FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Human Operator Tracking System for Safe Industrial Collaborative Robotics

Eduardo Fonseca

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Pedro Gomes da Costa, PhD

Co-Supervisor: Luís Freitas Rocha, PhD

July 27, 2021

Abstract

With the advent of the Industry 4.0 paradigm, manufacturing is shifting from mass production towards customisable production lines. While robots excel at reliably executing repeating tasks in a fast and precise manner, they lack the now desired versatility of humans. Human-robot collaboration (HRC) seeks to address this issue by allowing human operators to work together with robots in close proximity, leveraging the strengths of both agents to increase adaptability and productivity.

Safety is critical to user acceptance and the success of collaborative robots (cobots) and is thus a focus of research. Typical approaches provide the cobot with information such as operator pose estimates or higher-level motion predictions to facilitate adaptive planning of trajectory or action. Therefore, locating the operator in the shared workspace is a key feature.

This dissertation seeks to kickstart the development of a human operator tracking system that provides a pose estimate that can be used as input for safety features like speed and separation monitoring. Two state-of-the-art bottom-up methods for human pose estimation in two-dimensional RGB images, OpenPifPaf and OpenPose, are evaluated using a custom dataset. The results are then analysed considering real-time capability in the use case of a single operator performing industrial assembly tasks in a collaborative robotic cell equipped with a robotic arm.

While both methods are acknowledged as viable solutions, based on the resulting evaluation metrics, OpenPose was selected for its higher recall and 55% higher inference speed, in favour of OpenPifPaf's higher precision, which is believed to compromise the method's robustness to occlusions and therefore, the operator's safety. It is also proposed that future work should consider: training custom models for each method; fusing 2D pose estimates from different views of the scene into a refined 3D pose estimate that is robust to occlusions; and further investigate the effect of image input resolution on performance.

Resumo

Com o advento do paradigma da Indústria 4.0, a manufatura tem transitado da produção em massa para linhas de produção customizáveis. Apesar de robôs serem excelentes na execução rápida e precisa de tarefas repetitivas, não apresentam a versatilidade humana que é agora desejada. A colaboração humano-robô (HRC) procura resolver este desafio, permitindo que operadores humanos trabalhem em grande proximidade com robôs, aproveitando os pontos fortes de ambos os agentes para aumentar a adaptabilidade e a produtividade.

A segurança é um fator crítico para a aceitação do utilizador e para o sucesso dos robôs colaborativos (cobots), sendo, portanto, um foco de investigação. Abordagens típicas fornecem ao cobot informações como estimativas da localização do operador, ou previsões de mais alto nível do seu movimento para facilitar o planejamento adaptativo de trajetória ou ação. Sendo assim, localizar o operador na área de trabalho compartilhada é vital.

Esta dissertação procura iniciar o desenvolvimento de um sistema de monitorização do operador humano que forneça uma estimativa de posição que possa ser usada para recursos de segurança como monitorização de velocidade e distância. Dois métodos estado-da-arte para estimativa de posição humana em imagens RGB bidimensionais, OpenPifPaf e OpenPose, são avaliados usando um conjunto de dados personalizado. Os resultados são então analisados considerando execução em tempo real, no caso de uso de um único operador executando tarefas de montagem industrial numa célula robótica colaborativa equipada com um braço robótico.

Embora ambos os métodos sejam reconhecidos como soluções viáveis, com base nas métricas de avaliação alcançadas, o OpenPose é selecionado pelo seu maior *recall* e 55% maior velocidade de inferência, a favor da maior precisão do OpenPifPaf, que se acredita comprometer a robustez do método para oclusões e, portanto, a segurança do operador. Propõe-se ainda que trabalhos futuros considerem: o treino de modelos customizados para cada método; a fusão de estimativas de posição 2D de diferentes perspetivas da cena para obtenção de uma estimativa 3D refinada e robusta a oclusões; e a investigação adicional do efeito da resolução da imagem de entrada no desempenho dos métodos avaliados.

Acknowledgements

I would like to gratefully acknowledge my supervisors, Professor Pedro Gomes da Costa and Professor Luís Freita Rocha, for choosing me to conduct this work, and for advising and encouraging me throughout this year.

My sincere gratitude to Carlos Costa for his contribution as the main subject of the images used throughout this dissertation, as well as to everyone at INESC TEC that supported this research.

My best wishes to all the members of my extended NEEEC family.

My deepest love:

To my QueerBots - Gala Humblot-Renaux, my inspiration to become the best researcher I can be, and Carolina Gomez Salvatierra, for accompanying me through hardship, even if remotely.

To all my friends that walked this path to graduation alongside me, and to all among them who made me who I am today.

To my family for their infinite support, and most of all to my parents, João and Susana, for their wisdom and reassurance.

And, finally, to new beginnings.

Eduardo Fonseca

Contents

List of Figures

List of Tables

Abbreviations and Symbols

2D	Two-Dimensional
3D	Three-Dimensional
AA	Average Accuracy
AP	Average Precision
AR	Average Recall
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
FN	False Negative
FP	False Positive
FPS	Frames Per Second
HRC	Human-Robot Collaboration
HRI	Human-Robot Interaction
JPE	Joint Position Error
JSON	JavaScript Object Notation
MPJPE	Mean Per Joint Position Error
PCK	Probability of Correct Keypoint
RGB-D	Red Green Blue-Depth
ROS	Robot Operating System
TN	True Negative
TP	True Positive

Chapter 1

Introduction

With the advent of the Industry 4.0 paradigm, manufacturing is shifting from mass production towards customisable production lines. While robots excel at executing repeating tasks in a fast and precise manner, they lack the versatility that is currently desired and only easily achieved by humans. Recent research in the field of Human-robot collaboration (HRC) has seeked to address this issue by allowing human operators to work together with robots in close proximity, leveraging the strengths of both agents to increase adaptability and productivity [2].

Although humans and robots working together symbiotically is an enticing prospect, technical constraints have long restricted what can be accomplished. In particular, ensuring the safety of human operators in a shared workspace is still a limiting factor for collaborative robotics [1]. While safety solutions for collaborative robots - also known as *cobots* - are reliable and trusted in laboratories, they suffer from low acceptance among users of manufacturing systems, in large part due to a lack of standards and solutions for specific industrial applications [2].

In order to ensure applicability to industrial manufacturing settings, cobots must efficiently adapt to the unpredictability of human behaviour while adhering to strict safety standards. An often crucial first step to achieve this is to reliably locate the operator in the shared workspace in order to enable safety features like collision avoidance. In fact, typical methods involve tracking the operator in some form, often providing the cobot with information such as pose estimates - or even higher-level motion predictions - to facilitate adaptive planning of trajectory or action [3]. Therefore, a tracking system that provides a three-dimensional estimate of the human operator's pose throughout operation is an essential feature for safe industrial cobots.

1.1 Use Case

While safety is a concern for all types of cobots, we will focus on a specific use case: one of INESC TEC's research projects, *ScalABLE 4.0*, which sought to address other challenges brought on by the Industry 4.0 paradigm shift. Namely, the project's stated main objective was "the development

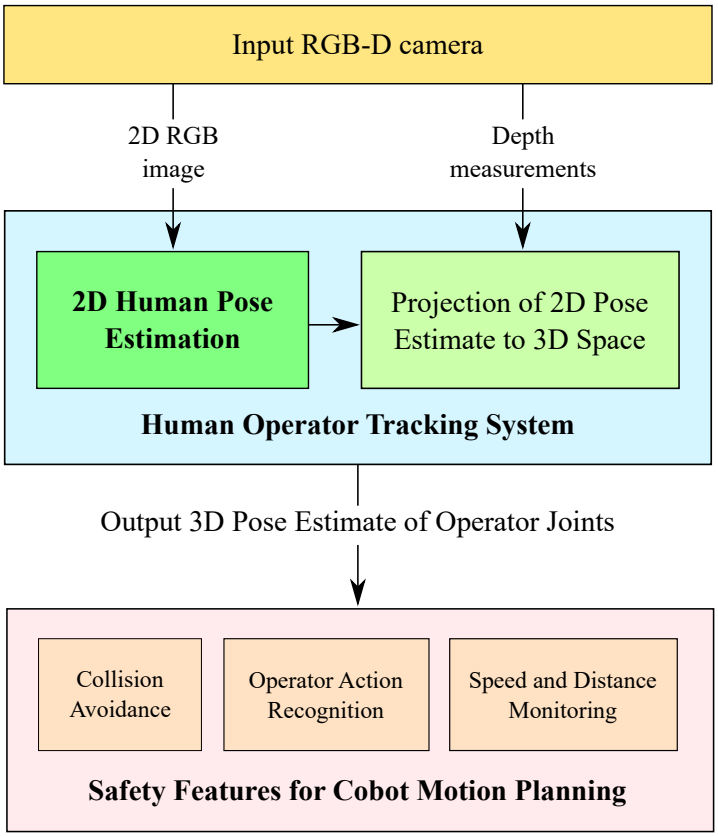


Figure 1.1: Block diagram overview of high-level features related to a human operator tracking system for safe collaborative robots.

and demonstration of an open scalable production system framework (OSPS) that enables optimization and maintenance of production lines ‘on the fly’, through visualization and virtualization of the line itself" [4].

In particular, the ’PSA’ use case focused on a motor assembly plant and the integration of robotic solutions into a pilot line, in a less intrusive manner [5]. One of the solutions that was developed consisted of an augmented reality interface that enabled HRC by teaching operators how to coordinate tasks with the robotic platform for complex assembly operations [6].

As was previously addressed, safety is crucial in order to establish the operator’s trust in the cobot. While projection mapping allows the agents to share the workspace and work in close proximity, the lack of active safety measures still pose a threat to collaboration in industrial scenarios. As such, the proposed work will expand on *ScalABLE 4.0* by kickstarting the development of a human operator tracking system for its collaborative robotic workstation.

1.2 Tracking System and Scope

Figure 1.1 provides a high-level overview of how a human operator tracking system can be implemented, using RGB-Depth images as input in order to output a 3D pose estimate of the operator's joints. In turn, this information can be used to implement safety features for motion planning or control in collaborative robots like, for example, collision avoidance, operator action recognition, or speed and distance monitoring.

This dissertation seeks to kickstart the development of a human operator tracking system with similar architecture to what is depicted in figure 1.1. To accomplish this, the scope of this work is focused on the selection of a 2D human pose estimation methods that is appropriate to implement in this system for our use case.

1.3 Outline and Contributions

This document is structured as follows: Chapter 1 contextualises the work in the use case of safety in industrial collaborative robotics. Chapter 2 provides an overview of background and related works regarding HRC and Safety, and briefly explores some higher level approaches to cobot trajectory planning and control which rely on the output of a tracking system. Chapter 3 clarifies key concepts related to the field of human pose estimation research and introduces the methods that we considered for implementation in a tracking system, OpenPifPaf and OpenPose. Chapter 4 addresses the methodology of this work, detailing how these human pose estimation methods were evaluated for use in the *ScalABLE 4.0* collaborative robotic workstation. Chapter 5 presents the results obtained from the evaluation procedure of the previous chapter and discusses them in the context of our use case. Finally, in Chapter 6, we review the work, draw conclusions, and propose future work.

Chapter 2

Background and Related Work

2.1 Human-Robot Collaboration

Human-Robot Collaboration (HRC) is a highly active field of research in robotics, in large part thanks to the demand of more flexible robotic solutions for industrial manufacturing. Collaborative robots, otherwise known as a *cobots* [7], are desirable since they leverage the advantages of both robot and human agents - respectively, speed and precision in repeating task execution, as well as flexibility and cognitive ability [2]. In HRC, the human operator works in close proximity with the robot, in a shared workspace, towards accomplishing the same common goal [1].

Wang et al. [2] provide an overview of the current state of HRC in manufacturing, as well as insight into industrial scenarios and multimodal robot control through high-level commands like gestures or voice commands. Four different human-robot relationships - Coexistence, Interaction, Cooperation, and Collaboration - are also characterized in table 2.1 according to five different perspectives:

1. Workspace: If the human and robot share a workspace, without separation through physical or virtual fences.

2. Contact: If the human and robot have direct physical contact.

3. Shared working task: If the human and robot share operations when working towards the same objective.

4. Simultaneous process: If the human and robot work at the same time, wether on the same or different tasks.

5. Sequential process: If the human and robot do not have overlapping operations and instead work one after the other in the temporal scale.

These definitions are particularly useful as there is no consensus on how to distinguish these complex scenarios. In fact, human-robot interaction (HRI) is also considered a different branch of robotics research and is used as a more general term. A simpler way to distinguish HRC from

Table 2.1: Features of different human-robot relationships [2].

		Coexistence	Interaction	Cooperation	Collaboration
Shared	Workspace		✓	✓	✓
	Direct contact		✓		✓
	Working task		✓		✓
	Resource			✓	✓
	Simultaneous process	✓		✓	✓
	Sequential process		✓	✓	

HRI is based on wether the human and robot are working together towards a common goal - while it is required in collaboration, it is not necessarily so in interaction [1].

2.2 Safety

The issue of safety is critical to the success of collaborative robotics and a main focus of research [1]. While safety solutions for collaborative robots (cobots) in laboratories are reliable and trusted, the lack of standards and safety certification for industrial applications results in low acceptance among users [2]. As such, the alternative of physically separating agents through safety fences has traditionally been favoured due to its relative simplicity. In this approach, optical or mechanical barriers are used to detect if the operator breaches the robot's delineated workspace boundaries, reducing the speed or entirely stopping the robot's motion accordingly.

While physical separation ensures safety, it also severely limits collaboration through interaction, either with or without direct contact. Therefore, in order to overcome this limitation, the cobot must efficiently adapt to the unpredictability of human behaviour while adhering to strict safety standards. In particular, one method to achieve safety in a shared workspace consists of monitoring the operator for the purposes of collision avoidance [3].

Additionally, market demands have emphasised research on HRC in assembly tasks with robotic manipulators. Thus, the proposed work will focus on the development of a continuous operator monitoring system, based on computer vision techniques, for the purpose of adaptive trajectory planning of cobots in assembly tasks, according to the operator's pose.

As was discussed in chapter 1, safety solutions for HRC suffer from insufficient standardisation and certification, especially for industrial applications. Thus, research in recent years has heavily focused on this issue.

Mehta et al. [8, 9] achieves real-time 3D pose estimation from RGB images, using convolutional neural networks (CNNs). However, VNect and XNect are not robust to big occlusions and are significantly less accurate than other slower methods.

In [3], an extensive survey of methods for safe HRI was conducted, which were then classified into four major categories: Control, Motion planning, Prediction, and Consideration of Psychological Factors. The consideration of both physical and psychological safety is also highlighted

as a critical factor for HRI success. Similarly, [10] highlights human trust in robotic collaborators and reviews different trust modelling methods.

