$$

5: Hence, $\overrightarrow{X}$ is solved up to a scale factor α_N. The full system can be determined by imposing one arbitrary constraint, for example $||\overrightarrow{X}|| = 1$.

Once the projection matrix **P** has been found, it needs to be decomposed into intrinsic and extrinsic parameters, in the form:

$$\begin{aligned} \mathbf{P} &= \mathbf{K}\ \mathbf{H} \\ &= \mathbf{K}\ (\mathbf{R}\,|\,\mathbf{T}) \\ &= \left(\begin{array}{c|c} \mathbf{K}\,\mathbf{R}_{3\times3} & \mathbf{K}\mathbf{T}_{3\times1} \end{array} \right) \end{aligned} \tag{2.31}$$

K is an upper-triangular matrix, and **R** is orthogonal by virtue of being a rotation matrix ($\mathbf{R}^T = \mathbf{R}^{-1}$). Hence, **K** and **R** can be found with an RQ-decomposition (derived from QR-decomposition) from the first 3×3 block of **P**. This is done with Givens rotations in OpenCV, and will not be detailed here: see [**?**, **?**] for more details. Then, finding $\overrightarrow{T}$ from the last column of **P** is trivial. This decomposition can be done in OpenCV with the decomposeProjectionMatrix(). However, one needs to bear in mind that this decomposition is not unique. Forcing f_{pu} and f_{pv} to be positive solves this issue, providing that the camera and image axes point in the same direction.

Camera calibration from several images of a checkerboard, with Zhang method

In practice, a more accurate method would use more than one single image of a checkerboard. It would also estimate distortion coefficients. A classic approach is the one proposed by [**?**]. It can be separated into 2 steps: first, initializing calibration parameters with a DLT method similar to the above-mentioned. Second, refining intrinsic, extrinsic, and distortion parameters by using several images of the checkerboard taken from various view points. This is treated as a nonlinear optimization problem, solved by the Levenberg-Marquardt algorithm [**?**, **?**]. The function that needs to be minimized is the reprojection error for all M points on each N checkerboard images, defined as such:

$$\sum_{i\in[1,N]} \sum_{j\in[1,M]} || \overrightarrow{q_{ij}} - \overrightarrow{\widehat{q_{ij}}(\mathbf{K}, k_1, k_2, \overrightarrow{R_i}, \overrightarrow{T_i}, \overrightarrow{Q_j})} ||^2 \tag{2.32}$$

The reprojection error is the Euclidian distance between the detected 2D point $\overrightarrow{q_{ij}}$, and $\overrightarrow{\widehat{q_{ij}}}$, the estimated projection of the 3D point $\overrightarrow{Q_j}$. Projecting 3D points on the 2D plane will be addressed in the next section on **??**. In any case, $\overrightarrow{\widehat{q_{ij}}}$ is dependent on fixed intrinsic parameters (**K** matrix, distortion coefficients such as k_1 and k_2, as well as others if needed), on extrinsic parameters depending on the position of the camera when taking each checkerboard image (here, the rotation is expressed as a Rodrigues vector of size 3 [**?**]), and on each 3D checkerboard point. This can be done in OpenCV with the function calibrateCamera(), which takes matching 3D and 2D points as input, and returns extrinsic parameters for each image view, as well as intrinsic parameters and distortion coefficients.

Calibration of all cameras together with a wand, with Sparse Bundle Adjustment (SBA)

In biomechanics, numerous cameras are usually needed, in order to obtain accurate 3D reconstruction in spite of occlusions. Hence, cameras are often placed around the subject. This can be problematic for the determination of a common frame of reference with a checkerboard, since it needs to be observed by all cameras simultaneously. When laying flat in the center, the checkerboard is positioned at a long distance from the camera, and oriented at a challenging angle. This makes it hard for it to be detected accurately, which is consequently susceptible to deteriorate extrinsic calibration.

On the other hand, spherical markers placed on a wand are more likely to be detected by all cameras simultaneously, regardless of their position and orientation in space. However, they only represent either a single point, or a 1D length if they are placed at both extremities of a wand.

A planar object like a checkerboard, on the contrary, is sufficient to determine a 3D coordinate system. This implies that using a wand does not allow for direct 2D image point to 3D coordinate correspondences. Hence, 3D positions also need to be optimized, which involves calibrating all cameras together instead of one by one. This has a few implications: first, cameras need to be synchronized, and then, more parameters have to be optimized, which tends to make this method less stable. It should also make the optimization extremely computationally intensive. Luckily, one can take advantage of the sparsity of the Jacobian of the objective function, which allows for considerable computation savings. This method, called Sparse Bundle Adjustment (SBA), will not be described in details, but extensive explanations can be found in [**?**].

In short, this calibration problem is solved by using a sparse variant of the Levenberg-Marquardt algorithm. The estimated $\overrightarrow{\widehat{q_{kj}}(\overrightarrow{a_k}, \overrightarrow{b_j})}$ point is dependent on each 3D point parametrized by $\overrightarrow{b_j}$, and on each cameras image parametrized by $\overrightarrow{a_k}$. The main difference with the function used in the checkerboard approach, is that this one is minimized over all M points of all K images from all cameras, instead of for each camera independently. In order to save resources, the projection is only calculated if the point j is actually visible in camera image k.

$$\sum_{k\in[1,K]}\sum_{j\in[1,M]} v_{kj} \,||\, \overrightarrow{q_{kj}} - \overrightarrow{\widehat{q_{kj}}(\overrightarrow{a_k}, \overrightarrow{b_j})} \,||^2, \text{ with } v_{kj} = \begin{cases} 1 \text{ if point } j \text{ detected by camera image} k \\ 0 \text{ if unseen} \end{cases} \tag{2.33}$$

In practice

Both checkerboard Zhang and wand SBA methods show similar residual errors, of the order of a millimeter in a roughly $4 \times 2 \times 1.5\ m^3$ capture volume [**?**,**?**] (see Tables **??**). This is better than the classic DLT approach. Nonetheless, the wand approach does not estimate distortion coefficients as accurately as the checkerboard method. If a fisheye or a wide lens is used, the checkerboard method should be favored.

Unlike a checkerboard, wand points are visible from all angles. Moreover, the wand method has the advantage of estimating the relative position of all cameras together, and then setting the global coordinate system with another object. This represents an additional step, and it implies that cameras are synchronized, although it also means that the reference object does not need to be visible by all cameras. Conversely, with the checkerboard approach, all cameras are calibrated independently: if the checkerboard representing the global origin cannot be detected in a certain viewpoint, the corresponding camera cannot be used for 3D reconstruction.

Nevertheless, the wand approach also optimizes over more parameters, which makes it less stable and involves a careful initialization, e.g., with an additional object of known position and dimensions, or simply by reducing the number of parameters, e.g., by fixing intrinsic parameters, which would have to be calibrated priorly. Moreover, while a checkerboard has a clearly recognizable pattern, two points of a wand are more likely to be subject to false positives errors.

In practice, a wand is much less cumbersome and much easier to transport than a checkerboard, especially for sports motion analysis where capture volumes can be large, and the checkerboard size needs to be scaled accordingly. In fact, a wand is not even necessary, and it can be replaced with the automatic detection of body segments by a 2D pose estimation model such as OpenPose [**?**, **?**, **?**]. This is, however, less accurate. The wand method is widely used by all commercial software solutions, such as Qualisys [**?**], as well as for free GUI such as EasyWand [**?**] in Matlab, or its Python version, Argus [**?**, **?**]. These are free, but not open-source. However, SBA is not implemented in OpenCV, and there is actually no widely supported Python version of the algorithm. Lastly, autocalibration procedures are being developed, which usually consist in automatically tracking features of interest in images, matching them across camera views, and then estimating intrinsic and extrinsic parameters in a method similar to SBA [**?**, **?**]. This approach is promising, but it is still in its infancy, and will not be described.

Issue	Checkerboard calibration [?]	Wand calibration (SBA) [?]
Accuracy	≈ 1 mm	≈ 1 mm, unless large distorsions (e.g., fisheye)
Transportation & manipulation	Large & cumbersome	Lightweight & foldable
Availability	Open-source (e.g., OpenCV)	Mostly commercial (e.g., Qualisys), or freeware but not open-source (e.g., Argus)
Detection	Recognizable pattern but sensitive to orientation	Sensitive to false positives but not to orientation
GCS estimation	All cameras need to detect the reference object (which is sensitive to orientation)	Not all cameras need to detect it, since camera positions known relative to each other
Optimization robustness	Robust	Need for careful initialization or reducing on parameters number
Computational cost	A few seconds	A few seconds

Table 2.1: Comparison of the main two approaches for accurate camera calibration: with a checkerboard [?], or with a wand [?]. Not that some hybrid approaches exist, e.g., those which use a checkerboard for intrinsic calibration, and a wand for extrinsic calibration. SBA: Sparse Bundle Adjustment. GCS: Global Coordinate System.

Other options, explored in **??**, would be to compute intrinsic and extrinsic parameters independently. Intrinsic parameters and distortion coefficients can be calculated an any convenient moment, since they are in theory constant for a given camera. This can be done with a checkerboard, either with OpenCV raw coding, or by using a GUI like Argus [**?**]. On the other hand, extrinsic parameters depend on the camera positions: they need to be computed upon every new capture setting. One way to do this consists in performing an extrinsic calibration on the spot with a wand, with fixed intrinsic parameters. Another option, if wand calibration is not possible, is to use a method analogous to Zhang's, with any 3D object of known dimensions, large enough to cover a substantial field of view for each camera. This would leverage a Perspective-n-Point approach, which consists in a Levenberg-Marquardt optimization similar to Equation **??**, but with fixed intrinsic parameters [**?**]. This can be done with the solvePnP OpenCV function.

2.3.3 Triangulation

After 2D points are detected on all cameras, and considering that camera positions, orientations, and intrinsic properties are known, points can be reconstructed in the 3D space. This procedure is called triangulation, although it is a slightly abusive use of the term in case more than two cameras are used. In theory, the 3D point should lay at the intersection of all lines going from the camera centers to the 2D points (referred to as epipolar lines). In practice, the 2D points are not detected with perfect accuracy, and lines do not intersect (see Figure **??**). The 3D point is then approximated as the point that minimizes the distance between all epipolar lines.

Triangulation with a linear-eigen approach

The most straightforward, and computationally efficient method, is a DLT approach, sometimes called homogeneous or linear-eigen [**?**]. The procedure is slightly different from the one previously detailed for calibration. Indeed, since 3D coordinates are now the unknown variables, they are the ones factorized rather than the projection parameters. From Equation **??** applied to a

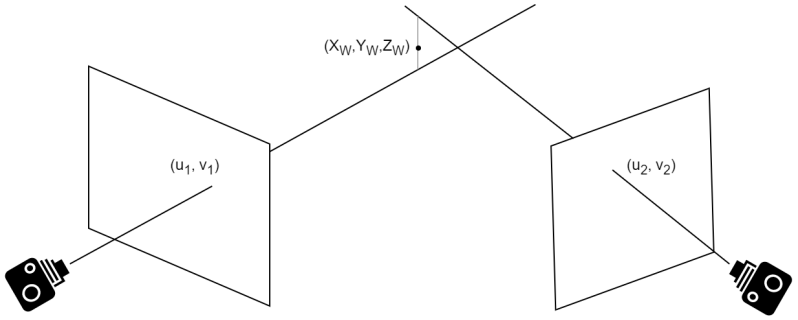


Figure 2.15: The 3D reconstructed point can be approximated as the minimal distance between epipolar lines, which pass by each camera center and 2D detected points.

point $\overrightarrow{Q_W}$ projected on one camera, we have:

$$\begin{cases} Z_C\, u_p = \overrightarrow{P_1} \cdot \overrightarrow{Q_W} \\ Z_C\, v_p = \overrightarrow{P_2} \cdot \overrightarrow{Q_W} \\ Z_C \quad = \overrightarrow{P_3} \cdot \overrightarrow{Q_W} \end{cases} \tag{2.34}$$

Which can be reduced as a system of two equations:

$$\begin{cases} (\overrightarrow{P_1} - u_p\, \overrightarrow{P_3}) \cdot \overrightarrow{Q_W} = 0 \\ (\overrightarrow{P_2} - v_p\, \overrightarrow{P_3}) \cdot \overrightarrow{Q_W} = 0 \end{cases} \tag{2.35}$$

Or:

$$\begin{pmatrix} \overrightarrow{P_1} - u_p\, \overrightarrow{P_3} \\ \overrightarrow{P_2} - v_p\, \overrightarrow{P_3} \end{pmatrix}_{2\times 4} \begin{pmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{pmatrix}_{4\times 1} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}_{2\times 1} \tag{2.36}$$

With C cameras, we obtain the following system of 2C equations:

$$\begin{pmatrix} \overrightarrow{P_1^1} - u_p^1\, \overrightarrow{P_3^1} \\ \overrightarrow{P_2^1} - v_p^1\, \overrightarrow{P_3^1} \\ \vdots \\ \overrightarrow{P_1^C} - u_p^C\, \overrightarrow{P_3^C} \\ \overrightarrow{P_2^C} - v_p^C\, \overrightarrow{P_3^C} \end{pmatrix}_{2C\times 4} \begin{pmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{pmatrix}_{4\times 1} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}_{2C\times 1} \tag{2.37}$$

This can be written in the form $\mathbf{A}\overrightarrow{X} = \overrightarrow{0}$, and thus a least-square solution can be found (See Algorithm **??**). Let $\mathbf{V}$ be the orthonormal basis of size 4×4 obtained by SVD of $\mathbf{A}$, the algorithm gives:

$$\begin{pmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{pmatrix}_{4\times 1} = \alpha_4 \begin{pmatrix} V_{14} \\ V_{24} \\ V_{34} \\ V_{44} \end{pmatrix} \tag{2.38}$$

As a consequence, the 3D coordinates of the triangulated point are:

$$\boxed{\overrightarrow{Q_W} = \begin{pmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{pmatrix} = \begin{pmatrix} V_{14}/V_{44} \\ V_{24}/V_{44} \\ V_{34}/V_{44} \\ 1 \end{pmatrix}} \tag{2.39}$$

All points can be triangulated in a similar fashion, frame after frame. Since the procedure entirely relies on linear algebra, it is very fast. However, it is not robust to outliers. Other methods have been proposed, such as the Iteratively Reweighted MidPoint method (IRMP) which looks the smaller distance between all epipolar lines [**?**], the L_2 method which minimizes the sum of the squared reprojection errors [**?**, **?**], or the L_∞ method which minimizes the maximum reprojection error [**?**], or the Q-sweep method which minimizes the median of the reprojection error, and thus is more robust to outliers [**?**]. Among all of these, the linear-eigen is the fastest by at least an order of magnitude, but the IRMP method may be an other good compromise in terms of speed and accuracy [**?**]. Some approaches also use a fast linear-eigen method, which is then refined with a (much) slower learning approach [**?**].

Note that these approaches do not take distortions into account, and assume either that distortions are negligible, or that videos have been priorly undistorted based on the coefficients found during intrinsic calibration [**?**]. Alternatively, some action cameras such as the GoPro 5+ offer a linear mode, which undistorts videos on the fly upon capture.

Moreover, cameras not only need to share a 3D global reference frame, but also a common time reference. Otherwise, the same frame risks relating to a different instant in the athlete's motion, in which case the 3D reconstruction would not make sense. This is classically done with a wired hardware trigger, but other approaches use a flash, a sound, or a wireless signal such as Wi-Fi, Bluetooth or GPS [**?**]. In **??**, we proposed another approach based on cross-correlation of 2D feature speeds.

Person matching

In case of multi-person detection, an additional step needs to be undertaken prior to triangulation: the 2D detected points of each person need to be matched across all views. A common approach consists in choosing one or two keypoint (e.g., the neck and the hip), and clustering together the persons for whom epipolar lines cross at a small enough distance from each other [**?**, **?**, **?**]. This works well and efficiently, unless an epipolar line passes through several people, in the case of a crowded scene for example. In this case, it is possible to use more keypoints (or different keypoints) to solve ambiguities. Other methods use a combinatory approach, slower but more robust, which tries out all possible combinations, and minimizes the reprojection error [**?**, **?**, **?**]. Other procedures use a spatio-temporal neural network, which take advantage of the information gathered in previous frames instead of working frame-by-frame [**?**]. Lastly, when multiple persons have been triangulated, they need to be tracked in time in order to avoid them swapping from one frame to another. This can be done by setting a threshold on 3D keypoint speed, below which a person is deemed to be correctly associated with the previous frame [**?**].

Reprojection error

The reprojection error is the distance between the projection $(\widehat{u_p}, \widehat{v_p})$ of a triangulated keypoint Q_W on an image, and its actual estimated 2D coordinates (u_p, v_p). It can be used to jointly evaluate the accuracy of the 2D pose estimation and of the 3D reconstruction. From Equation **??**, we have:

$$\begin{cases} \overrightarrow{P_1} \cdot \overrightarrow{Q_W} = \widehat{u_p}\ \overrightarrow{P_3} \cdot \overrightarrow{Q_W} \\ \overrightarrow{P_2} \cdot \overrightarrow{Q_W} = \widehat{v_p}\ \overrightarrow{P_3} \cdot \overrightarrow{Q_W} \end{cases} \tag{2.40}$$

Which leads to the coordinates of the reprojected point:

$$\begin{cases} \widehat{u_p} = \frac{\overrightarrow{P_1} \cdot \overrightarrow{Q_W}}{\overrightarrow{P_3} \cdot \overrightarrow{Q_W}} \\ \widehat{v_p} = \frac{\overrightarrow{P_2} \cdot \overrightarrow{Q_W}}{\overrightarrow{P_3} \cdot \overrightarrow{Q_W}} \end{cases} \tag{2.41}$$

And then, with $\overrightarrow{\widehat{q_p}} = (\widehat{u_p}, \widehat{v_p})$ and $\overrightarrow{q_p} = (u_p, v_p)$, the mean reprojection error of this point for all C cameras can be calculated:

$$\boxed{err = \frac{1}{C} \sum_{c \in [0,C]} ||\overrightarrow{\widehat{q_p}} - \overrightarrow{q_p}||^2} \tag{2.42}$$

Again, this formulation does not take distortions into account, so it should be taken with caution if wide lenses are used. Note that this formula is also used to calculate residual errors from calibration. Instead of reprojecting triangulated keypoints, one reprojects the 3D calibration points detected from a checkerboard or from a wand.

2.4 3D joint kinematics

2.4.1 The kinematics problem

Rough 3D joint coordinates alone are not enough to provide an insightful understanding of human motion. Unless some assumptions are made, one single keypoint per joint can at best describe planar flexion/extension angles. This conceals any abduction/adduction or internal/external rotation angles.

In order for its position and orientation to be fully determined, a segment needs to be equipped with at least 3 non-colinear markers. Additionally, a simple full human body includes at least 14 segments. Consequently, at least $14 \times 3 = 42$ markers should be required for full-body kinematics. And yet, OpenPose and other pose estimation models only provide 25 keypoints, which are even reduced to 21 if eyes and ears are excluded. This represents half of the minimum required. For this reason, it is important to find indirect ways to obtain 3D joint angles, from seemingly incomplete information.

2.4.2 The 6DoF approach

The traditional method for obtaining 3D joint angles consists in computing positions and orientations of all body segments independently: it is called the 6DoF (6 Degrees of Freedom) free-body approach. This assumes that there are at least as many known variables (marker positions) as unknown degrees of freedom (joint angles). This excludes most of the current pose estimation models from operating in this paradigm. However, as the biomechanics community make theirs the concepts of learning-based keypoint estimation, other models with more complete labelling may arise. Hence, for the sake of completeness, this approach is broached below.

If only 3 markers are used per segment, the problem can be solved analytically. The method is then called Direct Kinematics (DK, see Confusing concepts - Disambiguation) [**?**]. The local reference frame of each segment, i.e., its joint center and rotational axes, can be fully defined as regards to the locations of the markers. Simple trigonometric formulas can then be applied to infer inter-segmental rotations and translations.

In order for results to be consistent across subjects and across research studies, markers need to be carefully positioned and to follow standards. Such standards are given by the Vicon Plug-in Gait markerset for gait analysis [**?**], or by its successor Clinical Gait Module 2 (CGM2), which has been implemented in Python as pyCGM2 [**?**]. For more general purposes, the International Society of Biomechanics (ISB) proposes a standard for both lower and upper-body analysis [**?**, **?**] (see Figure **??**). Among other solutions, KineticsToolkit provides open-source Python tools to carry out analysis according to the ISB standards [**?**].

If more than 3 markers are placed on the segment, the problem needs to be solved numerically. These methods are then called Single-body Kinematic Optimization (SKO), or Segmental Optimization (SO) [**?**]. In practice, such a redundancy is advised, as it makes the analysis more robust to Skin Tissue Artifacts (STA), to measurement inaccuracies, and to potential marker loss. Once the local reference frame has been determined with one of the previously cited standards, minimizing the following function leads to a more robust approximation of the rotation and translation of the considered segment:

$$\sum_{m \in [1,M]} || \mathbf{R} \overrightarrow{X^m_{init}} + \overrightarrow{T} - \overrightarrow{X^m_{target}} ||^2 \tag{2.43}$$

With M the number of markers of the considered segment. The segment best matches its target when all of its initial marker coordinates $\overrightarrow{X^m_{init}}$, rotated by a matrix $\mathbf{R}$ and translated of a vector $\overrightarrow{T}$, are close to their measured coordinates $\overrightarrow{X^m_{target}}$.

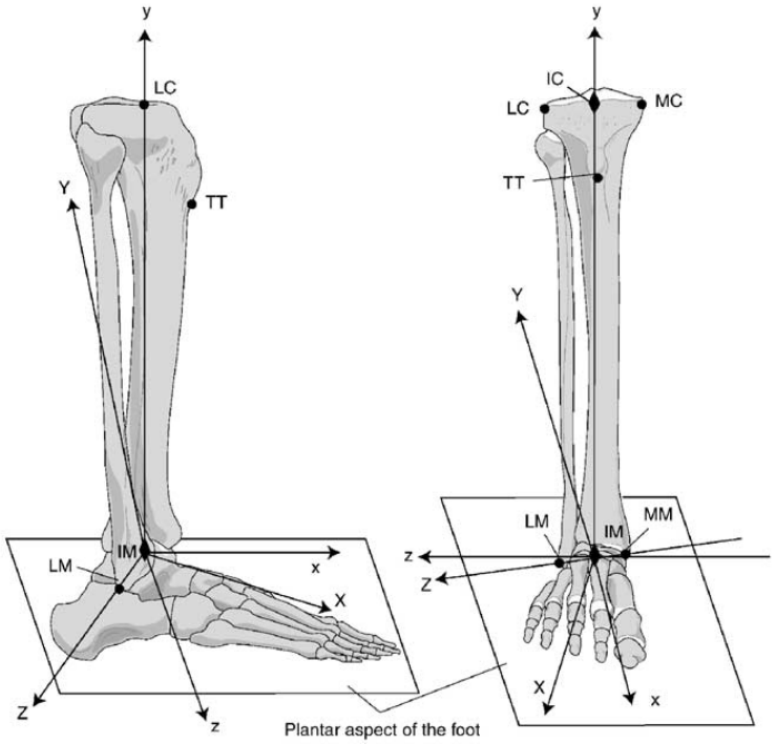


Figure 2.16: A few carefully placed markers are enough to determine the tibia/fibula coordinate system (XYZ) and the calcaneus coordinate system (xyz). Here, the ankle joint complex in the neutral position [?].

2.4.3 The inverse kinematics approach

Constrained system of rigid bodies / kinematic chain

Multibody Kinematic Optimization (MKO) or Global Optimization (GO) 2 méthodes: algorithme de Levenberg-Marquardt avec calcul du Jacobien pseudo-inverse, et optimisation sequentielle quadratique

inverse:numerical=iterative/model based/global optimization/multibody kinematic optimization -> trust model

theta = f-1(x)

Modele: joints with dof, scaling for segment lengths

Principles initially for robotics (simple), 3D animation applications (visually plausible rather than accurate) until models became more thorough and trustworthy

Inverse Jacobian and cost function

Model definition and scaling

- Kinematic chain (joint types, ranges of motion) - segment data (length, inertial properties) - constraints - forces / muscles / contacts / controllers - markers

Model-based (good enough? [**?**]. Neuromuscular, inertial properties, aspect not addressed here, as we are mostly interested in the kinematics at that stage). Grown rapidly since the beginning of the 2000s [**?**].

Model based: - model, kinematic chain: rigid multibody skeletal structure, with joint and segment definitions. Potentially with inertial properties, as well as neural and/or muscular system. - scaling - IK IK vs forward kinematics (joint angles -> segment orientation)

Theory: weighted least square/convex/non-linear optimization by OpenSim (vs. Jacobian inverse/Levenberg-Marquardt, vs. heuristic methods, same overall problem as in calibration, equations 2.32 and 2.33: least-square optimization: more unknown parameters in function q than

measured (known) variables. Non-linear because q is not a linear function of its parameters. Thus, classically solved by Levenberg-Marquardt algorithm. Hybrid between gradient descent and Gauss-Newton: converges faster than the first, and is more robust to bad initial estimates than the second. All Use Jacobian)

Global optimization / multibody kinematic optimization, reduces sensitivity to soft tissue artifact, reduces number of degrees of freedom to solve, and thus of markers needed (1 per joint if fully constrained [**?**], vs. 3 min for 6DoF, no need for interpolation or redo capture if marker falls) -> Fohanno2014?, more potential information on muscle activation for example [**?**, **?**] Slightly slower, assumes that all human beings have similar joint mechanics, based on *a priori* constraints inferred from observations of a population, not guaranteed to work on pathological population nor on unusual body structures (joint laxity cannot be qualified for example)

Résultats similaires si STA limités et modèle approprié [**?**] Aristidou2017: - analytical: if low DoF, not lots of STA, if looking at pathological individuals (missing clinically relevant movements), needs to be fast - numerical: more general, if less markers. Jacobian: if biomechanical laws and weighted constraints; heuristic: needs to be fast; data driven:

slower than direct kinematics

close loop?

As opposed to forward kinematics

Compare with 2D angles between 3 points

Different methods (model based vs autres) for angles (cf mail starred)

Model constrained, based on mechanical joint modeling, and on constraints inferred from optimize the posture of a physically consistent skeleton, scaled to each individual subject. In particular, this allows for obtaining 3D joint angles at each point in time, commonly referred to as inverse kinematics (IK.)

See OpenSim confluence for ID, Stat optim, RRA, CMC.. kinematics: No account taken of physical laws that cause motion such as gravity, collisions, or more generally inertial properties and forces: only geometric constraints

6DoF -> trust markers (assumes that markers are positioned very accurately, by one single trained operator, no STA.) -> no need for assumptions. Free bodies ->renders dislocations (good if there is some laxity, bad if none-> interpenetration, etc) Comes from biomechanics rather than robotics (simple joints)

kineticstoolkit (6DoF ISB), pyCGM2 (6DoF gait), pyomeca (opensim, biorbd), custom (jacobien et optim sqp), pinocchio (Jacobian), OpenSim (sqp?) Vicon, QTM (Visual3D)

2.4.4 Other approaches

Learning based, ...

3

Proposed solution: Pose2Sim Python package

We propose the Pose2Sim python package, as an alternative to the more usual marker-based motion capture methods. Pose2Sim stands for "OpenPose to OpenSim", as it uses OpenPose inputs (2D keypoints coordinates obtained from multiple videos) and leads to an OpenSim result (physically consistent full-body 3D joint angles). Code is available at https://github.com/perfanalytics/pose2sim.

This chapter is adapted from the article published in the Journal of Open Source Software: "Pose2Sim: An Open-source Python Package for multiview markerless kinematics" [?]. See Figure **??** *for a visual abstract.*

Contents

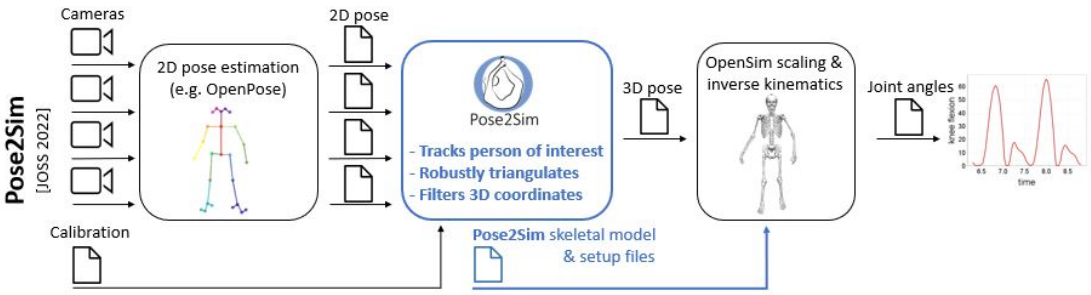


Figure 3.1: Visual abstract for the Pose2Sim workflow [?].

3.1 Introduction to the workflow

After having inspected the state of the art, and according to the **??**, it was apparent that there was both a need for better motion analysis for sports, and opportunities brought to existence by the recent advances in machine learning and computer vision (Figure **??**). Markerless pose estimation had become increasingly fast and accurate, while being non-invasive, and robust to clothing and background conditions. However, it did not yet offer a satisfying level of accuracy for biomechanics usage, and was not accessible to non computer-science experts.

As a consequence, we proposed a package written in Python, which would be entirely open-source, easy to install and use, and offer a fine level of control to the user. This package would take advantage of two of the most recognized and widespread tools in their respective fields: OpenPose and OpenSim. It would involve several cameras, without any hardware restrictions, although they would have to be synchronized and calibrated. This way, fewer assumptions would have to be made for 3D reconstruction of the 2D pose estimations. It would constrain 3D coordinates to an individually scaled, biomechanically coherent 3D model, in order to provide reliable and usable kinematic data in a sports context. There would not be any Graphical User Interface (GUI), at least in the first version.

Our package, Pose2Sim [**?**], is an alternative to the more usual marker-based motion capture methods. Pose2Sim stands for "OpenPose to OpenSim", as it uses OpenPose inputs (2D coordinates obtained from multiple videos) [**?**] and leads to an OpenSim result (full-body 3D joint angles) [**?**, **?**]. Pose2Sim is accessible at https://github.com/perfanalytics/pose2sim.

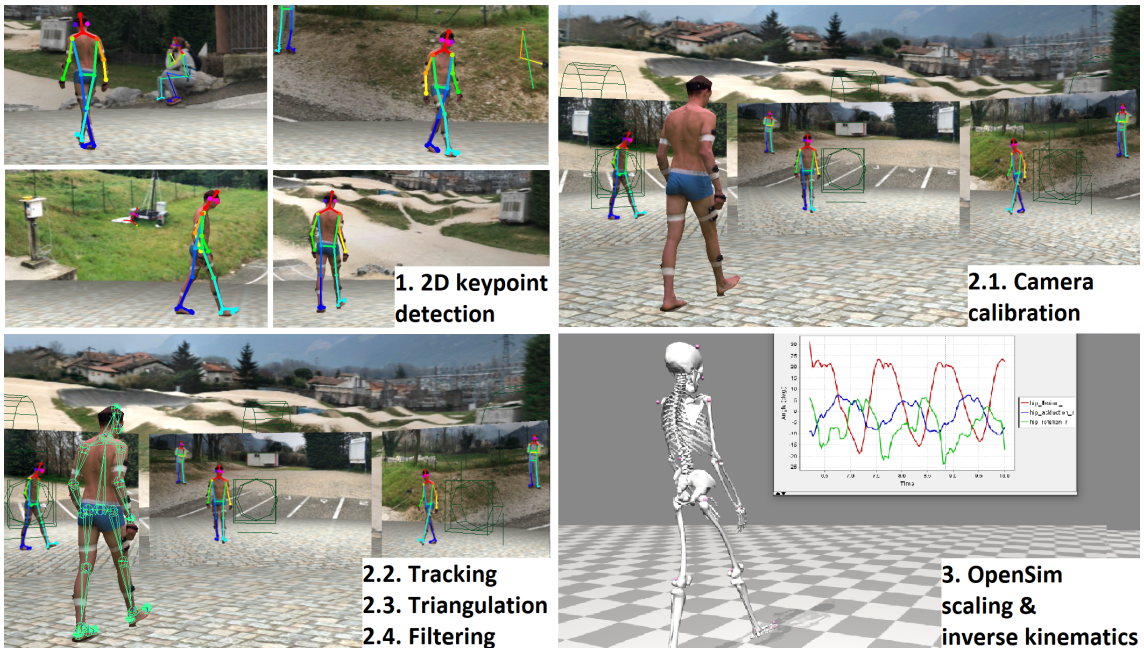


Figure 3.2: Pose2Sim full pipeline: (1) 2D keypoint detection; (2.1) Camera calibration; (2.1-2.4) Tracking of the person of interest, Triangulating of keypoint coordinates; and Filtering; (3) Constraining the 3D coordinates to an individually scaled, physically consistent OpenSim skeletal model.

The repository presents a framework which consists in (Figures **??**):

1. Preliminary 2D joint coordinate detections from multiple videos, e.g. with OpenPose.
2. Pose2Sim core, including 4 customizable steps:
   2.1. Camera calibration.
   2.2. 2D tracking of the person of interest.
   2.3. 3D keypoint triangulation.
   2.4. 3D coordinate filtering.
3. Scaling a full-body skeleton to each individual subject, and computing inverse kinematics via OpenSim so as to obtain 3D joint angles.

Each task is easily customizable, and requires only moderate Python skills. The whole workflow runs from any video cameras, on any computer, equipped with any operating system (although OpenSim has to be compiled from source on Linux). It requires no marker placement, and the scaling and inverse kinematic steps are simpler than they are with markers-based methods. Overall, human intervention is scarce, which makes it more robust to human error. Pose2Sim has already been used and tested in a number of situations (walking, running, cycling, dancing, balancing, swimming, boxing), and published in peer-reviewed scientific publications assessing the quality of its code [**?**], its robustness (see Chapter 4 on **??**) [**?**] and its accuracy (see Chapter 5 on **??**) [**?**]. Its results for inverse kinematics were deemed good when compared to marker-based ones, with errors generally below 4.0° across several activities, on both lower and on upper limbs. The combination of its ease of use, customizable parameters, and high robustness and accuracy makes it promising, especially for "in-the-wild" sports movement analysis.

3.2 Method details

3.2.1 Project

Pose2Sim is meant to be as fully and easily configurable as possible, by editing the `User/Config.toml` file. First of all, the user can specify the project path and folder names, the video frame rate, and the range of analyzed frames.

Optional tools are also provided for extending its usage (Figures **??**). Among others, DeepLabCut 2D files can be converted to the OpenPose formalism, and calibration files from other platforms can also be converted to the AniPose [**?**] formalism we use. 2D keypoint files can be displayed and stored as a video. Gait events can be detected from kinematic data thanks to [**?**] algorithm. Some other scripts allow for further processing of 3D coordinates. More practical information can be found on the GitHub repository.

3.2.2 2D keypoint detection

We recommend using the OpenPose body_25B model, as it provides foot keypoints, is as fast as the standard body_25 one, and as we have extensively tested it [**?**]. However, it requires manual installation [**?**]. Note that only 21 of the 25 detected keypoints are tracked, since eye and ear keypoints would be redundant in the determination of the head orientation.