Villani et al. [1] reviewed HRC solutions in industrial settings, regarding safety, user interface and applications. Both [1] and [11] address the currently available international safety standards and certification procedures. In particular, both present the four levels of collaborative operation modes identified in the ISO 10218-1/2:2011 [12, 13] standards, pictured in figure 2.1, and implementations of safety that apply these principles.

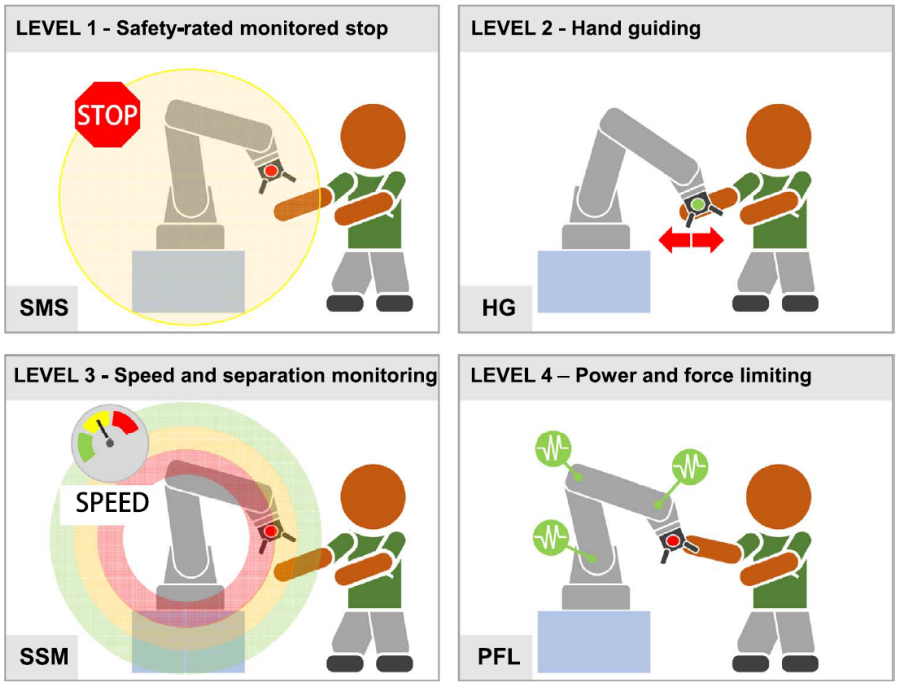


Figure 2.1: The collaborative operation modes identified by safety standards 10218-1/2:2011 [1].

These can be summarised as such:

1. Safety-rated Monitored Stop (SMS) - the robot stops moving if the operator ocupies the shared workspace.
2. Hand Guiding (HG) - the operator teaches the robot positions by moving it.
3. Speed and Separation Monitoring (SSM) - a vision system monitors the operator and adjusts robot motion according to the occupied zone.
4. Power and Force Limiting (PFL) - the robot's motor power and force are constrained to guarantee operator safety.

2.3 Control

Cheng et al. [14] proposes an integrated HRC framework with plan recognition and trajectory prediction modules to generate safe and efficient robot actions. The authors leverage the hierarchical

relationships between plans and trajectories in order to predict both the human's future motion - for the purpose of collision avoidance and safe trajectory planning - and action sequence. This plan recognition algorithm is based on Neural Networks and Bayesian interference and shows promising results, reducing average task completion time in an industrial assembly task experiment. Furthermore, the article provides useful insight into categorization of intelligent cobots:

1. Low-level collision avoidance algorithms that consider humans as simple moving obstacles.
2. Mid-level efficient cooperation through human trajectory prediction and task plan recognition.
3. High-level full collaboration through mode selection and automatic task assignment.

In [15], an integrated HRI framework is proposed for collaborative assembly in manufacturing cells. Also, [16, 17, 18] address implementations of some of the concepts from the previous subsections.

Chapter 3

Human Pose Estimation and Key Concepts

Enabling safe human-robot collaboration through active safety measures like safety-rated monitored stop or speed and separation monitoring typically requires the knowledge of where the operator is located in the workspace. Human pose estimation, which can be broadly defined as the localisation of a person's body parts or joints, can be used to accomplish this very task. Usually, this is done either in two-dimensional (2D) images or videos, by estimating pose in pixel coordinates; or in the three-dimensional (3D) space by inferring 3D coordinates from 2D pose or through fusion of depth information and/or distance measurements.

While 3D pose estimation is desired, it is still a relatively new field of research with few benchmarks or large datasets available, whereas most state-of-art human pose estimation methods approach the issue exclusively using 2D RGB images as input. Therefore, since 3D pose can be obtained from 2D, we focused our research on 2D RGB methods and their respective implementations.

In this chapter, we give an overview of related work and present key concepts related to the research field of human pose estimation. In section 3.1 we review early influential works and the vital role that deep learning plays in the current landscape. Section 3.2 clarifies the differences between top-down and bottom-up methods and establishes the state-of-the-art. Section 3.3 lists different implementations of human pose estimation methods that were considered for this work and justifies the pre-selection of OpenPifPaf and OpenPose as the methods to evaluate for use in a tracking system for our use case. Finally, section 3.4 introduces several types of metrics that are used to evaluate the performance of the human pose estimation methods and comments on their applicability for our purposes.

3.1 Early Work and the Introduction of Deep Learning

The problem of human pose estimation is often difficult to address due to the high variability of factors that influence it. For example, the human's activity (what they are doing), physical

attributes (their body type, skin tone), or even the context they are in (if they are in a crowd of other people, or a dimly lit room) are all variables that may have to be considered, depending on the desired use case. This has resulted in the development of a variety of methods with different strengths and weaknesses.

Early work often relied on hand-crafted features like tree models [19], pictorial structures [20, 21] or histograms of oriented gradients (HOG) [22, 23, 24]). However, these methods could not meet increasing demands of detection speed, accuracy and robustness [23, 25]. In turn, DeepPose [26] introduced the first application of Deep Neural Networks (DNN) to the human pose estimation problem, formulating it as a DNN-based regression to joint coordinates, and using a cascade of Convolutional Neural Networks (CNN) to solve it. This work kickstarted a tidal wave of deep learning methods, those based on DNN [27], that outperformed non-deep state-of-the-art methods [28, 19, 20, 22] by wide margins and quickly developed to become the state-of-the-art of the field.

Dang et al. [25] conducted a comprehensive survey on deep learning based human pose estimation methods, presenting useful methodology-based taxonomy. First, they distinguish between single-person and multi-person pipelines. Single-person pipelines usually work by inferring keypoints of human parts based on a bounding box that is provided in advance. Evidently, this constraint is too restrictive for our use case because it essentially requires us to accomplish our goal of locating the operator in the 2D image in advance, all to simply refine this pose using the detected keypoints. In contrast, multi-person pipelines must achieve the goal of keypoint detection without *a priori* knowledge of either number or location of people, thus requiring the detection of the humans themselves. As such, our analysis focuses on multi-person methods for human pose estimation.

3.2 Top-down and Bottom-up State-of-the-art Methods

Dang et al. [25] established an important distinction between two different types of approaches to 2D multi-person pose estimation, top-down and bottom-up. These are related to the order in which a method identifies people and their respective body part keypoints, also known as joints:

- **Top-down approaches** first estimate the location of the person or body part regions and then regress joint estimates. While these methods require a person detector, recent advances have resulted in outstanding performance. However, these methods struggle with occlusions and scenarios where person bounding boxes overlap, like crowded scenes.

  Examples of top-down methods include: PoseNet [29], RMPE [30], CFN [31], Mask R-CNN [32], CPN [33] MSRA [34]; as well as 3D pose estimation methods like Mesh R-CNN [35], V-nect [8],and X-nect [9].

- **Bottom-up approaches** first identify all keypoints/joints in the image and then associate them to build a skeleton. Recent methods often perform on par with top-down approaches, while benefiting from wider applicability due to increased robustness to occlusions.

Examples of bottom-up methods include: DeepCut and DeeperCut [36, 37], OpenPose [38, 39], MultiPoseNet [40] and OpenPifPaf [41, 42].

In recent years, Mask R-CNN [32] and OpenPose [39] have been cited as state-of-the-art methods for 2D multi-person human pose estimation, as well as cornerstones for the top-down and bottom-up approaches respectively. Kreiss et al. [41, 42] proposed OpenPifPaf, a new method that extends the notion of fields discussed by Cao et al. [38] and which they claim outperforms Mask R-CNN and OpenPose, especially in lower resolution images and densely crowded scenes where humans partially occlude each other.

3.3 Pre-selection of Methods to Evaluate

Having surveyed the field of human pose estimation, we identified different implementations of some of the methods discussed and considered them as possible candidates to evaluate for use in a tracking system for our use case:

- **Detectron2**[1] [43] is Facebook AI Research's library of detection and segmentation algorithms and the successor of Detectron[2] and maskrcnn-benchmark[3], which implements Mask R-CNN among other methods.
- **OpenPose**[4] [39] - the official implementation of OpenPose with extended detection for fance, hand and foot keypoints with the pre-trained model *BODY_25*.
- **OpenPifPaf**[5] [42] - the official implementation of OpenPifPaf, which provides three different pre-trained models - *resnet50*, *shufflenetv2k16* and *shufflenetv2k30* - using the COCO training dataset [44], that are further discussed in subsection 4.4.1.
- **Intel RealSense Skeleton Tracking SDK**[6] - a proprietary method integrated into the Intel RealSense software development kit for their depth cameras that they claim to offer "fast and highly accurate 2D and 3D human pose estimation with 18 joints" without GPU.

In order to limit the scope of the work, two 2D human pose estimation methods were selected for evaluation: **OpenPifPaf** and **OpenPose**. Both methods rely on bottom-up approaches, sharing similarities in methodology that might streamline the discussion of results, and report outstanding performance metrics in widely-used benchmarks, such as MPII [45] or COCO [44]. Furthermore, OpenPose is considerably more mature and widely adopted by users, but both implementations facilitate integration by offering thorough documentation and APIs for Python development.

[1] https://github.com/facebookresearch/detectron2
[2] https://github.com/facebookresearch/Detectron/
[3] https://github.com/facebookresearch/maskrcnn-benchmark/
[4] https://github.com/CMU-Perceptual-Computing-Lab/openpose
[5] https://github.com/openpifpaf/openpifpaf
[6] https://www.intelrealsense.com/skeleton-tracking/

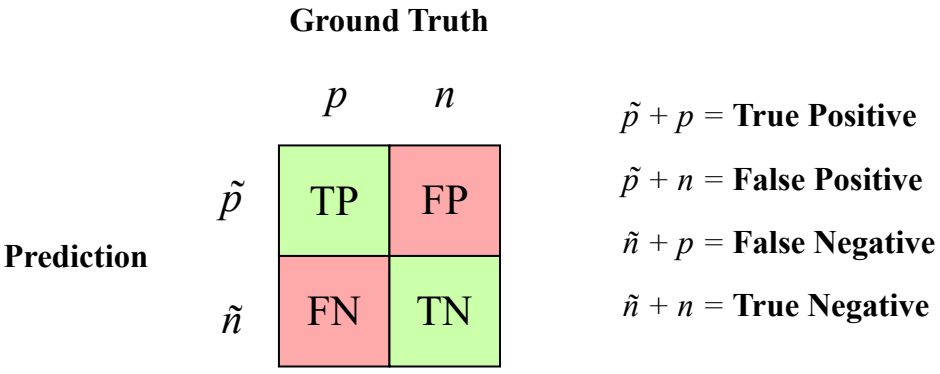


Figure 3.1: Binary confusion matrix.

These methods were selected over the top-down approaches provided by Detectron2 since they are supposedly more robust to occlusions - a challenge that is particularly relevant to our use case because the operator is frequently occluded by the equipment of the collaborative workstation. Moreover, while direct integration with RGB-D cameras is a desirable feature, the Intel RealSense Skeleton Tracking SDK was excluded in favour of methods that are open-source.

In the future, extending this work to include the evaluation of these other methods would provide a more thorough perspective and motivate a more informed choice of human pose estimation method for an operator tracking system.

3.4 Evaluation Metrics

In order to properly evaluate the performance of a human pose estimation method, we must first choose evaluation metrics that will clearly convey information that is relevant to our use case. As the choice of performance metric is highly dependant on the problem that is being analysed, there is no single metric that can address all facets and real-world constraints of pose estimation. Therefore, this section presents the different performance metrics that were considered for implementation in the evaluation procedure of this work. These can all be broadly categorised as measures relating to either accuracy or computational speed/cost.

3.4.1 Precision, Recall and Accuracy

Traditionally, the performance of a Deep Learning (DL) model is characterized by how well it generalises to data that it has not encountered before, that is, whether it can accurately predict the correct output from new input data. To assess this, most metrics are, in some way, based on a confusion matrix, which is a matrix that categorises the model's predictions into correct or incorrect classes, with dimensions equal to the number of possible ground truth or prediction outcomes. Figure 3.1 depicts a binary confusion matrix - the simplest, and also most common, type of confusion matrix, of dimensions 2 by 2, where both the ground truth and prediction can either be positive or negative.

As depicted, each prediction can be classified as: a true positive (**TP** - positive prediction, $\tilde{p}$, and positive ground truth, p); a false positive (**FP** - positive prediction, $\tilde{p}$, and negative ground truth, n); a false negative (**FN** - negative prediction, $\tilde{n}$, and positive ground truth, p); or a true negative (**TN** - negative prediction, $\tilde{n}$, and negative ground truth, n). We can determine the total numbers of: positive predictions $\tilde{\mathbf{P}} = TP + FP$; negative predictions $\tilde{\mathbf{N}} = FN + TN$; positive ground truths $\mathbf{P} = TP + FN$; and negative ground truths $\mathbf{N} = FP + TN$.

Using these variables, the following metrics can be defined:

- **Precision** measures the proportion of positive predictions that were correctly identified: $Prec = \frac{TP}{\tilde{P}} = \frac{TP}{TP+FP}$
- **Recall**, also known as *true positive rate*, measures the proportion of ground truth positives that were correctly predicted as positive: $Recall = TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$. We can also define other complimentary rates:
  - *False negative rate*: $FNR = 1 - TPR = \frac{FN}{P}$
  - *True negative rate*: $TNR = \frac{TN}{N} = \frac{TN}{TN+FP}$
  - *False positive rate*: $FPR = 1 - TNR = \frac{FP}{N}$
- **Accuracy** measures the proportion of correct predictions made over the ground truth total: $Acc = \frac{TP+TN}{P+N}$

Powers [46] introduced the concepts of informedness, markedness, correlation and significance as a way to address bias related to the use of recall and precision. However, in our use case, these two metrics may be sufficient to determine which human pose estimation method performs better at correctly identifying the human operator's skeletons and joints. In fact, variations of Average Precision (AP) and Average Recall (AR) are frequently used as performance metrics for pose estimation methods [32, 41, 39] and benchmarks [45, 44, 47].

3.4.2 Probability of Correct Keypoint

An important step in the quantitative evaluation of human pose estimation was the protocol described by Ferrari et al. [48] that relied on the Probability of a Correct Pose (PCP) metric, which measured the percentage of correctly localized body parts.

Yang et al. [49] addressed limitations of PCP by introducing the Probability of Correct Keypoint (PCK), a threshold metric that defines a candidate keypoint to be correct if it falls within $\alpha \cdot max(h,w)$ pixels of the ground truth keypoint. Since h and w are, respectively, the height and width of the bounding box and α is an arbitrarily selected value between 0 and 1, the use of this metric explicitly requires that the people in test images are annotated with a tightly cropped bounding box. Several variations of PCK have been proposed but we highlight PCKh, introduced by Andriluka et al. [45] with the "MPII Human Pose" benchmark, which modifies the matching threshold to be equal to 50% of the head segment length.

Similarly to recall and precision, PCK and its variants are, ultimately, measures of whether keypoints are being correctly predicted or not. However, PCK differentiates itself by providing a useful quantitative definition of what a true or false prediction is, in a way that is mostly robust to challenges like foreshortening. Nonetheless, it is still a fairly limited solution since it relies on an arbitrarily defined scalar factor (α), as well as the additional requirement of annotated bounding boxes. Thus, because it scales with the size of the human bounding box, it is particularly useful if testing with a large dataset and high variance of sizes of humans in frame.

3.4.3 Mean Per Joint Position Error

In 3D human pose estimation, another metric that is commonly used to determine a method's accuracy is Mean Per Joint Position Error (MPJPE). This metric is typically defined as the average of the physical distance between a predicted joint and its ground truth counterpart. For our purposes of evaluating human pose estimation in 2D images, this metric can be defined as the average euclidean distance in pixels between the predicted and ground truth joints.

MPJPE has the distinct advantage of clearly defining a quantitative measure of estimation accuracy that can be easily implemented with standardized input. Another interesting metric that is similarly based on distance is Object Keypoint Similarity (OKS) [50], which was introduced in the COCO 2017 Keypoint Detection Task [51]. While worthy of mention for the relevance of the annual COCO challenges, OKS was considered unnecessarily complex for our purposes.

3.4.4 Inference Speed, Frame Rate and Computational Cost

While the accuracy of a prediction is extremely important, our use case relies on real-time operation, thus requiring the human pose estimation method to output its predictions quick enough for use in other tasks, such as cobot motion planning for safety purposes. The meaning of real-time operation is context-dependant and discussed later in section 5.2. Regardless, achieving it requires a compromise between accuracy and speed.

Therefore, a standard metric for evaluating human pose estimation methods in tracking applications is inference speed, which is defined as the average inference time per image - the time it takes to output all pose estimates for an image. Additionally, we can measure how many image frames a method can analyse per second by computing the frame rate, measured in frames per second (FPS), based on inference speed.

Furthermore, while inference speed already addresses the most typical measure of computational cost, computation time, recording the usage of computing resources, such as GPU memory, during execution can be a useful complimentary metric.

Chapter 4

Methodology

This chapter details how the selected human pose estimation methods, OpenPifPaf [41, 42] and OpenPose [38, 39] were evaluated for use in a tracking system for the *ScalABLE 4.0* collaborative robotic workstation, clarifying, among others, the methodology of this work. Section 4.1 establishes the conditions and constraints of the setup in question that justify several decisions made throughout development. Section 4.2 summarises our assumptions and goals resulting from the previous section and describes the steps taken to build custom datasets for evaluation. Section 4.3 relates how the images in each custom dataset were annotated to provide a ground truth to which to compare the predictions of OpenPifPaf and OpenPose. Section 4.4 details how we obtained each method's pose predictions for all images in the custom datasets. Finally, section 4.5 clarifies the evaluation procedure that processes these predictions into results that can be analysed. Additionally, appendix A provides some implementation details regarding the Python scripts that were created for this work[1], as well as examples of the files that are obtained and mentioned throughout this chapter.

4.1 Setup

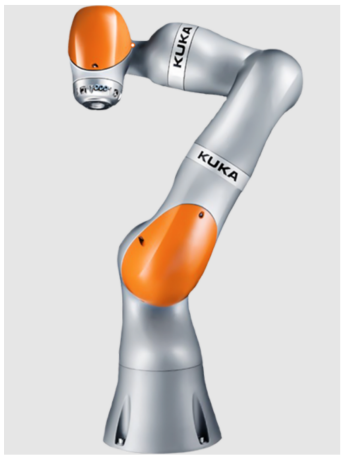
As in real industrial use cases, the development of safety features is constrained by several limitations, both in terms of the technologies used and the scenario in question. While different methods may be applied to several settings, the one which will optimize our system's success in our specific use case must be determined. Therefore, we must clearly establish what limitations this work was developed under, as well as which conditions led to each of the decisions that were made throughout development.

[1] These scripts are publicly available at https://github.com/eduardocaldasfonseca/cobot-monitor/

4.1.1 Collaborative Robotic Workstation

The *ScalABLE 4.0* collaborative robotic workstation is located in INESC TEC's iiLab. It is typically mounted with a KUKA LBR iiwa 14 R820 robot[2], a robotic arm model that benefits from several user safety features. These can be categorized in the *Hand Guiding* and *Power and Force Limiting* collaborative operative modes that were previously discussed in section 2.2. In fact, KUKA claims that these render obsolete many of the safety precautions that are traditionally associated with industrial robots, such as safety fences.

Alternatively, when the KUKA robot is unavailable, the Universal Robots UR5[3] collaborative robotic arm may replace it instead. The two models are depicted in figure 4.1, alongside some technical specifications according to Villani et al. [1] that may prove useful.



KUKA LBR iiwa 14 R820

- DOFs: 7
- Payload: 14 kg
- Reach: 820 mm
- Repatability: ±0.15 mm
- Weight: 30 kg
- Velocity Joints: 70 → 180 °/s

UR5

- DOFs: 6
- Payload: 5 kg
- Reach: 850 mm
- Repatability: ±0.1 mm
- Weight: 18.4 kg
- Velocity: 1000 mm/s

Figure 4.1: KUKA LBR iiwa 14 R820 (left) and UR5 (right) robots and respective technical specifications, according to Villani et al. [1].

Additionally, an Intel RealSense D435 RGB-Depth Camera[4] was provided as one of the sensors to be installed in the workstation. This is a stereo depth camera that outputs RGB video at a frame rate of 30 Frames Per Second (FPS) and a frame resolution of 1920 × 1080 pixels, as well as

[2] https://www.kuka.com/en-de/products/robot-systems/industrial-robots/lbr-iiwa
[3] https://www.universal-robots.com/products/ur5-robot/
[4] https://www.intelrealsense.com/depth-camera-d435/

depth frames at up to 90 FPS and 1280×720 pixels, using active IR. These technical specifications in particular are useful benchmarks for input image resolution and inference speed requirements.

Regarding software, the *ScalABLE 4.0* collaborative station is currently using the Melodic distribution[5] of the Robotic Operating System (ROS) 1 software framework - an open-source robotics middleware of widespread use in both academic and industrial settings - in the Ubuntu 18.04 LTS (Bionic Beaver) release[6] of the Linux OS family, as well as the INESC TEC ROS-based software framework for industrial robot control. It is expected that these will eventually be upgraded into their most recent long-term support releases - respectively, ROS 1 Noetic[7] and Ubuntu 20.04[8].

4.1.2 Hardware and Software Architecture

While iiLab was available, the work was developed using another computer outside the laboratory, in part due to restrictions related to the COVID-19 pandemic. This computer is equipped with the following:

- Hardware:
  - CPU - Intel Core i5-6600K @ 3.50GHz
  - GPU - NVIDIA GeForce GTX 1060 (with 6070MiB)
- Software:
  - Ubuntu 18.04.05
  - Python 3.6.9
  - OpenCV 4.5.1
  - OpenPifPaf 0.12.10
  - OpenPose 1.3
  - CUDA version 11.3 (for NVIDIA GPU)

When necessary, we prioritised matching the collaborative robotic workstation's setup and the requirements of the human pose estimation methods in question. For example, the Ubuntu distribution matches that of the workstation, and the CUDA GPU drivers were installed with regards to the OpenPose requirements.

It is worth noting that the OpenPose documentation[9] highly recommends the use of cuDNN, the NVIDIA CUDA Deep Neural Network library[10] that is used to accelerate GPU performance. While it is believed that this would greatly impact the results obtained regarding the inference speed of OpenPose, technical difficulties in its installation prevented us from using it.

[5] http://wiki.ros.org/melodic
[6] https://releases.ubuntu.com/bionic/
[7] http://wiki.ros.org/noetic
[8] https://releases.ubuntu.com/20.04/
[9] Installation section of the OpenPose Doc can be found here
[10] https://developer.nvidia.com/cudnn

Having established the technologies that are used and some of the restrictions that were faced, we must now clarify how the evaluation datasets were built.

4.2 Dataset Building

Based on our setup and what was discussed in chapter 3, we can now narrow down the extent of our analysis, which will allow us to, among others, build a custom dataset that is useful for evaluation. The following subsection 4.2.1 summarises our assumptions and goals.

4.2.1 Assumptions and Goals

- **Sensor type and data fusion** - Since an RGB-Depth camera was provided, only single-view methods for human pose estimation in 2D RGB images are considered. As was previously discussed in chapter 3, while it is not in scope, the fusion of depth information to obtain 3D joint coordinates in the future - or even further fusion with other type of sensor data, such as laser scan point cloud - is considered.
- **Point of view** - Initially, only one camera is available to be mounted. However, mounting multiple cameras in order to obtain and fuse several perspectives of the scene is desirable.
- **Robustness to occlusions** - The system must be robust to, at least, partial occlusion since, during operation, the robotic arm may obstruct the camera's view of the human operator.
- **Number of operators** - Assuming a single operator is working with the cobot, multiple person detection is not required. However, robustness to multiple people in frame is still desirable since, typically, many human workers are present in industrial workspaces, even if only in the background.
- **Real-time operation** - The system must run online, in real-time, to be useful for safety features in industrial applications.

4.2.2 Image Collection

In order to evaluate a human pose estimation method for use in a tracking system for our collaborative workstation, a dataset must be built that is applicable to this use case or, even better, that accurately represents a wide array of real-life scenarios relating to it. Moreover, it must also take into account rarer occurrences, namely occlusions, motion blur, foreshortening of limbs, and varying number of humans in frame.

For this purpose, we had originally intended to use the RealSense D435 camera to record a custom dataset of RGB-D frames, including several different people that would simulate the same assembly task in collaboration with the robotic arm of the workstation. However, not only was the KUKA robotic arm unavailable and UR5 not mounted - which would prevent us from recording an accurate workflow and the most common type of occlusions - time and sanitary lockdown-related restrictions would severely limit the variety of data that could be gathered, as well as introduce factors, like the use of face masks, that might impact the results obtained.

Therefore, we instead opted to build the custom dataset from footage that had been recorded previously to create a demonstration video[11] of one of the features of the collaborative robotic workstation. The unedited footage captures one of INESC TEC's researchers performing assembly and then disassembly operations of a motor engine, in collaboration with the KUKA robotic arm - our desired use case. Even though evaluation with a dataset consisting entirely of the same operator restricts the generalisation of our results, as it does not account for factors such as varying body types and skin tones, this solution also provides unique opportunities regarding point of view.

Namely, the scene was filmed from two different angles: in one, the camera is positioned to the operator's left, above eye level and angled down, capturing the workspace and both agents; while in the other, the camera is to the right of the operator but at approximately eye level and closer to parallel to the ground. Each of these videos has a duration of around 27 minutes, comprising a total of 55 minutes of footage. These were converted from video into two different subsets of RGB images in the *JPG* file format, and named according to the position of the camera relative to the operator's point of view - creating the 'left' and 'right' datasets respectively.

(a) Operator occlusion in 'left' dataset.

(b) Operator occlusion in 'right' dataset.

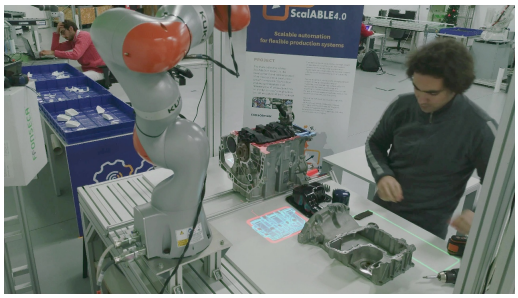
Figure 4.2: Example images from 'left' and 'right' datasets depicting operator occlusions.

One frame was selected for every 20 second of footage[12], compiling a total of 167 frames: 84 for the 'left' dataset; and 83 for the 'right' one. These include several occurrences of operator occlusion, motion blur, and varying number of humans in frame, as well as cases of foreshortening where, due to the relative angle of the operator's limbs to the camera, the perception of their depth is limited and they appear shorter - examples of these challenging scenarios from each dataset are depicted, respectively, in figures 4.2, 4.3, 4.4, and 4.5.

However, the fact that the recording was not specifically set up for the purposes of image collection presents additional challenges: not only was the exact location of the cameras not determined, a different model of camera was used that does not provide depth information. Consequently, we are unable to accurately assess a ground truth pose in a real world frame and are limited to evaluating the human pose estimation methods based on pixel distance errors in the image frame. As such, instead of determining the human operator's pose in the 3D workspace,

[11] Demonstration video 'H2020 ScalABLE4.0 - Spatial augmented reality interface for human-robot collaboration' can be found here.

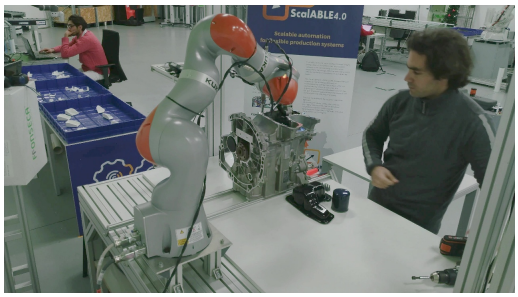
[12] Using a *Video to JPG Converter* software that can be found here.



(a) Motion blur in 'left' dataset.

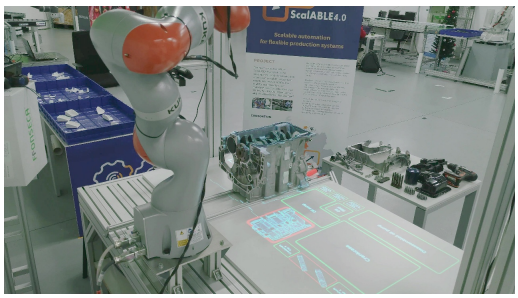
(b) Motion blur in 'right' dataset.

Figure 4.3: Example images from 'left' and 'right' datasets depicting motion blur.



(a) Several humans in frame in 'left' dataset.

(b) Several humans in frame in 'right' dataset.