This being said, the user can choose to use any deep-learning pose estimation network. This choice will affect how keypoint indices will be mapped to model markers in OpenSim, corresponding to anatomical landmarks or joint centers. The OpenPose body_25, body_25B, body_135, COCO, and MPII models are fully supported. The AlphaPose COCO, COCO-WholeBody, and full-body HALPE models are also supported, as well as the full-body but single-person detection BlazePose model. COCO and MPII model are the ones generally used by other networks such as OpenPifPaf [**?**], YOLO-pose [**?**, **?**], and others, which means that they are also supported. It is also

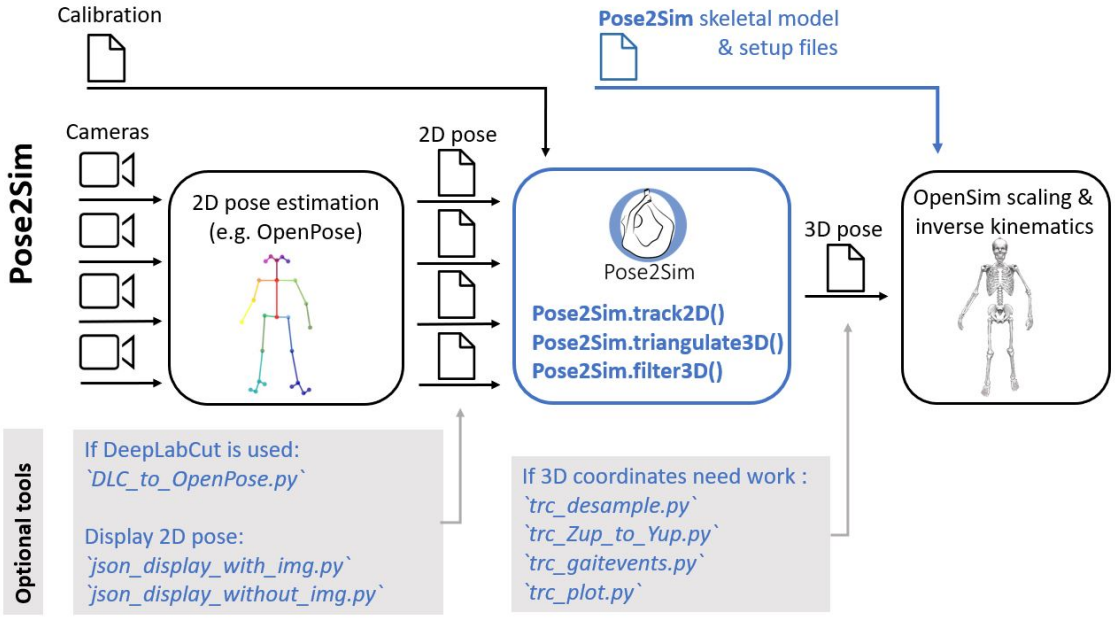


Figure 3.3: The Pose2Sim workflow, along with some optional utilities provided in the package.

possible to build custom skeletons in the `skeleton.py` file, trained for example with DeepLabCut [**?**, **?**] or SLEAP [**?**]. They will be triangulated, but the user will need to build an OpenSim model and set the keypoints in the right place before being able to perform inverse kinematics.

Two optional standalone scripts are also provided if the user desires a visual display of the 2D pose estimation, as well as a tool for converting DeepLabCut data to OpenPose formalism (Figure **??**).

3.2.3 Camera calibration

The user can indicate whether cameras are going to be calibrated with a checkerboard, or if a preexisting calibration file (such as one provided by a Qualisys system) will simply be converted.

If checkerboard calibration is chosen, the number of corners and the size of the squares have to be specified. In this case, the operator needs to take about 20 pictures or one video of the checkerboard per camera. Corners are then detected and refined with OpenCV [**?**]. Detected corners can optionally be displayed for verification. Each camera is finally calibrated using OpenCV based on an algorithm proposed by [**?**]. For intrinsic parameters' determination, the checkerboard needs to be taken from different angles, as close as possible to the camera, however without being cropped. For extrinsic parameters, only one frame is used. The checkerboard is classically laid flat on the ground, in a place visible by all cameras. The user can choose the index of the image which they want to be used as a reference for calculating extrinsic parameters. Residual calibration errors are given, and stored in a log file.

3.2.4 Tracking the person of interest

One needs to differentiate the people in the background from the actual subject. The tracking step examines all possible triangulations of a chosen keypoint among all detected persons, and reprojects them on the image planes. The triangulation with the smallest reprojection error is considered to be the one associated with the right person on all cameras. If the reprojection error is above a predefined threshold, the process is repeated after taking off one, or several cameras. This happens, for example, if the person of interest has exited the field of a camera, while another person is still in the background.

We recommend choosing the neck point or one of the hip points. In most cases they are the least likely to move out of the camera views.

3.2.5 Triangulating

Pose2Sim triangulation is robust, largely because instead of using classic Direct Linear Transformation (DLT) (see previous chapter on **??**) like other approaches do [**?**, **?**, **?**, **?**], we propose a weighted DLT, i.e., a triangulation procedure where each OpenPose keypoint coordinate is weighted with its confidence score [**?**]. This is a good compromise between losing all confidence information, and triangulating the whole heatmap in a volumetric way [**?**], which is very slow. The weighted DLT has been independently introduced by [**?**], in the context of camera calibration. See Algorithm **??** for the proposed solution.

Algorithm 3 Weighted DLT

Our weighted DLT simply consists of weighing Equation **??** with the confidence c given by OpenPose for each camera. The rest of the procedure remains unchanged.

$$\begin{aligned} c \times (P_1^T - uP_3^T)\overrightarrow{Q} = 0, \\ c \times (P_2^T - vP_3^T)\overrightarrow{Q} = 0 \end{aligned} \tag{3.1}$$

Other parameters can be specified, such as:

- The minimum likelihood $conf_{min}$ below which a 2D point will not be taken into account for triangulation. Each detected keypoint is attributed a confidence score by OpenPose.
- The maximum in reprojection error $error_{max}$ above which triangulation results will not be accepted. This can happen if OpenPose provides a bad 2D keypoint estimate, or if the person of interest leaves the camera field. Triangulation will then be tried again on all subsets of all cameras minus one. If the best of the resulting reprojection errors is below the threshold, it is retained. If it is still above the threshold, one more camera is excluded.
- The minimum number of "good" cameras, $ncams_{min}$ (i.e., cameras remaining after the last two steps) required for triangulating a keypoint. If there are not enough cameras left, the 3D keypoint is dropped for this frame.

See Algorithm **??** for the full algorithm of triangulation of a keypoint. Once all frames are triangulated, the ones with missing keypoint coordinates are interpolated. The interpolation method can also be chosen from among linear, slinear, quadratic, and cubic. The mean reprojection error over all frames is given for each point and saved to a log file, as well as the number of cameras excluded to reach the demanded thresholds. The resulting 3D coordinates are formatted as a .trc file, which is a tabulation-separated text format used by OpenSim.

3.2.6 Filtering and other operations

Different filters can be chosen, and their parameters can be adjusted. The user can choose a zero-phase low-pass Butterworth filter [**?**] that they can apply either on keypoint positions or on their speeds, a LOESS filter [**?**], a Gaussian filter, or a median filter. We recommend choosing the first option, as it is the most customary method used in biomechanic sciences. Waveforms before and after filtering can be displayed and compared.

Algorithm 4 Pose2Sim triangulation of a keypoint

Let $error_{max}$ be the maximum allowed reprojection error, $ncams_{min}$ the minimum number of cameras demanded for triangulation (among $ncams_{tot}$ available cameras), and $conf_{min}$ the minimum confidence accepted for a keypoint to be used.

while $error > error_{max}$ AND $ncams > cams_{min}$ **do**
  for all combinations $\binom{ncams_{tot}}{ncams}$ **do**
    for all cameras **do**
      if $conf > conf_{min}$ **then**
        $weighted\_DLT \leftarrow add\_keypoint$
      end if
    end for
    $Q \leftarrow solve\_weighted\_DLT$
    $error \leftarrow calculate\_reprojection\_error$
  end for
  if $min(error) < error_{max}$ **then**
    return $Q[min(error)], min(error)$
  else
    $ncams \leftarrow ncams - 1$
  end if
end while

If needed, other standalone tools are provided to further work on the .trc 3D coordinate files (Figure **??**). Among others, it is possible to undersample a file from a higher to a lower framerate, or to convert a file from Z-up to Y-up axis convention. The resulting 3D coordinates can be plotted for verification. Additionally, a tool is provided to detect gait events from point coordinates, according to the equations given by [**?**].

3.2.7 OpenSim scaling and inverse kinematics

OpenSim [**?**,**?**] is a widespread open-source software which helps compute consistent 3D joint angles, usually from marker coordinates. It lets scientists define a detailed musculoskeletal model, scale it to individual subjects, and perform inverse kinematics. Results are accurate and robust since biomechanical constraints can be adjusted and weighted, bones are set to a constant length, and joints limited to coherent angle limits. OpenSim provides other features such as net calculation of joint moments or resolution of individual muscle forces, although this is beyond the scope of our contribution.

The main contribution of Pose2Sim is to build a bridge between OpenPose and OpenSim. The OpenSim model needs to be carefully crafted. Indeed, inverse kinematics is an under-constrained problem, that can be guided with carefully chosen joint constraints. Pose2Sim provides a full-body model, adapted from the human gait full-body model [**?**] and the lifting full-body model [**?**]. The first one has a better definition of the knee joint, with a coupling relationship between knee flexion and abduction/adduction and internal/external rotation. These joint constraints allow for reducing the 3D knee kinematics to a problem with only one degree of freedom, assuming that the observed individual does not bear any strong anatomical divergence to the norm. In the same way, the latter model has a better definition of the spine: each lumbar vertebra is constrained to the next one, which makes it possible for the spine to bend in a coherent way with only a few tracked keypoints, without having to make it a rigid single bone. Combining those two models allows for ours to be as versatile as possible. Hand movements are locked, because the standard OpenPose models don't provide any hand detection. A ball joint was added between head and torso so that the rotation of the head could be roughly rendered. Of course, the user can also build their own

OpenSim model, and use the markerset of their choice.

The placement of markers on the model is also of paramount importance, especially with so little of them. OpenPose keypoints do not necessarily coincide with joint centers, probably because of systematic labelling errors in the training dataset [**?**]. Moreover, they may be located in a different anatomical position when limbs are fully extended in comparison to when they are fully flexed. However, once model markers are positioned in accordance with those of the triangulated 2D pose estimations, the scaling step is very fast and straightforward since the marker placement will not change from one session, subject, or operator to another. Our model takes these labelling errors into account, and offsets model markers as regards true joint centers accordingly (Figure **??**).

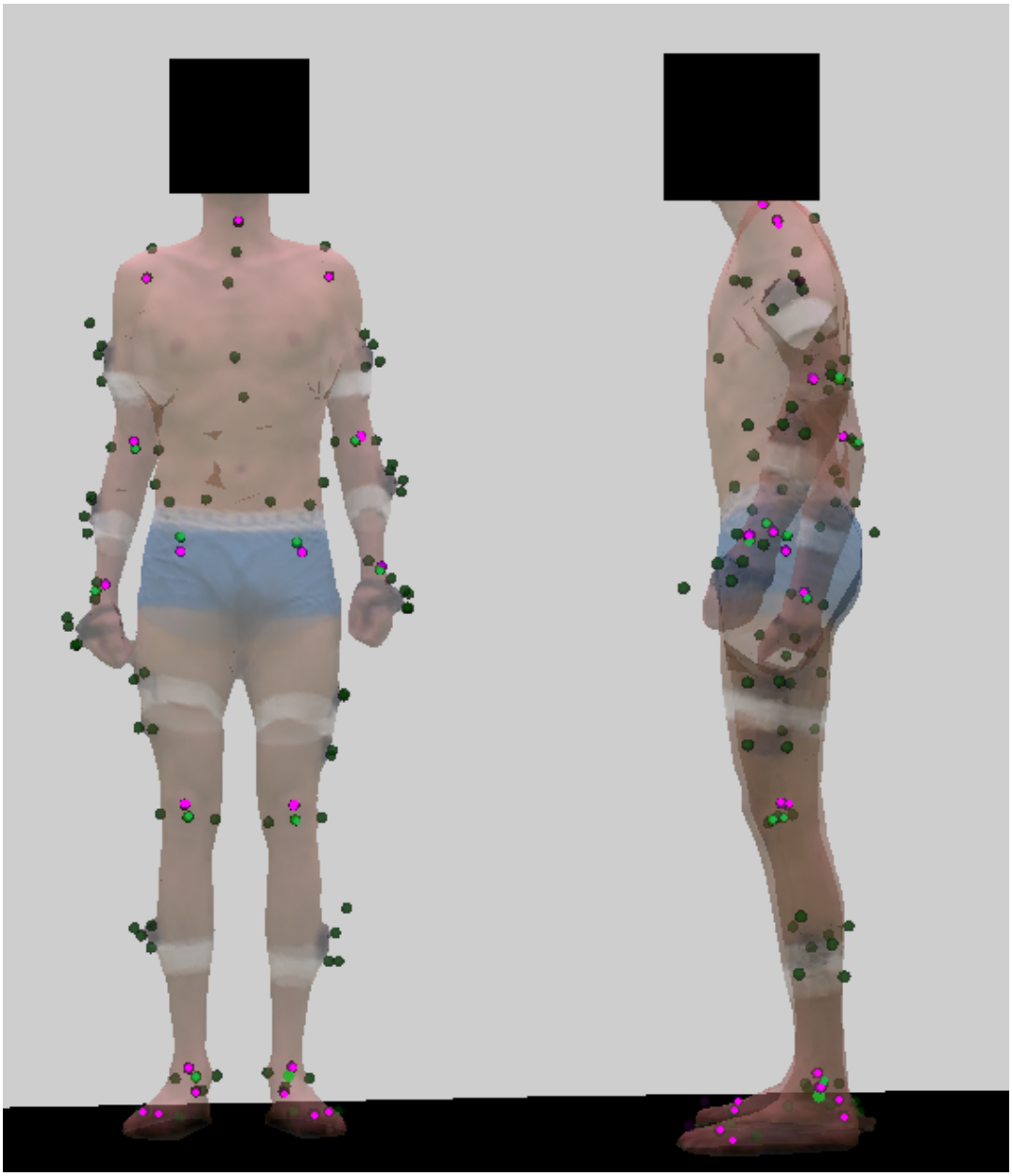


Figure 3.4: Triangulated anatomical markers and clusters (dark green), calculated joint centers (light green), and OpenPose body_25B keypoints (pink) on a textured mesh. OpenPose's eyes and ears keypoints were excluded. Mesh opacity was set to 0.5 in order to make all points visible. This view made it possible to precisely place OpenPose triangulated keypoints on the OpenSim model.

Scaling in OpenSim can be broken down into two parts: first, the proper scaling of the model,

which adjusts bone dimensions and inertial properties according to the distances between the joint markers; second, the adjustment of the other markers on the model, especially anatomical and cluster markers. Joint centers are not trivial to obtain in marker-based approaches, since they must be calculated from the position of skin markers or from clusters: this is not an issue for OpenPose triangulated keypoints, which already represent joint centers in the first place. Moreover, if the model is defined properly, there is no need for further marker adjustment, since markers will not be subject to placement variation due to human error or to skin movement artifacts. Hence, only the first step of scaling must be undergone, and it is simpler than in marker-based approaches. Each body is scaled according to a factor computed as a ratio of distances, defined by the distance between pairs of model markers over distance between corresponding pairs of experimental markers. The markers used for scaling are chosen as follows:

- Arm: distance between shoulder and elbow markers;
- Forearm: distance between elbow and wrist;
- Thigh: distance between hip and knee;
- Shank: distance between knee and ankle;
- Foot: distance between heel and big toe;
- Pelvis: distance between left and right hip;
- Torso: distance between neck and hip;
- Head: distance between head and nose;

All weights of joint coordinates are set to 1, apart from those of the nose and the head, which are set to 0.2, and from those of the other head markers, which are set to 0. We recommend scaling on a standing pose, which is easy to keep for a second, usually not too far from sports poses, and accurately estimated by 2D pose detection models. Moreover, if this condition is satisfied, we can add as a scaling assumption that ankle flexion should be fixed at a neutral 0° angle.

The inverse kinematic tool is used with the same marker weights as in the scaling step. Unlike in marker-based capture, keypoints detection hardly depends on the operator, the subject, nor the context. For this reason, the scaling and the inverse kinematic steps are straightforward, and the provided setup files require little to no adjusting.

3.3 Installation and demonstration

3.3.1 Installation

1. Install **OpenPose** (instructions here).
   Windows portable demo is enough.
2. Install **OpenSim 4.x** from there.
   Tested up to v4.4-beta on Windows. Has to be compiled from source on Linux (see there.
3. *Optional:* Install **Anaconda** or Miniconda.
   Open an Anaconda terminal and create a virtual environment by typing:

   ```
   conda create -n Pose2Sim python=3.7
   conda activate Pose2Sim
   ```

4. Install **Pose2Sim**
   If you don't use Anaconda, type `python -V` in terminal to make sure python>=3.6 is installed.

   - OPTION 1: *Quick install.* Type in terminal:

```
pip install pose2sim
```

- OPTION 2: *Build from source and test the last changes.* Open a terminal in the directory of your choice and clone the Pose2Sim repository:

```
git clone https://github.com/perfanalytics/pose2sim.git
cd pose2sim
pip install .
```

3.3.2 Demonstration Part-1: Build 3D TRC file on Python

This demonstration provides an example experiment of a person balancing on a beam, filmed with 4 calibrated cameras processed with OpenPose.

Open a terminal and check package location with `pip show pose2sim | grep Location`. Copy this path and go to the Demo folder with `cd <path>\pose2sim\Demo`.
Type `python`, and test the following code (Figures **??**):

```
from Pose2Sim import Pose2Sim
Pose2Sim.calibrateCams()
Pose2Sim.track2D()
Pose2Sim.triangulate3D()
Pose2Sim.filter3D()
```

You should obtain a plot of all the 3D coordinates trajectories (Figures **??**). You can check the logs in `Demo\Users\logs.txt`. Results are stored as .trc files in the `Demo\pose-3d` directory (Figures **??**). Note that when the functions are called without any argument, the Config file is searched in the default `Users\Config.toml` location. These parameters can be edited by the user.

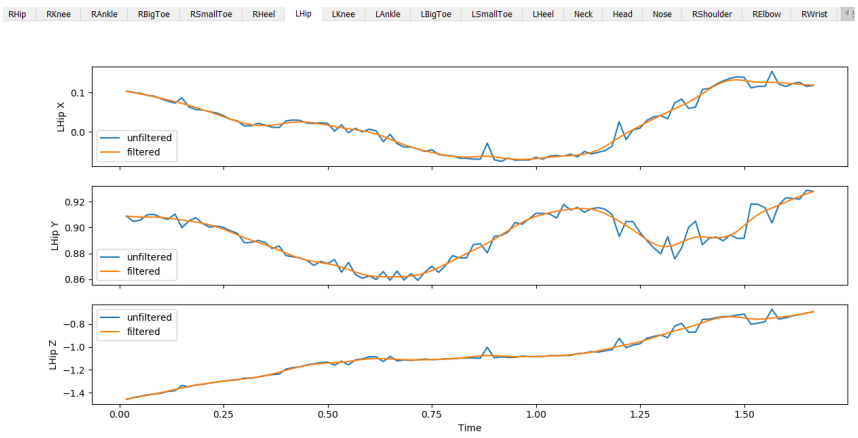


Figure 3.5: Filtered results. Each keypoint trajectory is displayed in a different tab.

PathFileType		4 (X/Y/Z)	Demo_0-100.trc											
DataRate	CameraRate	NumFrames	NumMarkers	Units	OrigDataRate	OrigDataStartFrame	OrigNumFrames							
60	60	100	21	m	60	0	100							
Frame#	Time	RHip			RKnee			RAnkle			RBigToe			
		X1	Y1	Z1	X2	Y2	Z2	X3	Y3	Z3	X4	Y4	Z4	
1	0.0166666667	-0.064972148	0.9015045551	-1.4005886926	-0.0396263662	0.4930973651	-1.4485228257	0.0437901401	0.1438754982	-1.5950846772	-0.0169084827	0.4930973651	-1.4485228257	
2	0.0333333333	-0.0740068294	0.9044249595	-1.3887505524	-0.0396716745	0.4930610544	-1.4481465416	0.039886297	0.1434935164	-1.5931576221	-0.0169371875	0.4930610544	-1.4481465416	
3	0.05	-0.0799400351	0.9088363315	-1.3905580361	-0.0372341964	0.4955765014	-1.4425663246	0.0424186998	0.1500265582	-1.5949272593	-0.0157499669	0.4955765014	-1.4425663246	
4	0.0666666667	-0.0834212999	0.9118448511	-1.3790501541	-0.0378442483	0.4986109124	-1.4352189865	0.041841984	0.1506211073	-1.5951903296	-0.0159322546	0.4986109124	-1.4352189865	
5	0.0833333333	-0.0821238866	0.910708592	-1.3705528594	-0.0415058215	0.4929167908	-1.4391550118	0.0368223321	0.1492766387	-1.5990002244	-0.0214821932	0.4929167908	-1.4391550118	
6	0.1	-0.0870228272	0.9113384284	-1.3568997099	-0.0434174115	0.4981952646	-1.4247995005	0.0306840105	0.1528813191	-1.5954295987	-0.0237343535	0.4981952646	-1.4247995005	
7	0.1166666667	-0.0920100974	0.9116316951	-1.3447088632	-0.0445856424	0.5002300425	-1.4213497807	0.0290451125	0.1540803887	-1.5892342384	-0.0245350272	0.5002300425	-1.4213497807	
8	0.1333333333	-0.0906673188	0.9161769285	-1.3309245886	-0.046053813	0.5073858348	-1.4072542077	0.0334937714	0.1591193395	-1.5866813244	-0.0225906144	0.5073858348	-1.4072542077	

Figure 3.6: An example .trc file of triangulated keypoint coordinates, directly usable in OpenSim.

```
In [6]: Pose2Sim.calibrateCams('User/Config.toml')

Calibrating cameras...
--> Residual (RMS) calibration errors for each camera are respectively [0.221, 0.235, 0.171, 0.191] px,
 which corresponds to [0.402, 0.445, 0.45, 0.505] mm.
Calibration file is stored at [redacted]
```

(a) Calibration can either be done from a checkerboard, or by simply converting a Qualisys calibration file. Calibration errors are computed and provided.

```
In [11]: Pose2Sim.track2D('User/Config.toml')

Tracking the person of interest for Demo, for frames 0 to 100.
100%|[redacted]| 100/100 [00:00<00:00, 383.53it/s]
--> Mean reprojection error for Neck point on all frames is 12.3 px, which roughly corresponds to 22.4 mm.
--> In average, 0.01 cameras had to be excluded to reach the demanded 20 px error threshold.
Tracked json files are stored in [redacted]
```

(b) If several persons are detected in the scene, a tracking step can be carried out in order to make sure that the right person from each camera will be triangulated.

```
In [12]: Pose2Sim.triangulate3D('User/Config.toml')

Triangulation of 2D points for Demo, for frames 0 to 100.
D:\softs\github_david\Pose2Sim\Demo\calib-2d\Calib_qca.toml
100%|[redacted]| 100/100 [00:02<00:00, 33.71it/s]
Mean reprojection error for RHip is 8.0 px (~ 0.015 m), reached with 0.99 excluded cameras.
Mean reprojection error for RKnee is 9.4 px (~ 0.017 m), reached with 0.61 excluded cameras.
Mean reprojection error for RAnkle is 10.8 px (~ 0.02 m), reached with 0.1 excluded cameras.
Mean reprojection error for RBigToe is 10.9 px (~ 0.02 m), reached with 0.57 excluded cameras.
Mean reprojection error for RSmallToe is 10.6 px (~ 0.019 m), reached with 0.44 excluded cameras.
Mean reprojection error for RHeel is 11.1 px (~ 0.02 m), reached with 0.31 excluded cameras.
Mean reprojection error for LHip is 8.8 px (~ 0.016 m), reached with 0.83 excluded cameras.
Mean reprojection error for LKnee is 10.6 px (~ 0.019 m), reached with 0.8 excluded cameras.
Mean reprojection error for LAnkle is 12.3 px (~ 0.022 m), reached with 0.15 excluded cameras.
Mean reprojection error for LBigToe is 10.2 px (~ 0.019 m), reached with 0.33 excluded cameras.
Mean reprojection error for LSmallToe is 11.2 px (~ 0.02 m), reached with 0.46 excluded cameras.
Mean reprojection error for LHeel is 10.6 px (~ 0.019 m), reached with 0.38 excluded cameras.
Mean reprojection error for Neck is 11.1 px (~ 0.02 m), reached with 0.17 excluded cameras.
Mean reprojection error for Head is 9.8 px (~ 0.018 m), reached with 0.56 excluded cameras.
Mean reprojection error for Nose is 8.4 px (~ 0.015 m), reached with 1.95 excluded cameras.
Mean reprojection error for RShoulder is 9.4 px (~ 0.017 m), reached with 0.61 excluded cameras.
Mean reprojection error for RElbow is 9.0 px (~ 0.016 m), reached with 0.63 excluded cameras.
Mean reprojection error for RWrist is 9.7 px (~ 0.018 m), reached with 0.49 excluded cameras.
Mean reprojection error for LShoulder is 10.2 px (~ 0.019 m), reached with 0.5 excluded cameras.
Mean reprojection error for LElbow is 12.1 px (~ 0.022 m), reached with 0.39 excluded cameras.
Mean reprojection error for LWrist is 11.6 px (~ 0.021 m), reached with 0.38 excluded cameras.
--> Mean reprojection error for all points on all frames is 10.3 px, which roughly corresponds to 18.8 mm.
--> Cameras were excluded if likelihood was below 0.3 and if the reprojection error was above 15 px.
In average, 0.55 cameras had to be excluded to reach these thresholds.
3D coordinates are stored at [redacted]
```

(c) The triangulation is weighted by the OpenPose likelihood, and constrained by some thresholds defined in the Config.toml file. If these constraints are not met, e.g., if the reprojection error is too large or if the likelihood of a keypoint is too low, one or several cameras are excluded. The mean reprojection error and the number of cameras that have been excluded to meet the constraints is printed, for each keypoints.

```
In [13]: Pose2Sim.filter3D('User/Config.toml')

Filtering 3D coordinates for Demo, for frames 0 to 100.
--> Filter type: Butterworth low-pass. Order 4, Cut-off frequency 6 Hz.
Filtered 3D coordinates are stored at [redacted]
```

(d) Triangulated data can be filtered, either with a low-pass Butterworth filter or with other types, and parameters can be adjusted.

Figure 3.7: First steps of Pose2Sim pipeline in Python. Calibration can either be done from a checkerboard, or by simply converting a Qualisys calibration file. Note that the functions can be used without any arguments if the Config.toml file is left in the default location.

3.3.3 Demonstration Part-2: Obtain 3D joint angles with OpenSim

In the same vein as we would do with marker-based kinematics, the model first needs to be scaled to each individual, and then inverse kinematics can be performed (Figures **??**).

Scaling:

1. Open OpenSim.
2. Open the provided `Model_Pose2Sim_Body25b.osim` model from `pose2sim/Demo/opensim`. (File ↦ Open Model)
3. Load the provided `Scaling_Setup_Pose2Sim_Body25b.xml` scaling file from `pose2sim/Demo/opensim`. (Tools ↦ Scale model ↦ Load)
4. Run. You should see your skeletal model take the static pose.

Inverse kinematics

1. Load the provided `IK_Setup_Pose2Sim_Body25b.xml` scaling file from `pose2sim/Demo/opensim`. (Tools ↦ Inverse kinematics ↦ Load)
2. Run. You should see your skeletal model move in the Vizualizer window.

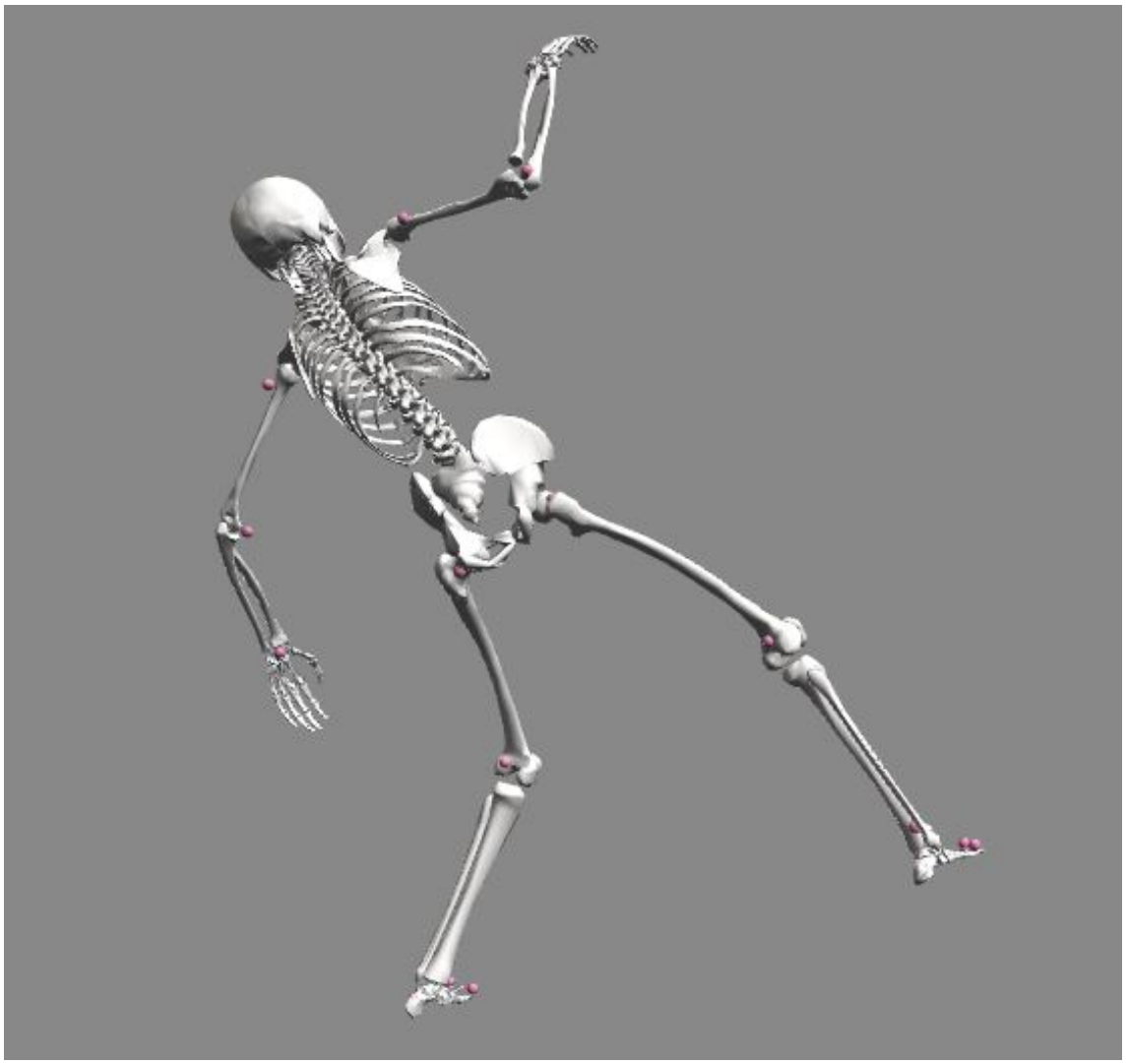


Figure 3.8: At the end of the demonstration, you should have a skeleton balancing on a beam in OpenSim.

Alternatively, OpenSim can be run in command-line:

1. Open an Anaconda terminal in your `OpenSim/bin` directory, typically `C:\OpenSim <Version>\bin` on Windows.
   You will need to adjust the time_range, output_motion_file, and enter the full paths to the input and output `.osim, .trc, and .mot` files in your setup file.

```
opensim-cmd run-tool <PATH TO YOUR SCALING OR IK SETUP FILE>.xml
```

2. You can also run OpenSim directly in Python:

```
import subprocess
subprocess.call(["opensim-cmd", "run-tool",
      "<PATH TO YOUR SCALING OR IK SETUP FILE>.xml"])
```

3. Or take advantage of the full the OpenSim Python API. See there for installation instructions. Note that it is easier to install on Python 3.7 and with OpenSim 4.2.

3.4 Limits and perspectives

3.4.1 2D keypoint detection

Pose2Sim is currently primarily used with OpenPose as a 2D pose detection network. Despite it is very robust, it suffers from issues when used for full-body kinematic analysis. First, keypoint localization suffers from systematic offsets when compared to actual joint center positions [**?**]. Constraining these coordinates to a skeletal model largely reduces the detrimental impact of low-quality 2D joint center estimations. Nevertheless, these offsets have been taken into account in the provided OpenSim model, by shifting OpenPose keypoint placements with regard to marker-based calculated joint centers. This was done manually, but precisely, thanks to our overlayed view (Figure **??**). However, OpenPose's offset may not be the same when a limb is extended as when it is bent, which may influence kinematic results on extreme poses, such as seen in some sports. Hence, using a 2D pose estimation model free from systematic biases on all ranges of motion would certainly improve kinematic accuracy. The body_25B model is more accurate than the default body_25 one, but it is still biased.

Furthermore, both models only detect 25 keypoints. This makes inverse kinematics an under-constrained problem, which has to be guided with carefully chosen joint constraints, and with precise placement of markers on the model. Currently, all pelvis, lumbar, and thoracic angles are solely determined by the detection of the hip keypoints on the lower part, and of the shoulder and neck keypoints on the upper part. As a consequence, even if joint centers were perfectly estimated, the optimization procedure would admit two solutions for the spine curvature, both mathematically and kinematically correct: one with a lordotic posture, and the other with a kyphotic posture. Additionally, there is no marker for the hand, which does not allow for capture of any pronation/supination movement, let alone of any hand or finger movement. The shoulder is also defined as a ball joint, whereas the pectoral girdle is much more complex. Internal/external rotations are solved with difficulties despite the use of kinematic constraints. Using the experimental body_135 OpenPose model would solve the hand issue, but it would also greatly increase the computational cost and would leave the shoulder and spine problem unaddressed. As a consequence, and provided that they are reliably labeled, OpenPose needs more keypoints to solve these indeterminations, and potentially several per joints, in the same way as markersets are designed in marker-based methods. Pose2Sim could operate with such a model, although new keypoints should then be placed afresh on the unscaled OpenSim model.

Moreover, OpenPose struggles to accurately detect pose when the person is upside-down, or

taking an unusual pose. As a consequence, it could be interesting to build a new dataset, with more accurate labelling, more keypoints, trained on specific sports poses.

On a different note, in a sports context, not only the human pose is of interest: sports gear can also be considerably important to detect, such as a ball [**?**], skis [**?**], or bike parts in the context of cycling (see Chapter 7 on Joint OpenPose and DeepLabCut detection). This can help to analyze game dynamics, and to quantify posture cues related to a specific sports discipline. This can be done, for example, by separately process the video with OpenPose, as well as with a custom-trained DeepLabCut model. Resulting .trc coordinate files can be merged, and used in OpenSim. However, the DeeLabCut keypoints must be referenced on an OpenSim model, which may need to be crafted from scratch, such as a ball, skis, or bike, depending on the detected object.

3.4.2 User-friendly calibration

Calibration remains a challenging task in daylight, at a distance, and with non research-grade cameras. It could be useful to make it more robust, either by implementing the Aniposelib library [**?**], by importing calibration files from an Argus wand calibration [**?**], or by automatically calibrating on people's limb length [**?**]. Along with synchronization of light-weight cameras, this topic will be detailed in Chapter 6 on **??** [**?**].

3.4.3 Multi-person analysis

Pose2Sim bears several limitations. First, despite it is not altered by people entering the field of view, it can currently only analyze the movement of one single person. For races, team sports, and combat sports, it would be useful to be able to analyze the movement of several athletes at the same time. This could be achieved in two steps: first, by triangulating all the persons whose reprojection error is below a certain threshold, instead of taking only the one with minimum error, in a similar way as carried out by [**?**]; then, by tracking the triangulated persons in time, e.g., by limiting the displacement speed of each person's neck keypoint from one frame to the next one (see **??** in previous chapter for more insight on available methods).

3.4.4 Real-time analysis

Currently, Pose2Sim does not work in real time. This is a drawback for coaches and athletes, who need feedback in a timely manner. However, a timely analysis of athletes' movements directly on the sports field appears to be achievable. Indeed, OpenPose is faster than most of its competitors [**?**], and the rest of the process is not computationally costly. Moreover, the pose detection, the triangulation, and the OpenSim inverse kinematic optimization work frame by frame. As a consequence, it is conceivable to calibrate and scale the model first, and then to feed the GUI frame by frame. This would allow the workflow to operate with only a few seconds of delay.