(c) No humans in frame in 'left' dataset.

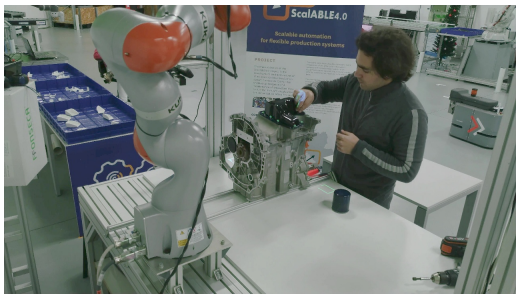
(d) No humans in frame in 'right' dataset.

Figure 4.4: Example images from 'left' and 'right' datasets depicting varying number of humans in frame.

evaluation is based on the accuracy of the pixel position estimate of the human operator's joints that are provided by the two 2D human pose estimation methods.

4.2.3 Image Resizing

Since input size greatly impacts the inference speed of the pose estimation methods in question, the resolution of the images in the datasets must be so that the real-time operation goal is met without compromising the accuracy of the estimate. Therefore, the images extracted from the original video files, which were filmed in a resolution of 3840 x 2160 pixels, were conservatively



(a) Foreshortening in 'left' dataset.

(b) Foreshortening in 'right' dataset.

Figure 4.5: Example images from 'left' and 'right' datasets depicting foreshortening of operator's arms.

downscaled by a factor of 3, using the OpenCV library[13] for computer vision in the Python script 'resize_dataset.py'[14].

The reduced resolution of 1280 × 720 pixels keeps the original aspect ratio of 16 x 9 but matches the resolution of the depth output of the D435 camera, allowing simple mapping of depth measurements to image frame. The resized images were compiled into new datasets, 'left-small' and 'right-small'[15]. These datasets will be used by default so, from here on, the designations 'left' dataset or 'right' dataset refer to these new 'small' datasets unless otherwise stated. The effect of image input size is further discussed in section 5.2.

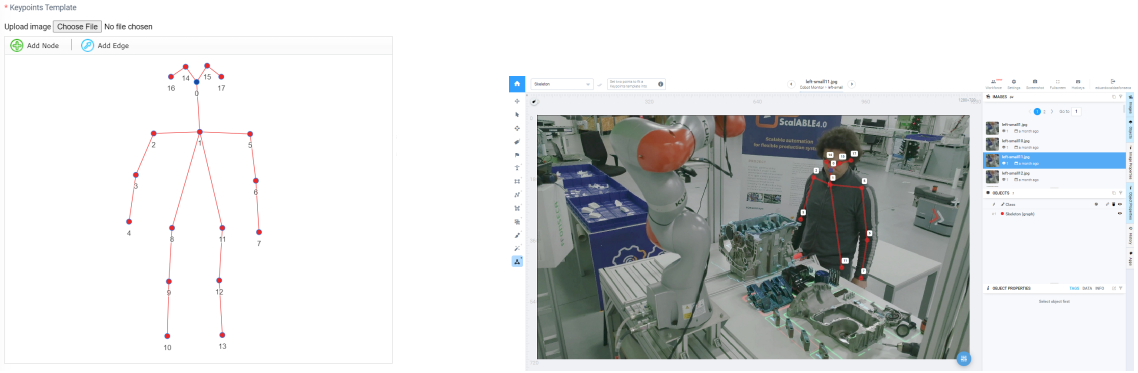
4.3 Ground Truth Annotation

Once the 'left-small' and 'right-small' datasets were built, they had to be annotated to provide a ground truth, i.e., a reference to which the human pose estimates can be compared to for the purposes of evaluation. Using the Supervisely web interface[16], the operator's pose was annotated with a skeleton object - a set of keypoints representing joints, configured according to the template in figure 4.6a - for all images in both datasets.

This annotation skeleton consists of 18 joint keypoints that are shared in the skeleton configurations for the outputs of OpenPifPaf and OpenPose, with the notable exceptions of an absent 'Chest' keypoint in OpenPifPaf and several additional OpenPose keypoints that are ignored - one that is located between the left and right 'Hip' keypoints, and 6 additional foot keypoints specific to the 'BODY_25' model. Figure 4.7 shows the different skeleton keypoint configurations.

It is worth noting that, in all cases where one of the operator's joint is occluded, we opted to omit that particular joint keypoint from the annotation, even if its location could be reasonably approximated. This decision was made because, while an estimate of the joint's pixel location in the two-dimensional image might be correct, the object that is occluding it would prevent a

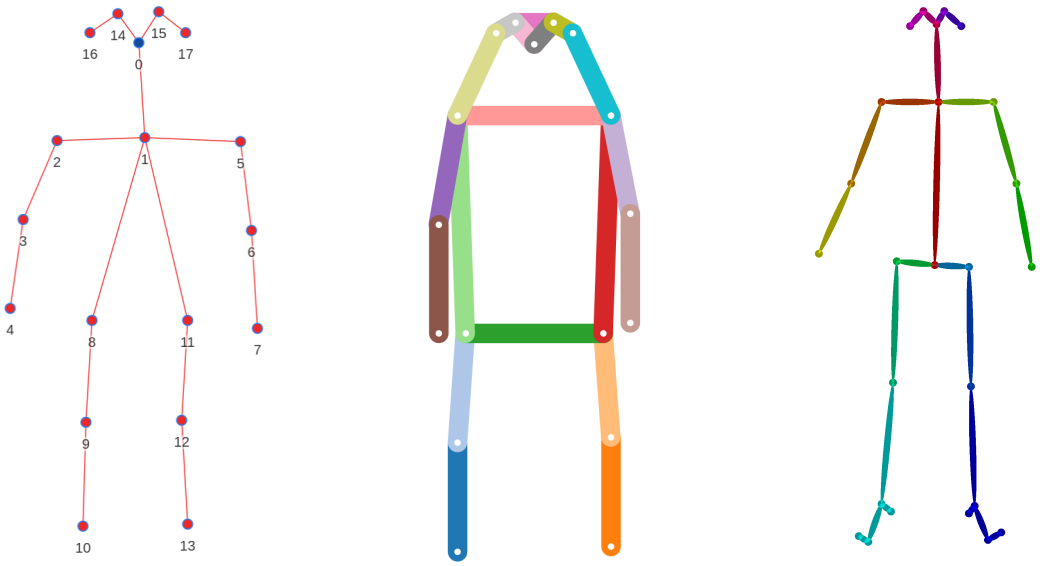
[13] https://opencv.org/
[14] Link to 'resize_dataset.py' script in GitHub repository
[15] The suffix '-small' was added to the dataset names to refer to the resizing.
[16] https://app.supervise.ly/



(a) Skeleton keypoint template. (b) Example of an annotated image.

Figure 4.6: Screenshots of Supervisely web interface used for annotation.

depth camera from correctly determining that point's depth, resulting in an incorrect estimate of the real-world joint position. Figure 4.8 demonstrates one such case where OpenPose made reasonable pose estimates for joints that are occluded by the workstation's projector and the motor that is being assembled. Both of these objects are much closer to the camera than the operator.



(a) Annotation skeleton. (b) OpenPifPaf COCO. (c) OpenPose BODY_25.

Figure 4.7: Skeleton keypoint configurations for (a) Annotation skeleton, (b) OpenPifPaf COCO, and (c) OpenPose BODY_25.

Additionally, the joint keypoints related to the operator's hips were particularly challenging to annotate due to both the clothing that was worn, the operator's orientation in relation to the camera, and the fact that the workstation's table is at approximately the same height as his hips, frequently occluding them.

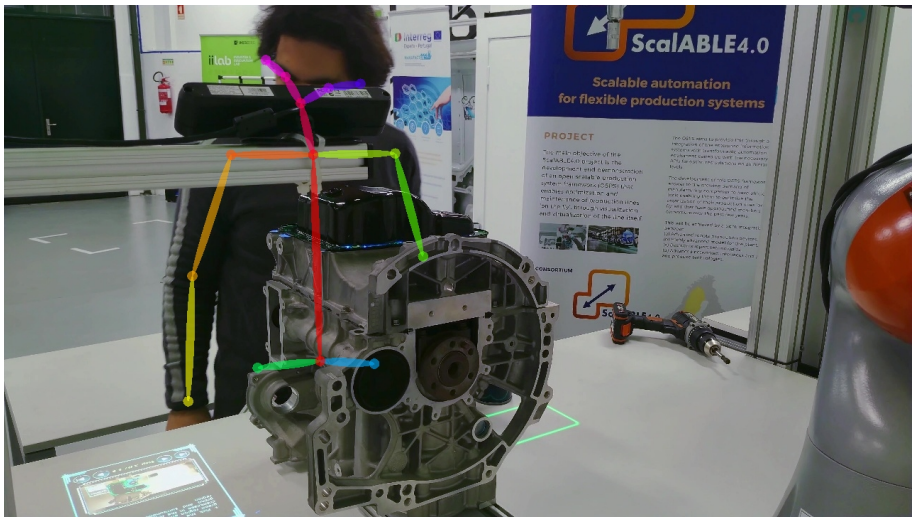


Figure 4.8: Example of rendered frames where OpenPose predicted reasonable estimates for occluded joints.

After concluding the annotation of both datasets, exporting from Supervisely results in, among others, two important types of *JSON* files: a 'meta.json'; and individual files for each annotated image, that are named after them (for example, 'left-small1.jpg.json') and contain the annotated skeleton object with every joint keypoint's pixel location. However, in these individual files, the joint keypoints are not directly referred to by the labels presented in figure 4.6a. Instead, Supervisely generates a code for each and the 'meta.json' file associates this code with the joint's label. This will be relevant later in section 4.5.1 where we will map the different joint labels of the skeletons that Supervisely, OpenPifPaf and OpenPose use. Examples for both types of files mentioned and implementation details are given in appendix A.

4.4 Operator Pose Estimation in Datasets

In order to obtain the operator pose estimates of OpenPifPaf and OpenPose for our 'left-small' and 'right-small' datasets, two Python scripts were created - 'run_openpifpaf.py'[17] and 'run_openpose.py'[18]. These allow the user to customise useful attributes as well as measure the inference time for each image, using the *timeit*[19] Python library to measure execution time.

To facilitate the implementation of the evaluation procedure of section 4.5, these scripts output a standardized set of files using the nomenclature of '*method*_run*X*_' - where *method* is either 'pifpaf' or 'openpose' and *X* is an assigned number to represent that run, i.e., one execution of the script - followed by:

[17] The 'run_openpifpaf.py' script in the GitHub repository can be found here
[18] The 'run_openpose.py' script in the GitHub repository can be found here
[19] https://docs.python.org/3/library/timeit.html

- 'meta.json' - a *JSON* file that contains useful information about the run, such as the version of the pose estimation method, which model was used, among others. This file also contains both the individual and total inference time measurements that are later used to determine inference speed and frame rate.
- 'keypoints.json' - a *JSON* file for each image in the dataset used containing the method's predictions for each joint.
- 'rendered.jpg' - if the *image_creation* variable is enabled, a *JPG* file for each image will be created where the original image is overlayed with the predicted skeleton keypoints. Whenever possible, this feature was disabled so that the rendering of these images would not influence the time measurements.

Examples for each of these types of *JSON* output files can be found in appendix A, as well as additional explanations regarding some of the features implemented in the scripts.

4.4.1 Pre-trained Model Selection

Both OpenPifPaf and OpenPose provide different pre-trained models for pose estimation. Therefore, we conducted some preliminary tests to determine which models should be evaluated.

Checkpoint	AP	AP^M	AP^L	t_{total} [ms]	t_{dec} [ms]
resnet50	**68.2**	65.8	72.7	64	22
shufflenetv2k16	**67.2**	62.7	74.6	51	19
shufflenetv2k30	**71.0**	66.6	78.5	92	16

AP refers to 'Average Precision', with superscripts M and L referring to scale - respectively, 'medium' and 'large' humans. The **t** values refer to inference time averages.

Table 4.1: Performance metrics of the different OpenPifPaf pre-trained models, according to OpenPifPaf Guide.

In the case of OpenPifPaf, their online documentation, OpenPifPaf Guide[20], reports the performance metrics of their pre-trained models on the COCO validation set [44] - *resnet50*, *shufflenetv2k16*, and *shufflenetv2k30* - which are depicted in table 4.1. From our preliminary tests, *shufflenetv2k16* was found to be the fastest among these options and was thus selected.

In turn, OpenPose Doc[21] reports three models: *BODY_25*, *COCO*, and *MPI*. The authors report that *BODY_25* is both more accurate and faster than the remaining models. Additionally, our

[20] https://openpifpaf.github.io/intro.html
[21] https://github.com/CMU-Perceptual-Computing-Lab/openpose/tree/master/doc

preliminary attempts to run OpenPose using the *COCO* and *MPI* resulted in 'Out of memory' error reports. Therefore, almost by default, *BODY_25* was selected as the model used for evaluation of OpenPose.

While this option was considered to be out of scope for this work, it is worth noting that both methods provide ways to train custom models. This prospect should be considered for future work because, while training a custom model could result in overfitting and, thus, a potential loss of applicability to different use cases, it could also result in big improvements to performance.

4.5 Evaluation Procedure of Human Pose Estimation Methods

Having obtained both a ground truth annotation and operator pose estimates from both OpenPifPaf and OpenPose, the output files obtained from each must now be processed in order to obtain the evaluation metrics from 3.4 that we decided to consider - Average Precision (AP), Average Recall (AR), Average Accuracy (AA), Mean Per Joint Position Error (MPJPE).

Figure 4.9 provides an overview of the evaluation procedure that was implemented in the Python script 'eval.py'[22]. In the following subsections, it will be generally explained how this procedure is executed.

4.5.1 Skeleton Extraction and Keypoint Mapping

Initially, each image's corresponding annotation *JSON* file is used to extract the annotated skeleton, using the 'meta' file to translate the Supervisely code of each keypoint into the annotation skeleton joint label. In parallel, the 'keypoints' *JSON* files containing the predictions of either OpenPifPaf or OpenPose are used to extract all the skeletons that were detected by these methods.

Both OpenPifPaf and OpenPose output each skeleton's joint keypoints as an array of values in the format of (x0, y0, c0, x1, y1, c1, ...), where x0 and y0 are the pixel coordinates and c0 is the confidence score of joint 0, while (x1, y1, c1) is the triplet corresponding to joint 1, so and so forth for all joints in their respective skeleton configuration. As such, the order of joints in these arrays are different, thus, the index number does not necessarily correspond to the annotation joint labels. For example, the 'Left ear' keypoint is annotated with the label 17, but it is the joint with index 3 in OpenPifPaf and index 18 in OpenPose.

In order to properly compare the predictions obtained from OpenPifPaf and OpenPose, it is necessary to correctly associate each predicted joint to its annotated counterpart. Therefore, the annotation skeleton of figure 4.7 was designated as the standardised skeleton to which we map every joint keypoint. Table 4.2 is used to map joint designation[23] to the annotation/standardised keypoint labels and to the OpenPifPaf and OpenPose indexes.