3.4.5 Visualization tools

Pose2Sim does not provide a GUI yet. This can make it complicated for coaches to adopt the tool. However, the code has been adopted by other entities. The 3D animation "CEB" studio built a Blender [**?**] extension using Pose2Sim for realistic 3D markerless animation. However, it is not free nor open-source [**?**] (Figure **??**). In addition, the CAMERA laboratory of the University of Bath is currently developing a GUI around Pose2Sim, which would make the tool more accessible.

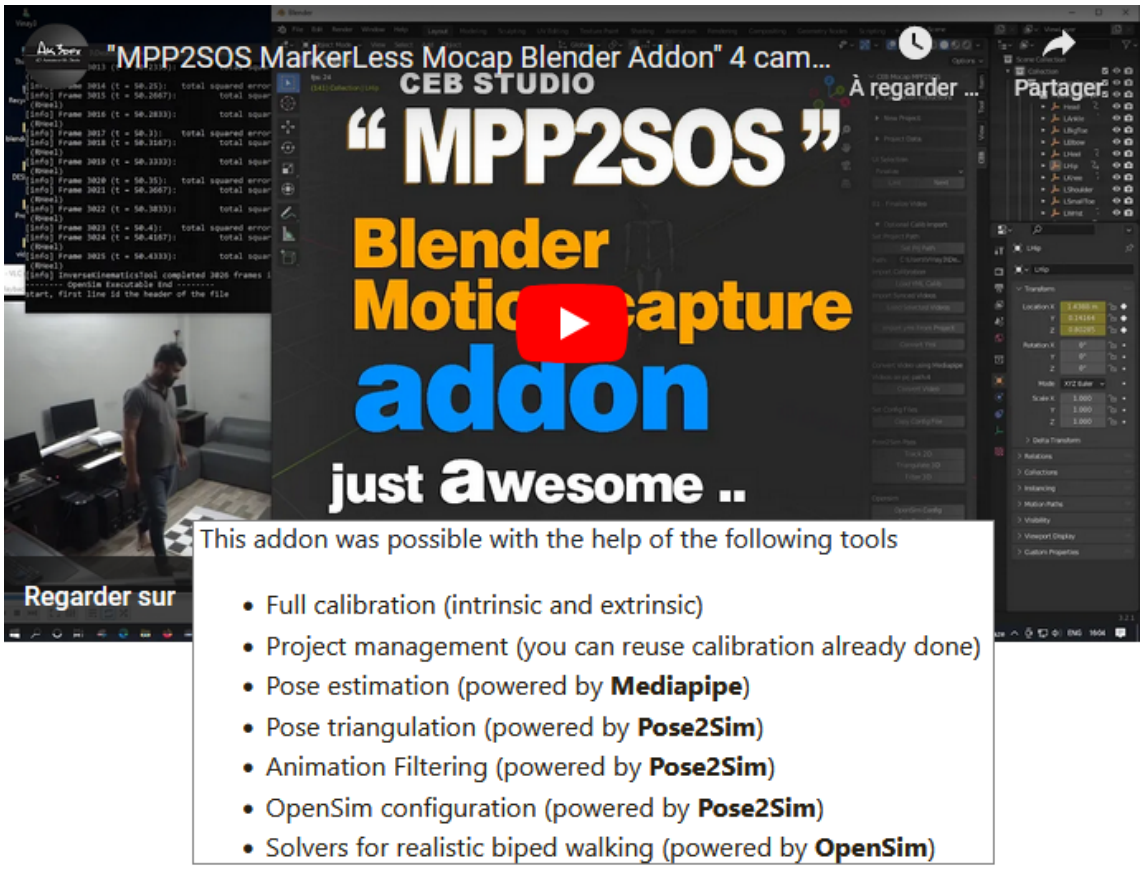
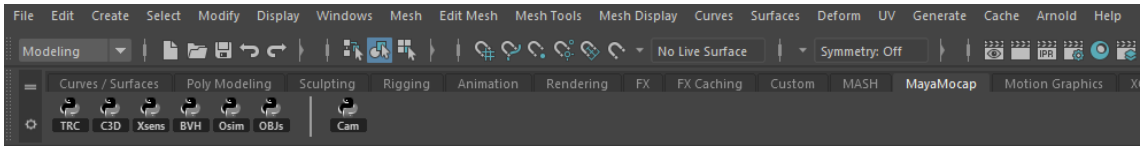
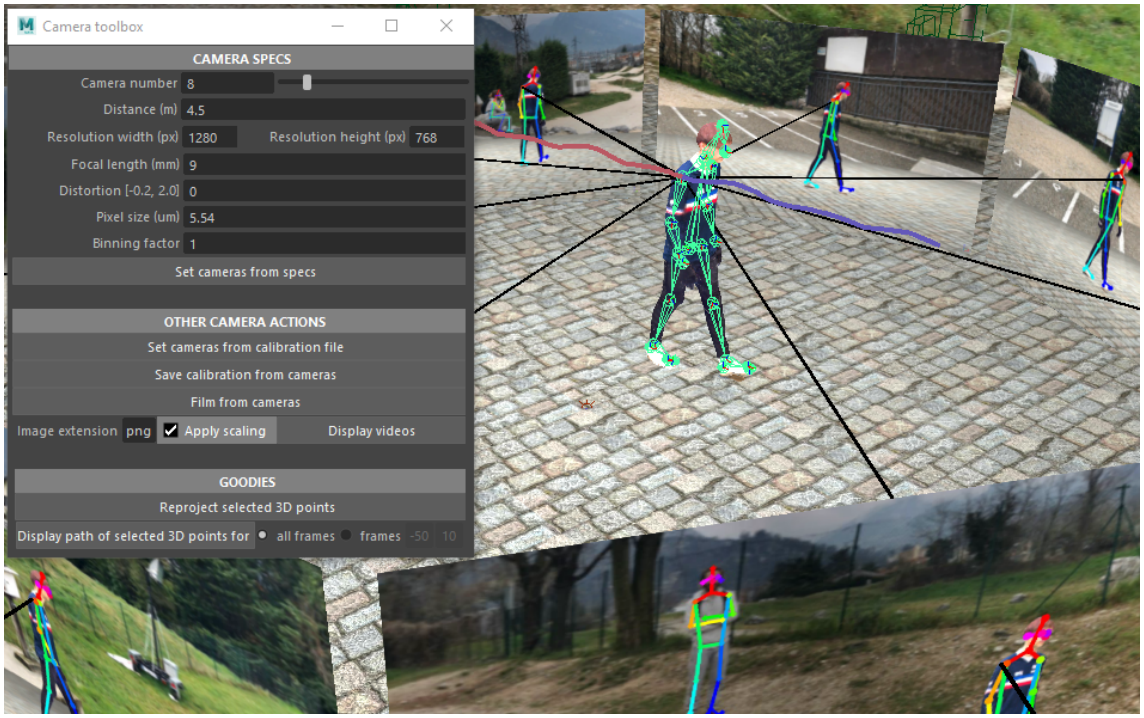


Figure 3.9: CEB Studio built the MPP2SOS Blender add-on, which uses Pose2Sim for realistic 3D markerless animation.

I also developed a toolbox for Maya [**?**] called Maya Mocap [**?**]. First, it can import and display various types of motion files. Then, it can load cameras from a calibration file, film with them, and import the filmed image sequences. It can also help to make sure that triangulated points are well reprojected on the camera plane, by tracing a line from the point to the camera center. In addition, it can display the 3D trajectory of a point (see Figure **??**). Next objectives would be to make it able to import an OpenSim model and its motion files, and to present it as a cleaner package, ready to be released. Since all the tools used in Pose2Sim are open-source, it would also be more consistent to translate it into Blender instead of Maya, in order to offer an entirely operational and open-source tool.



(a) The Maya-Mocap add-on is displayed as an additional toolbar in Maya.



(b) Maya-Mocap can import several file types, e.g., a .trc motion file (bright green) or a textured animated 3D mesh. It can also load cameras from a calibration file, film with them, and display the filmed image sequences. In addition, it can reproject a selected point onto the camera plane (black lines), to make sure that it has been correctly triangulated. The 3D trajectory of a point can also be highlighted (red and blue lines).

Figure 3.10: The Maya-Mocap add-on

3.4.6 Other perspectives

Other minor adjustments could be made in order to improve the triangulation and the filtering steps. Implementing Random Simple Consensus (RANSAC) triangulation [**?**] as an alternative to our weighted Direct Linear Transform (DLT) [**?**], and opting for optimal fixed-interval Kalman smoothing instead of low-pass filtering [**?**, **?**], may reduce errors, especially in large outliers. Performing the pose estimation, the triangulation, and the inverse kinematics on the cloud rather than on a local computer could allow athletes and coaches to use the software in a web application. Adding muscles which were stripped from the skeleton in the OpenSim model could allow for joint kinetics prediction. Neural networks could be trained to estimate ground reaction forces from kinetics on specific tasks, without the use of a force platform [**?**, **?**, **?**].

4

Robustness assessment

A markerless motion capture method is satisfying if it is accurate, fast, and robust. Robustness is deemed good when results are unchanged while adding constraints on the subject or on the environment. We challenge our workflow on walking, running, and cycling tasks, by adding people in the background, and by simulating challenging conditions: (Im) alters image quality (11-pixel Gaussian blur and 0.5 gamma compression); (4c) uses fewer cameras (4 vs. 8) which leads to unsolved occlusions; and (Cal) introduces calibration errors (1 cm vs. perfect calibration).

When averaged over all joint angles, stride-to-stride standard deviations lay between 1.7° and 3.2° for all conditions and tasks, and mean absolute errors (compared to the reference condition—Ref) ranged between 0.35° and 1.6°. For walking, errors in the sagittal plane were: 1.5°, 0.90°, 0.19° for (Im), (4c), and (Cal), respectively. As a consequence, Pose2Sim is robust enough for the 3D joint angle analysis of walking, running, and cycling, under challenging conditions.

*This chapter is adapted from the article published in the Sensors: "Pose2Sim: An End-to-End Workflow for 3D Markerless Sports Kinematics—Part 1: Robustness" [**?**]. See Figure **??** for a visual abstract.*

Contents

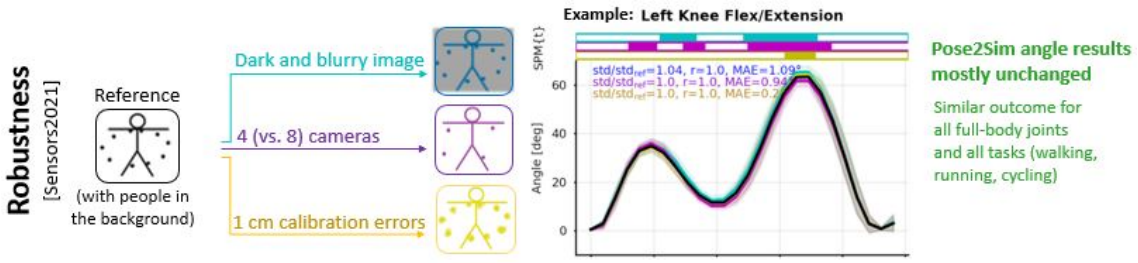


Figure 4.1: Visual abstract for Pose2Sim robustness assessment [?].

4.1 Introduction

4.1.1 Robustness definition

According to the review of [**?**], the performance of a method can be ranked regarding its accuracy, speed, or robustness. Accuracy is mostly assessed with MPJPE (Mean Per Joint Position Error); speed is evaluated either regarding computing complexity, or framerate when possible; and robustness is gauged through differences in the results while changing the system parameters only. [**?**] points out that authors usually only consider accuracy, sometimes speed, but rarely robustness. However, robustness is of paramount importance in the context of sports, especially "in the wild". This chapter will focus on robustness, the next one on **??**, and we will not focus on speed in this thesis (although chapter 3.4.5 broaches Real time considerations).

[**?**] proposed to express robustness as the number of constraints on the subject or on the environment required for a motion capture system to be operational. Some of the assumptions they proposed have already been universally overcome by deep-learning-based methods. For example, no markers are involved anymore, the subject can wear their usual clothes (including loose pants or dresses [**?**]), and the background does not need to be static or uniform. Some other items remain an open problem.

For instance, most 3D methods assume that only one person lies in the camera field of view. This is a strong assumption, especially outdoors where people and athletes pass by and an operator is often present. Although it is starting to be addressed, standard solutions are yet to be determined [**?**, **?**, **?**, **?**].

Other open questions lie in the environment, much less controlled in a sports context than in a lab, which can result in poor image qualities. [**?**] experienced that OpenPose is very robust to extreme lighting conditions. However, research has shown that pose estimations models are more robust to noise or brightness changes, while less robust to motion or to defocus blur [**?**]. And yet, in sports, the movement is not usually slow, continuous, nor limited to the sagittal plane.

Occlusions are, for the most part, solved by using a network of calibrated cameras. Since triangulation is computed using a least square method, a large amount of cameras will also blunt imprecision on the 2D joint estimations. [**?**] showed that once correctly trained for 3D macaque pose estimation, eight cameras were enough to correctly infer 80% of the 13 considered keypoints, while four cameras decreased the performance to about 50%. However, a correct estimation of extremities such as feet and hands required more than eight cameras.

Camera calibration can be challenging outside, due to large volume spaces, bright light, and contrasting shades. As a consequence, it is close to impossible with the classic approach using a wand equipped with retro-reflective markers. Moreover, simple calibration with a checkerboard may cause errors on intrinsic and extrinsic camera parameters estimation [**?**]. A calibration is generally considered acceptable if the average residuals of each camera (i.e., the root mean square error of the reprojection of the 3D reconstructed coordinates on the 2D image plane) is below 1 pixel. In metric terms, the markers-based Qualisys Track Manager software recommends redoing a calibration when the average residuals exceed 3 millimeters [**?**]. The pinhole camera model gives an equivalence between pixel values on the image plane, and metric values on the object plane at the center of the scene, as demonstrated by Figure **??** and Equation **??**. See Chapter 2.2.1 on **??** or [**?**] for in-depth explanations.

$$Err^{Img} = \frac{f \times Err^{Obj}}{Z_C} \tag{4.1}$$

4.1.2 Assessing robustness

Because [**?**] showed that the quality of markerless results were task specific, we will examine walking, running, and cycling. Before assessing the robustness of the workflow, the relevance of

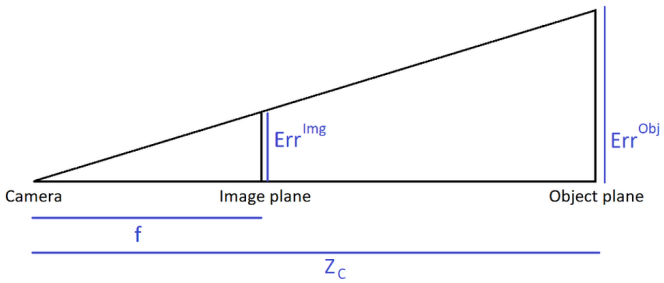


Figure 4.2: The pinhole camera model permits a correspondence between image coordinates and object coordinates. f: focal distance, Z_C: object to camera distance, Err^{Img}: error on image plane, Err^{Obj}: error on object plane. f and Err^{Img} are usually expressed in pixels, while Z_C and Err^{Obj} are expressed in meters.

the computed 3D full-body angles needs to be estimated. This will be done by comparing our angle results to those of a normative walking database. Further concurrent validation of the accuracy will be determined in the next chapter on **??**. Assuming that the gait of a young and healthy person is repeatable, we assume that most of the variability between cycles should be caused by a lack of repeatability of the system. Thus, repeatability will be evaluated by comparing the kinematics of between cycles, within each task and each capture condition.

Robustness itself will be assessed through all three types of movements, in accordance with the open problems previously described. In addition to the person of interest, some people will be present in the background in all examined conditions. We will compare results more in depth when:

1. Image quality is altered, by simulating a dark scene captured with defocused cameras objectives.
2. Self- and bike-occlusions are becoming more challenging, as we decrease the number of cameras.
3. Calibration errors are introduced, by corrupting the calibration files.

The underlying idea presented in this article is to verify whether modifying external environment parameters significantly impacts variability in joint angle estimation.

4.2 Methods

See video here for a visual description of the protocol.

4.2.1 Experimental setup

To guarantee a tight control of the environment parameters, we captured our data in a dedicated studio platform called Kinovis [**?**], from which we were able to create realistic virtual views similar to outdoor video. This platform is a $10m \times 10m \times 5.6m$ green room equipped with 68 video cameras recording at 30 fps in 4 Mpixels resolution, for a practical acquisition space of about $5m \times 5m \times 3m$. The system computes 3D textured meshes by convex visual hull reconstruction [**?**]. The meshes were inserted in a virtual environment composed of an environment texture map captured from a BMX racetrack, and a custom-made virtual floor. It should be noted that three people were present in the background, which introduced a realistic artifact of multiple subjects.

We developed a script for Autodesk Maya [**?**] (see **??**) that allows us to render the textured mesh files, as well as to virtually set any cameras with specific properties (position, orientation, resolution, focal length, distortion, pixel size, binning factor). Views seen through virtual cameras

can be saved as video files and visualized into a global 3D environment (Figure **??**). The generated video files were used as input to the 3D kinematics pipeline.

Figure 4.3: The Kinovis room allows for the capture of 3D textured meshes. These meshes are placed in a virtual environment, and then filmed from virtual cameras.

For the purpose of this study, we created 8 virtual video cameras. Resolution was set to 1280×768 pixels, focal length to 9 mm, pixel size to 5.54 µm, and no distortion nor binning was introduced. Binning refers to the process of grouping pixels in order to increase sensitivity to light, at the expense of decreasing resolution. Cameras were regularly distributed 8 m away from the center of the captured volume, at a height of 1 m, so that the whole body could be detected for a maximum of movement cycles. We then rendered the scene as video files from our virtual cameras and saved the exact calibration parameters. We applied a 3×3 pixel Gaussian blur afterwards to reduce sharp edges of the virtual scene compositing (Figure **??**). This resulting image quality was considered as "standard".

4.2.2 Participant and protocol

One healthy adult male subject (1.89 m, 69 kg) participated in the study. He provided his informed written consent prior to participating. The study was conducted in accordance with the

Figure 4.4: To smooth out sharp edges due to compositing, we applied a 3×3 *pixel Gaussian blur to the videos filmed from our virtual scene.*

Declaration of Helsinki [**?**]. No requirement was given to him regarding his outfit. He was asked to perform three basic sports tasks: walking, running, and cycling. For all three tasks, the subject was given a moment beforehand to warm up and find a comfortable and regular pace, which he could then follow owing to the sound of a metronome:

- *Walking:* The subject walked in a straight line back and forth over the 10 m diagonal of the room. His body mesh could be fully reconstructed only in the central 5 m of the acquisition space, i.e., only roughly 2 gait cycles were acquired per walking line. His comfortable stride pace was 100 BPM (Beats per Minute). The stride length was not monitored.
- *Running:* The subject jogged in a straight line back and forth along the 10m diagonal of the room. His comfortable stride pace was 150 BPM (Beats per Minute). The stride length was not monitored.
- *Cycling:* The subject cycled on a road bike placed on a home trainer. He himself adjusted the resistance and the height of the saddle prior to the capture. His comfortable cadence was 60 BPM.

As obtaining the textured meshes of the subject in the green Kinovis room involved filming simultaneously with 68 4 Mpixels cameras that generated a flow of over 8 gigabytes per second, the capture design limited the acquisition time to 45 s.

4.2.3 Challenging robustness

We challenged robustness with 3 challenging conditions, compared to a reference one.

- *Reference Condition (Ref):* The reference condition under which our 3D markerless kinematic system had to operate took advantage of the standard image quality, 8 available virtual cameras, and a perfect calibration. The standard quality corresponded to the unaltered images of the 3D scene filmed from our virtual cameras. The reference condition involved 8 virtual cameras, as a good compromise of what is feasible in real outdoor conditions. As a comparison, a study on macaques in a similar context showed that 8 cameras were enough to correctly infer 80% of the 13 considered keypoints [**?**]. Since the virtual cameras were explicitly specified in the virtual environment, calibration could be considered perfect.
- *Poor Image Quality (Im):* Video quality was made blurrier and darker: a Gaussian blur ($11 \times 11px$) was applied, as well as a 0.5 gamma compression (Figure **??**). This simulated a dark scene captured with defocused camera objectives. These parameters were chosen empirically.
- *Less Cameras (4c):* The 2D joint coordinates were triangulated with only 4 cameras, instead of 8 in the reference condition: one on each side, one in the front, and one in the back, set 90° apart from each other.
- *Calibration Errors (Cal):* Calibration residuals are classically supposed to be under 1 px on the image plane or under 3 mm on the object plane. Using Equation **??** demonstrates that in our case 3 mm corresponds to 0.61 px. We chose to simulate a calibration error of 2 px, which corresponds to about 1 cm (Equation **??**).

$$Err^{Obj} = \frac{Err^{Img} \times D}{f} = \frac{2 \times 8}{\frac{9 \times 10^{-3}}{5.54 \times 10^{-6}}} = 9.8 \times 10^{-3} m \tag{4.2}$$

  The calibration error was simulated by translating the extrinsic parameters of each camera in a random direction. The norm was randomly picked in a normal distribution of mean 2 px and a standard deviation of 1 px. The mean of these 8 translations was ensured to be equal to 210^{-3} px.

4.2.4 Markerless kinematics

Then, we used Pose2Sim to robustly triangulate OpenPose outputs and feed the resulting 3D joint coordinates to OpenSim. The exact same parameters were used for all 4 conditions and all 3 movement tasks, in order to make sure the process did not induce any supplementary deviation to the compared results.

2D pose estimation

We applied OpenPose (version 1.6) on all the captured videos. We used the experimental body_25B model (see Figure**??**) with highest accuracy parameters, which is more accurate than the default body_25 one and reduces the number of false positives [**?**].

Tracking, triangulation, and filtering

We used the Tracking function of Pose2Sim to try out every possible triangulation of the neck keypoint of all detected persons. The person with the smallest reprojection error was deemed to be the one to be tracked. If the reprojection error was above 10 pixels, one camera was removed and the process was repeated. See **??** Chapter 3 for more details.

The weighted DLT provided by the Triangulation function of Pose2Sim was employed. Some keypoints were sometimes occluded to some cameras, either by the subject himself, by his cycling

Figure 4.5: The image under poor image quality (Im) conditions. A Gaussian blur ($11 \times 11 px$) was applied, and a 0.5 gamma compression made the image darker.

gear, or simply because the subject left the camera field of view. In such a case, OpenPose usually gave a low (or zero) confidence to the estimated point, which was dealt with by setting a confidence threshold above which the camera in question was not used for the triangulation. However, OpenPose occasionally wrongly detected the occluded keypoint with a relatively high confidence. Under such circumstances, the point was erroneously triangulated. This issue was spotted and solved by reprojecting the 3D point on the camera planes. If the reprojection error between the reprojected points and the OpenPose detection was higher than a predefined threshold, the process was repeated after removing one, or several, cameras. If less than 3 cameras remained, the frame was dropped for this point, and missing frames were later interpolated with a cubic spline. 3D joints positions were then exported as an OpenSim compatible .trc file. We chose a confidence threshold of 0.3 and a reprojection error threshold of 10 px.

3D coordinates were finally filtered with a Butterworth filter [**?**]. We chose a zero-lag fourth order low-pass Butterworth filter, with a 6 Hz cutoff frequency.

Gait events determination

Cycles were determined using one of the methods given by [**?**] for the determination of gait events using kinematic data. Zeni et al. suggested defining heel strike as the time of maximum distance between the heel and the sacrum. Here, the sacrum marker was assimilated to the ipsilateral hip joint coordinate. Although there is no heel strike in cycling, we used the same definition as a reference for the start of a cycle. An implementation of the algorithm is provided by Pose2Sim in Utilities/trc_gaitevents.py. A cycle was defined as starting from heel strike, with a duration t_{cycle}:

$$t_{cycle} = \frac{2}{cadence/60} seconds \tag{4.3}$$

The duration of a cycle t_{cycle} was 1.2 s for walking, 0.8 s for running, and 1.0 s for cycling.

The reduced area of acquisition and the limit of 45 s of capture restricted the analysis to 8, 9, and 15 cycles for walking, running, and cycling, respectively.

Kinematic analysis

The full-body musculo-skeletal model proposed by [**?**] was adjusted to our needs. OpenPose model markers were carefully placed on the body, as closely as possible to their average triangulated 3D coordinates. Muscles were removed since they were not considered at that stage. A ball joint was added between head and torso so that the rotation of the head could be roughly rendered. Pelvis translation and subtalar angle rotation were unlocked. Lumbar extension and lumbar bending were clamped between -15° and 15°, and constraints on hip flexion/extension ranges of motion were released since they were too strict for our cycling task. This model uses a coupling relationship between knee flexion and the other planes of movement, which allows for 3D knee kinematics with only one degree of freedom. *Note that the model was later improved when assessing the accuracy of the workflow.*

See previous chapter on **??** for more details on the scaling and inverse kinematics procedures. We visually verified at the end of the inverse kinematic step that model markers approximately overlaid experimental markers (Figure **??**), and we batched inverse kinematics for all gait cycles with the command line utilities. We then concatenated all results in a .csv file analyzed them using python.

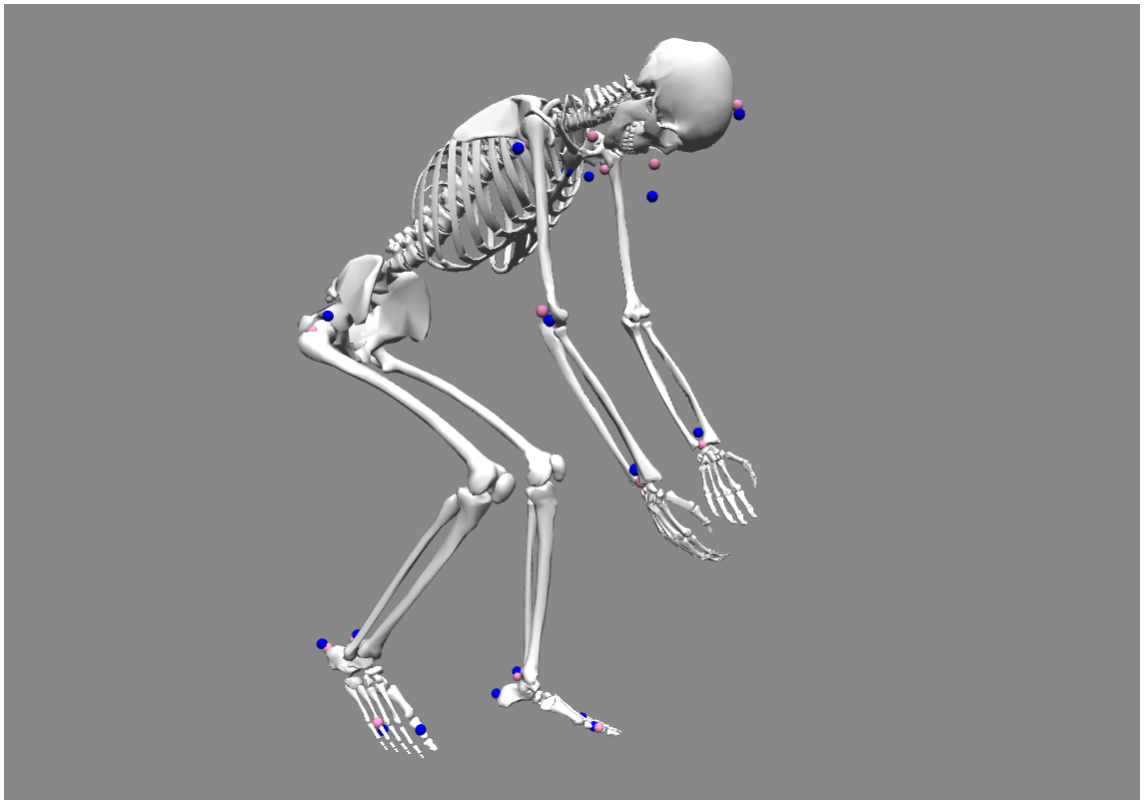


Figure 4.6: OpenSim inverse kinematics on cycling (C). Model markers are pink; experimental markers are blue.

4.2.5 Statistical analysis

As similar results were expected for both sides, as more strides were captured on the left one, and for the sake of clarity, the analysis was performed on the left side only. Descriptive

statistics were calculated at the triangulation stage, for each (Ref), (Im), (4c), and (Cal) condition. We calculated mean, standard deviation (std), and maximum (max) number of cameras excluded for performing triangulation. Triangulation was deemed acceptable in Pose2Sim when OpenPose confidence was above 0.3, and when reprojection error was below 10 pixels. We also calculated mean, std, and max reprojection errors. Descriptive statistics of the OpenSim scaling and inverse kinematics steps were also rendered: we retrieved mean, std, and max root-mean-square error (RMSE) of the experimental and model markers.

Next, we investigated the relevance of our results. The angles of the Ref condition on the walking task were compared to those of a normative walking gait database [**?**], from which we took a subset of 14 young and healthy males walking at a "comfortable" speed (Norm). Pearson's correlations (r) and mean absolute error (MAE) were calculated. As the angle definition was different from that in the OpenSim model [**?**], only the flexion/extension angles were compared.

Repeatability was estimated by computing stride-to-stride standard deviations within (Ref), (Im), (4c), and (Cal) conditions. These stride-to-stride variability results were then compared to [**?**], obtained with a marker-based approach and averaged over 18 young adults. The robustness of the workflow was assessed by calculating the standard deviation ratio between each degraded condition and the Ref one, as well as the r coefficient and the MAE. Statistical parametric analysis (SPM-1D) was lastly performed between Ref and degraded conditions to convey statistical differences along time (See [**?**]).

4.3 Results

4.3.1 Data collection and 2D pose estimation

Each trial took several hours to process on a cluster of high-end computation units. Then, filming the resulting 3D meshes in a virtual scene with virtual cameras also took a few hours per sequence, as well as about 50 Go of storage space. Although all of this allowed us to perfectly control the parameters of the capture, this step would not be carried out in the wild, where capture would simply be carried out by filming with calibrated cameras.

Apart from the capture design, which is very particular to this study, the OpenPose 2D pose estimation was the most computationally costly step. On a standard computer with an Nvidia GeForce GTX 1080 graphic card (8 Go memory), the detection for each camera ran at a little over 0.5 fps (i.e., 0.07 fps for 8 cameras).

4.3.2 Pose2Sim tracking, triangulation, and filtering

On our standard computer, tracking was performed at an average of 5 fps depending on the number of undesired persons present in the background; triangulation was at about 10 fps; and filtering was almost instantaneous in comparison.

Depending on the considered frame, the reprojection error of a joint coordinate sometimes exceeded the prescribed threshold. In this case, the triangulation process was executed another time after excluding the 2D joint coordinates estimated from one, or several, cameras. Over all frames, an average of approximately 0.5 cameras were excluded in walking and in running and 1.6 in cycling (Table **??**). The mean of the reprojection error was about 3.5 px ($\approx$1.6 cm) in walking and running and 6 px ($\approx$3 cm) in cycling. More cameras had to be excluded for the nose and for the extremities such as wrists and toes, and the reprojection error was mostly large on hips, head, and extremities.

Within each task, almost twice as many cameras had to be excluded in the (Im) condition as in the (Ref) condition, with nearly one camera excluded in walking and running, and nearly 2.5 in cycling. However, the reprojection error was increased by only about 10%. About twice as few

Task	Conditions	Mean number of excluded cameras			Mean absolute reprojection error		
		mean	std	max	mean	std	max
Walking	Ref	0.47	0.57	2.0 (Nose)	3.3 px (1.6 cm)	1.1 px (0.54 cm)	5.3 px (2.6 cm, LHip)
	Im	0.91	0.8	2.4 (LWrist)	3.7 px (1.8 cm)	1.0 px (0.52 cm)	5.2 px (2.6 cm, LSmallToe)
	4c	0.27	0.34	1.0 (Nose)	2.9 px (1.4 cm)	0.93 px (0.47 cm)	4.5 px (2.2 cm, LSmallToe)
	Cal	0.47	0.57	2.0 (Nose)	5.1 px (2.5 cm)	0.91 px (0.45 cm)	6.9 px (3.4 cm, LHip)
Running	Ref	0.48	0.64	2.2 (LWrist)	3.5 px (1.7 cm)	1.2 px (0.57 cm)	5.6 px (2.8 cm, LWrist)
	Im	0.94	1.2	4.5 (LWrist)	4.0 px (2.0 cm)	1.4 px (0.69 cm)	7.2 px (3.6 cm, RWrist)
	4c	0.22	0.31	1.0 (LWrist)	3.3 px (1.6 cm)	0.97 px (0.48 cm)	4.7 px (2.3 cm, LWrist)
	Cal	0.47	0.65	2.2 (LWrist)	5.4 px (2.7 cm)	1.0 px (0.52 cm)	7.2 px (3.6 cm, LWrist)
Cycling	Ref	1.62	1.4	4.2 (RBigToe)	6.1 px (3.0 cm)	1.2 px (0.58 cm)	8.5 px (4.2 cm, Head)
	Im	2.41	1.9	5.7 (RBigToe)	6.3 px (3.1 cm)	1.3 px (0.60 cm)	8.5 px (4.2 cm, Head)
	4c	0.76	0.67	2.1 (RBigToe)	5.3 px (2.6 cm)	1.6 px (0.82 cm)	8.4 px (4.2 cm, LElbow)
	Cal	1.68	1.4	4.24 (RBigToe)	6.9 px (3.4 cm)	1.0 px (0.51 cm)	8.9 px (4.4 cm, Head)

Table 4.1: Descriptive statistics on the triangulation step performed by Pose2Sim from OpenPose body_25B model. Mean absolute reprojection error and mean number of excluded cameras were calculated over time. Mean (mean), standard deviation (std), and maximum (max) in each of these variables are displayed. Walking, running, and cycling tasks were investigated in each four conditions: reference (Ref), poor image quality (Im), four cameras instead of eight (4c), and calibration errors (Cal).

cameras had to be excluded in the (4c) condition as in (Ref) condition, and the reprojection error was decreased by about 10%. The (Cal) condition did not involve more excluded cameras; indeed, calibration errors had no effect on the OpenPose confidence, and reprojection error stayed within the threshold of 10 px. However, a 2 px average error in calibration logically resulted in about a 2 px ($\approx$1 cm) larger reprojection error than in the Ref condition.

4.3.3 OpenSim scaling and inverse kinematics

Scaling took a few hours, which is in line with the usual expectations in marker-based methods. However, markers would be positioned in the same place by the pose estimation algorithm regardless of the subject and of the operator: as a consequence, in the future, only bones would have to be scaled, and markers will not have to be further adjusted. Due to the small number of markers whose positions had to be optimized, Opensim's inverse kinematics ran at more than 0.5 fps.

OpenSim recommends the RMS of experimental vs. model marker errors to be below 1 cm for scaling. Our best average RMS error was 1.9 cm. During inverse kinematics, it is recommended for it to stay below 2–4 cm. The average RMS error was typically below 4 cm, but it could reach much more for some markers at certain phases of the cycle, especially in the cycling task. Within each task, changing conditions made very little difference in RMS marker errors, including in mean, standard deviation (std), or maximum error (Table **??**).

Note that one should not compare the values of the reprojection errors in the triangulation step, and of the marker errors in the inverse kinematics one. Primarily, the first one is calculated over time, while the second one is calculated over markers. Additionally, the first one is a mean absolute error (MAE), while the second one is a root mean square error (RMSE). RMSE squares the errors, which always makes it larger than the MAE. It penalizes large errors more, which has some implications that are not addressed here but are documented in [**?**]. These errors should only

Task	Condition	RMS marker error		
		mean	std	max
Walking	Ref	2.8 cm	0.13 cm	3.2 cm (Mid stance)
	Im	2.8 cm	0.11 cm	3.1 cm (Mid stance)
	4c	2.8 cm	0.12 cm	3.2 cm (Mid stance)
	Cal	2.9 cm	0.13 cm	3.2 cm (Mid stance)
Running	Ref	2.2 cm	0.22 cm	2.6 cm (Mid stance)
	Im	2.4 cm	0.21 cm	2.8 cm (Mid stance)
	4c	2.5 cm	0.30 cm	2.4 cm (Mid stance)
	Cal	2.2 cm	0.21 cm	2.6 cm (Mid stance)
Cycling	Ref	3.4 cm	0.11 cm	3.6 cm (Dead center)
	Im	3.8 cm	0.18 cm	4.2 cm (Dead center)
	4c	3.9 cm	0.60 cm	5.9 cm (Dead center)
	Cal	3.4 cm	0.11 cm	3.6 cm (Dead center)

Table 4.2: Descriptive statistics on the inverse kinematics step performed by OpenSim with a full body model adapted from Rajagopal's. Root mean square (RMS) errors between experimental and model markers were calculated over all markers. Mean, standard deviation (std), and maximum (max) are displayed. Dead center refers to the phase where the crank is near the vertical position.

be used to compare conditions within each step.

4.3.4 Relevance, repeatability and robustness of angles Results

Flexion/extension lower-limb angles were compared to a normative walking gait database [**?**] (Figure **??**). Ankle movement differed noticeably (r = 0.35, MAE = 5.4°), especially between 40% and 70% of the gait cycle. There was a good agreement for the knee (r = 0.93, MAE = 5.7°) and hip angles (r = 0.97, MAE = 9.0°) despite some notable offset in hip angle. A similar shift occurred in the pelvis ante/retroversion angle (not further analyzed).

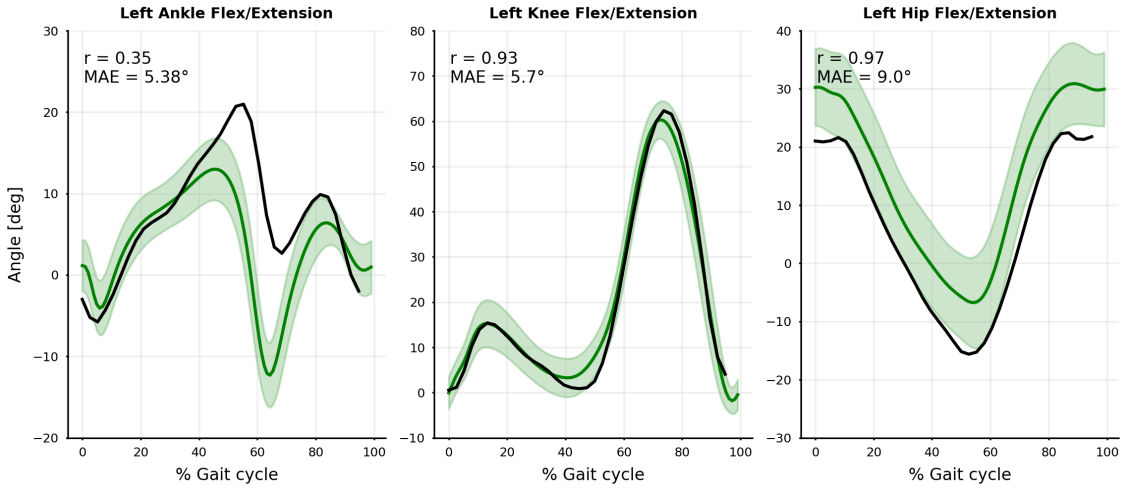


Figure 4.7: Comparison between our markerless results (black line: mean stride-to-stride results for our subject) and the normative marker-based database (green line and area: mean and standard deviation across 14 young, healthy, male subjects). Mean absolute error (MAE) and Pearson correlation coefficient (r) are represented.

When averaged over all joint angles, the stride-to-stride standard deviations lay between 1.7° and 3.2° for all conditions and tasks (Table **??**). The walking task was the most variable, while the cycling one was the least variable; however, in the latter the upper body was static and artificially drew the mean down, which was revealed after removing it from the statistics (Table **??**). In the walking task, the stride-to-stride standard deviation of our individual lower body angles was higher than [**?**], who used a marker-based approach (Table **??**).

There was a good agreement between the Ref condition and the degraded ones, at the same time across all tasks, movement planes, and joints. Mean absolute errors (MAE) averaged over all joint angles (compared to the reference condition—Ref) ranged between 0.35° and 1.7° for all conditions and tasks (Table **??**). Individual angle MAE lay between 0.07° and 5.2° (Figure **??** for the walking task, and Figure **??** and Figure **??** in the Appendix for the running and cycling tasks). However, the cycling task was more challenging: ankle joint angles were especially less robust regarding all considered dependent variables.

Angle results were particularly little affected by the 1 cm calibration error (Cal condition). The standard deviation between cycles (std) virtually did not increase, the average Pearson's r correlation coefficient was 1.00 in all tasks but the cycling one (r = 0.98), and the mean absolute angle error stayed below 0.5° (Table **??**).

The standard deviation in Im and 4c conditions increased by about 10 to 30% as compared to Ref condition, apart from the cycling 4c condition where it increased by 100%. The Pearson's r correlation coefficient was about 0.98 in all tasks but the cycling one. Mean absolute angle errors across all joints lay between 0.9 and 1.8° (Table **??**).

Task	Condition	std (°)	std/stdref	r	MAE (°)
Walking	Ref	2.56	-	-	-
	Im	3.03	1.19	0.97	1.55
	4c	3.24	1.27	0.97	1.50
	Cal	2.60	1.02	1.00	0.35
Running	Ref	2.59	-	-	-
	Im	2.76	1.07	0.99	0.92
	4c	2.79	1.10	0.97	1.60
	Cal	2.54	0.98	1.00	0.47
Cycling	Ref	1.78	-	-	-
	Im	1.89	1.08	0.88	1.72
	4c	3.04	1.93	0.81	1.54
	Cal	1.80	1.02	0.99	0.50
Cycling (lower-body only)	Ref	2.09	-	-	-
	Im	2.41	1.22	0.94	1.69
	4c	3.82	2.31	0.90	1.84
	Cal	2.13	1.03	0.99	0.51

Table 4.3: Summary of angles statistics, averaged for all joints. Each condition is represented: reference condition (Ref), degraded image quality (Im), four cameras instead of eight (4c), degraded calibration (Cal). Comparisons between each Im, 4c, Cal conditions, and Ref are accounted for with standard deviation (std), the standard deviation ratio (std/stdref), the Pearson's correlation coefficient (r), and the mean absolute error (MAE).

Joint	Method	Flexion/Extension	Abduction/Adduction*	Internal/External rotation
Ankle	[?]	2	2.5	-
	Ours	**2.07**	**4.84**	**-**
Knee	[?]	0.7	-	-
	Ours	**4.85**	**-**	**-**
Hip	[?]	1.2	1.8	1.1
	Ours	**2.61**	**1.5**	**3.72**

*Table 4.4: Stride-to-stride standard deviations of lower-body angles. Comparison between our markerless approach, and a marker-based one (averaged over 18 young subjects). * Ankle subtalar angle is assimilated to an abduction/adduction angle.*

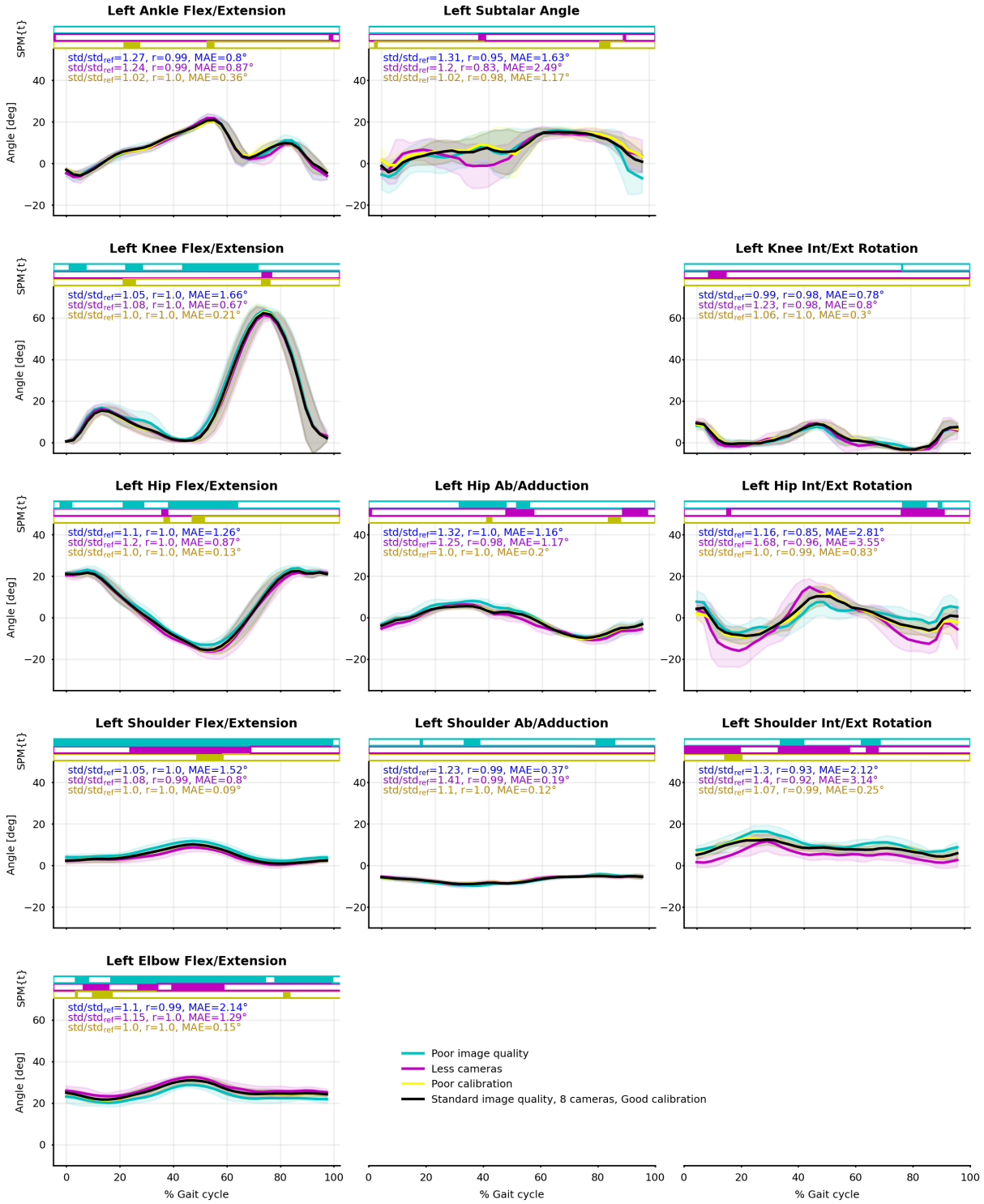


Figure 4.8: Joint angle means (solid line) and standard deviations (shaded area) from the eight captured cycles of walking. Reference condition (Ref) is black; degraded image quality (Im) is blue; four cameras instead of eight (4c) is purple; degraded calibration (Cal) is yellow. Pearson's correlation coefficient (r) and mean absolute error (MAE) between Ref and Im, 4c, Cal are reported. Paired t-tests along time were computed by SPM-1D and are represented as bar plots above the curves: a color rectangle means that there was a cluster of statistically significant differences ($\alpha = 0.05$) at that moment. Running and cycling figures can be found in the Supplementary Material.

Kinematics were more robust in the flexion/extension plane (Table **??**), where std generally did not increase by more than 10% as compared to the Ref condition, and r was mostly equal to 1.00. The difference in robustness was not as obvious in the MAE variable since the range of motion was much smaller in other planes.

Task	**Cond.**	**Flex./Ext.**			**Abd./Add.***			**Int./Ext. rot.**		
		std/std_{ref}	r	MAE (°)	std/std_{ref}	r	MAE (°)	std/std_{ref}	r	MAE (°)
Walking	Im	1.11	1.00	1.48	1.28	0.98	1.05	1.23	0.89	2.47
	4c	1.15	1.00	0.90	1.29	0.93	1.28	1.54	0.94	3.35
	Cal	1.01	1.00	0.19	1.04	0.99	0.50	1.04	0.99	0.54
Running	Im	1.03	1.00	0.98	1.08	0.98	0.48	1.13	0.98	1.41
	4c	1.03	1.00	0.98	1.18	0.93	1.00	1.14	0.97	4.06
	Cal	1.00	1.00	0.30	0.99	0.99	0.53	0.92	1.00	0.80
Cycling	Im	0.99	0.97	1.89	1.27	0.66	1.45	0.99	0.97	1.71
	4c	1.31	0.96	1.43	3.10	0.46	1.76	1.71	0.97	1.47
	Cal	1.00	1.00	0.39	1.06	0.96	0.36	1.02	1.00	1.00

*Table 4.5: Summary of angles statistics in the three rotation planes, averaged over all joints (n = 5, 3, 2 for Flexion/Extension, Abduction/Adduction, and Internal/External Rotation, respectively.). All conditions are represented: degraded image quality (Im), four cameras instead of eight (4c), and degraded calibration (Cal). These conditions were compared to the reference one (Ref) by calculating the ratio of standard deviation (std/stdref), the Pearson's correlation coefficient (r), and the mean absolute error (MAE). *Ankle subtalar angle is assimilated to an abduction/adduction angle.*

The SPM showed some statistical differences along time between the reference condition and the degraded ones; however, they did not happen at any particular moment during the gait or pedaling cycle (Figure **??** for the walking task, and Figure **??** and Figure **??** in the Appendix for the running and cycling tasks).

4.4 Discussion

4.4.1 Relevance, repeatability and robustness

Results of joint angles in the walking condition seemed to be realistic when compared to the normative database. The noticeable discrepancies may be due to a difference between the angle definition of both models, especially in the ankle: the normative database calculated 6 degrees-of-freedom free-body angles between limb segments, while our OpenSim model took bone geometries into consideration. The shift in the hip angle was related to an excessive pelvis anteversion, which may have been induced by the model being underconstrained due to OpenPose dearth of keypoints. This lack of marker constraints could be partly compensated by adding joint constraints, especially around the spine (See next chapter on **??**, combining the [**?**] and the [**?**] models). In any case, the angle results seem to be relevant, but accuracy needs to be further investigated by comparing Pose2Sim markerless method to a reference marker-based one exploiting a similar model definition.

Repeatability was simply assessed by the standard deviation of angle time series across gait cycles, although it also includes the participant's intrinsic variability. However, the subject being young and healthy, his gait and cycling patterns were thought to be consistent. Hence, the results were repeatable as the standard deviation of angles across all strides within all tasks and conditions mostly stayed below 3°. Nevertheless, this standard deviation was still higher than the one previously reported by [**?**] for a young and healthy participant using a marker-based approach. It may be partly caused by our lack of a force plate and our low sampling rate, which did not let us measure a very accurate heel strike event, and thus could have induced some additional variability. In addition, our videos were sampled at 30 Hz instead of hundreds of Hertz as is usual in motion analysis. It is to be noted that high-speed video cameras can sample at a such speed and solve at least a part of this issue. Depending on the sports task investigated, a 3° standard deviation can be more or less of an issue.

Robustness was investigated based on the criteria exposed by [**?**]. First, people could be in the background. Then, several movement tasks have been investigated. Last, the workflow was confronted to three degraded conditions regarding image quality, camera number, and calibration. The results were also robust, since degraded conditions still gave consistent joint angles. A further SPM study showed occasional statistical differences along time, but did not reveal any part of the movement cycle to be more problematic than another. There was no apparent outlier in the 3D angle results, even in the degraded conditions. This was in spite of the presence of other people in the field of view and in spite of the subject walking out of it.

All stages of the workflow contributed to its robustness. First, we never had to deal with limb swapping, unlike other studies [**?**, **?**]. This may be due to our use of the experimental Body_25B OpenPose model, which is claimed to decrease the number of false positives [**?**]. Then, we used Pose2Sim to robustly sort the detected people and to triangulate 2D coordinates. These coordinates were the least precise in the Im condition where the image was strongly blurred and darkened, but Pose2Sim partly solved it by taking the OpenPose likelihood into account and by excluding the most inaccurate cameras: twice as many cameras were excluded as compared to the reference condition, which resulted in a reprojection error only 10% higher.

Finally, we used OpenSim to take advantage of a physically consistent full-body model and of an inverse kinematics optimization algorithm, which allowed us to refine our 3D data coordinates and to obtain 3D joint angles. The Cal condition simulated calibration errors, which bypassed the Pose2Sim safeguards since the OpenPose likelihood was unaffected, and the reprojection error stayed below the threshold. Despite it producing the largest reprojection error, virtually no difference with the reference condition was observed after the OpenSim pass. A 2 px calibration error ($\approx$1 cm) is much worse than the maximum 1 px, or 3 mm usually recommended, but the mean absolute error it induced for us stayed below 0.5°.

Using four cameras rather than eight still gave relevant angle results, but the individual errors of each camera cannot be blunted by the multiplicity of views, especially when faced with occlusion issues such as in the cycling task. The ankle joint angles were especially less robust regarding all considered dependent variables because the feet were more easily occluded by the cycling gear.

Ultimately, all conditions challenged the workflow at different stages, but the results remained stable and very close to those of the reference condition, with an average mean absolute error mostly lower than 1.5°, a correlation coefficient largely above 0.95, and a standard deviation usually increased by less than 20%. This demonstrates the robustness of the presented method in conditions that would have probably caused marker-based approaches to fail. Moreover, our reported angle deviations seem quite reasonable compared to errors in marker-based approaches, which can propagate up to 10° because of inter-operator differences [**?**, **?**], to 3° because of tissue artifacts [**?**, **?**], or to 3° depending on the joint center calculation method [**?**]. Essentially, the findings presented here seem to indicate that the slight decline in repeatability is an acceptable compromise when put into perspective with the increase in robustness, in ease of use, and in new opportunities for analysis of sports performed "in-the-wild".

Nonetheless, it would be interesting to check for even more challenging conditions. We have investigated global Gaussian blur, but not movement blur, such as when the camera shutter speed is not high enough. It would also be interesting to investigate the effect of even larger calibration errors, even less cameras used, different camera placements, different camera types (such as GoPros with strong fisheye distortions), or different framerates or image definitions.

4.4.2 Limits and perspectives

Our use of a virtual scene helped us perfectly control all parameters of the capture. Although this scene is synthetically crafted, it looks similar to a real one. This is not thought to hinder the performance of 2D pose detection since deep-learning models are occasionally trained on synthetic datasets that look empirically less real and still successfully transfer to real-world data (see [**?**, **?**, **?**, **?**, **?**]). However, an image with a better definition and with a framerate over 30 fps may improve results. Furthermore, the 68 video camera capture design uses up very large storage space, and the restricted space offered by the Kinovis room constrained us to capture only a few cycles per capture. This limited us to a low amount of data, with only one subject and about 10 cycles analyzed per task.

We noticed that the cycling task was particularly challenging, probably because of self-occlusions and gear occlusions. Setting some cameras closer to the ankles and in a lower position would have slightly improved this issue. Unlike [**?**], we did not find running to be more challenging for our system than walking. In the context of sports sciences, it would be useful to test other tasks, such as boxing (a typical 3D movement, explored in chapter 6 on **??** [**?**]), flipping (the 2D pose estimator may have trouble with upside-down people), swimming (in an environment with a different refractive index, with splashes around limb extremities), or a sport discipline with unusual outfits and large occlusions (such as motorbiking or fencing.)

Moreover, the [**?**] model used as a basis does not appear to be well suited for road cycling since the spine is designed as a single rigid bone that cannot bend. This results in sharp shifts in hip abduction/adduction and internal/external rotation angles, which are not biomechanically coherent (see Figure **??**). However, the same issue would prevail even with marker-based methods and would only be solved by changing the spine definition in the OpenSim model. The full-body model with a fully articulated spine and ribcage developed by [**?**] has far too many degrees of freedom for our small amount of detected keypoints; however, the lifting full-body (LFB) model validated by [**?**] solved the spine challenge by constraining vertebra movements to each other, without adding any degrees of freedom. Pose2Sim now provides a combination of these two models (see Pose2Sim package) in order to benefit from the most realistic options (assuming a healthy subject), both in the knee and in the spine. This is the one used in the next section on **??**.

For further use in a sports context, it would also be useful to support multi-person analysis, to have a more accurate and extensive 2D pose estimator, to work in real time, and to provide a GUI. All these considerations are discussed in the **??** section of the previous chapter on the Pose2Sim package.

[**?**] proposed a taxonomy of 3D pose estimation algorithms based on accuracy, speed, and robustness. Although more modalities could be tested, our workflow was robust in the range of our tested conditions. Speed was unaddressed, but although real-time analysis seems out of reach, a timely analysis of athletes' movements directly on the sports field appears to be achievable. Yet, ultimately, the accuracy of the workflow must be concurrently validated with a reference marker-based method. This is the topic of the next chapter.

5

Accuracy assessment

Besides robustness, accuracy needs to be addressed. Two-dimensional deep-learning pose estimation algorithms can suffer from biases in joint pose localizations, which are reflected in triangulated coordinates, and then in 3D joint angle estimation. Pose2Sim, our robust markerless kinematics workflow, comes with a physically consistent OpenSim skeletal model, meant to mitigate these errors.

Its accuracy was concurrently validated against a reference marker-based method. Lower-limb joint angles were estimated over three tasks (walking, running, and cycling) performed multiple times by one participant. When averaged over all joint angles, the coefficient of multiple correlation (CMC) remained above 0.9 in the sagittal plane, except for the hip in running, which suffered from a systematic 15° offset (CMC = 0.65), and for the ankle in cycling, which was partially occluded (CMC = 0.75). When averaged over all joint angles and all degrees of freedom, mean errors were 3.0°, 4.1°, and 4.0°, in walking, running, and cycling, respectively; and range of motion errors were 2.7°, 2.3°, and 4.3°, respectively. Given the magnitude of error traditionally reported in joint angles computed from a marker-based optoelectronic system, Pose2Sim is deemed accurate enough for the analysis of lower-body kinematics in walking, cycling, and running.

*This chapter is adapted from the article published in the Sensors: "Pose2Sim: An End-to-End Workflow for 3D Markerless Sports Kinematics—Part 2: Accuracy" [**?**]. See Figure **??** for a visual abstract.*

Contents

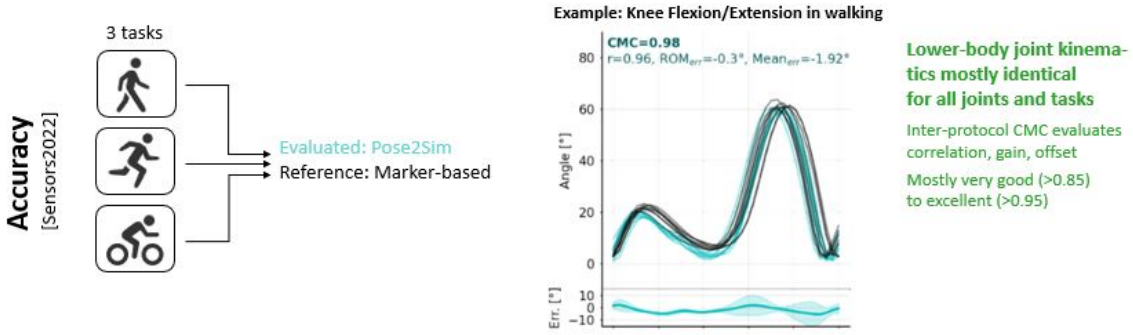


Figure 5.1: Visual abstract for Pose2Sim accuracy assessment [?].

5.1 Introduction

Several markerless technologies are currently being developed for sports motion analysis. We showed that they could be particularly robust (see previous chapter on **??** [**?**]). However, their accuracy has not fully been assessed yet, especially when constrained to a skeletal model. Despite marker-based solutions are not usually adapted to sports analysis in context, they are very accurate and are often considered as a silver-standard, if not the gold-standard in motion analysis. Hence, they can be a good tool to evaluate markerless systems.

A large part of studies investigating 3D joint center estimation choose to triangulate the output of OpenPose [**?**]. Their Mean Per Joint Position Error (MPJPE) usually lies between 30 and 40 mm [**?**, **?**, **?**]. Ankle MPJPEs are within the margin of error of marker-based technologies (1–15 mm), whereas knee and hip MPJPEs are greater (30–50 mm). These errors are systematic and likely due to "ground-truth" images being mislabeled in the training dataset [**?**]. Triangulation from other 2D deep-learning algorithms (such as AlphaPose [**?**] and DeepLabCut [**?**]) have also been compared [**?**]. AlphaPose results are similar to OpenPose's; however, DeepLabCut errors are substantially higher.

Numerous studies have focused on the accuracy of 3D joint center estimation, but far fewer have examined 3D joint angle estimation. D'Antonio et al. computed direct flexion-extension angles for the lower limb from two cameras processed with OpenPose [**?**]. Range of Motion (ROM) errors lay between 2.8° and 14.1°. Wade et al. calculated frontal and sagittal knee and hip angles with OpenPose, AlphaPose, and DeepLabCut [**?**]. They deemed the method accurate enough for assessing step length and velocity, but not for joint angle analysis. AniPose offers a toolkit for triangulating 2D poses from DeepLabCut [**?**]. To our knowledge, it has only been concurrently validated for index finger angles in the sagittal plane, resulting in a root-mean-square error of 7.5 degrees [**?**]. Theia, a commercially available software package for markerless analysis, uses its own patent-protected 2D pose estimator and triangulation procedure, and runs a skeletal model to constrain the results to physically consistent poses and movements [**?**]. Their root-mean-square error (RMSE) compared to a marker-based method ranged between 2.6° and 13.2°.

Pose2Sim (see **??** [**?**]) robustly triangulates OpenPose results [**?**], and computes 3D joint angles using OpenSim scaling and inverse kinematics [**?**, **?**]. We showed that Pose2Sim was robust people in the background, to different sorts of mouvements (walking, running, and cycling), to dark and blurry images (0.5 gamma compression and 5.5 cm Gaussian blur), to 1 cm random calibration errors, and to using as few as four cameras (see previous chapter on **??** [**?**]). Now, the objective of this present study is to concurrently evaluate Pose2Sim's lower-limb 3D accuracy, with a marker-based method, and on the same tasks.

5.2 Methods

5.2.1 Data collection

The same adult male participant as in the previous chapter on **??** (1.89 m, 69 kg) was equipped with 83 reflective markers inspired from the CAST marker set [**?**], composed of 35 anatomical markers, and 12 clusters of 4 markers (Figure **??**). He was asked to perform three tasks: walking, running, and cycling, at a regular pace, back and forth across the capture space, following a regular pulsing sound (see previous chapter for further details). He provided his written consent prior to participating.

All tasks were performed in a room equipped with a green background for optimal segmentation of the subject with respect to the background, and 3D animated mesh extraction using a visual hull approach at each video frame [**?**]. Twenty opto-electronic cameras captured the 3D coordinates of the markers, and 68 video cameras allowed retrieval of 3D textured meshes of the

participant, which we subsequently placed in a virtual environment and filmed from 8 virtual cameras (Figure **??**). This gave us the opportunity to overlay our reflective markers, calculated joint centers, and OpenPose keypoints to the extracted mesh. This was particularly useful to correctly place OpenPose keypoints on the OpenSim model, i.e., with a systematic offset as regards true joint centers (**??**). The acquisition was restricted in terms of 3D volume covered by both systems and data storage, resulting in the analysis of 8, 13, and 13 cycles of walking, running, and cycling, respectively. Once 3D point coordinates were retrieved, both systems underwent processes that were as close to each other as possible: coordinates were sampled at 30 Hz, then they were filtered with a 4th-order 6 Hz low-pass Butterworth filter (which efficiently filtered out noise without underestimating peak values, including in extremities); heel strikes were detected in both cases with the method proposed by [**?**]; stride duration was determined as the inverse of the frequency of the metronome followed by the participant; and inverse kinematics were optimized with the same OpenSim skeletal model.

5.2.2 Markerless analysis

All videos from our virtual cameras were processed by OpenPose (version 1.6), which delivered 2D joint coordinates for each view. We used the OpenPose experimental body_25B model (Figure **??**) with the highest accuracy parameters [**?**]. The Pose2Sim workflow was then used to track the person of interest, robustly triangulate the OpenPose 2D joint coordinates, and filter the resulting 3D coordinates. Then this output was fed to our OpenSim setup to constrain the results to physically consistent kinematics (more details in previous chapter).

Pose2Sim comes with a generic OpenSim skeletal model, that has been slightly improved as regard to the previous chapter. It was adapted from the human gait full-body model [**?**] and the lifting full-body model [**?**]. While the spine of the gait model is represented as a single rigid bone, it is articulated in the lifting model, and each lumbar vertebra is constrained to the next one. This is more accurate for activities for which the spine is bent, such as cycling. However, the knee joint is more accurately defined in the gait model: abduction/adduction and internal/external rotation angles are constrained to the flexion/extension angle, whereas they are simply ignored in the lifting one. This also improves the estimation of knee flexion. All else being equal, as we want our model to be as versatile as possible, we used the spine definition of the lifting model, and the knee definition of the gait model. Since we did not investigate muscle-related issues, they were removed to decrease computation time. Since no keypoint would have accounted for it, wrist flexion and deviation were locked at 0°, and arm pronation/supination was locked at 90°. Conversely, the translation of the pelvis was unlocked, in addition to the subtalar angle; and hip flexion was limited to 150° instead of 120° (which was not enough for the pedaling task). With regards to the previous chapter, marker placement was also improved in the OpenSim model. The average systematic offset between OpenPose-triangulated keypoints and MoCap-calculated joint centers was measured on our 3D overlay view (Figure **??**), and was taken into account when manually placing OpenPose keypoints onto the OpenSim unscaled model.

OpenSim (version 4.2) was used to scale the model to the participant on a T-pose, and then inverse kinematics was performed. Scale factors were computed with measurement-based scaling, i.e., by computing the ratio of distances between keypoints on the model, and experimental keypoints provided by the coordinates file of triangulated OpenPose data. Static pose weights were all set to 1, apart from Nose and Head keypoints which were set to 0.1, and Shoulder and Hip keypoints were set to 2. The participant was standing upright with feet flat during his T-pose, so we set a weight of 1 for a zero angle in pelvis list, pelvis tilt, L5-S1 flexion, and ankle angles. The offset in machine-learning-based joint center estimations has been shown to be systematic and not dependent on the subject [**?**] (nor on the operator); hence, once this bias has been taken into account in the generic model, the markers' adjustment step is unnecessary. Keypoint weight markers for inverse kinematics were the same as for scaling.

5.2.3 Marker-based analysis

The captured markers were automatically identified with an AIM procedure within the Qualisys Track Manager software (version 2019.1). Joint centers were then calculated. The centers of ankles, knees, wrists, and elbows were defined as the midpoints between the malleoli/epicondyles/styloids since it has been shown that when executed on a lean participant, functional methods do not improve the reliability of the kinematics of running [**?**]. Hip joint center was defined with a functional method [**?**]. The OpenSim model used for marker-based scaling and inverse kinematics was the same as the Pose2Sim model. Scale factors were computed similarly, but with marker data rather than with OpenPose keypoints. Weights proposed by the inverse kinematics solver of OpenSim were set to 5 for joint centers, to 1 for cluster markers, and to 2 for other anatomical markers. Inverse kinematics was processed with the same marker weights.

5.2.4 Statistical analysis

Since the participant did not report any locomotion impairment and the captured movements were mostly symmetrical, we only analyzed the right side. Our study focuses on the lower limb, but results for upper limb and sacro-lumbar joints are detailed in the Appendix B for information (Figure **??** and Figure **??** for walking, Figure **??** and Figure **??** for running, and Figure **??** and Figure **??** for cycling). The analyzed angles were ankle flexion/extension, subtalar angle, knee flexion/extension, and hip flexion/extension, abduction/adduction, and internal/external rotation provided by the OpenSim inverse kinematics procedure.

First, Pose2Sim scale factors were compared to marker-based ones, and RMS errors were reported and compared to OpenSim's best practice rules. Then, the overall similarity of paired angle waveforms was assessed with a special formulation of the coefficient of multiple correlation (CMC), specifically designed to compare different protocols or measurement systems [**?**]. The CMC gives a single result taking into account differences in correlation, gain, and offset. It reaches 1 if the curves are perfectly overlapped, and drops to zero if the curves are very dissimilar, or even to complex values (reported as "nan" hereafter). This is, for example, the case if the mean inter-protocol offset (averaged over time and trials) exceeds the grand mean ROM, which results in taking the square root of a negative number. CMC values are deemed good if between 0.75 and 0.84, very good if between 0.85 and 0.94, and excellent if above 0.95 [**?**]. The CMC results were then broken down to take an in-depth look into correlation, gain, and offset, separately. The strength of the linear relationship between kinematic analysis systems was assessed with the Pearson's r correlation coefficient. Gain was evaluated by computing the paired ROM differences. Once normality of the ROM errors was checked with a Shapiro–Wilk test [**?**], we computed paired t-tests to verify whether the error was significant [**?**]. Mean inter-protocol offset angle, hereafter called mean error, was one of the outputs of the subsequent Bland–Altman analysis [**?**, **?**]. Once normality of the paired means of the angle differences was verified, we determined if the mean markerless angles were significantly different to the mean marker-based ones.

The Bland–Altman analysis gives some more information about the agreement between the considered markerless and marker-based systems [**?**, **?**]. It consists of plotting the difference between the values given by both systems against their mean, for all angular points at all time instances. Limits of agreement were defined as the interval within which 95% of data will be found, i.e., between mean difference ±1.96 standard deviation, provided that the differences follow a normal distribution. Bland–Altman plots also help to identify the potential presence of heteroscedasticity, i.e., the fact that the spread of the error may depend on the angle magnitude [**?**].

Finally, root-mean-square errors (RMSE), mean errors ($Mean_{err}$), and ROM errors (ROM_{err}) were computed for the walking task to enable comparison with previously published metrics obtained using Theia3D, a commercial markerless solution [**?**], and Xsens [**?**], a commercial system based on IMUs. Theia3D's ROM results were approximated from reported graphics along the flexion/extension degree of freedom.

5.3 Results

5.3.1 Concurrent validation

Inverse kinematics is successful when OpenSim's global optimizer keeps the model markers close to experimental markers, i.e., when RMSE is less than 2–4 cm according to OpenSim's best practices. This was the case for both systems (Table **??**). However, RMSE was particularly higher in cycling than in walking or running.

Task	Marker-based RMSE (cm)	Markerless RMSE (cm)
Walking	1.1—1.2	1.4—1.9
Running	1.5—1.7	1.5—2.1
Cycling	≈ 3.5	3.0—3.6

Table 5.1: RMSE between experimental and theoretical markers during OpenSim inverse kinematics, for both marker-based and Pose2Sim models.

CMC assesses waveform similarities between Pose2Sim and the marker-based reference method, by jointly evaluating correlation, gain, and offset. It was mostly very good (CMC > 0.85) to excellent (CMC > 0.95) in all tasks, all degrees of freedom, and all lower-body joints (Table **??**, Figure **??**, Figure **??**, and Figure **??**). This was especially the case along the flexion/extension degree of freedom, except for angles of the hip in running and of the ankle in cycling, for which CMC results suffered from an offset compared to the marker-based method. Hip abduction/adduction and internal/external rotation waveforms were not in good agreement (CMC < 0.75), except for the hip internal/external rotation angles in running. In cycling, all non-sagittal angles had complex CMCs, which means that no agreement was found at all.