[22] The 'eval.py' script in the GitHub repository can be found here

[23] The 'Left' or 'Right' sides in the joint designation are defined from the point of view of the human facing towards us. Which is why, in the keypoint template of the annotation skeleton, the 'Right Shoulder' joint, labelled 2, is on our left.

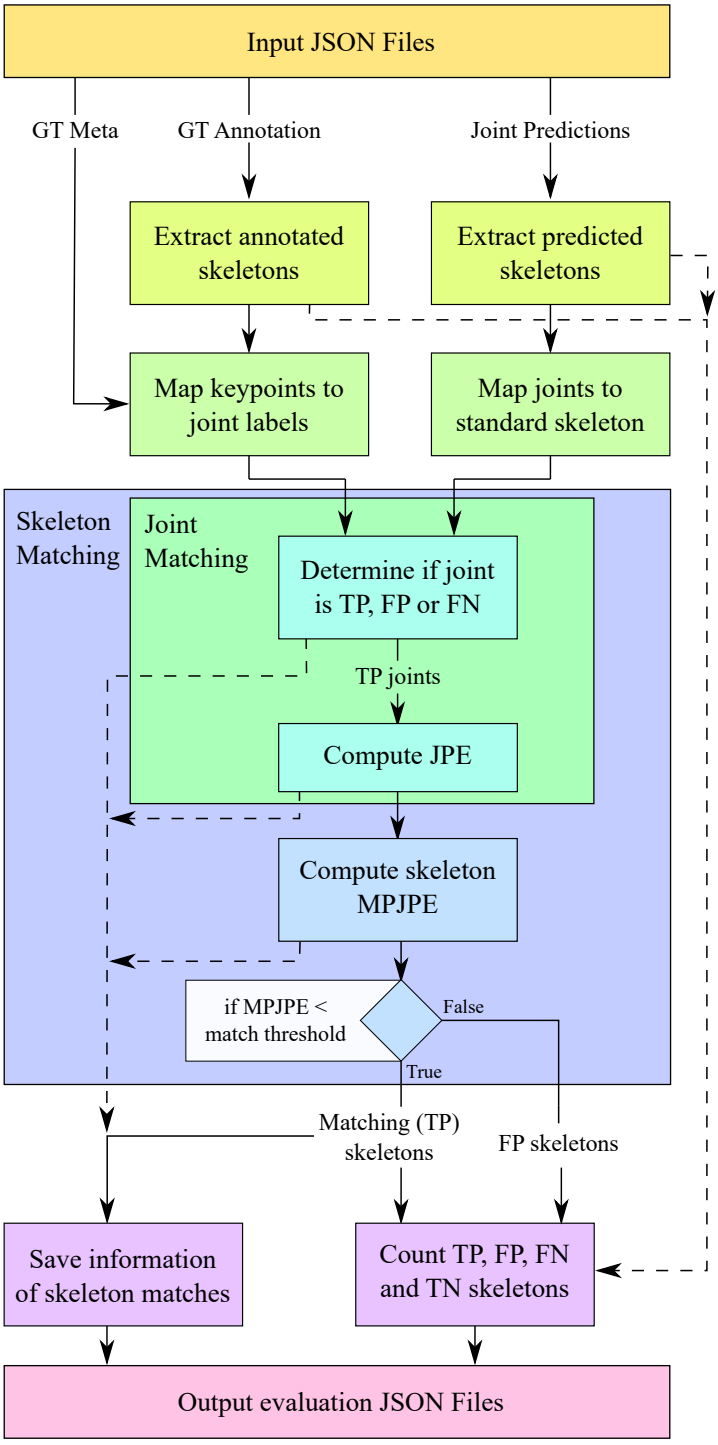


Figure 4.9: Block diagram of the implemented evaluation procedure.

4.5.2 Skeleton Matching - Filtering False Positives

Having extracted both the annotated and predicted skeletons, as well as established how their joint keypoints are associated, we must now identify which of the predicted skeletons are actually pose

Table 4.2: Keypoint label mapping of joints for each skeleton

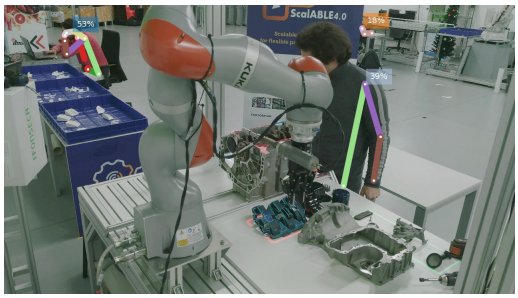
Joint Designation	Annotation	OpenPifPaf	OpenPose
Nose	0	0	0
Chest	1	/	1
Right Shoulder	2	6	2
Right Elbow	3	8	3
Right Wrist	4	10	4
Left Shoulder	5	5	5
Left Elbow	6	7	6
Left Wrist	7	9	7
Right Hip	8	12	9
Right Knee	9	14	10
Right Ankle	10	16	11
Left Hip	11	11	12
Left Knee	12	13	13
Left Ankle	13	15	14
Right Eye	14	2	15
Left Eye	15	1	16
Right Ear	16	4	17
Left Ear	17	3	18

The '/' symbol indicates that the 'Chest' point is not present in the OpenPifPaf skeleton.

estimates for the operator. This process is designated as skeleton matching and, in our evaluation procedure, is designed to filter out False Positive (FP) skeletons, as in skeletons that the pose estimation method detected but are not an estimate of the operator pose. Figure 4.10 shows one image from each dataset and each method with examples of FP skeletons - in them, other humans that are not the operator, the robotic arm, one of the operator's tool, and even a backpack in the background are all detected as humans present in the image.

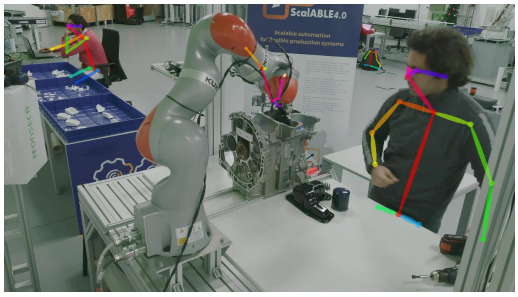
It is known that, for each annotated image, the operator is either in frame and their pose is annotated as a singular skeleton, or the operator is not in frame and no skeletons are annotated. The number of annotated skeletons and the number of annotated joints are both saved, respectively as 'annotated_skeleton_total' (either 0 or 1) and 'annotated_joint_total' (between 0 and 18) to be part of the final output. These values are used later in comparison to the number of detected skeletons and joints to determine the number of FP skeletons and joints.

In order to distinguish between True Positive (TP) and FP skeletons, we decided to rely on the MPJPE value of all predicted skeletons in each image. Initially, the skeleton with the lowest MPJPE would be designated as the matching skeleton. However, this approach proved to be overly simplistic since it is possible that there is more than one predicted skeleton that correctly detects the operator. This is due to the fact that OpenPifPaf and OpenPose rely on bottom-up approaches to human pose estimation - where joints are first detected separately and then associated to build a skeleton. Figure 4.11 demonstrates this, as OpenPifPaf detected several joints that can be attributed to the operator but, probably due to heavy occlusion, could not fully associate them as part of the same skeleton.

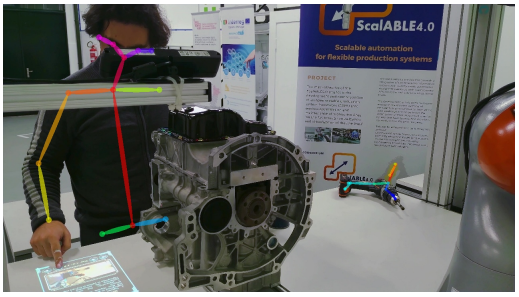


(a) Left, OpenPifPaf - detected other humans in background.

(b) Right, OpenPifPaf - operator is not in frame but poster is detected.



(c) Left, OpenPose - human, robotic arm and backpack detected.



(d) Right, OpenPose - tool and poster are incorrectly detected.

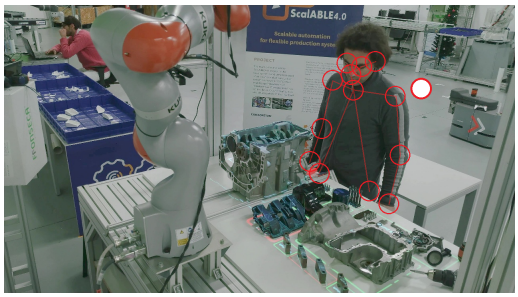
Figure 4.10: Examples of rendered frames with false positive skeletons.

Figure 4.11: Example of rendered frame where OpenPifPaf did not fully associate detected joints as part of the same skeleton.

Since more than one predicted skeleton can be a match, our solution instead defines a matching skeleton as one with a MPJPE below a designated threshold. Ideally, this threshold value would be based on criteria similar to that of the PCK metric - either a fraction of one of the bounding box dimensions of the annotated skeleton, or a percentage of the length of a segment that connects two

keypoints. However, as the original annotation did not include a bounding box and the size of the operator is fairly consistent between the whole datasets, we opted to arbitrarily define a value and then individually verify all images for incorrect matches. For visual reference, figure 5.1 depicts one of the images in the 'left' dataset overlayed with its annotated skeleton, and the same image with circles with radius approximately equal to 25 pixels centered around each of the operator's annotated joints, in order to demonstrate that, with a matching threshold of 25 pixels, to match a predicted skeleton, its average joint must be within those boundaries.

(a) Image with overlaid annotation skeleton. In order to match a predicted skeleton, the average pixel distance error must be below an arbitrary threshold.



(b) The same image with added circles of 25 pixels radius centered around each joint, representing a possible matching threshold for distance error.

Figure 4.12: Visual demonstration of the matching threshold criteria.

Throughout skeleton matching, the total number of predicted skeletons is saved as the variable 'pifpaf_skeleton_total' or 'openpose_skeleton_total' (depending on the method in question), along with the number of matching skeletons as 'matching_skeleton_total', to be part of the final output. These values are especially useful to determine False Negative (FN) skeletons, the cases in which there is an annotated skeleton but the operator is not detected by the pose estimation method, such as the one presented in figure 4.13.

4.5.3 Joint Matching and MPJPE Computation

As was discussed in the previous section, our skeleton matching implementation relies on computing its MPJPE value, the mean value of the Joint Position Error (JPE) for all joints in that skeleton. To do this, we must first match each predicted joint to its equivalent annotated joint, the process we designate as joint matching. Similarly to skeleton matching, this procedure is designed to handle cases where joints are falsely detected, FP joints, or where joints are not detected despite its equivalents having been annotated, a FN joint.

To execute the joint matching procedure, we iterate through the predicted joints of each prediction skeleton in an image: first, we check if the corresponding annotated joint exists using the keypoint label mapping represented in table 4.2. If it does not, we register it as a FP joint. Otherwise, we compute this joint's position error by calculating the euclidean distance between the predicted pixel location and the annotated pixel location. After iterating through all predicted

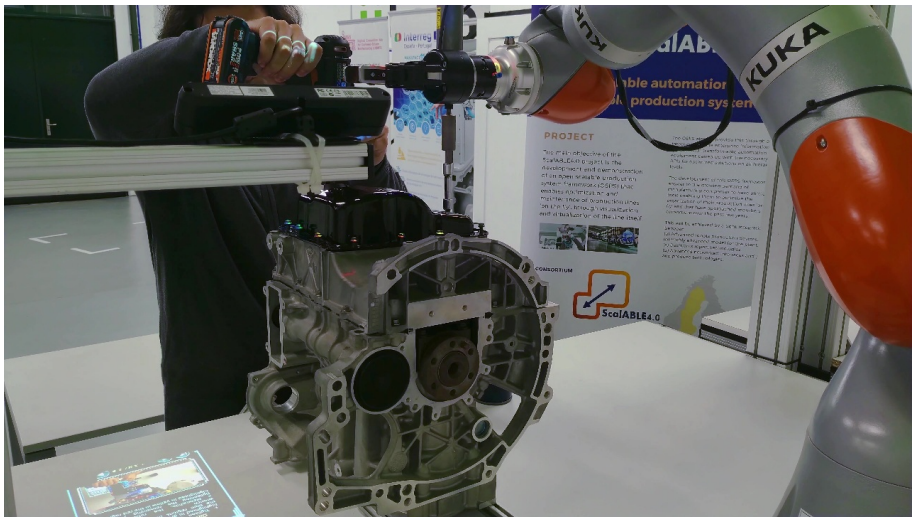


Figure 4.13: Example of rendered frame where OpenPifPaf did not detect the visible operator.

joints, we check the annotated skeleton of the image in question and register all the annotated joints that do not have a predicted counterpart as a FN joint.

Throughout joint matching, information about each joint is temporarily saved. This consists of a pair of values: the first one is either the JPE or a string indicating that joint to be a FP or a FN[24]; while the second is the confidence score that the pose estimation method attributed to that joint prediction. Moreover, four additional values are saved for each skeleton: the total number of predicted joints as 'joint_counter_total'; the MPJPE value as 'mpjpe'; the total number of FP joints as 'fp_joint_total'; and the total number of FN joints as 'fn_joint_total'. Finally, during the skeleton matching phase, if a particular skeleton is a match, this temporary set of variables is saved in its entirety in the array of 'matching_skeletons', to be part of the final output.

Additionally, once all skeletons have been either matched or discarded, four more variables are saved for the final output: the total number of predicted joints in the all matching skeletons of the image as 'joint_image_total'; the MPJPE average of all matching skeletons in the image as 'mpjpe_image_average'; as well as the total number of FP and FN joints in all matching skeletons as 'fp_joint_image_total' and 'fn_joint_image_total' respectively.

Once the predictions of a specific method have been evaluated for each individual image in one of the datasets, four final variables are saved: the average number of predicted, FP and FN joints per image of the dataset as 'joint_total_average_per_image', 'fp_joint_total_average_per_image' and 'fn_joint_total_average_per_image' respectively; as well as the average MPJPE per image as 'mpjpe_average_per_image'.

[24] In the cases were the joint should be ignored, this string can also be 'invalid'. For example, when the 'Chest' keypoint is annotated (labelled as 1), it should not be counted as a FN for OpenPifPaf as it does not have an equivalent.

4.5.4 Output Results

Throughout the evaluation procedure, several variables were saved to be used either to determine evaluation metrics or as supporting information to facilitate the discussion of the results obtained. The 'eval.py' script - that implements the evaluation procedure detailed in this section - outputs four different *JSON* files of results, one for each combination of dataset and method. An excerpt of the 'results_left_openpifpaf.json' output file that demonstrates the output format for all the variables previously mentioned can be found in appendix A.

These variables are then processed to compile the results that are discussed next, in chapter 5.