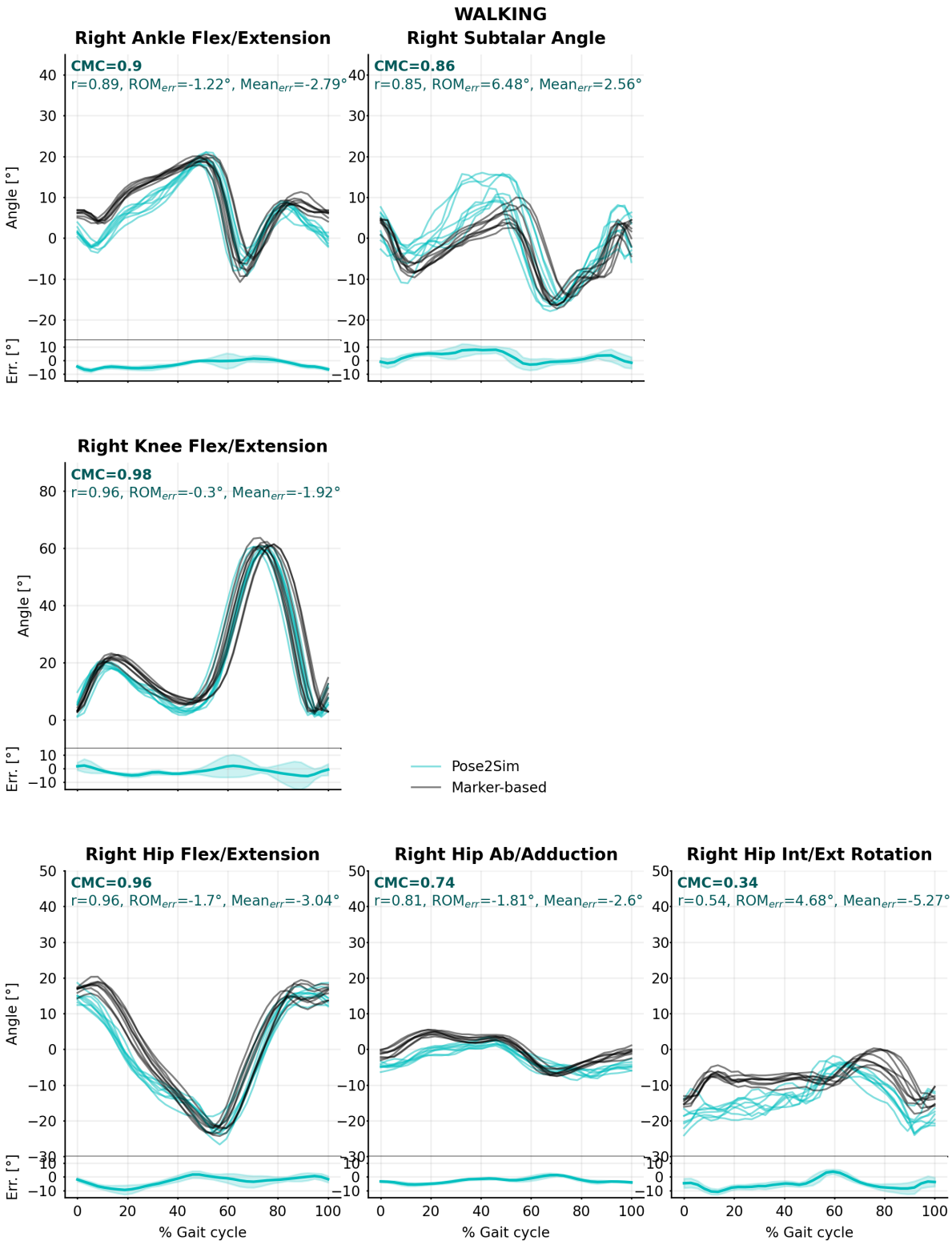


Figure 5.2: Pose2Sim (cyan) and marker-based (black) lower-body joint angles for the walking task. Coefficient of multiple correlation (CMC) is indicated, and broken down into, respectively, Pearson's coefficient (r) for correlation assessment, range of motion errors (ROM_{err}) for gain, and overall mean errors ($Mean_{err}$) for offset. Mean error and standard deviations are also represented at the bottom of the graphics. See Appendix B for running and cycling results, and for sacro-lumbar and upper-body results.

Task	Joint	Flex./Ext.				Abd./Add.[1]				Int./Ext. rot.			
		CMC	r	ROM_{err}	$Mean_{err}$	CMC	r	ROM_{err}	$Mean_{err}$	CMC	r	ROM_{err}	$Mean_{err}$
Walking	Ankle	0.90	0.89	-1.22	-2.79*	0.86	0.85	6.48*	2.56*	—	—	—	—
	Knee	0.98	0.96	-0.30	-1.92*	—	—	—	—	—	—	—	—
	Hip	0.96	0.96	-1.70	-3.04*	0.74	0.81	-1.81*	-2.6*	0.34	0.54	4.68*	-5.27*
Running	Ankle	0.99	0.99	-2.9*	-0.71*	0.97	0.96	2.20	0.96	—	—	—	—
	Knee	1.00	1.00	0.04	-0.65*	—	—	—	—	—	—	—	—
	Hip	0.65	0.95	4.01*	15.18*	0.37	0.65	-3.94*	-3.76*	0.93	0.95	1.25	-3.49*
Cycling	Ankle	0.75	0.85	1.93*	-6.73	nan	-0.32	10.27*	1.59*	—	—	—	—
	Knee	1.00	1.00	-2.94*	2.12*	—	—	—	—	—	—	—	—
	Hip	0.92	0.97	-5.91*	6.12*	nan	0.14	1.72*	-5.62*	nan	-0.07	3.07*	2.11*

*Table 5.2: Summary of comparisons between Pose2Sim and marker-based angle waveforms. A specific formulation of the Coefficient of Multiple Correlation (CMC) was used, specifically designed to compare different protocols or measurement systems. CMC jointly evaluates correlation, gain, and offset, which were respectively assessed with r-Pearson's coefficient, range of motion errors (ROM_{err}), and mean errors ($Mean_{err}$). *Significant at 5% level. [1]Despite ankle subtalar angle combines abduction/adduction and internal/external rotation, it is thereafter reported in the abduction/adduction column.*

Pearson's r correlation coefficient results were close to the CMC ones, albeit they became very good to excellent in the two angles that were affected by an offset. When averaged over all joint angles, errors in the range of motion (ROM_{err}) were 2.7° (sd = 2.1°), 2.3° (sd = 1.1°), and 4.3° (sd = 2.5°) in walking, running, and cycling, respectively. Along the flexion/extension degree of freedom, they were below 2°, 4°, and 6°, in walking, running, and cycling, respectively. Along the internal/external rotation degree of freedom, they stayed below 5°; however, they reached up to 10° along the abduction/adduction degree of freedom. Average mean angle errors ($Mean_{err}$) were 3.0° (sd = 1.0°), 4.1° (sd = 1.6°), and 4.0° (sd = 0.59°), in walking, running, and cycling, respectively. In walking and running, mean errors remained under 5.3° in all degrees of freedom, apart from the hip flexion/extension angle in running, which was offset by 15°. Although they were noticeably larger, mean errors were always under 7° in cycling (Table **??** and Table **??**, Figure **??**, Figure **??**, and Figure **??**)

Task	Joint	Flex./Ext.		Abd./Add.*		Int./Ext. rot.	
		$Mean_{err}$ (°)	95% LoA (°)	$Mean_{err}$ (°)	95% LoA (°)	$Mean_{err}$ (°)	95% LoA (°)
Walking	Ankle	-2.79	[-9.4,3.8]	2.56	[-6.9,12]	—	—
	Knee	-1.92	[-12,8.4]	—	—	—	—
	Hip	-3.04	[-11,5.2]	-2.6	[-6.9,1.7]	-5.27	[-14,3.5]
Running	Ankle	-0.71	[-4.5,3.0]	0.96	[-5.3,7.2]	—	—
	Knee	-0.65	[-2.9,1.6]	—	—	—	—
	Hip	15.18	[5.5,25]	-3.76	[-9.2,1.6]	-3.49	[-9.3,2.4]
Cycling	Ankle	-6.73	[-16,2.1]	1.59	[-9.8,13]	—	—
	Knee	2.12	[-1.1,5.3]	—	—	—	—
	Hip	6.12	[-1.7,14]	-5.62	[-10,-1.1]	2.11	[-4.5,8.7]

*Table 5.3: Bland–Altman analysis results of 3D angle errors between Pose2Sim analysis and the reference marker-based one. Mean errors ($Mean_{err}$) and 95% limits of agreement (LoA) are represented. *Although ankle subtalar angle combines abduction/adduction and internal/external rotation, it is hereafter reported in the abduction/adduction column.*

Limits of Agreement (LoA) values were relatively evenly and randomly distributed among all tasks, degrees of freedom, and joints, averaging to an interval of 15° within which 95% of the errors would lie (Table **??**, Figure **??**, Figure **??**, and Figure **??**). Due to the limited range of motion of sacro-lumbar and upper-body angles, limits of agreement were smaller in these joint angles (Figure **??**, Figure **??**, and Figure **??**). Angle magnitude did not have an influence on the spread of errors (hence the data are homoscedastic), except for the cycling task for ankle angles and flexion/extension hip angles.

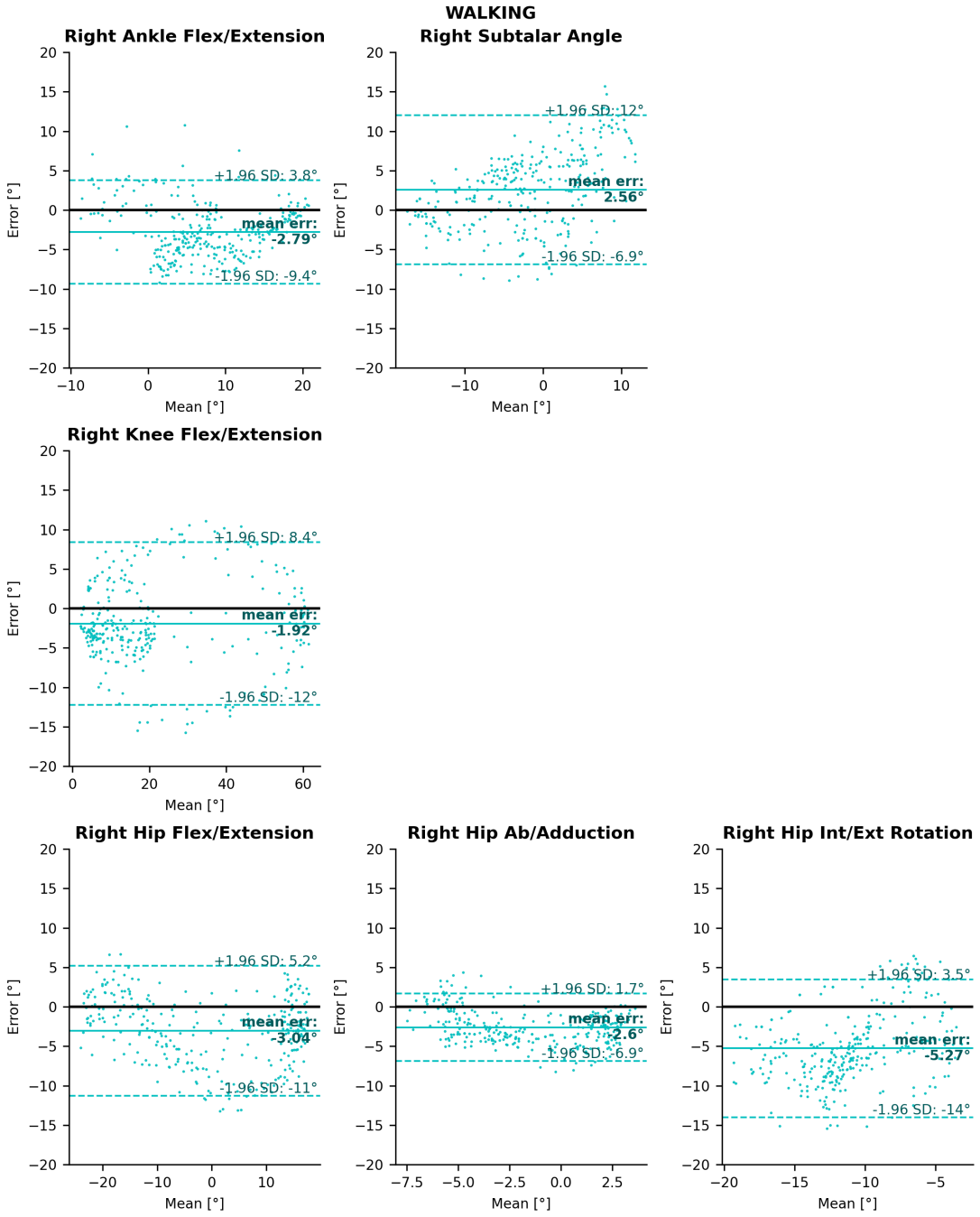


Figure 5.3: Bland–Altman analysis of lower-body joint angle errors for the walking task. Mean bias is represented as a horizontal solid, bold line, and 95% limits of agreement are represented as dotted lines. See Appendix B for running and cycling results, and for sacro-lumbar and upper-body results.

5.3.2 Comparison with other systems

The RMSE reported by Theia3D [**?**] was, on average, 1.5° higher than that of Pose2Sim, and its ROM errors were consistently higher, at least along the flexion/extension degree of freedom. However, the study using Xsens [**?**] reported mean errors 0.3° lower on average, and ROM errors 1.0° lower on average (Table **??**).

Error metric	Joint	Flex./Ext.			Abd./Add.			Int./Ext. rot.		
		Ours	Theia	Xsens	**Ours**	Theia	Xsens	**Ours**	Theia	Xsens
RMSE (°)	Ankle	4	6.7	—	5.1	8	—	—	—	—
	Knee	5.1	3.3	—	—	—	—	—	—	—
	Hip	5.6	11	—	3.1	2.6	—	6.6	6.9	—
$Mean_{err}$ (°)	Ankle	2.79	—	2.15	2.56	—	1.81	—	—	—
	Knee	1.92	—	1.87	—	—	—	—	—	—
	Hip	3.04	—	2.47	2.6	—	4.83	5.27	—	3.02
ROM_{err} (°)	Ankle	-1.22	≈ -10	0.4	6.48	—	1.38	—	—	—
	Knee	-0.3	≈ -1	0.8	—	—	—	—	—	—
	Hip	-1.7	≈ -10	2.42	-1.81	—	5.37	4.68	—	0.04

*Table 5.4: Pose2Sim results compared to Theia3D [**?**] and to Xsens [**?**] in the walking task. Root-mean-square error (RMSE), mean error (Meanerr), and range of motion (ROM) are examined. Theia3D's ROM results were approximated from reported graphics along the flexion/extension degree of freedom. Both studies to which we compared our results involved a different setup (participants, cameras, protocol, etc.), therefore the differences cannot be totally attributed to the different technologies (markerless or IMU) nor to the different algorithm (Pose2Sim or Theia3D). * Although ankle subtalar angle combines abduction/adduction and internal/external rotation, it is hereafter reported in the abduction/adduction column.*

5.4 Discussion

5.4.1 Strengths of Pose2Sim and of markerless kinematic

Pose2Sim offers a way to perform a markerless kinematic analysis from multiple calibrated views, taking OpenPose results as inputs, and giving biomechanically oriented results via OpenSim. Both OpenPose and OpenSim are open-source and among the most widespread and renowned tools in their respective fields. We compared Pose2Sim lower-body results to those of a reference marker-based method, over three tasks performed by one participant: walking, running, and cycling. Both protocols were as similar as possible, and used the same constrained skeletal model in order to ensure that there was no discrepancy in results caused by different definitions of anatomical frames [**?**]. Pose2Sim kinematic waveforms were very similar to marker-based ones, especially in the sagittal plane. One exception to this observation was the hip angle in running, which suffered from a 15° offset due to the dearth of keypoints in this area. This led the optimization procedure to admit two solutions for the spine curvature, both mathematically and kinematically correct: one with a lordotic posture, and the other with a kyphotic posture. There was also less agreement for ankle angles in cycling, most likely because for both Pose2Sim and marker-based kinematics, keypoint/marker detections suffered from occlusions from the bike. This is corroborated by the higher RMSE between experimental and theoretical markers observed in cycling (Table **??**). The similarity of waveforms among both protocols was assessed with the coefficient of multiple correlation

(CMC) [**?**], which takes into account the concurrent effects of correlation, gain, and offset. When averaged over all lower-limb joints and all degrees of freedom, mean errors amounted to 3.0°, 4.1°, and 4.1° in walking, running, and cycling, respectively, and range of motion errors were equal to 2°, 2.3°, and 4.3°. It should be noted that, unlike ours, Theia3D [**?**] and Xsens [**?**] studies to which we compared our results involved several subjects (30 and 10, respectively.) Our study recorded with eight virtual cameras, 1 MP definition, 30 Hz framerate, and perfect calibration, whereas the Theia system recorded with eight cameras, 3 MP definition, 85 Hz, with a marker-based calibration. Hence, the comparison between accuracies of Theia3D, Xsens and Pose2Sim are given for an overview of their order of magnitude, not as a claim for exact comparison. This study focused on lower-body kinematics, however we also reported upper-body and sacro-lumbar kinematics in Appendix B for reference. It may be noted that differences between the two approaches were larger than for the lower body, and especially for the sacro-lumbar flexion.

This shows that a carefully designed skeletal model, when correctly scaled and constrained, can lead to accurate results from a markerless approach, despite poorly labeled joint centers [**?**, **?**], and despite a low number of detected keypoints. Indeed, it has been shown that the triangulation of deep-learning-based pose estimation methods produces systematic errors up to 50 mm in 3D knee and hip joint center coordinates [**?**]. Without the use of a skeletal model, flexion/extension lower-body angle errors in cycling have been demonstrated to be as large as 3–12° [**?**]. Moreover, Pose2Sim still gave relevant results when using the coordinates of only 21 triangulated keypoints coordinates (after exclusion of eye and ear keypoints). This is in line with conclusions that were previously made for marker-based approaches, implying that constrained kinematic models are resilient to marker placement and quantity [**?**].

The setup of Pose2Sim can be installed anywhere, i.e., directly on-site rather than in a laboratory setting. No particular attention has to be devoted to the background color, to the participant's clothing, nor to the luminance of the recording area. No apparatus interferes with the athlete's movement, who can fully concentrate on their performance. This is of crucial importance in the context of sports analysis. Results are not operator or subject dependent, which makes labeling, scaling, and inverse kinematics both easy and robust. It is to be noted, however, that it does not leave room for adjustment if it is needed to better monitor a specific body part. However, the operator or scientist has access to fine control on most parameters at each step of the analysis: the deep-learning 2D pose estimation model can be changed; tracking, triangulation, and filtering parameters can be adjusted; and the OpenSim model, scaling, and inverse kinematics can be entirely controlled.

5.4.2 Limits and perspectives

Our study still has potential limitations, starting with those stated in the previous chapter on **??**: we used a virtual scene, and captured a limited amount of data (only 8–13 cycles per task were captured, performed by one participant).

Moreover, both systems were neither cross-calibrated nor synchronized by hardware. Concerning calibration,we were only looking for relative joint angles, thus sharing common coordinates was not important. Concerning the second issue, we visually synchronized the data to the frame on a sharp movement. However, our framerate was only 30 fps, which means that there could be up to a half frame (1/15 seconds) of delay. Moreover, gait events were calculated with a kinematic approach [**?**] rather than detected with a force plate. And yet, all of these limits would have likely deteriorated our results rather than artificially improved them.

On the other hand, movements were captured at 30 Hz. Given the relatively slow and steady movements we analyzed, we believe that this framerate did not impact our results, although both marker-based and markerless kinematics would beneficiate from a higher sample frequency on more demanding activities. Note that Pose2Sim can operate at any framerate, and this limitation is only due to the settings of the video acquisition system. Although results cannot be overly

generalized to other sports movements, we assume that conclusions would hold for other healthy subjects, first because the OpenPose training was done on numerous participants having different gender, race, body shape, and outfit [**?**, **?**, **?**]; second, because deep-learning-based pose estimation algorithms are not subject to inter-operator errors nor to soft-tissue artifacts; and, third, because the OpenSim kinematic model is scaled to the participant's anthropometry. Nonetheless, it would be worth assessing its accuracy on more challenging sports and with multiple subjects. Moreover, we used perfect virtual cameras instead of real ones. Real cameras could have induced errors due to motion blur, large distortions, or calibration errors. Our previous study, however, showed that the system was very robust to these issues, including with as little as four cameras, at least with movements such as walking, running, and cycling on an ergometer (see previous chapter on **??** [**?**]). It may be interesting to try Pose2Sim with light and versatile action cameras such as GoPros, calibrated with a checkerboard. The accuracy of these cameras has already been explored on marker-based data. Although the maximum point coordinate error was about 10 times as large as that with a motion capture system (2.47 versus 0.21 mm), knee joint angles were highly correlated (joint coordinates error below 2.5°) [**?**]. This is the topic of the next chapter, about GoPros used for boxing analysis.

OpenPose keypoint localization suffers from systematic offsets when compared to actual joint center positions [**?**]. This has been taken into account on a static pose in the OpenSim unscaled model, by shifting OpenPose keypoint placements with regard to marker-based joint centers. This was done manually, but precisely, thanks to our overlayed view (Figure **??**). The OpenSim model was then scaled to the participant's anatomy without the use of any MoCap procedure. However, OpenPose's offset may not be the same when a limb is extended as when it is bent, which may influence kinematic results on extreme poses. Hence, using a pose estimation model free from systematic biases on all ranges of motion would improve kinematic accuracy, even if applying a constrained skeletal model already largely reduces the detrimental impact of low-quality 2D joint center estimations. See **??** on Pose2Sim for more details on perspectives.

[**?**] proposed a taxonomy of 3D pose estimation algorithms based on accuracy, robustness, and speed. Accuracy was assessed in this chapter, and robustness was investigated in the previous one; however, speed has not yet been tested. Yet, a timely feedback is crucial for coaches and athletes to be able to use a motion analysis software. The bottleneck for computational costs, here, is by far the pose estimation system, but some neural networks are tackling this issue [**?**, **?**].

Deep-learning-based human pose estimation is making considerable and consistent progress. It is becoming more accurate, more robust, faster, and simpler to use, approaching Atha's 1984 definition of an ideal motion analysis system [**?**]. Pose2Sim takes advantage of these advances, and mitigates the remaining errors by constraining these outputs to obtain physically consistent kinematics.

6

Application to boxing, using action cameras

Providing Key Performance Indicators (KPIs) to coaches allows them to assess performance, and to tailor training to each athlete's specificities. Boxing is a challenging sport for motion analysis, because movements are fast, 3 dimensional, and involve the whole body. Moreover, it is usually not possible to equip a boxer with markers, lighting conditions and large capture volumes make the classic marker-based difficult, research-grade cameras are too cumbersome to set up in a timely way and lower-end cameras can't be synchronized by hardware, nor with a flash in the context of a competition.

The objective of the study was to verify whether it is possible to accurately measure Key Performance Indicators (KPIs) in boxing, with a markerless protocol and under suboptimal conditions. This was concurrently validated with a marker-based protocol. A secondary goal was to compare the impact on result quality of a markerless analysis with post-calibration and post-synchronization, to the impact of choosing a different 2D pose estimation model. We conclude that KPIs are remarkably well evaluated in all conditions. Using a different 2D pose estimation model had a similar, but mild impact, on results, but combining both factors lead to more imprecision.

*This chapter is a more detailed version of the poster presented at the congress of the European College of Sport Science (ECSS): "A 3D markerless protocol with action cameras – Key performance indicators in boxing" [**?**]. See Figure **??** for a visual abstract.*

Contents

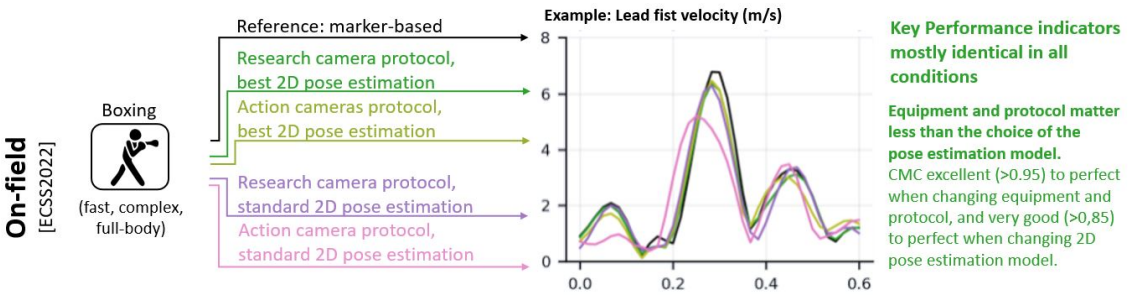


Figure 6.1: Visual abstract for the assessment of KPIs in boxing with Pose2Sim [?]. Research camera protocol: Qualisys cameras, calibration with markers, hardware synchronization.
Action camera protocol: GoPro cameras, calibration on boxing ring dimensions, synchronization on 2D keypoint speeds.

6.1 Introduction

6.1.1 Limits of research-grade systems in competitions

When fine kinematic analysis is needed, the de facto approach is marker-based motion capture. However, we have seen that it is not appropriate in sports, and markerless methods are favored.

Sports motion analysis involves other constraints, such as the capture system, which needs to be discrete and installed swiftly. As a consequence, research grade video systems such as Qualisys are not appropriate: they involve setting up large cameras and stands, with power and synchronization cables, which are obstructing and prevent cameras from being spread too far apart. These systems require at least two trained operators to set them up and to adjust their parameters. They are also very expensive, while their framerate and resolution are not very good. On the other end, consumer-grade action cameras such as GoPros are very small, don't necessitate any cabling, and the operator has nothing to do but pressing the recording button. They are also cheap, and offer remarkably high framerate, resolution, and image quality. However, their battery does not last more than an hour, they don't come with calibration, and until recently, no synchronization solution existed. There is also no centralized visual feedback on the recording.

When outdoors, in direct sunlight and with large capture volumes, calibration of video cameras becomes delicate, too: markers on a calibration wand are inconstantly detected. No reliable solution currently exists, including with active LED markers. Checkerboard calibration is almost equally problematic, as corners are not always well detected unless the board is too large to conveniently carry around. Other solutions exist, such as calibrating intrinsic parameters with a checkerboard [**?**], and extrinsic parameters with manually clicked and semi-automatically tracked points on a wand [**?**, **?**]. Alternatively, it is possible to calibrate extrinsics on any object of known dimensions [**?**], be it a human being [**?**].

Lastly, synchronization can also be a problem. Using a trigger signal is the most accurate approach, but again, it generally involves using wired cameras. Other methods rely on using a flash, or an audio clap [**?**], but they are not possible in the context of a competition. Moreover, as the sound does not travel instantaneously, the synchronization will not be very accurate on a large capture space if the distance from camera to sound source is not taken into account [**?**]. For example, a 20 meters difference would lead to a 60 ms shift, which represents 7 frames at 120 fps. Identifying the instant of a sharp movement is sometimes used as a time reference, but it is generally not accurate enough for a synchronization to the frame, especially if the framerate is high. Other procedures can be explored, such as WiFi synchronization [**?**] or Bluetooth synchronization [**?**], but they usually require using external devices. GPS synchronization is compatible with GoPro 9 and plus, but signal may not be available everywhere, typically in some sports halls [**?**]. Another way to synchronize data streams is to cross-correlate them, and to infer the delay between them from the time of maximum correlation [**?**].

6.1.2 Key Performance Indicators in boxing

Key Performance Indicators (KPIs) are a set of variables which need to be measured in priority, in order to assess performance, or to evaluate main areas where work is needed. Although they are more known in the fields of management or of the industry, identifying them is also important in sports [**?**, **?**]: they allow coaches to assess performance, and to tailor training to each athlete's specificities. Determining them would typically be done in two steps. First, through discussions with coaches, who usually have a subjective, but nevertheless very fine and comprehensive understanding of their sport. Then, the most relevant of these variables can be selected, for example by Principal Component Analysis (PCA) [**?**]. PCA allows for grouping the variables that explain most of the performance, and excluding the others [**?**].

However, this approach has the disadvantage that prior to pursuing any further statistical analysis, all the potential variables first need to be extensively recorded. Only then, the most relevant

ones can be determined. Moreover, it might lead to the selection of indicators which can be hard to retrieve, or be not intuitively meaningful as they potentially consist in the linear combination of some seemingly unrelated variables. In this case, one can choose the performance indicator most associated with each principal component [**?**]. Other methods exist, such as exclusive expert coach opinion, hierarchical models obtained by breaking down individual techniques, or notational analysis based on scoring events and actions along a competition, or other inferential statistic methods such as regression analysis, and more [**?**, **?**].

KPIs must be meaningful and specific to each sport discipline, and it should be possible to capture them in a context as close as possible to the competition one. Boxing is one of the disciplines involved in the PerfAnalytics project, and it covers a wide range of movements, which makes its study relevant. KPIs in boxing can be separated into several categories: action annotations (such as scored points, number of recorded jabs punches or dodges by leaning backwards [**?**]), anthropometric KPIs (such as arm length and muscle percentage [**?**]), physiological (such as velocity at maximal oxygen uptake and anaerobic power [**?**]), or biomechanical (such as ground reaction force of the rear leg before execution of a cross punch, activation of the latissiums dorsi during the hook punch, or extension of the elbow at the end of the execution of the jab [**?**]).

Among all these KPIs, we focus on a subset of the biomechanic ones, namely kinematics. Ultimately, a decisive aspect in a punch is its speed. First, a fast punch is difficult for the opponent to dodge. Then, speed multiplied by force is equal to punch power. In addition, punch force is also correlated to hand velocity [**?**]. However, speed is not generated the same way in jabs as in hooks, since one is a mostly translational movement whereas the other is mostly rotational. [**?**] broke down the phases of both techniques. Among other body motions, the jab first involves translating the lead foot toward the target, then the pelvis and the torso, while the lead arm flexes at the elbow to store elastic energy prior to throwing the punch. Finally, the elbow extends, until the fist is brought into contact with the target. Regarding the rear hook, it starts with flexing the rear knee, while the pelvis and torso rotate horizontally away from the target. Then the motion is reversed, as the knee extends and the pelvis and torso rotate toward the target. Lastly, the attacking arm abducts at the shoulder until the fist reaches the target.

As a consequence, it appears that lead foot translation, pelvis translation, lead elbow extension, and lead fist velocity would consist in good KPI variables for the jab. Similarly, rear knee flexion, pelvis rotation, rear shoulder extension, and rear fist velocity would be valuable to characterize the rear hook.

6.1.3 Objectives

Evaluating kinematic KPIs in boxing is inconvenient with marker-based techniques, and challenging with markerless ones. Indeed, boxing movements are fast, 3 dimensional, and involve the whole body. Moreover, addressing motion capture in an ecologically valid context involves dealing with constraints on the protocol, and requires carefully thinking about the hardware, the calibration method, and the synchronization procedure. These three elements determine 3D reconstruction. However, before this stage comes the 2D pose estimation.

Numerous solutions exist in this regard. AlphaPose [**?**] and OpenPose [**?**] are both the most common and the most accurate [**?**, **?**]. OpenPose is even more widely used than AlphaPose, hence this is the one we choose to focus on. Its standard model is body_25, however the body_25B subset of the single-network whole-body pose estimation [**?**] also provides 25 points, is as fast as the standard one, and is claimed to be more accurate and to reduce false positives [**?**, **?**].

The objective of this study is to evaluate how accurately KPIs can be retrieved in real-life sports conditions, concurrently with a marker-based analysis. Results will be evaluated for a research-grade markerless system, for a consumer-grade markerless one with post-calibration and post-synchronization, as well as with a slightly less accurate pose estimation model in both previous conditions. A subsidiary question is whether the 3D reconstruction procedure has more or less

impact on accuracy than the choice of a 2D pose estimation model.

6.2 Methods

6.2.1 Participants and protocol

3 adult elite boxers were selected, from different weight categories and with different morphologies: one right-handed female, one right-handed male, and one left-handed male. Details are provided in Table **??**. They all provided informed written consent. Participants were equipped with 44 reflective markers, placed by one single operator, following the recommendations of the International Society of Biomechanics (ISB) [**?**, **?**]. Marker set and marker placement are detailed Figure **??**.

A professional coach instructed the athletes to perform a sequence of shadow-boxing 6 times, consisting in a jab, a high rear hook, and a low rear hook. They executed these movements in their usual training center, on a competition boxing ring.

Participant	Gender	Handedness	Age (years)	Height (m)	Weight (kg)
1	Male	Right-handed	20	1.72	54
2	Female	Right-handed	19	1.63	59
3	Male	Left-handed	18	1.90	78

Table 6.1: Demographic details on the participants.

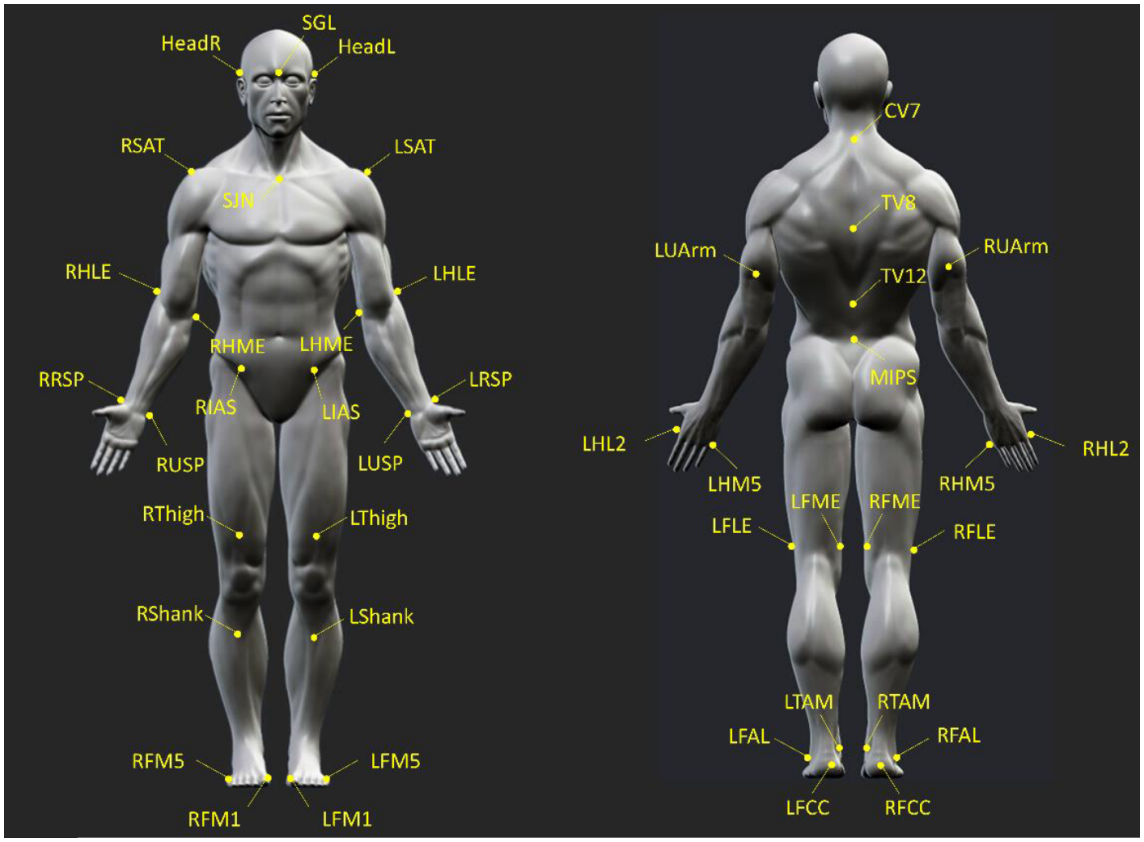


*Figure 6.2: Marker location on the body [**?**].*

The boxing sequences were recorded by 12 opto-electronic Qualisys cameras (7 Miqus M3, resolution 2 MP, framerate 300 fps; and 5 Arqus A5, resolution 5 MP, framerate 300 fps), 8 video Qualisys cameras (Miqus videos, resolution 2 MP, framerate 60 fps), and 8 GoPros (4 GoPro 7 and 4 GoPro 8, 2 MP, 120 fps, linear field of view). GoPro videos were down-sampled to 60 fps before pursuing further analysis. Marker-based and markerless Qualisys cameras were placed next to each other as pairs, except for the 2 surplus optoelectronic cameras. Aperture, focus, shutter speed, and other parameters were carefully adjusted by hand. Both data streams were recorded within the Qualysis QTM software, which allowed for calibration and synchronization, and for a common frame of reference. GoPro cameras did not need any adjustment prior to recording. However, calibration and synchronization were performed after the capture, following the methods proposed in the next two sections.

Marker-based 3D coordinates were calculated with the Qualisys QTM software. These marker data were augmented with joint center coordinates: shoulder centers were defined based on the regression equations adopted from [**?**], and elbow, wrist, knee, and ankle joints were defined as the midpoints between epicondyle markers [**?**]. Markerless videos from GoPro and Qualisys cameras were processed by OpenPose v1.6 [**?**], both with body_25 and body_25B models. Once calibration and camera synchronization were solved (see next two sections), tracking, triangulation, and filtering were done with Pose2Sim [**?**]. Both marker-based and markerless 3D coordinates were processed by a 10 Hz, 4th order low-pass Butterworth filter [**?**], which was deemed appropriate to filter out noise without missing peak values. Inverse kinematics of both were processed with the same OpenSim model [**?**], in order to not interfere with the other protocol variations which were actually investigated.

6.2.2 Post-calibration on ring dimensions

Unlike with Qualisys optoelectronic and video cameras, there is no live calibration procedure for GoPros. Intrinsic parameters were computed priorly by filming a checkerboard of known dimensions from different distances and orientations. Corners were found, and their locations were refined with OpenCV [**?**]. This allowed for focal length, optical center, and distortion parameters to be calculated [**?**]. See Table **??** for a summary of the GoPro 7 and 8 intrinsic parameters.

Model	Field of view	Resolution (px)	Focal length (px)	Distortion coefficients
GoPro 7	Linear	1920x1080	1029	[-0.004, 0.004, 0.0]
GoPro 8	Linear	1920x1080	915	[-0.01, 0.004, -0.0015]

Table 6.2: Intrinsic parameters of the GoPro 7 and GoPro 8 cameras. The optical center was assumed to be at the center of the image. Focal length was supposed to be identical in both directions, the pixel to be square and not skewed, and 6^{th} order radial and 2^{nd} order tangential distortion coefficient to be null.

Extrinsic parameters can then be solved by pairing global 3D coordinates of an object to its corresponding 2D coordinates on the image. This is classically done with a frame or a checkerboard laid on the floor (where it can be seen by all cameras), but when cameras are too low or too far from the center of the scene, image coordinates can be imprecise and lead to inaccurate extrinsic calibration. In this case, any object of known dimensions on the scene can be used: in our case, we measured the ring dimensions, and retrieved the corresponding 2D coordinates on each camera view. We then used the Perspective-n-Point (PnP) algorithm using non-linear Levenberg-Marquardt minimization scheme [**?**] to solve extrinsic parameters [**?**] (see Figure **??**). Dimensions of the ring were measured approximately, thus we iteratively adjusted its coordinates in order to minimize the grand average error for all cameras between 2D coordinates and projected 3D coordinates, until all camera errors stayed under 3 cm.

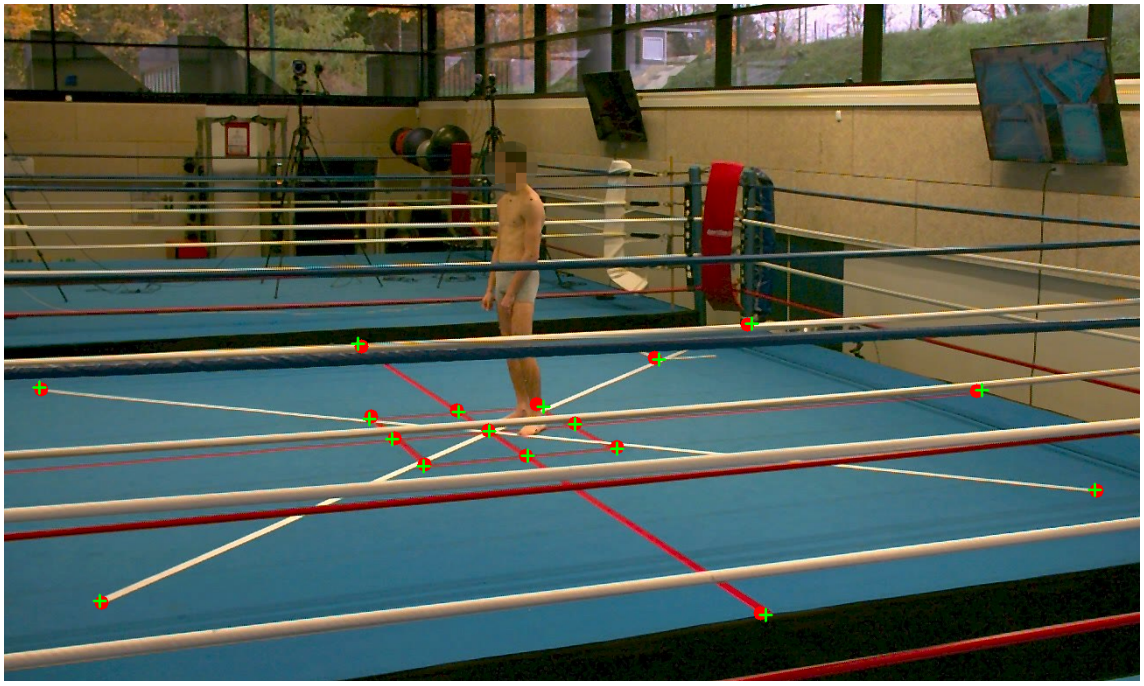


Figure 6.3: Extrinsic calibration on ring dimensions. Green crosses represent the 2D points of the ring clicked by the user, and red dots its 3D coordinates projected on the image plane.

6.2.3 Post-synchronization on 2D movement speeds

GoPro videos were roughly cut to select the approximate same sequence from all cameras. OpenPose 2D keypoint trajectories were differentiated, in order to obtain 2D keypoint speeds. We assumed that two nearby cameras were synchronized when 2D speeds were maximally correlated.

Hence, we used time-lagged cross-correlation to determine this offset (Figure **??**). This can be done for one specific point, or for all of them at once with attributing a different weight to each keypoint. Typically, larger weights could be attributed to fists or to other fast moving keypoints. In practice, all weights were set to 1, since it made results more robust without causing them to be less accurate.

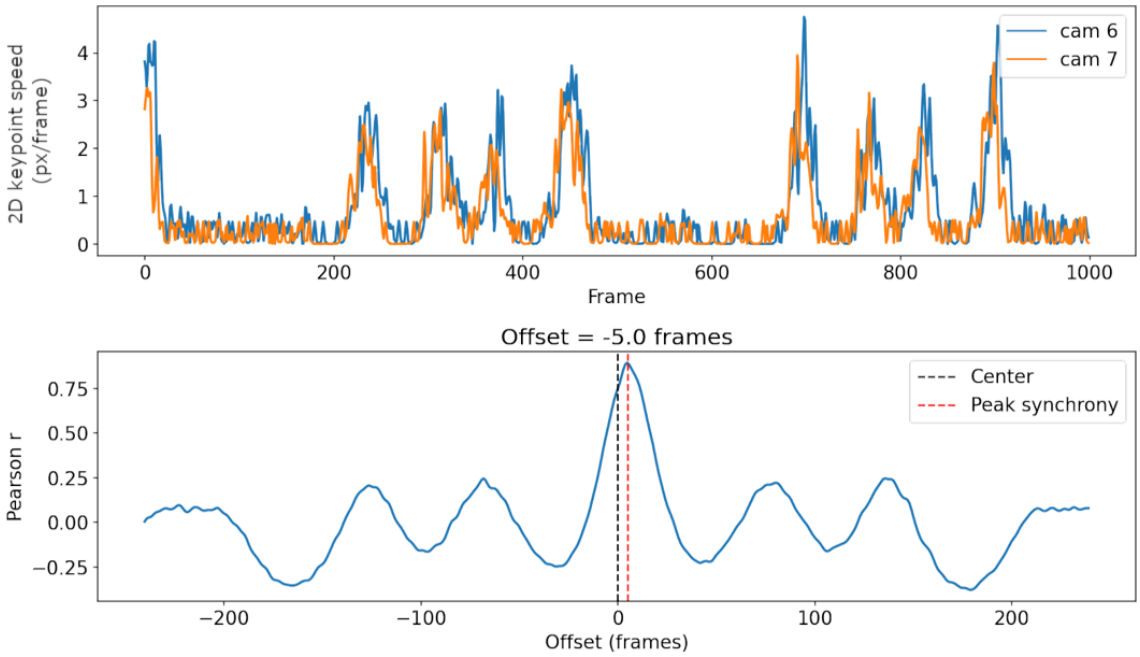


Figure 6.4: Top: 2D keypoint speed comparison between two cameras. Bottom: The offset frame with maximum correlation corresponds to the synchronization offset.

6.2.4 GoPro to Qualisys spatio-temporal coordinate system

In order to be able to compare GoPro results to Qualisys ones, they need to share a common spatio-temporal base. We synchronized both 3D coordinates outputs in the same way as previously described, however with 3D instead of 2D speeds. Then, we transformed GoPro 3D coordinate results to the Qualisys' coordinate system. The rotation and translation needed were found by minimizing the differenc between GoPro and Qualisys 3D coordinates.

6.2.5 Statistical analysis

Calibration accuracy was assessed by comparing residual reprojection errors with the GoPro protocol to those with the Qualisys one. There is no objective metric for assessing synchronization accuracy, however we estimated it indirectly. Once GoPro cameras were synchronized two by two, we retrieved the resulting Pearson correlation coefficient between 2D speeds, and averaged it across all cameras. Similarly, we calculated the same coefficient for the perfectly synchronized Qualisys system. Assuming that GoPro cameras are approximately spread out in the same way as Qualisys ones, and that the 2D pose estimation performed similarly well in both cases, any correlation difference would probably be due to the non-perfect post-synchronization of GoPro cameras. A correlation is generally considered to be weak if r<0.5, moderate if 0.5<r<0.7, and strong if r>0.7.

We examined time series for lead foot translation, pelvis translation, lead elbow extension, and lead fist velocity for the jab; and rear knee flexion, pelvis rotation, rear shoulder extension, and rear fist velocity for the rear hook. Waveform similarity between the reference marker-based method and all other markerless ones was assessed with the inter-protocol coefficient of multiple correlation (CMC) [**?**], which quantifies in one single value the concurrent effects of differences in correlation, gain, and offset (see **?? ??**). We compared the Root Mean Square Error (RMSE) with marker-based results to the findings of two previous studies done with the commercial markerless software Theia3D [**?**].

We also retrieved certain quantifiable indicators on the aforementioned recorded variables.

Ranges of motion and onset times were examined for translations, and peak values and times for angles and speeds. We computed mean differences and standard deviation (std) between marker-based results and all other protocol results. The impact of the camera type, and of the 2D pose estimation model, were similarly explored.

Normality of the distribution of differences was verified with a Shapiro-Wilk test [**?**]. If this condition was satisfied, we used paired t-tests to test the null hypothesis that there was no statistical difference between protocols [**?**]. If not, we used the Wilcoxon signed-rank test [**?**]. When some paired differences were equal to zero, the normal approximation was used. When less than 10 of the paired differences were non-zero, we could not reject the null hypothesis that mean paired results were equal. We used a significance level of 0.05.

6.3 Results

6.3.1 Calibration and synchronization accuracies

Calibrating GoPro cameras on ring dimensions led to an average of 2.59 px reprojection error, which corresponds to 1.93 cm of error at the center of the scene. Conversely, Qualisys average residuals for calibration stayed under 1 mm, which is approximately 20 times better. After synchronization, Pearson correlation coefficient between 2D speeds was 0.69 in average (moderate correlation) with the GoPro system, while it was 0.73 in average (strong correlation) with the Qualisys one, which denotes a non-perfect synchronization of the GoPro cameras.

6.3.2 Waveform comparison

Waveforms were very dissimilar across participants, especially for the hook technique. For example, no apparent peak could be found on hook trials for subject 2 on rear knee flexion nor on rear shoulder abduction. On the opposite, differences were almost imperceptible across protocols (Figure **??**). Results of the research-grade markerless setup, with specialized cameras, marker-based calibration, and hardware synchronization, were almost identical to those of the marker-based analysis (CMC > 0.99). Results from GoPro cameras with calibration on ring dimensions and synchronization on 2D keypoint speeds were also in excellent agreement (CMC > 0.95). The same was true with Qualisys cameras used with the default body_25 model (CMC > 0.95). However, the combined use of GoPro cameras and of the default body_25 OpenPose model led to a slight decrease in accuracy, even if results were still in very good agreement (CMC > 0.85). Velocities and shoulder rotation were the least in agreement, although CMC stayed over 0.90 (Table **??**).

More precisely, in the most favorable conditions, the RMSEs amounted to 8.5 mm for lead foot translation (which can be compared to 24 mm for ankle translation during gait for Theia3D [**?**]), 7.9 mm for pelvis translation (vs. 36 mm for hip translation), 4.1° for lead elbow extension (vs. 7.4° for Theia3D in the same boxing settings [**?**]), 0.3 m/s for lead and rear fist velocities (vs. 0.1° and 0.2 m/s [**?**]), 3.2° for knee flexion (vs. 3.3° [**?**]), 6.5° for pelvis rotation (vs. 8.5° [**?**]), 6.5° for shoulder abduction (vs. 6.3° [**?**]). Results were slightly degraded when using GoPro cameras *or* the standard body_25 model, but they still remained comparable to the Theia3D results. When using both GoPro cameras *and* the body_25 model, they became more clearly worse, except for translations for which errors remained comparable (Table **??**).

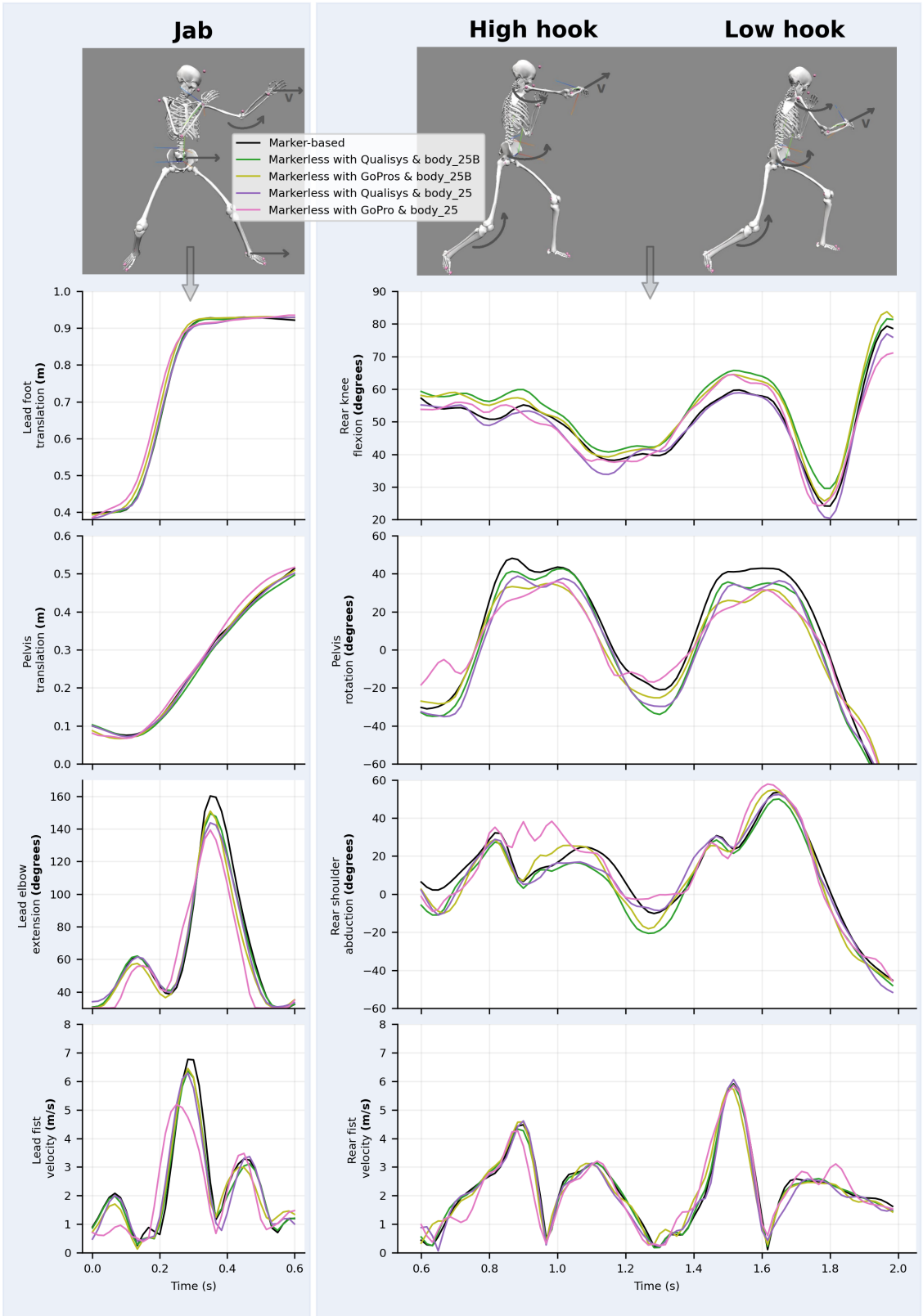


Figure 6.5: A representative trial by one of the athletes. Comparison of selected variables for a sequence of jab, high hook, and low hook in boxing. Waveforms look very similar across all protocols, even though the use of the standard OpenPose body_25 model instead of the experimental body_25B one seems to cause more discrepancy when compared to the reference marker-based analysis.

Marker-based vs. →	Markerless with Qualisys	Markerless with GoPros	Markerless with Qualisys & body_25	Markerless with GoPros & body_25
CMC		Jab punch		
Lead foot translation	1.00	1.00	1.00	1.00
Pelvis translation	1.00	1.00	1.00	1.00
Lead elbow extension	1.00	0.98	0.99	0.95
Lead fist velocity	0.99	0.97	0.98	0.91
		Hook punches		
Rear knee flexion	0.99	0.99	0.99	0.99
Pelvis rotation	0.99	0.99	0.99	0.98
Rear shoulder rot.	0.99	0.99	0.99	0.95
Rear fist velocity	0.99	0.97	0.98	0.90
RMSE		Jab punch		
Lead foot translation (mm)	8.5	20.5	9.5	34.6
Pelvis translation (mm)	7.9	8.3	8.3	15.6
Lead elbow extension (°)	4.1	10.3	6.2	17.4
Lead fist velocity (m/s)	0.3	0.5	0.5	0.9
		Hook punches		
Rear knee flexion (°)	3.2	4.1	3.2	3.9
Pelvis rotation (°)	6.9	8.6	8.1	11.4
Rear shoulder rotation (°)	6.5	8.0	7.3	14.2
Rear fist velocity (m/s)	0.3	0.5	0.4	0.9

Table 6.3: The Coefficient of Multiple Correlation (CMC) was used to assess the waveform similarity of the variables of interest. Agreement is deemed excellent if CMC>0.95, and very good if CMC>0.85 [?]. Root Mean Square Errors (RMSEs) were also reported.

6.3.3 KPI accuracy assessment

As KPI differences across protocols were never normally distributed, paired t-tests could not be rigorously performed. As a consequence, we only used Wilcoxon signed-rank tests. Nonetheless, in many cases, especially for time results, most paired differences were equal to zero. In these instances, we could not reject the null hypothesis that results were identical in average. It should be noted that for rear knee flexion and for rear shoulder abduction in hook punches, KPI means and standard deviations did not take participant 2 into account, since her movement patterns did not present any peak in these variables.

Range of motion errors usually remained under ± 10 mm, as compared to the marker-based protocol. Mean errors were even lesser with body_25 model, but with more variation around the mean, which did not lead to any significant differences. Elbow extension, knee extension (calculated as $180° - \textit{knee flexion}$), shoulder abduction, and pelvis rotation peak angles were generally underestimated, by 2.9° to 5.4° with the Qualisys cameras and body_25B model, by 2.7° to 10.2° when using GoPro cameras or body_25 model, and by more than 11.8° to 17.2° when using the

most suboptimal protocol with both GoPro cameras and body_25 model. Peak fist velocity errors did not broadly depend on the choice of protocol. However, they were underestimated by up to 0.4 m/s (1.4 km/h) for both hands (Table **??**).

Peak times were usually estimated too early with GoPro cameras, by about 10 to 40 ms, i.e., 0.6 to 2.4 frames at 60 fps. Most other onset and peak time errors were not statistically significant, especially for Qualisys cameras when used with the experimental Body_25B pose estimation model. Rear shoulder abduction demonstrated the largest peak time error, up to about 50 ms (3 frames) (Table **??**).

Mean$_{err}$ (±std) of Marker-based vs. →	**Markerless with Qualisys**	**Markerless with GoPros**	**Markerless with Qualisys & body_25**	**Markerless with GoPros & body_25**
ROM[1] *or* Peak value		Jab punch		
Lead foot translation[1] (mm)	-10.6 (±6.7) *	-10.0 (±7.4) *	-1.1 (±9.8)	4.9 (±12.0)
Pelvis translation[1] (mm)	-6.9 (±6.0) *	4.3 (±5.3) *	-4.1 (±7.4)	11.2 (±9.5) *
Lead elbow extension (°)	-2.9 (±2.0)	-8.6 (±3.5) *	-10.2 (±4.3) *	-17.2 (±6.1) *
Lead fist velocity (m/s)	-0.4 (±0.1) *	-0.2 (±0.2) *	-0.4 (±0.2) *	-0.4 (±0.2) *
		Hook punches		
Rear knee flexion[2] (°)	5.4 (±1.9) *	4.0 (±2.1) *	-0.5 (±1.9)	1.7 (±2.8)
Pelvis rotation (°)	-3.9 (±4.8) *	-7.9 (±2.7) *	-6.9 (±4.1) *	11.8 (±5.7) *
Rear shoulder abduction[2] (°)	-4.1 (±2.2) *	-2.7 (±2.9)	-3.4 (±2.7) *	-0.9 (±6.2)
Rear fist velocity (m/s)	-0.1 (±0.1) *	-0.4 (±0.2)	-0.3 (±0.1) *	-0.4 (±0.6)
Onset time[1] *or* Peak time		Jab punch		
Lead foot translation[1] (ms)	-0.9 (±2.3)	-14.8 (±11.4) *	-1.9 (±5.3)	-34.3 (±18.2) *
Pelvis translation[1] (ms)	6.5 (±8.7)	-7.4 (±13.8)	7.4 (±5.3) *	-14.8 (±18.2) *
Lead elbow extension (ms)	-3.7 (±5.3)	-2.8 (±11.9)	-3.7 (±6.4)	-19.4 (±19.8) *
Lead fist velocity (ms)	-8.3 (±8.8)	-13.0 (±9.8) *	-10.2 (±10.4) *	-20.4 (±15.1) *
		Hook punches		
Rear knee flexion[2] (ms)	15.3 (±27.0)	9.7 (±31.2)	-19.4 (±34.5)	-19.4 (±48.2)
Pelvis rotation (ms)	18.5 (±41.8)	28.5 (±49.5)	31.5 (±58.0) *	32.4 (±38.2) *
Rear shoulder abduction[2] (ms)	-58.3 (±52.1)	-40.3 (±45.0) *	-56.4 (±59.6)	-15.3 (±55.5)
Rear fist velocity (ms)	0.9 (±2.3)	-7.4 (±19.4) *	-2.8 (±9.0)	-14.8 (±28.0) *

*Table 6.4: Mean and standard deviation (±std) errors of the boxing KPIs captured through each protocol, as compared to marker-based reference results. [1]Peak values and times were observed for rotations and speeds, and ROMs and onset times for translations. [2]Participant 2 was not taken into account in rear knee flexion nor on rear shoulder abduction results, since her movement patterns did not lead to any peak in these variables. ROM: Range Of Motion. *indicates statistically significant differences (p<0.05).*

Using GoPros instead of Qualisys cameras had a comparable impact as switching back to the standard body_25 OpenPose model on ranges of motion (up to 13 mm difference), peak angles (up to 7°), and peak velocities (0.1 m/s). However, using GoPro cameras lead to larger delays than choosing the body_25 model: about 20 ms vs. 10 ms (1.2 vs; 0.6 frames) for translations, and 10 ms vs. 5 ms (0.6 vs. 0.3 frames) for velocities. No general conclusion could be drawn for time differences in joint angles, as peaks were more challenging to accurately detect, especially when

using the body_25 model which led to noisier waveforms (Table **??**).

Mean$_{err}$ (±std)	**Qualisys vs. GoPros**		**Body_25B vs. Body_25**	
	ROM[1] *or* Peak value (mm, °, or m/s)	**Onset[1] *or* Peak time** (ms)	**ROM[1] *or* Peak value** (mm, °, or m/s)	**Onset[1] *or* Peak time** (ms)
Jab				
Lead foot translation[1]	3.3 (±8.9) *	-23.1 (±18.9) *	12.2 (±10.9) *	-10.2 (±16.1) *
Pelvis translation[1]	13.3 (±10.4) *	-18.1 (±18.4) *	4.8 (±8.2) *	-3.2 (±15.8)
Lead elbow extension	-6.5 (±10.8) *	-7.4 (±16.6) *	-7.8 (±9.5) *	-8.3 (±17.1) *
Lead fist velocity	0.1 (±0.2) *	-7.4 (±15.7)	-0.1 (±0.2) *	-4.6 (±11.0) *
Hook				
Rear knee flexion[2]	0.5 (±3.2)	-2.8 (±29.8)	-4.1 (±2.6) *	-31.9 (±58.8) *
Pelvis rotation	-4.4 (±6.9) *	5.5 (±59.8) *	-3.5 (±5.2)	8.4 (±50.7)
Rear shoulder abduction[2]	2.0 (±5.2) *	29.9 (±80.6)	1.2 (±4.6)	13.2 (±68.9)
Rear fist velocity	-0.1 (±0.7)	-10.2 (±30.1) *	-0.1 (±0.5)	-5.6 (±21.8)

Table 6.5: Mean and standard deviation (std) errors of the boxing KPIs, compared between camera types, and 2D pose estimation models. Qualisys cameras are research-grade, while GoPro cameras are consumer-grade and involve different calibration and synchronization procedures. OpenPose Body_25B model is claimed to be more accurate than the standard body_25 model.[1]*Peak values and times were observed for rotations and speeds, and ROMs and onset times for translations.* [2]*Participant 2 was not taken into account in rear knee flexion nor on rear shoulder abduction results, since her movement patterns did not lead to any peak in these variables. ROM: Range Of Motion. *indicates statistically significant differences (p<0.05).*

6.4 Discussion

6.4.1 Accuracy assessment

The boxing capture session was operated in close to ecologically valid conditions, with minimal interferences both with the athlete and the environment, on a regular competition ring. The movements we investigated were challenging, as they were performed at high speeds, were 3 dimensional, and involved the whole body. We examined a broad range of variable types, such as translations, rotations, joint angles, and velocities. Moreover, aside from the foot and pelvis ranges of motions, the chosen KPIs evaluated spatio-temporal parameters at specific instants, rather than their average over a whole movement cycle. Thus, onset and peak times were determined rather than durations, peak angles rather their ROMs, peak speeds rather than their average. This made the study more sensitive to noise and to small errors in variable measurements.

And yet, it performed remarkably well in all conditions. The difference in waveforms was hardly noticeable, except in the last condition, under the combined effects of using consumer-grade cameras with a less accurate 2D pose estimation model. Even in these conditions, results were in very good agreement for velocities, excellent agreement for joint angles, and perfect agreement for translations. RMSE results were slighlty better than those previously reported for Theia3D in the most favorable conditions, and comparable under either the use of GoPro cameras, or of the standard OpenPose model. They were more noticeably degraded when being confronted with both challenging conditions, although they remained well within the same order of magnitude.

However, more significant imprecision was observed when looking at specific instants, i.e., when examining our chosen KPIs. Still, results were coherent. Indeed, peak velocities in jabs and hooks were fully in accordance with values previously reported by [**?**, **?**]. The magnitude of the difference with marker-based results were moderate. Translation errors remained sub-centimetric, peak joint angle errors were mostly lesser than 5°, and peak velocity errors stayed under 0.4 m/s. Peak and onset time errors were usually sub-frame (under 17 ms), aside from the pelvis rotation and the shoulder abduction, whose error could go up to 3.5 frames. This was probably due to the dearth of keypoints in the pelvic and trunk region, and to the fact that the shoulder girdle is much more complicated than a ball joint, as it has been defined in our OpenSim model.

Using a less accurate 2D pose estimation model had a similar impact as using consumer-grade cameras. This impact, however, was very mild, and only became more obvious once both factors were combined, which led to more noisy and unstable waveforms. This shows that a research team should choose their 2D pose estimation algorithms with as much care as their hardware. Some other models provided by OpenPose or other contributors are less accurate [**?**], and may consequently lead to divergent results. This also involves that the triangulated keypoints must be carefully placed on the OpenSim skeletal model. Indeed, we noticed that elbow and knee extension angles were systematically underestimated, which may imply that the corresponding keypoints were mispositioned.

Finally, we can infer from our study that a research-grade system is not necessarily needed for the determination of sports KPIs. Consumer-grade cameras may be sufficient, despite they imply greater calibration errors (centimetric rather than sub-millimetric), and approximate synchronization (based on the participant's 2D keypoint speeds rather than fixed by a hardware trigger). This is especially interesting, since calibrating with a wand equipped with retro-reflective markers is rarely successful in broad daylight, and since laying cables in the scene for hardware synchronization potentially interferes with natural sports movements. Moreover, action cameras such as GoPros are lightweight, easy to set up, and wireless. They also procure a larger field of view, often with higher frame rates and image definition, at considerably lower prices. In general, they are much more appropriate in a sports setting. Results being virtually the same as with research-grade cameras, practical considerations remain: they do not allow for any live feedback, they require careful planning for sparing battery life, and they potentially involve more complicated post-processing procedures in terms of calibration and synchronization. However, calibration and synchronization present the potential to be at least partly automatized.

All things considered, it is interesting to notice that regardless of the protocol, all results were very close to the marker-based ones. With the help of both IMUs and videos together, it has been shown that with fatigue, boxers tend to release their guard, lift their elbow, and increase their shoulder abduction [**?**]. Based on the outcomes of our study, it should be possible to monitor this without the use of any markers, IMUs, or any apparatus other than non-invasive and consumer-grade video cameras. This opens the way to sports kinematic analysis and to KPI determination in context, when marker-based analysis is not possible.

6.4.2 Limits and perspectives

Kinematic analysis from video cameras involves 3 main stages: 2D pose estimation, 3D reconstruction, and kinematic optimization. The last one has not been evaluated, and yet, it is of crucial importance. Choosing a good skeletal model with consistent joint constraints and accurate bone definition will determine whether the results can be trusted, or not. In particular, in this study the shoulder was defined as a ball joint, both on the marker-based and on the markerless analysis. Yet, the scapulothoracic girdle is a very complex multi-articulated joint, which allows for 3 rotations and 3 translations [**?**]. Considering the small amount of keypoints tracked by standard 2D pose estimation models, it is impossible to use a more complex model. Consequently, our shoulder abduction results can probably not be entirely trusted, neither for our markerless nor for

our marker-based protocol (see **??** in **??**).

In addition, GoPro 60 fps mode actually samples at 59.94 fps. This is a multiple of 29.97 fps, which is a holdout from the introduction of color on television in North America in the 1950s. This is also true for phone cameras and for most recording devices. It leads to a 3.6 frames delay per minute (0.06 s) when compared to the true 60 fps of Qualisys cameras. This temporal drift artificially hampered our results on peak time comparisons, which would otherwise be even better. However, this is not a problem in practice, as long as all cameras involved in the capture shoot at the same frame rate.

Most consumer-grade cameras also use a rolling shutter instead of a global shutter. This is known to cause some irreparable image distortion, commonly called "jello effects". However, unless lighting is particularly low, the rolling frequency for GoPro cameras appears to be approximately identical to the acquisition frequency. Considering that these cameras can film at frequencies as high as 240 fps in full-HD, and in the context of 3D reconstruction, cameras will always be firmly set on a sturdy surface, it is not likely to cause any issues, including in fast sports disciplines.

Despite the fact that boxing activities involve challenging constraints, kinematic analysis may not be as successful for other sports. For example, gymnastics involve athletes being often upside-down, which OpenPose does not handle well at all. In this case, other models can be used, or even custom-trained by users; however, as demonstrated, a lack of precision may lead to inaccuracies in results. A similar lack of precision will also occur if a large field is captured, such as in track and field or in team sports. To a certain extent, the body_25B model generalizes better to small people detection [**?**], although top-down method such as AlphaPose [**?**] usually perform better at smaller scales [**?**, **?**]. If the sports involves using additional equipment such as a bike, more occlusions may occur, which could also hamper results. We previously showed that Pose2Sim was robust to a decrease from 8 to 4 cameras, unless the person suffered from occlusions by equipment such as a bike [**?**]. Extracting not only keypoints, but also body shape, could provide enough additional information to allow for a decrease of the amount of cameras with less repercussion on the quality of results.

Lastly, currently Pose2Sim does not handle multi-person kinematic analysis, which is highly problematic for any team or opposition sports. This is planned to be implemented shortly. Our calibration and synchronization procedures are also projected to be added in the public release. Auto-calibration on people limb dimensions could also be proposed, even if all sizes would only be true up to a factor [**?**]. In the future, it would also be interesting to support joint kinetics prediction, by adding muscles which were stripped from the skeleton in the OpenSim model, as well as inverse dynamics, by training neural networks to estimate ground reaction forces on specific tasks [**?**, **?**, **?**, **?**].

7

Application to BMX racing, jointly capturing pilot and bike

Numerous sports disciplines are practiced with special equipment, such as a board in skateboarding, a racket and a ball in tennis, or a bike in BMX racing. The interactions between the athlete and their gear are often important to retrieve. We analyzed a BMX start sequence, by using OpenPose for 2D human pose estimation, and a custom trained DeepLabCut model for bike detection. We ran Pose2Sim on the joint pilot+bike 2D estimations, and performed 3D inverse kinematics on a custom OpenSim pilot+bike model. This showed that analyzing simultaneously the athlete and their equipment is possible, which provides additional perspectives for markerless sports motion analysis.
See Figure **??** *for a visual abstract.*

Contents

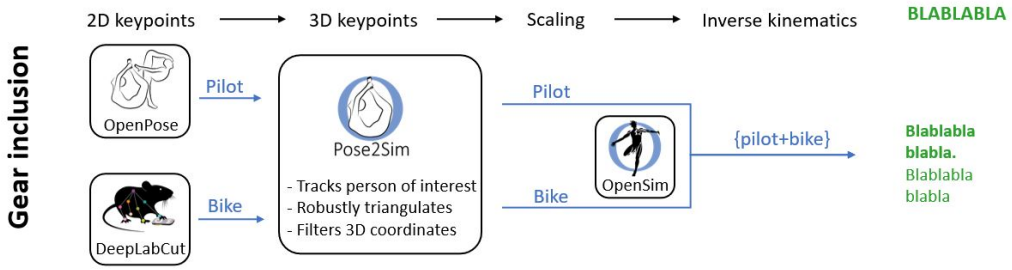


Figure 7.1: Visual abstract for joint analysis of the athlete with OpenPose, and their equipment with DeepLabCut.

Ski: lire intro sur equipement: https://sci-hub.se/https://ieeexplore.ieee.org/document/9105973 Formalisme joint vs constraint Part 3.1 https://sci-hub.se/https://ieeexplore.ieee.org/document/4587520 https://www.google.com/search?client=firefox-b-eq=constraint+vs+joint https://arxiv.org/pdf/2112.00627.pdf 42: ball as segmentation. eux: keypoint

Preliminary experiment

1. track equipment (shape or keypoints), Straight-forward with markers (hockey kays2017) or IMUs (boat ruffaldi2015), whereas specific training for markerless deep-learning (bike-snow Rosenhahn2008, ball Ghasemzadeh2021, skis ludwig2020) 2. compute joint kinematics of the equipment (6DoF, or definition of model with kinematic chain) 3. solve combined kinematics of the whole system (closed loop)

Chap3: On a different note, in a sports context, not only the human pose is of interest: sports gear can also be considerably important to detect, such as a ball [**?**], a hockey stick [**?**], a rowing boat [**?**], a snowboard [**?**] or skis [**?**], or bike parts in the context of cycling (see Chapter 7 on Joint OpenPose and DeepLabCut detection). This can help to analyze game dynamics, and to quantify posture cues related to a specific sports discipline. This can be done, for example, by separately process the video with OpenPose, as well as with a custom-trained DeepLabCut model. Resulting .trc coordinate files can be merged, and used in OpenSim. However, the DeeLabCut keypoints must be referenced on an OpenSim model, which may need to be crafted from scratch, such as a ball, skis, or bike, depending on the detected object.