Chapter 5

Results and Discussion

In this chapter we present and discuss the results obtained from the evaluation procedure of OpenPifPaf and OpenPose, highlighting important conclusions with bolded headers. Section 5.1 analyses the metrics obtained regarding the reliability and accuracy of the pose estimates. Section 5.2 presents inference speed results and reflects on which method would be more suitable for real-time operation in the real use case. Finally, section 5.3 argues which of the human pose estimation methods should be selected for further development of a human operator tracking system.

5.1 Reliability and Accuracy

As discussed previously, in order to develop a tracking system that relies on the output of one of the evaluated human pose estimation methods, the method that is ultimately chosen must be able to reliably identify the human operator in-frame, as well as accurately estimate their joint's pixel-position, all to determine where the operator is located in the workspace. Addressing this question requires analysing the output that is obtained from each method from two different perspectives: on a higher-level, if the operator is consistently detected and marked with a skeleton prediction; and, on a lower-level, if the individual joint predictions that make up the skeleton are correct. The following subsections address these issues individually.

Table 5.1: Evaluation metrics for skeleton predictions.

Method	**Dataset**	TP	FP	FN	TN	P	N	$\tilde{P}$	$\tilde{N}$	**AP**	**AR**	**AA**
OpenPifPaf	left	82	73	0	2	82	75	155	2	0.529	1.0	0.535
	right	77	9	3	1	80	10	86	4	0.895	0.963	0.867
	left + right	159	82	3	3	162	85	241	6	0.660	**0.981**	0.656
OpenPose	left	82	61	0	2	82	63	143	2	0.573	1.0	0.579
	right	77	17	3	1	80	18	94	4	0.819	0.963	0.796
	left + right	159	78	3	3	162	81	237	6	**0.671**	**0.981**	**0.667**

5.1.1 Skeleton Predictions

Regarding each method's skeleton predictions, table 5.1 reports the evaluation metrics that were obtained from the skeleton matching phase of the evaluation procedure (see subsection 4.5.2). Using the terminology discussed in section 3.4 to analyse the skeleton predictions results, each metric is specifically defined as the following:

- TP is the number of true positive skeletons - a predicted skeleton that was matched to the operator's annotated skeleton[1];
- FP is the number of false positive skeletons - predicted skeletons that do not represent the operator/were not matched to the annotated skeleton;
- FN is the number of false negative skeletons - an annotated skeleton that has no matching skeletons;
- TN is the number of true negative skeletons - when there are both no annotated nor predicted skeletons;
- P and N are, respectively, the number of cases with a positive or negative ground truth annotation of operator skeleton;
- $\tilde{P}$ and $\tilde{N}$ are, respectively, the number of cases with a positive or negative prediction of operator skeleton;
- **AP** is the average precision per image in the corresponding dataset ($Precision = \frac{TP}{TP+FP}$), the percentage of the total predicted skeletons that were correct predictions of the operator.
- **AR** is the average recall per image in the corresponding dataset ($Recall = \frac{TP}{TP+FN}$), the percentage of the annotated skeletons that were correctly predicted by the method;
- **AA** is the average accuracy per image in the corresponding dataset ($Acc = \frac{TP+TN}{P+N}$), the percentage of times that the method accurately predicted if the operator is present.

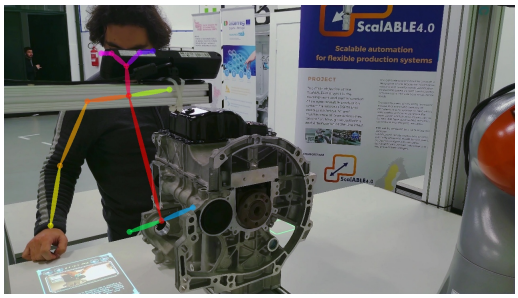
Both methods perform similarly in skeleton prediction - The metrics presented in table 5.1 indicate extremely similar performance, or even identical in the case of recall, from both methods in the task of skeleton prediction. Regarding precision and accuracy, while OpenPose slightly outperforms OpenPifPaf overall when considering both datasets, it overperforms in the 'left' dataset, but underperforms in the 'right' dataset.

The operator is reliably identified in-frame - Both methods reported perfect recall (AP = 1) for our 'left' dataset and near-perfect (AP = 0.963) for our 'right' dataset, meaning the operator is detected nearly every time he is in frame (159 out of a total of 162 frames). We analysed the images that reported FN skeletons[2] and concluded that they all suffered from heavy occlusion of the operator. Similarly to the example of figure 4.13, the operator is almost entirely occluded by the tool he is holding, the workstation's projector, the motor that is being assembled, and/or the robotic arm.

[1] We individually verified that, in all cases where there is more than one matching skeleton, they all correspond to the operator and do not overlap joint predictions. Therefore, we associated all matching skeletons, and considered them as a single TP skeleton.

[2] Images that report FN skeletons: 39, 56 and 82 for OpenPifPaf; and 39, 55 and 82 for OpenPose

(a) Failure - while smaller, the other person was selected over the operator.



(b) Success - while occluded, the operator is still selected over the person in the background

Figure 5.1: Example of images with rendered skeleton predictions from OpenPose, with *number_people_max* flag set to 1.

These results were expected since the 'right' dataset was built with the knowledge that the camera had not been positioned favourably for human pose estimation, resulting in frequent occlusions from several elements in the scene. Considering this factor, the results for skeleton prediction are, in isolation, impressive. While FN cases were indeed rare, repositioning the camera to a better view of the operator is a simple solution that could further mitigate this issue. Alternatively, the pose estimation methods could be trained with custom models to be more robust to these scenarios. However, because these occlusions are extremely challenging, this would likely result in overfitting. This issue of FN and these possible solutions are discussed again later, in the next subsection regarding joints.

Both methods report low precision due to abundance of FP - As demonstrated previously in figure 4.10, a major issue that both methods face in skeleton detection is the abundance of FP skeletons, which is reflected in lower precision (AP) scores. This is, in part, due to the incorrect detection of objects - for example, the operator's tools, the poster behind the operator, a backpack in the background, or even the robotic arm itself, among others - as humans, but, also, because of how we defined skeleton detection exclusively as the detection of the operator - and, thus, counting all other humans detected as FP skeletons.

In fact, both OpenPifPaf and OpenPose are methods for multi-person pose estimation and are, therefore, accomplishing their intended goals. Nonetheless, while detecting all humans in frame can be a useful feature (as the safety of everyone is important), it may also be a hindrance as, even if only actual humans were detected, it would requires us to filter out these other humans to identify the operator. We performed a similar procedure in the skeleton matching phase of the evaluation (see subsection 4.5.2). However, we relied on the average MPJPE, a measurement that is impossible to obtain in real operation, as, evidently, there is no ground truth annotation to which to compare the prediction.

We attempted some simple solutions to easily achieve the same effect of filtering FP skeletons: first by attempting to match skeletons based on the confidence score that OpenPifPaf and OpenPose attribute to each; and then by using an integrated feature of OpenPose, a flag that limits

the maximum number of people detected in an image, based on person area over the image, body part score, and joint score. Both attempts proved unsuccessful mostly because neither measure could be easily correlated to a controllable aspect of the scene, like pixel-size of skeleton, or the orientation of the human relative to the camera, as exemplified in figure 5.1. Still, these may prove as interesting possibilities to investigate further in the future.

Additional challenges that were not present in these datasets, like people crossing behind the operator or additional people at the workstation or at a similar distance from the camera, could further complicate the identification of the operator from the detected skeletons. Ideally, these variables could be controlled by, for example, covering the background behind the workstation, but, because this would hamper the adaptability of our system (one of the core motivations behind this work, especially in the context of flexible manufacturing and Industry 4.0), the development of a more complex subsystem for operator identification may be necessary in the future. A possible solution involves the introduction of time as a variable to allow for tracking between frames. In fact, new work by the original creators of the OpenPifPaf method, Kreiss et al., introduces temporal association [42].

In summation, even though both the 'left' and 'right' datasets are populated with a variety of challenging scenarios, both methods were able to reliably recognize when the operator is in frame. Even so, we inquire if this may be due to over-sensitivity towards possible humans in frame, as both methods were created with the purpose of multi-person pose estimation and suffer from lower precision/higher numbers of FP detections than desired. This is a concern namely because there is no inherent way for either of the pose estimation methods to distinguish between the human operator, other humans that might be present in-frame, or even objects that might be incorrectly detected. Therefore, we propose that future work investigates solutions like training these methods with customised models to be more fitted for the specific use case in question, or a skeleton FP filtering method to reliably identify which of the predicted skeletons corresponds to the operator.

5.1.2 Joint Predictions

Having considered the reliability of skeleton detection, we now present the evaluation metrics regarding the joint predictions of OpenPifPaf and OpenPose, respectively in tables 5.2 and 5.3. These results are subdivided according to type of joint to facilitate in-depth analysis. In the context of joint prediction, each metric is defined as the following:

- TP is the number of true positive joints - a joint that was both annotated and predicted by the method in question;
- FP is the number of false positive joints - a joint that was predicted but not annotated;
- FN is the number of false negative joints - a joint that was annotated but not predicted by the method in question;
- TN is the number of true negative joints - a joint that was neither annotated nor predicted by the method in question;

- P and N are the number of operator joints that were annotated, or not, from the 18 joints of our ground truth skeleton configuration;
- $\tilde{P}$ and $\tilde{N}$ are the number of operator joints that were predicted, or not, from the 18 joints of our ground truth skeleton configuration;
- **AP** is the average precision per image in the corresponding dataset ($Precision = \frac{TP}{TP+FP}$), the percentage of the total predicted skeletons that were correct predictions of the operator.
- **AR** is the average recall per image in the corresponding dataset ($Recall = \frac{TP}{TP+FN}$), the percentage of the annotated skeletons that were correctly predicted by the method;
- **AA** is the average accuracy per image in the corresponding dataset ($Acc = \frac{TP+TN}{P+N}$), the percentage of times that the method accurately predicted if the operator is present;
- **MPJPE** is the mean per joint position error, a measurement in pixels of the average distance between the annotated joint location and its predicted counterpart.

Previously, the sums of TP, FP, FN and TN; or P, N, $\tilde{P}$ and $\tilde{N}$ skeletons could not be determined in advance because it was not possible to know how many skeletons can potentially be predicted. However, the sum of the joint equivalents of these metrics is known since, for every image where the operator is present, there are always, at most, 18 different joints that could have been either annotated or predicted[3] - therefore, for a single image: $TP+FP+FN+TN = P+N = \tilde{P}+\tilde{N} =$ 18. In the same way, because both an annotated skeleton and a matching skeleton must be found in order to classify joints, we know that the sum $TP+FP+FN+TN = P+N = \tilde{P}+\tilde{N}$ for a single type of joint will always be equal to the number of skeleton TP, in the set of images that are considered. For example, the total number of 'Nose' joints is equal to 82 in the 'left' dataset, and 77 in the 'right' dataset - the number of images in each dataset where the operator was both annotated and predicted.

To compliment the following observations, figures 5.2, 5.3, 5.4 and 5.5 exemplify some of the conclusions reached and provide visual reference for some scenarios.

Lower body joints are heavily occluded - Table 5.4 compiles the joint metrics of both OpenPifPaf and OpenPose predictions for the operator's lower body joints, the right and left hips, knees and ankles - joints 9 to 13. These show that the operator's leg joints, knees and ankles - 9, 10, 12 and 13 - are almost always occluded by the workstation, and that this is correctly predicted by the methods.

Additionally, regarding the hip joints: the left hip joint (11) was almost always annotated in the 'left' dataset (P = 81) and correctly detected by both methods with high recall and precision. However, this joint suffered from very low precision in the 'right' dataset because it is often occluded - and therefore not annotated (N = 64) - but falsely predicted, especially by OpenPose, which reported FP = 61 and AP = 0.176.

In turn, the right hip joint (8) was detected with perfect recall by both methods in the 'left' dataset, as well as high precision and accuracy. In the 'right' dataset though, there are often more challenging heavy occlusions where the right hip joint is one of few joints that are not occluded.

[3] For OpenPifPaf, only 17 joints because of the non-existent 'Chest' keypoint.

Table 5.2: Evaluation metrics for OpenPifPaf joint predictions.

Joint	Label	**Dataset**	**TP**	**FP**	**FN**	**TN**	P	N	$\tilde{P}$	$\tilde{N}$	**AP**	**AR**	**AA**	**MPJPE**
Nose	0	L	78	0	1	3	79	3	78	4	1	0.987	0.988	5.66
		R	51	15	0	11	51	26	66	11	0.773	1	0.805	5.63
Chest	1	L	/	/	/	/	80	2	/	/	/	/	/	/
		R	/	/	/	/	36	41	/	/	/	/	/	/
Right Shoulder	2	L	75	5	0	2	75	7	80	2	0.938	1	0.939	10.72
		R	24	34	1	18	25	52	58	19	0.414	0.96	0.545	18.75
Right Elbow	3	L	67	5	6	4	73	9	72	10	0.931	0.918	0.866	8.25
		R	51	6	14	6	65	12	57	20	0.895	0.785	0.74	11.07
Right Wrist	4	L	62	4	6	10	68	14	66	16	0.939	0.912	0.878	4.94
		R	29	16	5	27	34	43	45	32	0.644	0.853	0.727	8.32
Left Shoulder	5	L	73	9	0	0	73	9	82	0	0.89	1	0.89	17.14
		R	44	9	0	24	44	33	53	24	0.83	1	0.883	14.33
Left Elbow	6	L	65	16	0	1	65	17	81	1	0.802	1	0.805	9.59
		R	35	12	6	24	41	36	47	30	0.745	0.854	0.766	9.15
Left Wrist	7	L	78	1	1	2	79	3	79	3	0.987	0.987	0.976	14
		R	26	7	7	37	33	44	33	44	0.788	0.788	0.818	7.34
Right Hip	8	L	70	6	0	6	70	12	76	6	0.921	1	0.927	22.81
		R	16	13	24	24	40	37	29	48	0.552	0.4	0.519	32.8
Right Knee	9	L	0	0	1	81	1	81	0	82	/	0	0.988	/
		R	0	0	0	77	0	77	0	77	/	/	1	/
Right Ankle	10	L	0	0	3	79	3	79	0	82	/	0	0.963	/
		R	0	0	1	76	1	76	0	77	/	0	0.987	/
Left Hip	11	L	73	1	8	0	81	1	74	8	0.986	0.901	0.89	29.14
		R	10	13	3	51	13	64	23	54	0.435	0.769	0.792	32.38
Left Knee	12	L	0	0	1	81	1	81	0	82	/	0	0.988	/
		R	0	0	0	77	0	77	0	77	/	/	1	/
Left Ankle	13	L	0	0	1	81	1	81	0	82	/	0	0.988	/
		R	0	0	1	76	1	76	0	77	/	0	0.987	/
Right Eye	14	L	58	16	0	8	58	24	74	8	0.784	1	0.805	4.38
		R	65	1	5	6	70	7	66	11	0.985	0.929	0.922	2.82
Left Eye	15	L	78	0	1	3	79	3	78	4	1	0.987	0.988	5.05
		R	53	12	1	11	54	23	65	12	0.815	0.981	0.831	2.74
Right Ear	16	L	4	0	3	75	7	75	4	78	1	0.571	0.963	8.33
		R	57	0	12	8	69	8	57	20	1	0.826	0.844	10.13
Left Ear	17	L	77	0	5	0	82	0	77	5	1	0.939	0.939	8.41
		R	17	0	8	52	25	52	17	60	1	0.68	0.896	4.9
Total		left	858	63	37	436	895	499	921	473	0.932	0.959	0.928	11.42
		right	478	138	88	605	566	743	616	693	0.776	0.845	0.827	12.34
		L+R	1336	201	125	1041	1461	1242	1537	1166	0.869	0.914	0.879	11.88

L and R represent the 'left' and 'right' datasets. The '/' symbol indicates that metric is not applicable. MPJPE is measured in pixels.