CF mail OpenPose DeepLabCut pour BMX pre manip, slt 6 cam, calib unsuccessful: on grid similar to other chapter, low image resolution other leads: bushing forces, ball joints scale avant et arriève vélo indépendament

7.1 Introduction

7.1.1 The importance of equipment

7.1.2 The start in BMX racing

7.2 Methods

7.2.1 Material and protocol

7.2.2 Pilot inverse kinematics

7.2.3 Bike inverse kinematics

Voir thèse Princelle

7.2.4 Joined pilot and bike inverse kinematics

Marche pas avec nos qualités de vidéo : simulations

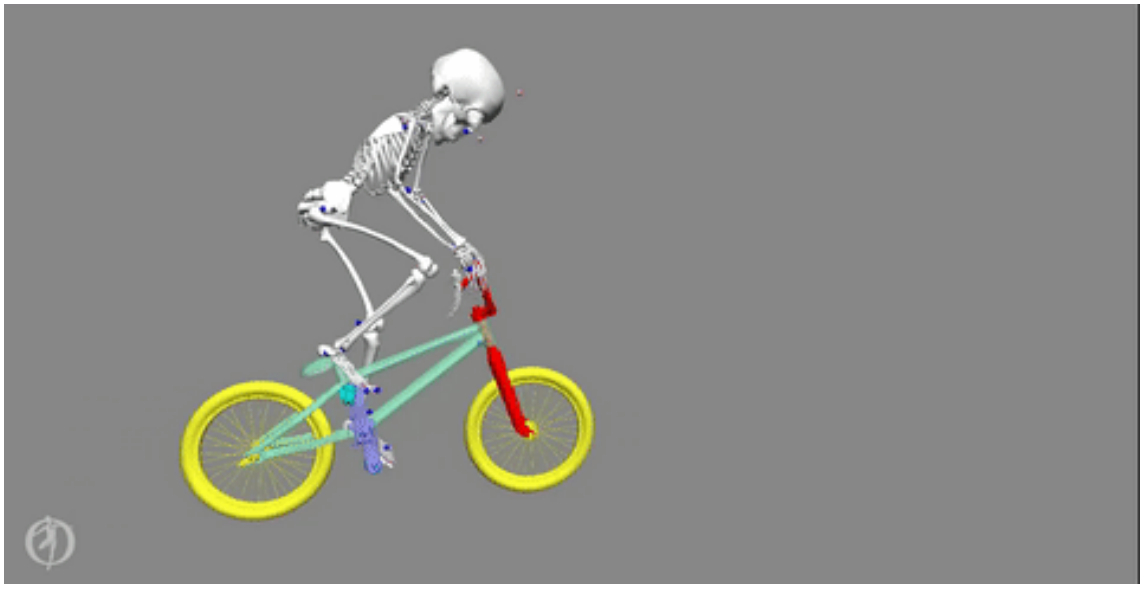


Figure 7.2: Note that the head is only big because the pilot was scaled with his helmet on. Since the head is at the end of the kinematic chain, and as kinetics are not addressed, it does not affect inverse kinematics. See animated version here.

7.3 Results

7.4 Discussion

7.4.1 On these data

7.4.2 Limits and perspectives

Rosenhahn2008 snowboard: closed kinematic chain: pas évident, voir [17] (reduced equations of robotic systems using Lie groups). Eux: soft constraints (invariances -> numeric) rather than joints (analytic)

Mathis2020 Principles, pitfalls and perspectives

splashes, occlusions, distance, etc

Voir Protocole doc:

General conclusion

Conclusion here.

The program also led to a collaboration with the CAMERA center (Center for the Analysis of Motion, Entertainment Research and Applications) of the University of Bath, to discussions with the Medical institute of University of Montreal, and to an in-person meeting with Matthew O'Neill from the Midwestern university. It has also been cited and reused in the field of 3D animation [**?**], and quoted by the Stanford University, which later developed their own 3D markerless motion analysis solution [**?**].

did a few experiments with balancing, dancing, and swimming, throwing

7.4.3 Issues related to OpenPose

it could be interesting to build a new dataset, with more accurate labelling, more keypoints, trained on specific sports poses.

One way to solve this is enhancing the OpenPose dataset, by augmenting it with larger rotations so that upside-down poses are recognized, or by training it on specific sports poses. One risk of this approach is that the model may perform better on specific extreme poses, but worse on standard ones [**?**].

Another approach, which could solve altogether the offsets in labeling, the dearth of keypoints, and the lack of accuracy on sports poses, could be to train on a whole new dataset. Note that this dataset should not base its labeling on marker positions, which could be interpreted as visual cues, that are not available in real sports situations. However, this condition is not sufficient: the dataset should also be large and diverse enough, represent a wide variety of body types and of sports movements [**?**], and include images with motion blur such as found in sports videos. One way to do it is to build a synthetic dataset. For example, a mass of c3D motion files could be gathered from various sports, and be used to fit an SMPL+H mesh [**?**] with AMASS [**?**]. These data could be augmented with already existing datasets for daily life activities, such as Agora [**?**]. Then, it would be possible to take advantage of the fact that the topology of an SMPL mesh is constant, and assume that only labelling the first frame of any given sequence should be sufficient: label positions should be consistently propagated to the next frames. At this stage, one could place as many virtual markers as needed, for a precise evaluation of any movement and pose. However, only an expert should perform this task, and make sure that markers are correctly positioned: crowd-sourcing this task, like it is done for more basic image classification and segmentation with ImageNet [**?**], has been proved to lead to systematic offset errors [**?**]. Finally, random clothing, background, and light could be added (see [**?**, **?**] for a detailed workflow), as well as variations in SMPL shape parameters. The scene would be filmed with numerous virtual cameras, in order to gather a large amount of diverse perspectives, and virtual markers would be automatically projected on the camera planes. This would result in an extensive sports dataset, created with minimal labelling work, on a potentially infinite amount of views. Nevertheless, before training the network, one should make sure that the generated data is as diverse as the real world, by using one of the metrics proposed by [**?**, **?**]. Additionally, keypoint positions need to be precise enough: SMPL shape vertices can sometimes be more than 5 cm apart, which could cause imprecision errors similar to skin artifacts. Besides, instead of constraining pose estimation results with a physically consistent skeletal model, it would be interesting to develop a physics-informed pose estimation model [**?**], which would offer the opportunity of embedding the kinematics priors as

early as possible in the learning process.

SHAPE?

CALIBRATION / AUTO-CALIBRATION

MULTIPERSON

7.4.4 inverse kinematics

True to all constrained models, as opposed to 6DOF approaches: only applicable with population close to the norm: limb missing, pathological joint, not okay

GUI

Bath, MPP2SOS, Maya-Mocap (Blender MoCapViz)

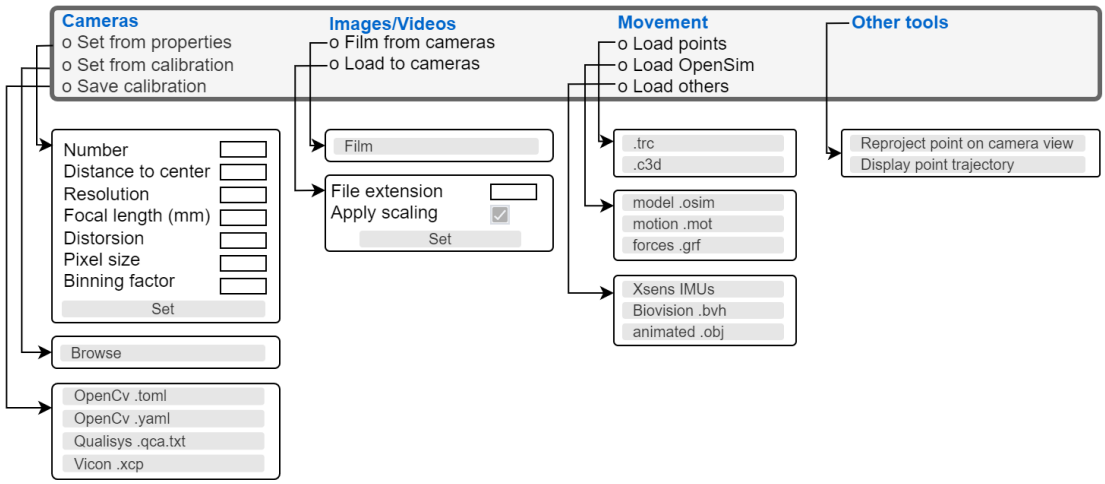


Figure 7.3: The organigram of Blender MoCapViz planned abilities.

List of Figures

List of Tables

A

Appendix A on Robustness assessment

Supplementary figures for Chapter 4 on **??***, for the running and cycling tasks.*

All details on methods and results are provided in the fore-mentioned chapter.

Contents

A.1 Robustness: Running task results

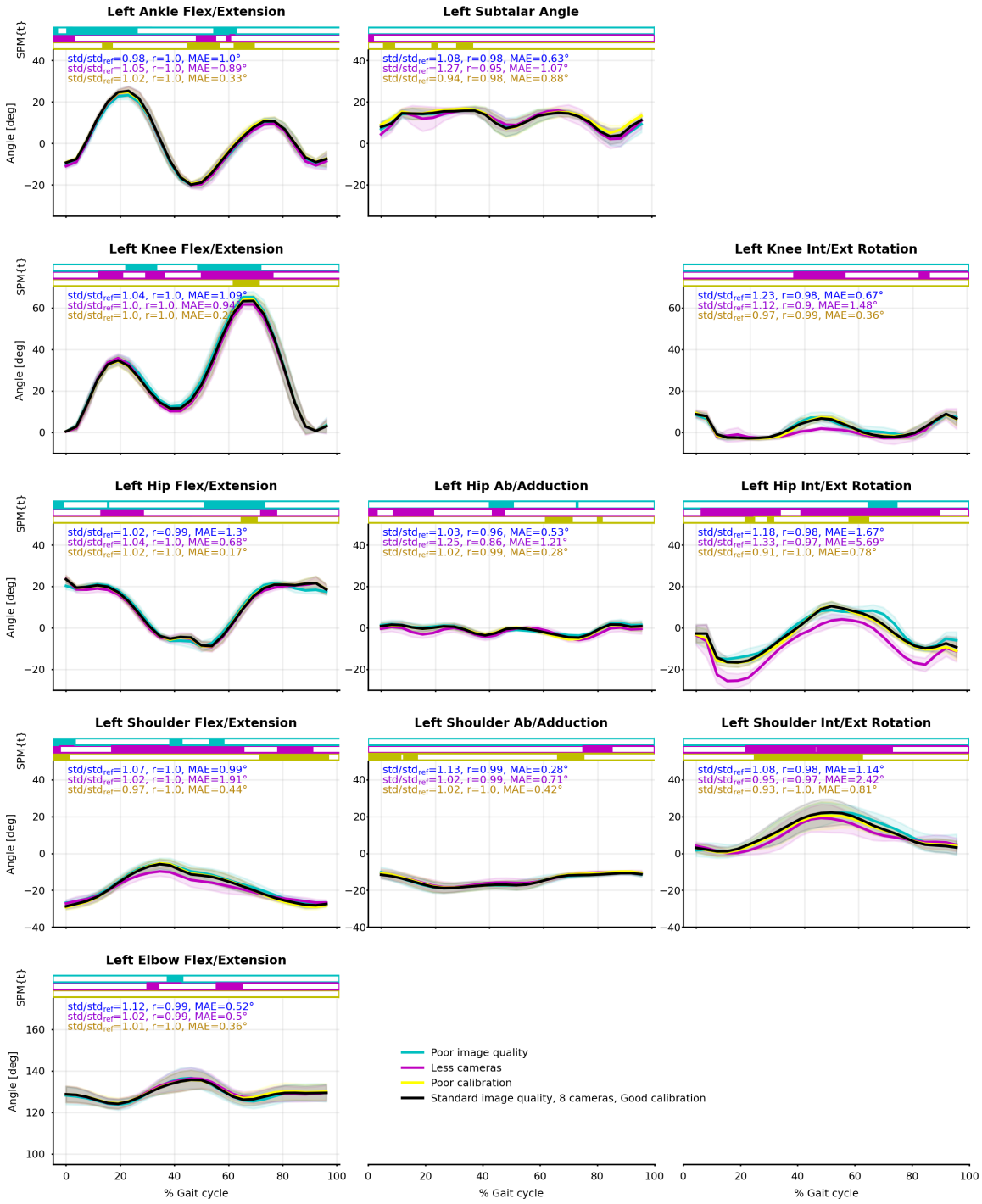


Figure A.1: Joint angle means (solid line) and standard deviations (shaded area) from the nine captured cycles of running. Reference condition (Ref) is black; degraded image quality (Im) is blue; four cameras instead of eight (4c) is purple; degraded calibration (Cal) is yellow. Pearson's correlation coefficient (r) and mean absolute error (MAE) between Ref and Im, 4c, Cal were calculated. Paired t-tests along time were computed by SPM-1D and are represented as bar plots above the curves: a color rectangle means that there was a cluster of statistically significant differences ($\alpha = 0.05$) at that moment.

A.2 Robustness: Cycling task results

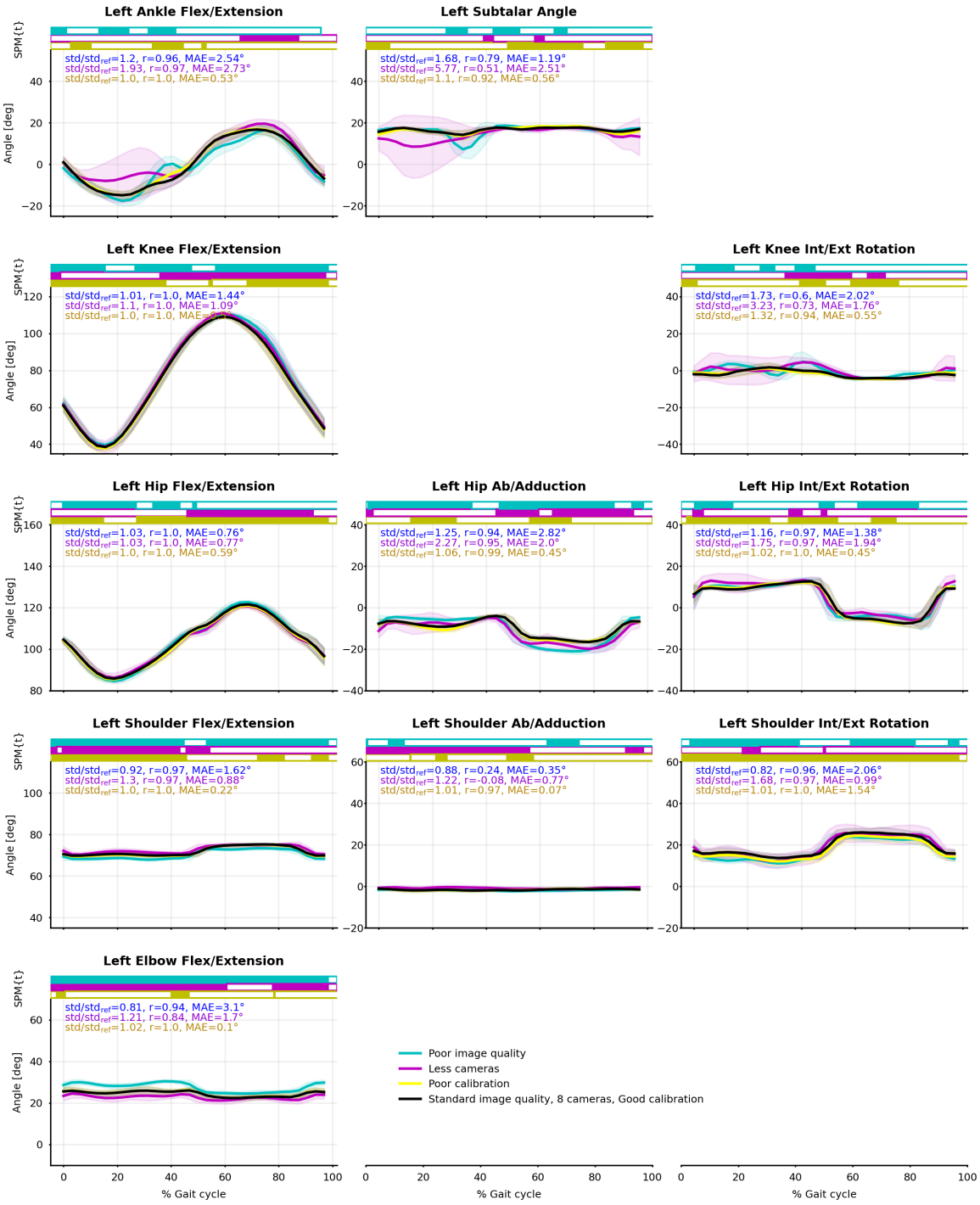


Figure A.2: Joint angle means (solid line) and standard deviations (shaded area) from the 15 captured cycles of cycling. Reference condition (Ref) is black; degraded image quality (Im) is blue; four cameras instead of eight (4c) is purple; degraded calibration (Cal) is yellow. Pearson's correlation coefficient (r) and mean absolute error (MAE) between Ref and Im, 4c, Cal were calculated. Paired t-tests along time were computed by SPM-1D and are represented as bar plots above the curves: a color rectangle means that there was a cluster of statistically significant differences ($\alpha = 0.05$) at that moment.

B

Appendix B on Accuracy assessment

Supplementary figures for Chapter 5 on **??**, *for the lower-body analysis of the running and cycling tasks, and for the upper-body analysis of all three tasks.*

All details on methods and results on the lower-body are provided in the forementioned chapter. Results and discussion on the upper-body are debated section **??**.

Contents

B.1 Lower-body results for the running and cycling tasks

Lower-body graphs are provided on Figure **??** and on Figure **??** for the running task, and on Figure **??** and on Figure **??**. However, all results are discussed in the main body of the article on **??**.

B.1.1 Accuracy: Running task results

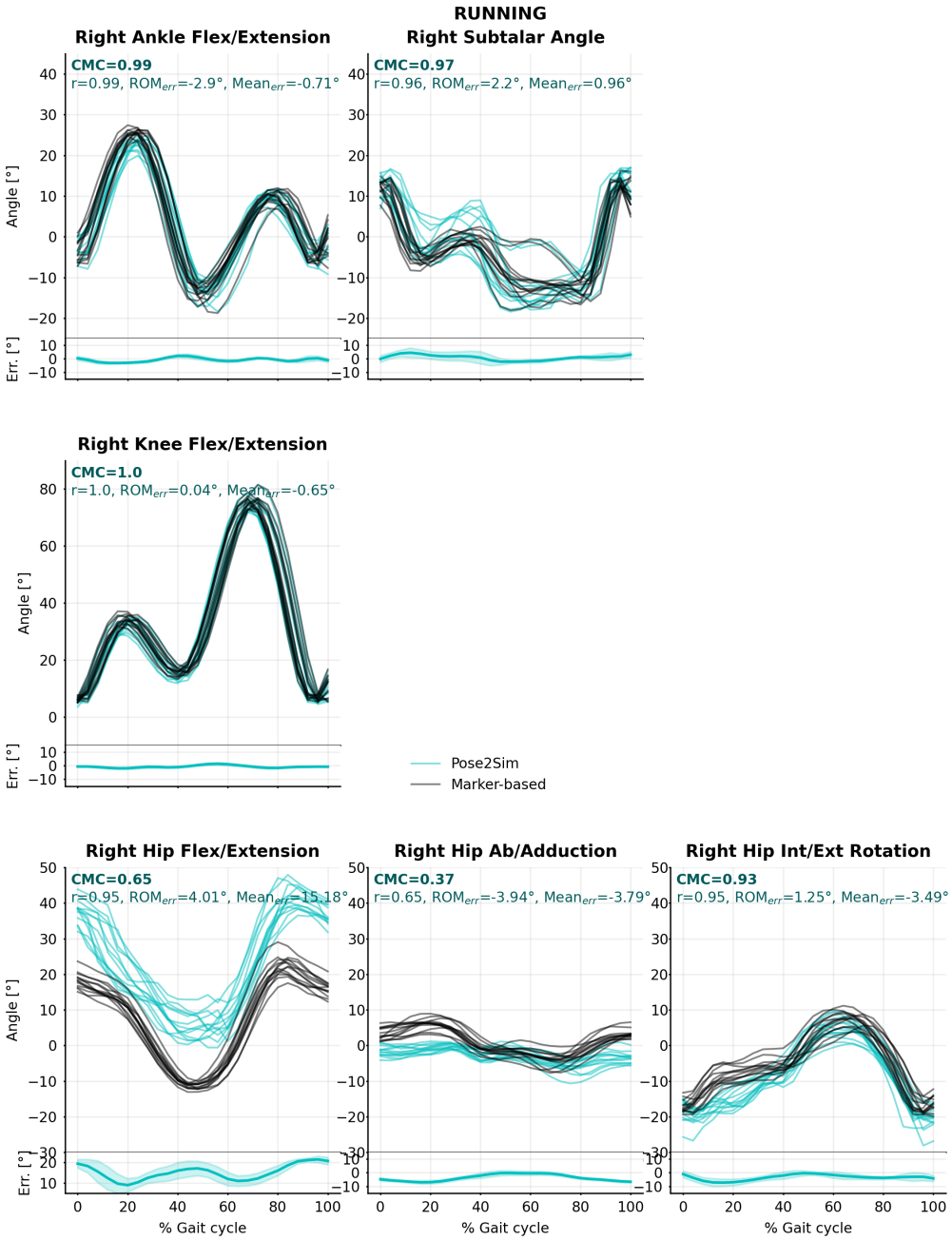


Figure B.1: Pose2Sim (cyan) and marker-based (black) lower-body joint angles for the running task. Coefficient of multiple correlation (CMC) is indicated, and broken down into, respectively, Pearson's coefficient (r) for correlation assessment, range of motion errors (ROM_{err}) for gain, and overall mean errors ($Mean_{err}$) for offset. Mean error and standard deviations are also represented at the bottom of the graphics.

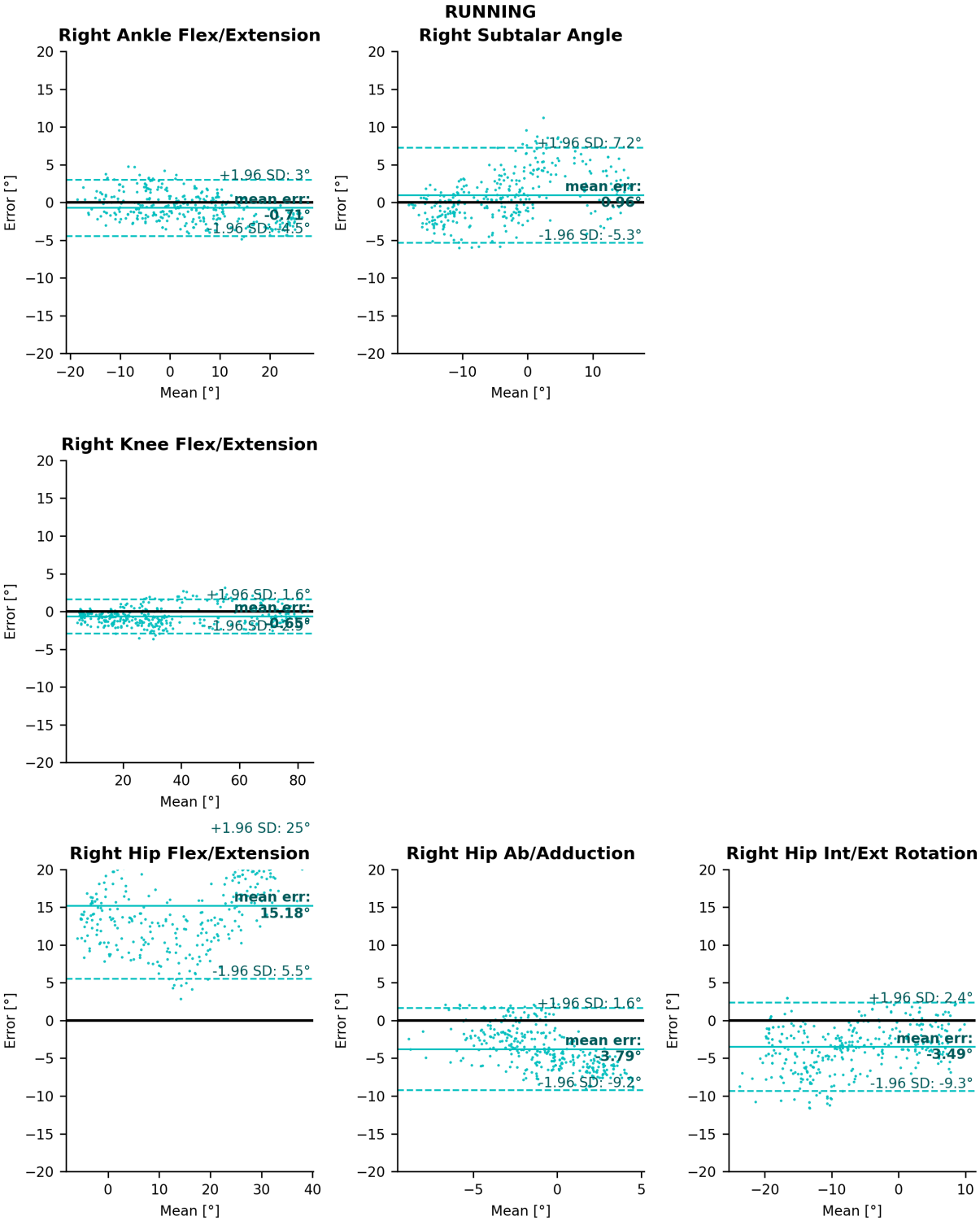


Figure B.2: Bland–Altman analysis of lower-body joint angle errors for the running task. Mean bias is represented as a horizontal solid, bold line, and 95% limits of agreement are represented as dotted lines.

B.1.2 Accuracy: Cycling task results

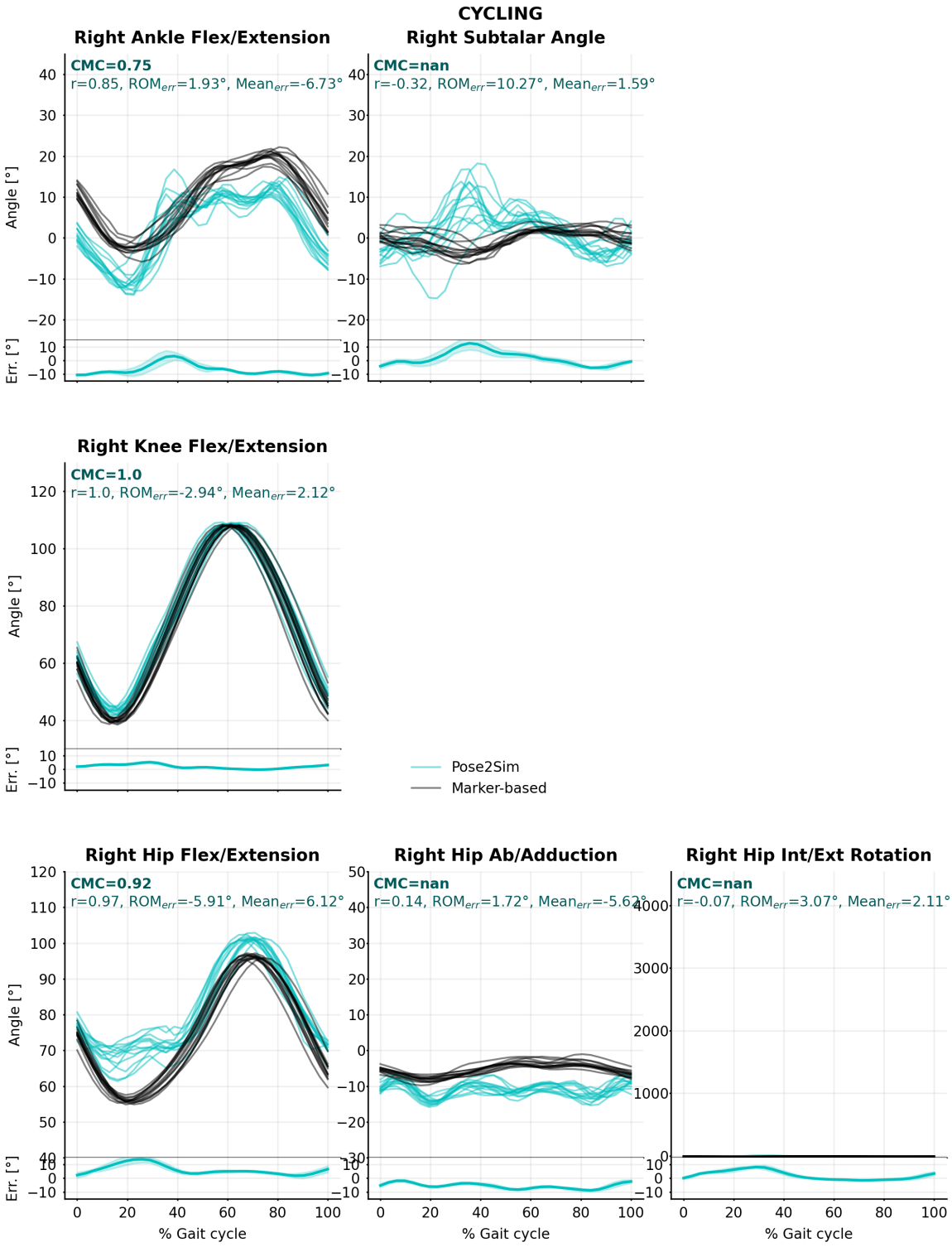


Figure B.3: Pose2Sim (cyan) and marker-based (black) lower-body joint angles for the cycling task. Coefficient of multiple correlation (CMC) is indicated, and broken down into, respectively, Pearson's coefficient (r) for correlation assessment, range of motion errors (ROM_{err}) for gain, and overall mean errors ($Mean_{err}$) for offset. Mean error and standard deviations are also represented at the bottom of the graphics.

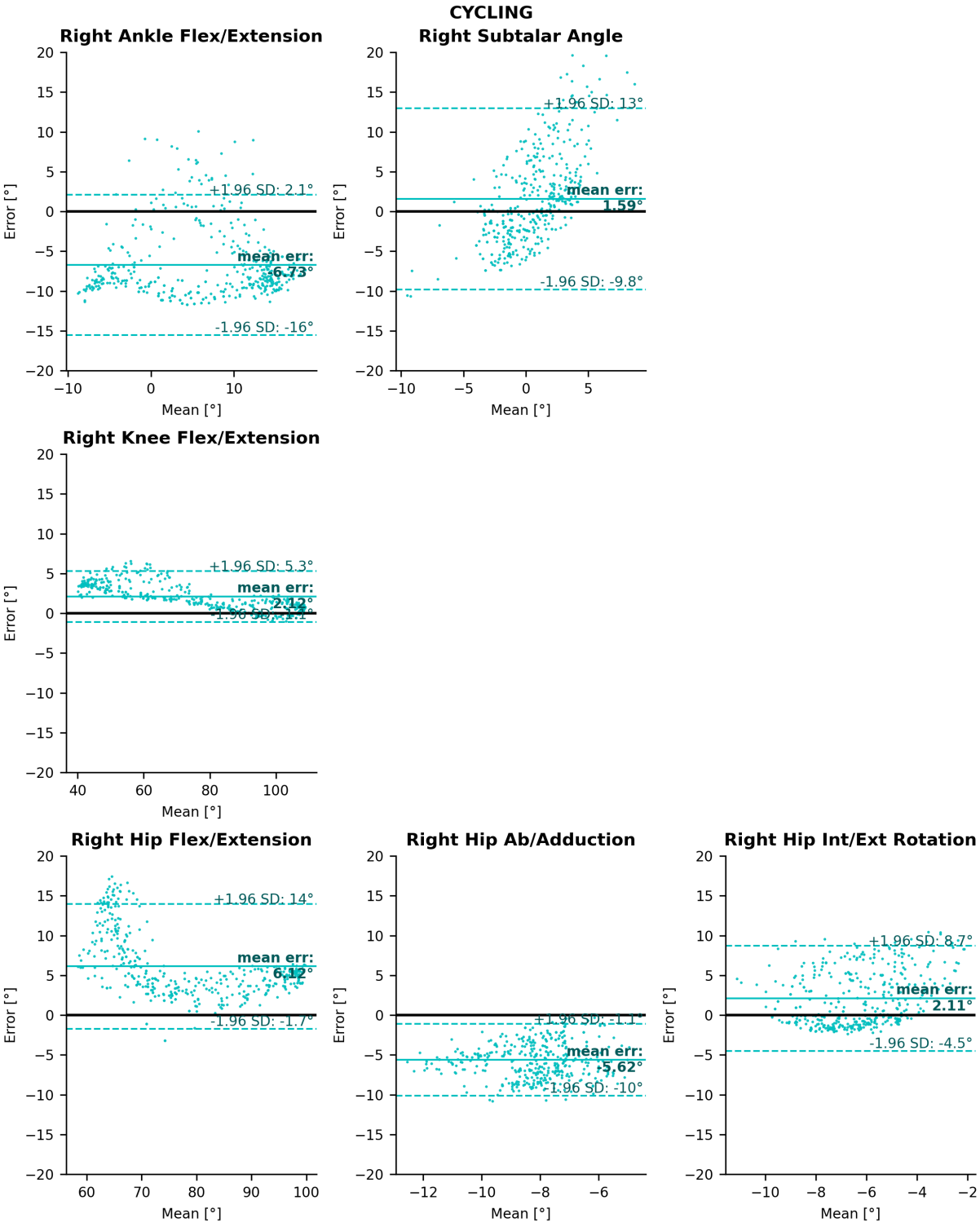


Figure B.4: Bland–Altman analysis of lower-body joint angle errors for the cycling task. Mean bias is represented as a horizontal solid, bold line, and 95% limits of agreement are represented as dotted lines.

B.2 Sacro-lumbar and upper-body results for all tasks

Although the article focused on lower limb kinematics, we ran the same analysis on sacro-lumbar, elbow, and shoulder joints for all three tasks (Figure **??** and Figure **??** for walking, Figure **??** and Figure **??** for running, and Figure **??** and Figure **??** for cycling). The OpenPose model we used does not allow for the capture of wrist deviation or pronation/supination, or of any hand or finger movement.

Results were generally less good than in the lower body, especially on sacro-lumbar flexion/extension, for which all CMC values were complex. This can be attributed both to the lack of OpenPose keypoints in this area, and to the simplicity of the OpenSim model in the upper-body part. Indeed, currently all pelvis, lumbar, and thoracic angles are solely determined by the detection of the hip keypoints on the lower part, and of the shoulder and neck keypoints on the upper part. Moreover, the skeletal model did not allow for any scapulo-thoracic degree of freedom. In addition to the sacro-lumbar joint, upper-body Pearson's correlation coefficients were mostly very good (>0.85) in most planes in walking and running. The range of motion error remained below 1° for shoulder and elbow angles in walking, while it reached almost 5° in the shoulder and 2° in the elbow in running. The mean error in the sagittal plane was below 1° in the shoulder angle in walking, but it reached 10° in the elbow; conversely, in running it reached 9° in the shoulder but remained under 1° in the elbow. In cycling, upper-body Pose2Sim angles were mostly not correlated to marker-based ones, and ROM errors and mean errors were much worse than in other tasks. Moreover, the Bland–Altman analysis showed that the data is heteroscedastic: the spread and magnitude of the errors varied as the joint angle evolved.

In conclusion, Pose2Sim does not evaluate some anatomical joint angles in the upper body, and is generally less accurate than for the lower body. This is mostly due to the lack of keypoints OpenPose detects. To date, OpenPose offers hand and face models but no detailed model of the upper limb exists. Pose2Sim could be used with other pose estimation algorithms, including custom ones leveraging DeepLabCut [**?**, **?**], for example, although it would involve manually labeling a large training dataset. This would enable the use of a more anatomically realistic kinematic model, such as the one proposed by [**?**] for the shoulder girdle.

B.2.1 Accuracy: Upper-body results for the walking task

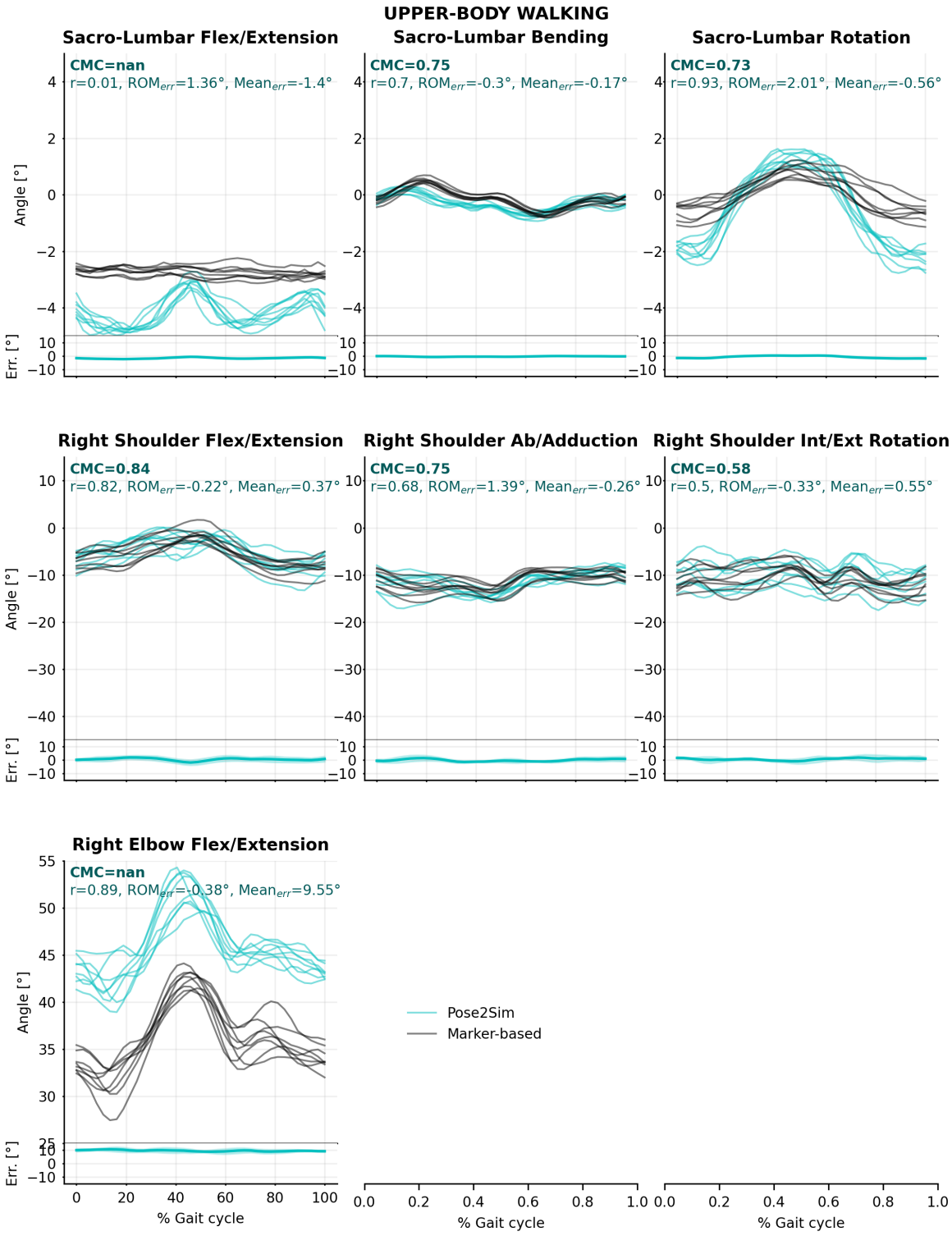


Figure B.5: Pose2Sim (cyan) and marker-based (black) sacro-lumbar and upper-body joint angles for the walking task. Coefficient of multiple correlation (CMC) is indicated, and broken down into, respectively, Pearson's coefficient (r) for correlation assessment, range of motion errors (ROM_{err}) for gain, and overall mean error ($Mean_{err}$) for offset. Mean error and standard deviations are also represented at the bottom of the graphics.

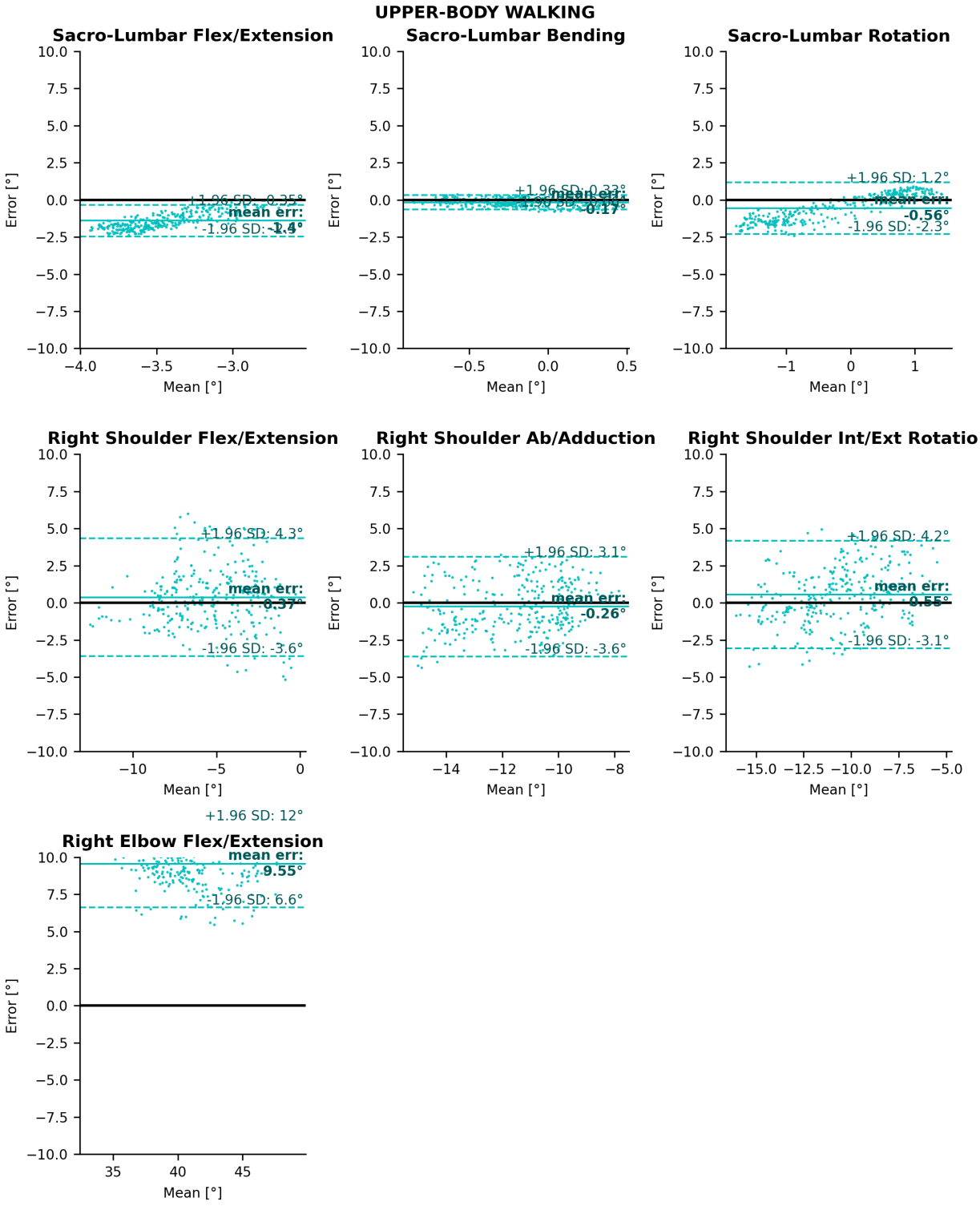


Figure B.6: Bland–Altman analysis of sacro-lumbar and upper-body joint angle errors for the walking task. Mean bias is represented as a horizontal solid, bold line, and 95% limits of agreement are represented as dotted lines.

B.2.2 Accuracy: Upper-body results for the running task

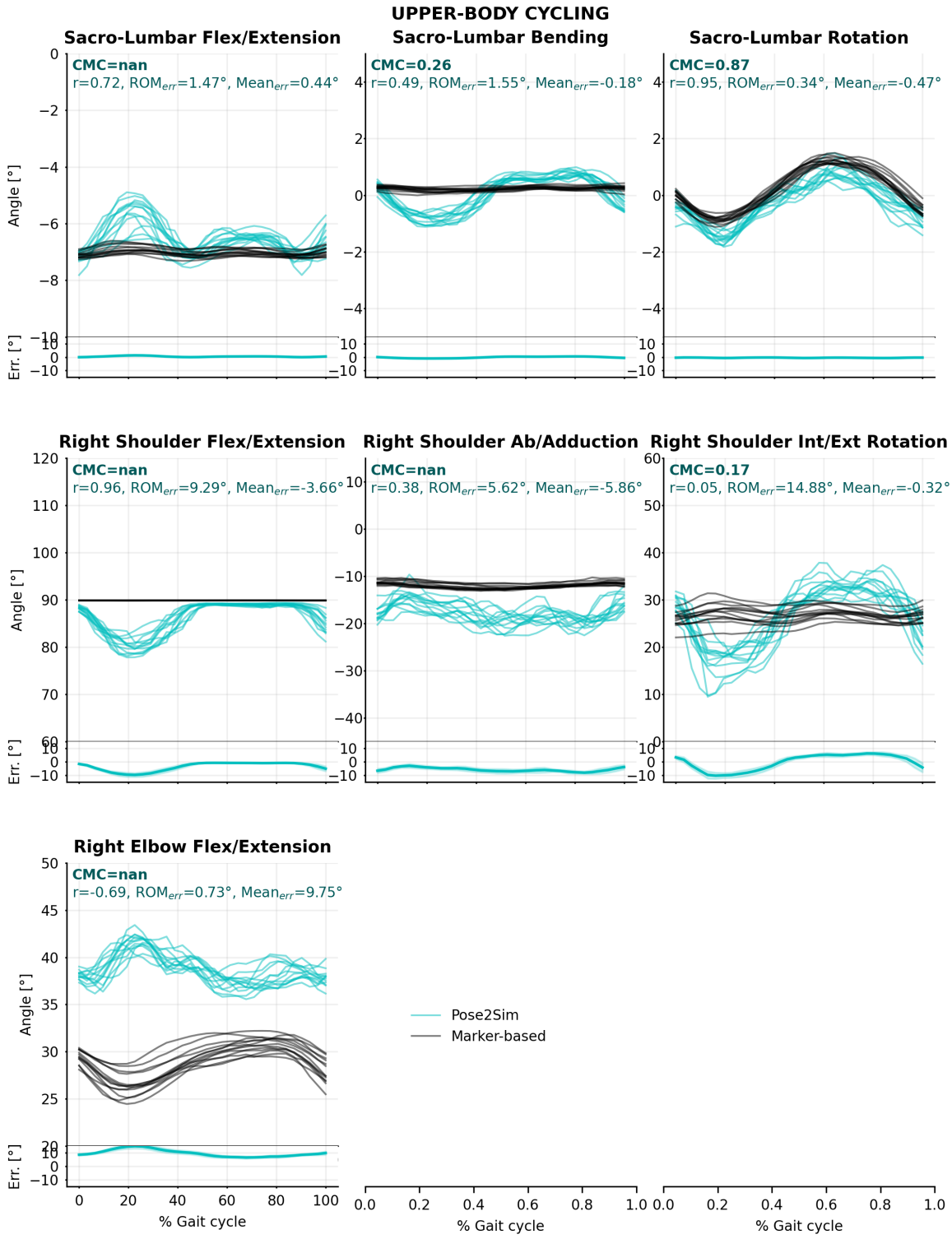


Figure B.7: Pose2Sim (cyan) and marker-based (black) sacro-lumbar and upper-body joint angles for the running task. Coefficient of multiple correlation (CMC) is indicated, and broken down into, respectively, Pearson's coefficient (r) for correlation assessment, range of motion errors (ROM_{err}) for gain, and overall mean error ($Mean_{err}$) for offset. Mean error and standard deviations are also represented at the bottom of the graphics.

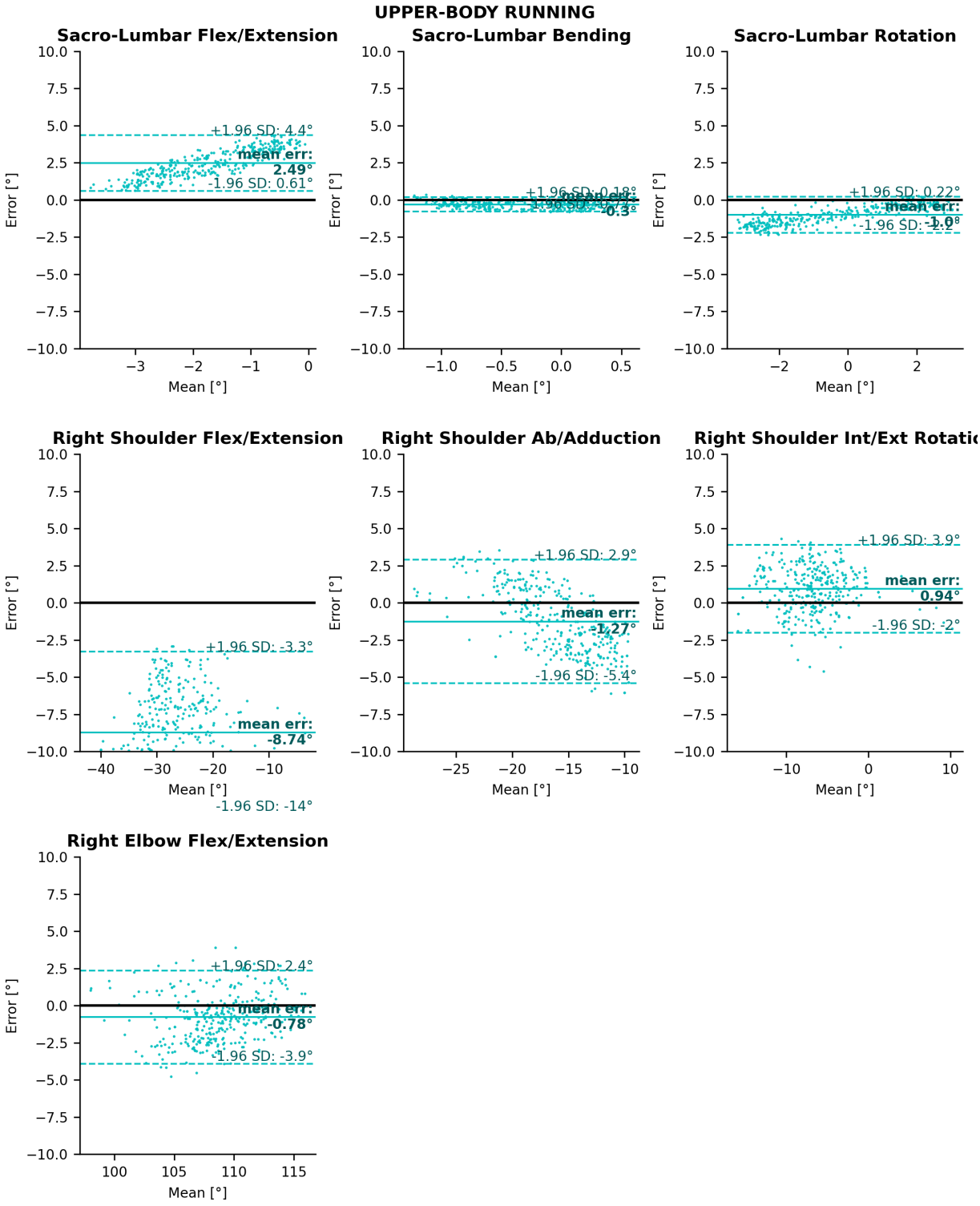


Figure B.8: Bland–Altman analysis of sacro-lumbar and upper-body joint angle errors for the running task. Mean bias is represented as a horizontal solid, bold line, and 95% limits of agreement are represented as dotted lines.

B.2.3 Accuracy: Upper-body results for the cycling task

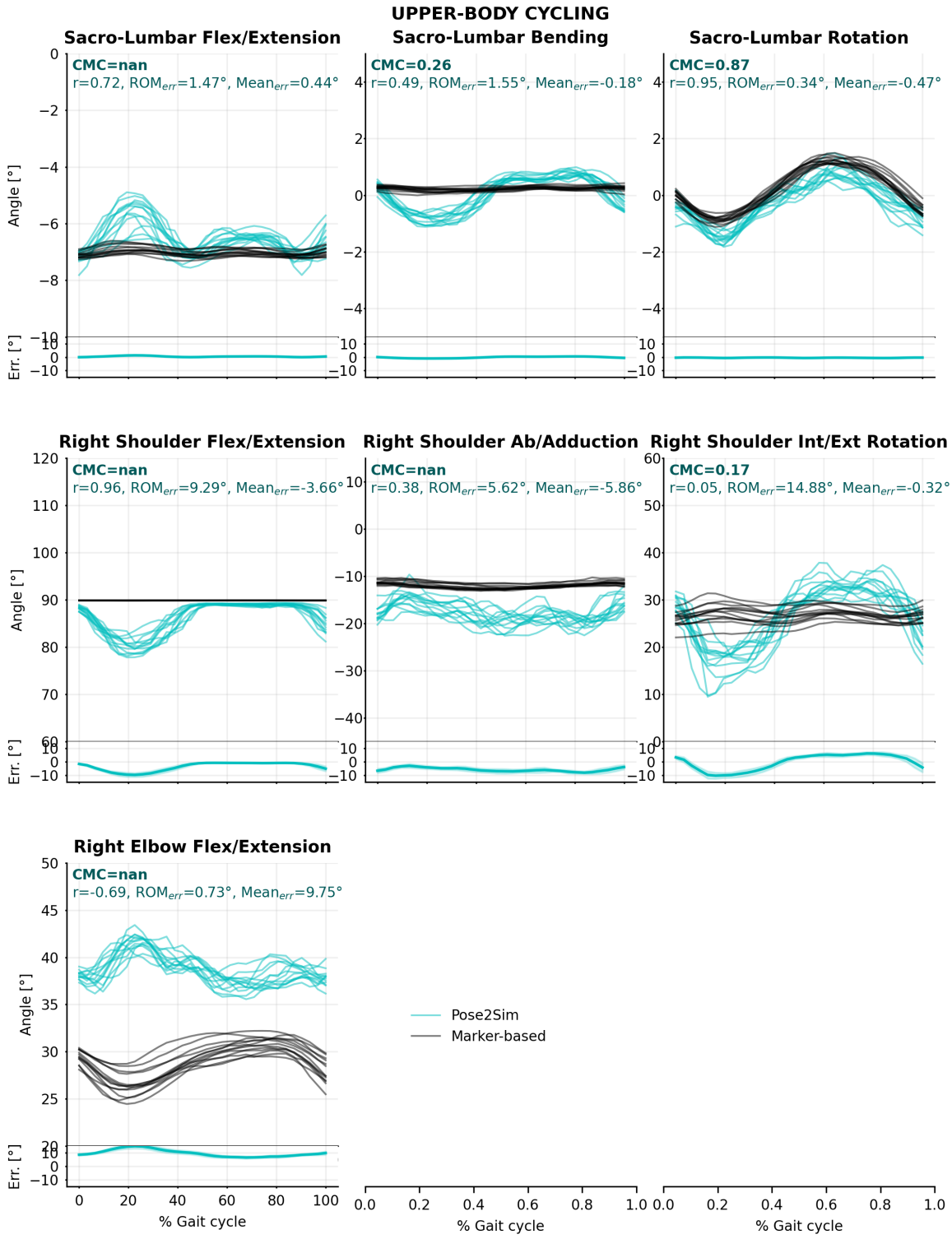


Figure B.9: Pose2Sim (cyan) and marker-based (black) sacro-lumbar and upper-body joint angles for the cycling task. Coefficient of multiple correlation (CMC) is indicated, and broken down into, respectively, Pearson's coefficient (r) for correlation assessment, range of motion errors (ROM_{err}) for gain, and overall mean errors ($Mean_{err}$) for offset. Mean error and standard deviations are also represented at the bottom of the graphics.

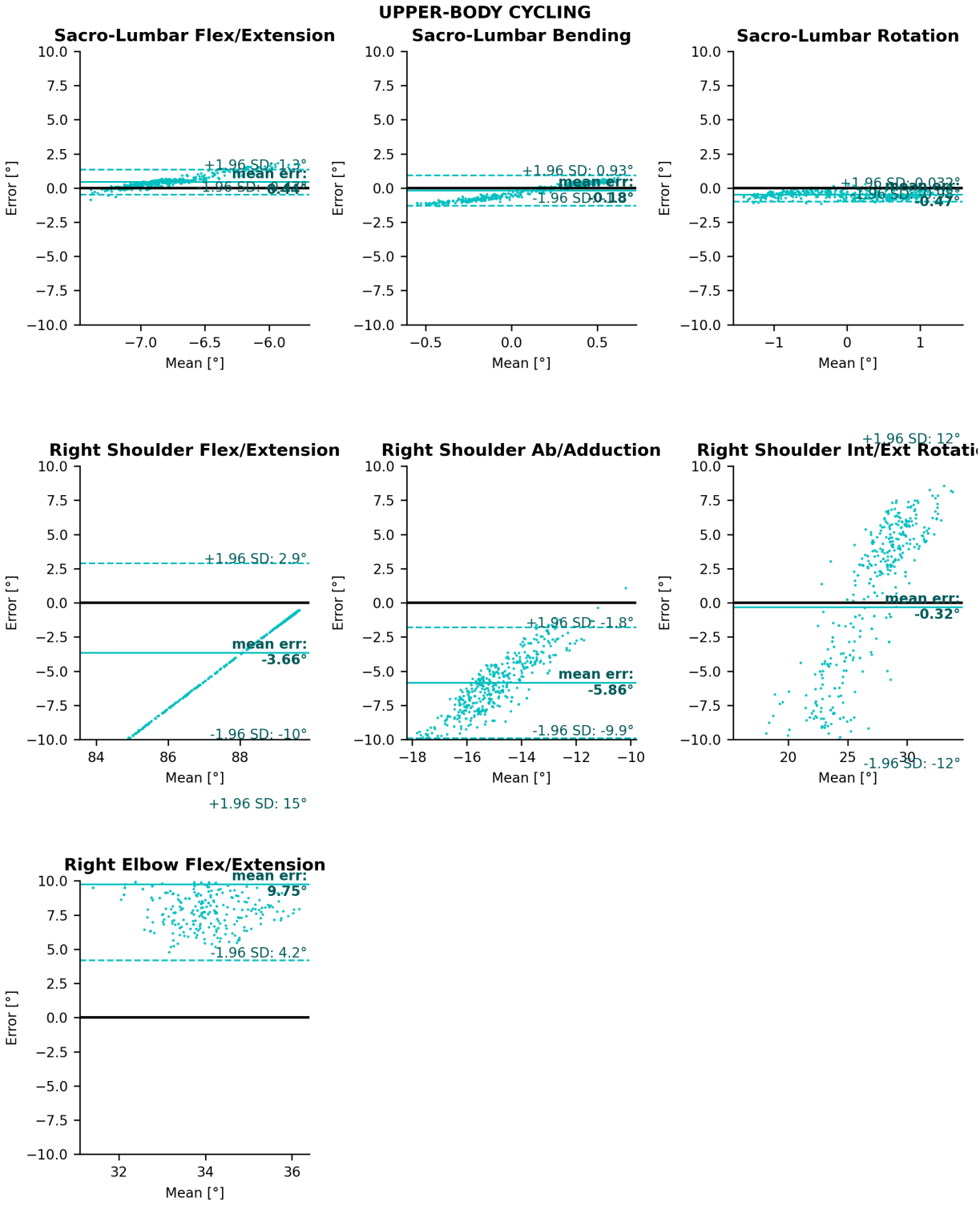


Figure B.10: Bland–Altman analysis of sacro-lumbar and upper-body joint angle errors for the cycling task. Mean bias is represented as a horizontal solid, bold line, and 95% limits of agreement are represented as dotted lines.

Appendix C : Protocol

Summary here on protocol.

Contents

Protocol

C.1 Section 1

C.1.1 Sous section 1

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

C.1.2 Sous section 2

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Confusing concepts - Disambiguation

Some terms used along this thesis may lead to confusion to an uninitiated person, as they are closely related, but describe slightly different concepts. This glossary compares and clarifies them.

Markers vs. Keypoints:
Markers are objects attached to the body of the user, often small and round. They are used to indirectly track the posture of a user. Markers can sometimes be active and emit light, instead of passive and reflect light. They can also be anchored to the bone, for a more precise analysis exempt from Soft Tissue Artifact (STA). A marker-based analysis commonly refers to the use of passive markers, glued to the skin of the user.
Keypoints are points of interest detected by a machine learning model, either in 2D or in 3D space. They can estimate the location of a human joint, or of a body part, or else of any point of interest of any object.
Once triangulated, keypoints can be treated as **virtual markers**, for example while running inverse kinematic (IK) analysis. Calculated markers are another kind of virtual markers, which can for example represent joint centers, or markers which have fallen off the body during capture.

Markerless vs. Sensorless:
Markerless systems don't use any markers, but they can use other sensors, such as Inertial Measurement Units (IMUs).
In contrast, **sensorless** systems don't involve any wearable markers nor any sensors. This also goes for sensorless dynamic analysis, which does not use any force sensors, or for muscle activation analysis, which does not use any Electromyography (EMG) sensor.
Note that approaches only using video (RGB), or depth-field video (RGB-D) sensors are usually considered sensorless, as they do not involve any particular alteration to the environment or behavior of the user.

Gold standard vs. Silver standard:
A **gold-standard** method refers to the most accurate of the currently available methods, which can be used as a reference to assess the performance of other methods. Bone-anchored pins, Magnetic Resonance Imaging (MRI), biplanar videoradiography, or 3D ultrasounds are usually considered as gold-standard techniques for posture analysis. In contrast, marker-based methods cannot be rigorously considered as such, since they are sensitive to Soft Tissue Artifact (STA) and to positioning variability from the operator. Hence, they are referred to as **silver-standard** methods.
However, the previously listed methods are not available for most sports activities, and thus cannot be considered as standards at all. Hence, in this case, marker-based approaches are sometimes referred to as gold-standard, as opposed to IMU or markerless systems, which are then considered as silver-standard.

Silhouette vs. Shape:
A **silhouette** is the 2D cutout of a human being on an image. Silhouette segmentation is a specific case of **object segmentation**. A **shape** is the equivalent of a silhouette in 3D space. A human shape can be used to characterize pose, like 3D keypoints do, but also morphology. A **mesh** is the geometric representation of a shape, divided into smaller 2D cells (Figure **??**).

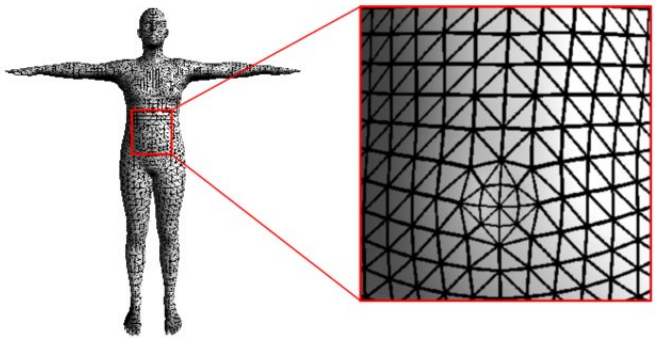


Figure C.1: The SMPL mesh model characterizes a human shape. Image from [?].

Machine learning vs. Deep learning:

Machine learning and Deep Learning, as well as Artificial Intelligence, Artificial Neural Networks, or Convolutional Neural Networks, are sometimes used interchangeably. However, they do not exactly refer to the same concepts (Figure **??**).

Artificial Intelligence (AI) concerns the general concept of machines being able to perform tasks that would seem to require human intelligence, or even to sense, reason, and act accordingly.

Machine learning (ML) is a subset of AI, and refers to the concept of machines being able to learn and improve as they are exposed to data. It is a data-driven approach, as opposed to knowledge-driven ones.

Artificial Neural Networks (ANN) are a specific way of performing ML, by using a network of units inspired from the natural neuron, and thus mimicking the brain.

Deep Learning (DL) refers to an ANN with more than 3 layers of neurons.

Convolutional Neural Networks (CNN) are a type of ANN which is particularly suited for image processing.

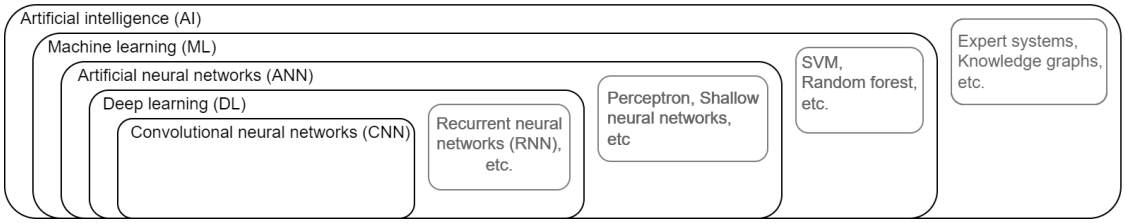


Figure C.2: Diagram of Artificial intelligence, Machine learning, Artificial neural networks, Deep learning, and Convolutional neural networks. The greyed out elements will not be described.

Dataset vs. Labels:

A **dataset** is a collection of data used to train a machine learning model.

These data need to be labelled prior to training. **Labels** and **annotations** can generally be considered as synonyms. Certain datasets come with several kinds of annotations: keypoints, instance segmentation, or image classification for example. Note that a same dataset can be annotated with several keypoint instances, e.g., one focusing on hands, and the other one on the full body.

Once labeled, a dataset can be fed to a deep learning **architecture**, or **algorithm**, upon which is appended a **feature extractor**, or **backbone**, specific to the chosen labelling convention. A **model** results from this training, which can then be used to predict the labels of new data. Hand-labeled dataset can be used as **benchmarks** for assessing the model accuracy or speed.

In the everyday use, datasets, labeling or annotation, architecture, and models, tend to be used in a confusing way. For instance, the standard OpenPose model has been trained on the COCO and MPII dataset, annotated with COCO, MPII, and additional foot keypoints, on the OpenPose architecture. This results in the OpenPose body_25 model.

Single-view vs. Multi-view:
A **multi-view** (or multiview) system uses several cameras, while a **single-view**, or **monocular** one, uses only one. While multi-view systems are always used to infer 3D information, monocular ones can either focus on 2D or 3D information retrieval.
Single-image, or **single-frame** approaches, specifically analyze one single image. In practice, video approaches often perform their analysis image by image.

Kinematics vs. Kinetics:
Kinematics describes the motion of points (e.g., heel marker), bodies (e.g., foot segment), or systems of bodies (e.g., human whole body). This can involve studying positions or angles, as well as linear and angular velocities or accelerations. However, kinematics studies motion only in a geometric way, without considering its causes. It deals with the *equations of motion*.
On the other hand, **kinetics** deals with the causes of motion. According to Newton's second law of motion, the forces that induce the motion of a body are obtained by multiplying its acceleration by its mass. Hence, unlike kinematics, kinetics takes masses into account (or more generally, inertial properties). it involves considering external forces (e.g., gravity or ground reaction forces), internal forces (e.g., joint moments or individual muscle forces), or energy or power (e.g., kinetic energy). More generally, kinetics deals with the *laws of motion*.
Dynamics includes both kinematics and kinetics. However, note that in practice, kinetics and dynamics are often used as synonyms. **Statics** is a special case of dynamics where the sum of forces is zero, and consequently, so is the acceleration. Thus, either the velocity of the object is constant, or it is not moving at all.

Spatio-temporal parameters vs Joint kinematics:
Spatio-temporal parameters are handcrafted indicators used for the kinematic analysis of a specific task. Among other, they can include step length or cadence for gait analysis, or the reach of the jab or the excursion of the center of motion for boxing analysis.
Joint kinematics is the study of joint motion, such as joint angle or joint laxity, or joint velocities and accelerations.
Technically, both spatio-temporal parameters and joint kinematics fall under the umbrella of kinematic data; however, in the customary usage, kinematics describes joint kinematics.

Forward kinematics vs. Inverse kinematics:
In the multi-body analysis of human motion, **forward kinematics** generally involves finding the pose $\overrightarrow{x}$ of segments of known dimensions, from all joint angles $\overrightarrow{q}$. In other words, it aims to solve the equation $\overrightarrow{x} = f(\overrightarrow{q})$. In this case, there are as many known joint parameters as unknown pose variables. Thus, an exact solution can be solved analytically, with simple trigonometric calculations. The location of markers can also be inferred, since they are attached to rigid body segments (soft tissue artifacts being ignored). Unlike in robotics, joint angles are hardly directly measurable in human motion analysis, which makes forward kinematics of little interest in this field.
Conversely, **inverse kinematics** deals with finding joint angles from the pose of some segments (determined with marker positions), and thus with solving the equation $\overrightarrow{q} = f^{-1}(\overrightarrow{x})$. Instead of prescribing the movement joint by joint, the idea is to provide an end target, that will be reached with a certain joint configuration. In this context, segment poses are usually not all known, and

there are then more unknown joint degrees of freedom than known segment poses: the system is called redundant. This implies that the target can be reached with different joint configurations. Consequently, no exact solution can be found, and one needs to turn to numerical methods in order to find an "optimal" solution.

Direct kinematics is usually considered to be a synonym of forward kinematics. However, when there are exactly as many unknown variables as known parameters, joint angles and segment pose can be equally solved both ways. Hence, the term direct kinematics is sometimes also employed to describe the analytical calculation of joint angles from the position of markers.

"Design, evaluation, and application of a workflow for biomechanically consistent markerless kinematics in sports"

"Conception, évaluation, et application d'une méthode biomécaniquement cohérente de cinématique sans marqueurs en sport"

Résumé

Les contraintes inhérentes au mouvement sportif ne permettent généralement pas son analyse à l'aide de marqueurs. En conséquence, des méthodes sans marqueurs sont de plus en plus proposées. L'une d'entre elles, Pose2Sim, a été développée et publiée en open-source dans le cadre de cette thèse. Pose2Sim fait le lien entre les deux programmes les plus utilisés de leurs domaines respectifs : OpenPose pour l'estimation de pose 2D, et OpenSim pour la cinématique 3D physiquement réaliste. La robustesse de cette solution a été évaluée, ainsi que sa précision. Pose2Sim a aussi été mis à l'épreuve de conditions expérimentales plus complexes, malgré lesquelles les indicateurs de performance clé en boxe ont pu être correctement évalués. Enfin, des séquences de départ de BMX ont été analysées. Il a été démontré que si l'estimateur de pose 2D est assez performant, il est possible d'utiliser Pose2Sim pour fournir aux entraîneurs la cinématique de l'ensemble {vélo+pilote}.

Mots-clés :

Abstract

Marker-based motion capture is hardly compatible with the inherent constraints of sports. As a consequence, numerous markerless methods are being proposed. One of them, Pose2Sim, has been developed and published during this doctoral program. Pose2Sim bridges the most renown programs in their respective fields: OpenPose for 2D pose estimation, and OpenSim for physically consistent 3D kinematics. The robustness of this solution has been evaluated, as well as its accuracy. Pose2Sim has also been tested in more challenging experimental conditions, in spite of which boxing key performance indicators have been correctly evaluated. Finally, BMX start sequences have been analyzed. It has been shown that if the 2D pose estimator was good enough, it would be possible to use Pose2Sim to provide joint {bike+pilot} kinematics to coaches.

Keywords : Markerless motion capture; Sports performance analysis; Kinematics; Computer vision; OpenPose; OpenSim; Python package