This is reflected in the much lower values of each metric of both methods for the right hip joint in this dataset. Similarly to the left hip joint, OpenPose reports higher numbers of FP right hip joints than OpenPifPaf in either dataset.

Furthermore, the MPJPE value for the hip joints of either method, which range from 22.81 to 49.45 pixels, are significantly higher than every other type of joint that is located from the waist up, which, with the exception of the 'Chest' keypoint, range from 2.74 to 18.75 pixels for OpenPifPaf and from 4.06 to 23.32 pixels for OpenPose. As was discussed in 4.3, this may be partly due to the challenges faced in the annotation of these joints, mainly the operator's clothing, orientation

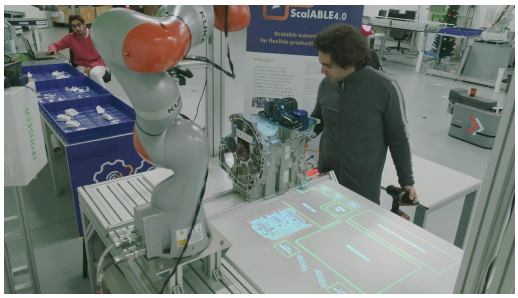
Table 5.3: Evaluation metrics for OpenPose joint predictions.

Joint	Label	**Dataset**	**TP**	**FP**	**FN**	**TN**	P	N	$\tilde{P}$	$\tilde{N}$	**AP**	**AR**	**AA**	**MPJPE**
Nose	0	L	79	3	0	0	79	3	82	0	0.963	1	0.963	7.2
		R	50	26	0	1	50	27	76	1	0.658	1	0.662	7.29
Chest	1	L	80	2	0	0	80	2	82	0	0.976	1	0.976	30.92
		R	36	40	0	1	36	41	76	1	0.474	1	0.481	35.04
Right Shoulder	2	L	75	7	0	0	75	7	82	0	0.915	1	0.915	12.23
		R	26	51	0	0	26	51	77	0	0.338	1	0.338	23.32
Right Elbow	3	L	73	7	0	2	73	9	80	2	0.913	1	0.915	8.55
		R	61	12	4	0	65	12	73	4	0.836	0.938	0.792	17.87
Right Wrist	4	L	68	9	0	5	68	14	77	5	0.883	1	0.89	7.03
		R	32	33	1	11	33	44	65	12	0.492	0.97	0.558	9.39
Left Shoulder	5	L	73	9	0	0	73	9	82	0	0.89	1	0.89	18.78
		R	44	32	0	1	44	33	76	1	0.579	1	0.584	16.45
Left Elbow	6	L	65	17	0	0	65	17	82	0	0.793	1	0.793	10.48
		R	41	28	1	7	42	35	69	8	0.594	0.976	0.623	8.85
Left Wrist	7	L	79	3	0	0	79	3	82	0	0.963	1	0.963	14.39
		R	32	26	0	19	32	45	58	19	0.552	1	0.662	16.52
Right Hip	8	L	70	12	0	0	70	12	82	0	0.854	1	0.854	28.42
		R	36	37	4	0	40	37	73	4	0.493	0.9	0.468	44.54
Right Knee	9	L	0	0	1	81	1	81	0	82	/	0	0.988	/
		R	0	1	0	76	0	77	1	76	0	/	0.987	/
Right Ankle	10	L	0	0	3	79	3	79	0	82	/	0	0.963	/
		R	0	1	1	75	1	76	1	76	0	0	0.974	/
Left Hip	11	L	81	1	0	0	81	1	82	0	0.988	1	0.988	32.99
		R	13	61	0	3	13	64	74	3	0.176	1	0.208	49.45
Left Knee	12	L	0	2	1	79	1	81	2	80	0	0	0.963	/
		R	0	0	0	77	0	77	0	77	/	/	1	/
Left Ankle	13	L	0	0	1	81	1	81	0	82	/	0	0.988	/
		R	0	0	1	76	1	76	0	77	/	0	0.987	/
Right Eye	14	L	58	21	0	3	58	24	79	3	0.734	1	0.744	5.33
		R	69	7	0	1	69	8	76	1	0.908	1	0.909	4.06
Left Eye	15	L	79	3	0	0	79	3	82	0	0.963	1	0.963	6.45
		R	53	23	0	1	53	24	76	1	0.697	1	0.701	7.55
Right Ear	16	L	5	7	2	68	7	75	12	70	0.417	0.714	0.89	5.55
		R	68	8	0	1	68	9	76	1	0.895	1	0.896	10.95
Left Ear	17	L	82	0	0	0	82	0	82	0	1	1	1	11.11
		R	25	13	0	39	25	52	38	39	0.658	1	0.831	16.14
Total		left	967	103	8	398	975	501	1070	406	0.904	0.992	0.925	12.96
		right	586	399	12	389	598	788	985	401	0.595	0.98	0.703	17.88
		L+R	1553	502	20	787	1573	1289	2055	807	0.756	0.987	0.818	15.42

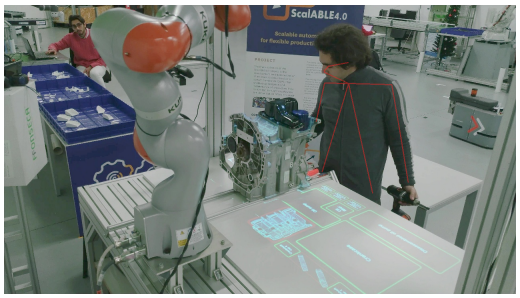
L and R represent the 'left' and 'right' datasets. The '/' symbol indicates that metric is not applicable. MPJPE is measured in pixels.

in relation to the camera, as well as occlusion from the workstation's table.

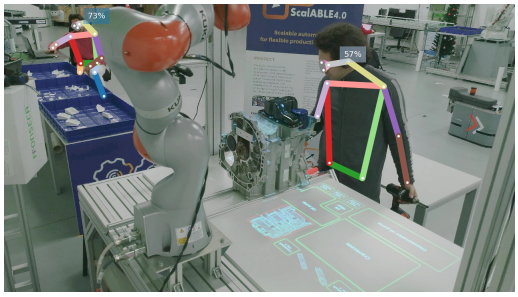
Arm joints are more accurately predicted by OpenPifPaf - In the use case in question, the arms are the operator's body parts that more often occupy the shared workspace and more rapidly change position over time. Therefore, these joints, and especially the wrist joints which are typically those that move closest to the robotic arm (see figure 5.3), require high accuracy to enable 3D pose estimation in the future. Table 5.5 reports the evaluation metrics of OpenPifPaf and OpenPose's predictions for the operator's arm joints, the right and left shoulders, elbows and wrists - joints 2 to 7. While both methods perform favourably for both datasets, OpenPifPaf



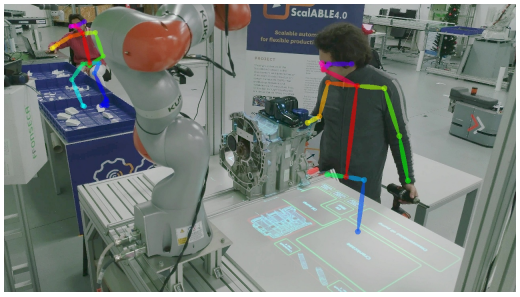
(a) Original image of 'left' dataset.



(b) Image with overlayed annotation skeleton.



(c) Rendered image of OpenPifPaf predictions.



(d) Rendered image of OpenPose predictions.

Figure 5.2: Example images of false positive (FP) predictions: skeletons of humans in background, and lower-body joints occluded by workstation table.

outperforms OpenPose across the board, reporting equal or higher precision, recall and accuracy, as well as lower MPJPE for all shoulder, elbow and wrist joints.

Regarding MPJPE specifically, both methods report the lowest position error for the left wrist joints in comparison to the remaining arm joints, possibly because it is often closer to camera and less occluded. We suggest that, in the future, further analysis is done on the relation between MPJPE and the remaining metrics to determine if one might compromise the other - for example, if higher recall results in higher position error.

Both methods typically report more FP than FN arm joints - Low accuracy is typically a result of FP rather than FN. This once again indicates that both methods might be oversensitive, predicting occluded joints more often than not detecting one that was annotated.

Head joints are generally accurately predicted by both methods - Table 5.6 compiles the joint metrics of both OpenPifPaf and OpenPose predictions for the operator's head joints, the nose, right and left eyes and ears - joints 0, and 3 to 6. Once again, both methods report favourable performance in head joint pose estimation, with OpenPifPaf slightly outperforming OpenPose in total AP, AA, and MPJPE; and underperforming in AR.

Moreover, with the exception of the right ear joint (5), which is often not visible to the camera in the 'left' dataset due to the operator's orientation (P = 7), the head joints, and especially the nose joint, report the lowest MPJPE of all joints which would, in theory, result in more accurate 3D pose estimates.

Table 5.4: Evaluation metrics for OpenPifPaf and OpenPose predictions of lower body joints.

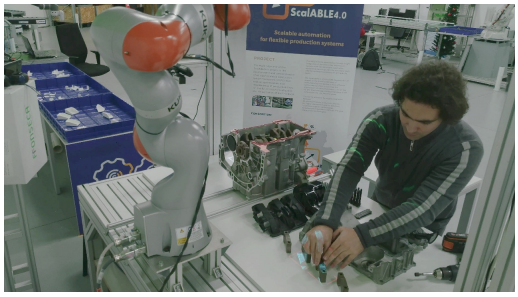
Joint	Label	**Method**	**Dataset**	**TP**	**FP**	**FN**	**TN**	P	N	$\tilde{P}$	$\tilde{N}$	**AP**	**AR**	**AA**	**MPJPE**
Right Hip	8	OpenPifPaf	L	70	6	0	6	70	12	76	6	0.921	1	0.927	22.81
			R	16	13	24	24	40	37	29	48	0.552	0.4	0.519	32.8
		OpenPose	L	70	12	0	0	70	12	82	0	0.854	1	0.854	28.42
			R	36	37	4	0	40	37	73	4	0.493	0.9	0.468	44.54
Right Knee	9	OpenPifPaf	L	0	0	1	81	1	81	0	82	/	0	0.988	/
			R	0	0	0	77	0	77	0	77	/	/	1	/
		OpenPose	L	0	0	1	81	1	81	0	82	/	0	0.988	/
			R	0	1	0	76	0	77	1	76	0	/	0.987	/
Right Ankle	10	OpenPifPaf	L	0	0	3	79	3	79	0	82	/	0	0.963	/
			R	0	0	1	76	1	76	0	77	/	0	0.987	/
		OpenPose	L	0	0	3	79	3	79	0	82	/	0	0.963	/
			R	0	1	1	75	1	76	1	76	0	0	0.974	/
Left Hip	11	OpenPifPaf	L	73	1	8	0	81	1	74	8	0.986	0.901	0.89	29.14
			R	10	13	3	51	13	64	23	54	0.435	0.769	0.792	32.38
		OpenPose	L	81	1	0	0	81	1	82	0	0.988	1	0.988	32.99
			R	13	61	0	3	13	64	74	3	0.176	1	0.208	49.45
Left Knee	12	OpenPifPaf	L	0	0	1	81	1	81	0	82	/	0	0.988	/
			R	0	0	0	77	0	77	0	77	/	/	1	/
		OpenPose	L	0	2	1	79	1	81	2	80	0	0	0.963	/
			R	0	0	0	77	0	77	0	77	/	/	1	/
Left Ankle	13	OpenPifPaf	L	0	0	1	81	1	81	0	82	/	0	0.988	/
			R	0	0	1	76	1	76	0	77	/	0	0.987	/
		OpenPose	L	0	0	1	81	1	81	0	82	/	0	0.988	/
			R	0	0	1	76	1	76	0	77	/	0	0.987	/
Lower Body Total		**OpenPifPaf**	left	143	7	14	328	157	335	150	342	0.953	0.911	0.957	25.975
			right	26	26	29	381	55	407	52	410	0.5	0.473	0.881	32.59
			L + R	169	33	43	709	212	742	202	752	0.837	0.797	0.92	29.2825
		OpenPose	left	151	15	6	320	157	335	166	326	0.91	0.962	0.957	30.705
			right	49	100	6	307	55	407	149	313	0.329	0.891	0.771	46.995
			L + R	200	115	12	627	212	742	315	639	0.635	0.943	0.867	38.85

L and R represent the 'left' and 'right' datasets. The '/' symbol indicates that metric is not applicable. MPJPE is measured in pixels.

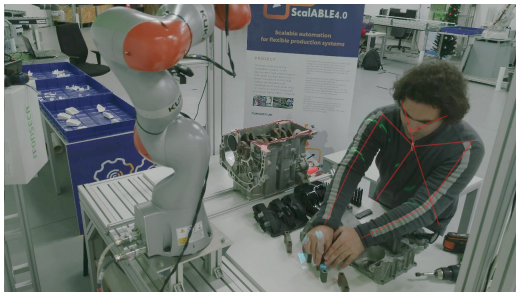
Overall, OpenPose reports more FP and OpenPifPaf more FN - Across all joints, OpenPose tends to report lower values of precision than OpenPifPaf due to an abundance of FP joints ($FP_{OpenPose} = 502 > FP_{OpenPifPaf} = 201$), which are typically joints that have been occluded. Inversely, while OpenPifPaf reports lower MPJPE and FP, it also reports many more FN joints ($FN_{OpenPifPaf} = 125 > FP_{OpenPose} = 20$), those that are in fact, not occluded but were still not detected. This seems to suggest that OpenPifPaf predicts joints more conservatively than OpenPose and with lower position error ($MPJPE_{OpenPifPaf} = 11.88 < MPJPE_{OpenPose} = 15.42$), a factor that may be useful depending on context and application.

Considering the assumption made during ground truth annotation (see section 4.3), predicting FP joints may be considered detrimental to accurate 3D pose estimation since depth measurements of occluded joints would be incorrect. However, in the context of our use case, not detecting a joint may seriously jeopardize the operator's safety and is, thus, much less desirable. Therefore, similarly to the previous analysis of skeleton FP, it might be necessary to develop a method to filter out FP joints in favour of preventing occurrences of FN.

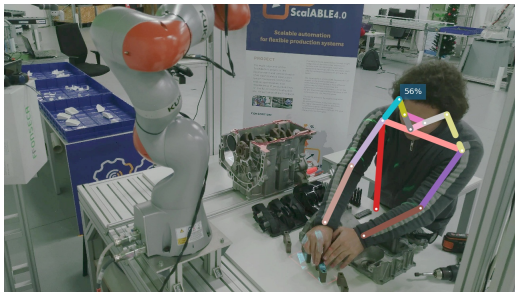
For example, a possible solution could rely on the fusion of depth information from two or



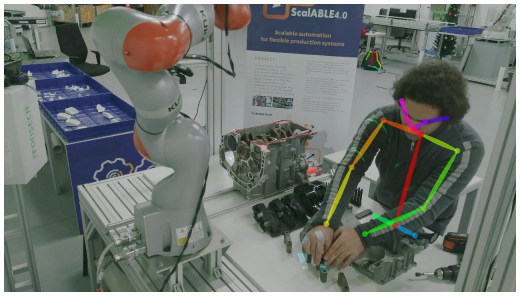
(a) Original image of 'left' dataset.



(b) Image with overlayed annotation skeleton.



(c) Rendered image of OpenPifPaf predictions.



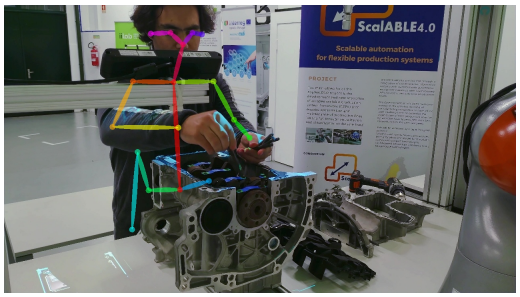
(d) Rendered image of OpenPose predictions.

Figure 5.3: Example images of false positive hip joint predictions and accurate predictions of arm and head joints.

(a) Original image of 'left' dataset.

(b) Image with overlayed annotation skeleton.

(c) Rendered image of OpenPifPaf predictions.



(d) Rendered image of OpenPose predictions.

Figure 5.4: Example images of OpenPifPaf's higher precision and OpenPose's seemingly accurate false positive predictions for occluded arm joints.

Table 5.5: Evaluation metrics for OpenPifPaf and OpenPose predictions of arm joints.

Joint	Label	**Method**	**Dataset**	**TP**	**FP**	**FN**	**TN**	P	N	$\tilde{P}$	$\tilde{N}$	**AP**	**AR**	**AA**	**MPJPE**
Right Shoulder	2	PifPaf	L	75	5	0	2	75	7	80	2	0.938	1	0.939	10.72
			R	24	34	1	18	25	52	58	19	0.414	0.96	0.545	18.75
		Pose	L	75	7	0	0	75	7	82	0	0.915	1	0.915	12.23
			R	26	51	0	0	26	51	77	0	0.338	1	0.338	23.32
Right Elbow	3	PifPaf	L	67	5	6	4	73	9	72	10	0.931	0.918	0.866	8.25
			R	51	6	14	6	65	12	57	20	0.895	0.785	0.74	11.07
		Pose	L	73	7	0	2	73	9	80	2	0.913	1	0.915	8.55
			R	61	12	4	0	65	12	73	4	0.836	0.938	0.792	17.87
Right Wrist	4	PifPaf	L	62	4	6	10	68	14	66	16	0.939	0.912	0.878	4.94
			R	29	16	5	27	34	43	45	32	0.644	0.853	0.727	8.32
		Pose	L	68	9	0	5	68	14	77	5	0.883	1	0.89	7.03
			R	32	33	1	11	33	44	65	12	0.492	0.97	0.558	9.39
Left Shoulder	5	PifPaf	L	73	9	0	0	73	9	82	0	0.89	1	0.89	17.14
			R	44	9	0	24	44	33	53	24	0.83	1	0.883	14.33
		Pose	L	73	9	0	0	73	9	82	0	0.89	1	0.89	18.78
			R	44	32	0	1	44	33	76	1	0.579	1	0.584	16.45
Left Elbow	6	PifPaf	L	65	16	0	1	65	17	81	1	0.802	1	0.805	9.59
			R	35	12	6	24	41	36	47	30	0.745	0.854	0.766	9.15
		Pose	L	65	17	0	0	65	17	82	0	0.793	1	0.793	10.48
			R	41	28	1	7	42	35	69	8	0.594	0.976	0.623	8.85
Left Wrist	7	PifPaf	L	78	1	1	2	79	3	79	3	0.987	0.987	0.976	14
			R	26	7	7	37	33	44	33	44	0.788	0.788	0.818	7.34
		Pose	L	79	3	0	0	79	3	82	0	0.963	1	0.963	14.39
			R	32	26	0	19	32	45	58	19	0.552	1	0.662	16.52
Arm Joints Total		**PifPaf**	left	420	40	13	19	433	59	460	32	0.913	0.97	0.892	10.773
			right	209	84	33	136	242	220	293	169	0.713	0.864	0.747	11.49
			L + R	629	124	46	155	675	279	753	201	0.835	0.932	0.822	11.13
		Pose	left	433	52	0	7	433	59	485	7	0.893	1	0.894	11.91
			right	236	182	6	38	242	220	418	44	0.565	0.975	0.593	15.4
			L + R	669	234	6	45	675	279	903	51	0.741	0.991	0.748	13.66

PifPaf and Pose are abbreviations for OpenPifPaf and OpenPose. L and R represent the 'left' and 'right' datasets. MPJPE is measured in pixels.

more different cameras, reducing vulnerability to occlusions at the cost of increasing computational cost by requiring us to run the selected 2D human pose estimation on more frames at a time. Additionally, favouring FP joints would only be viable if the pose estimates for occluded joints still reported reasonable position error, a requirement that is particularly difficult to verify using our methodology because ground truth annotation would require us to estimate occluded joints, introducing more uncertainty. Figures 5.4 and 5.5 provide examples that indicate that OpenPose is not always reliable to provide useful FP predictions through heavy occlusion.

In short, OpenPifPaf slightly outperforms OpenPose in the task of joint prediction but at the cost of a higher number of FN, which are undesirable for operator tracking systems designed for safety. Therefore, a case could be made for the selection of either method, depending on what factors can be compromised on and further research should be conducted into this matter.

Table 5.6: Evaluation metrics for OpenPifPaf and OpenPose predictions of head joints.

Joint	Label	**Method**	**Dataset**	**TP**	**FP**	**FN**	**TN**	P	N	$\tilde{P}$	$\tilde{N}$	**AP**	**AR**	**AA**	**MPJPE**
Nose	0	PifPaf	L	78	0	1	3	79	3	78	4	1	0.987	0.988	5.66
			R	51	15	0	11	51	26	66	11	0.773	1	0.805	5.63
		Pose	L	79	3	0	0	79	3	82	0	0.963	1	0.963	7.2
			R	50	26	0	1	50	27	76	1	0.658	1	0.662	7.29
Right Eye	3	PifPaf	L	58	16	0	8	58	24	74	8	0.784	1	0.805	4.38
			R	65	1	5	6	70	7	66	11	0.985	0.929	0.922	2.82
		Pose	L	58	21	0	3	58	24	79	3	0.734	1	0.744	5.33
			R	69	7	0	1	69	8	76	1	0.908	1	0.909	4.06
Left Eye	4	PifPaf	L	78	0	1	3	79	3	78	4	1	0.987	0.988	5.05
			R	53	12	1	11	54	23	65	12	0.815	0.981	0.831	2.74
		Pose	L	79	3	0	0	79	3	82	0	0.963	1	0.963	6.45
			R	53	23	0	1	53	24	76	1	0.697	1	0.701	7.55
Right Ear	5	PifPaf	L	4	0	3	75	7	75	4	78	1	0.571	0.963	8.33
			R	57	0	12	8	69	8	57	20	1	0.826	0.844	10.13
		Pose	L	5	7	2	68	7	75	12	70	0.417	0.714	0.89	5.55
			R	68	8	0	1	68	9	76	1	0.895	1	0.896	10.95
Left Ear	6	PifPaf	L	77	0	5	0	82	0	77	5	1	0.939	0.939	8.41
			R	17	0	8	52	25	52	17	60	1	0.68	0.896	4.9
		Pose	L	82	0	0	0	82	0	82	0	1	1	1	11.11
			R	25	13	0	39	25	52	38	39	0.658	1	0.831	16.14
Head Joints Total		**PifPaf**	left	295	16	10	89	305	105	311	99	0.949	0.967	0.937	6.366
			right	243	28	26	88	269	116	271	114	0.897	0.903	0.86	5.244
			L + R	538	44	36	177	574	221	582	213	0.924	0.937	0.899	5.805
		Pose	left	303	34	2	71	305	105	337	73	0.899	0.993	0.912	7.128
			right	265	77	0	43	265	120	342	43	0.775	1	0.8	9.198
			L + R	568	111	2	114	570	225	679	116	0.837	0.996	0.858	8.163

PifPaf and Pose are abbreviations for OpenPifPaf and OpenPose. L and R represent the 'left' and 'right' datasets. MPJPE is measured in pixels.

5.2 Real-time Operation and Inference Speed

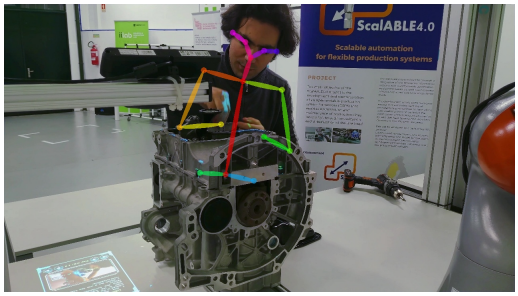
Besides the reliability and accuracy of their predictions, another requirement for the implementation of either of the human pose estimation methods in an operator tracking system is that it operates in real-time. Depending on the context, this can have different meanings: in computer vision, real-time operation is typically defined as matching the frame rate of the camera output which, in this case, would mean the 30 FPS (frames per second) of the Intel RealSense D435 camera's RGB output. However, for the purpose of tracking the operator, this may be an unnecessarily high FPS requirement. In fact, both the minimum requirement for inference speed and the actual performance of the pose estimation module will be entirely dependent of the other modules of the tracking system that rely on its output. Therefore, to determine which method would be more suitable for real-time operation, we simply consider which one is faster.

OpenPose reports faster inference speed for 1280 x 720 resolution - To this end, the inference time of each method for every image in both datasets was measured, as described in section 4.4. Each combination of pose estimation method and dataset was executed 5 times each, in the same controlled setup, to ensure we had not measured an outlier. Table 5.7 reports both the average inference time of each image in each of the 5 runs, as well as the corresponding average frame rate in frames-per-second (FPS). OpenPose reports considerably faster inference speed than OpenPif-

(a) Original image of 'left' dataset.

(b) Image with overlayed annotation skeleton.

(c) Rendered image of OpenPifPaf predictions.



(d) Rendered image of OpenPose predictions.

Figure 5.5: Example images of OpenPifPaf's higher precision and OpenPose's inaccurate false positive predictions for occluded arm joints.

Paf ($7.567 FPS > 4.857$ - an increase of over 55%), even though we did not use the cuDNN library that OpenPose Doc highly recommends. However, the authors of OpenPifPaf claim in [41] that they outperform OpenPose in smaller resolutions.

Since the average frame rate we obtained is relatively low, we inquire if the image resolution could be further reduced without compromising estimation accuracy. Additionally, the inference speed of the two methods might not vary in the same proportion with image resolution. Therefore, we suggest that the effect of image input size should be further investigated to determine, for example, the relationship between inference speed and input scale factor, as well as its effect on the evaluation metrics reported previously. In the meantime however, our results regarding inference speed favour the selection of OpenPose as the human pose estimation method that should be integrated into a tracking system for our use case.

Table 5.7: Average inference speed of OpenPifPaf and OpenPose and average time measurements per image for each run.

Method	**Dataset**	**FPS**	**Inference time p/image (ms)**				
			1	2	3	4	5
OpenPifPaf	left	**4.848**	2069	2058.4	2068.7	2056.7	2060
	right	**4.865**	2051.2	2055.4	2052.8	2049.3	2068.5
OpenPose	left	**7.549**	1349.5	1320.4	1316.5	1318.4	1318.8
	right	**7.584**	1313.2	1315.9	1321.7	1317.9	1324.3

5.3 Method Selection

While OpenPifPaf slightly outperformed OpenPose in joint prediction accuracy, both methods were reliable in skeleton prediction, and OpenPose outperformed OpenPifPaf in inference speed by 55%. This faster speed is critically important considering that real-time operation is a crucial factor to the success of a tracking system and that the remaining modules that will be developed will further increase execution time. Additionally, we believe that the solutions proposed for FP filtering, like better positioning of cameras and fusion of information from more than one perspective, are preferable to risking FN detections that compromise the operator's safety.

Therefore, based on the results presented throughout this chapter, we would select **OpenPose** as the human pose estimation method to use in the development of a human operator tracking system for safe industrial collaborative robotics. However, we acknowledge both methods as viable options and propose that future work should train custom models for each method to increase performance (if possible to accomplish without overfitting), attempt to fuse 2D pose estimates from different views of the scene into a refined 3D pose estimate that is robust to occlusions, as well as further investigate the effect of image input resolution on the performance of both metrics (especially OpenPifPaf's speed) and the relationship between the accuracy metrics of AP, AR, and MPJPE.

Chapter 6

Conclusion and Future Work

The introduction of the Industry 4.0 paradigm is shifting manufacturing from mass production towards customisable production lines. In response to this change in market demands, collaborative robots that work simultaneously with human operators are sought after for the increase flexibility they provide. However, the safety of the operator remains a crucial concern with no standardised solutions.

In this dissertation, we analysed the development of a human operator tracking system for safe Human-Robot Collaboration, in the context of industrial assembly tasks with robotic manipulators. For this, we investigated 2D methods for multi-person human pose estimation that would provide the pixel location of the operator's joints in the workspace in order to facilitate tracking. In turn, a refined pose estimate in the 3D space will enable the development of safety features in the vein of collision avoidance and motion planning.

Two methods were considered, OpenPifPaf and OpenPose, which are state-of-the-art bottom-up approaches that rely on deep neural networks to predict joint estimates in 2D RGB images and then associate them into a skeleton configuration. We presented procedures for the evaluation of these selected human pose estimation methods in custom built datasets.

Evaluation metrics regarding the reliability of person detection, the accuracy of joint predictions, and inference speed in the context of real-time operation were presented and discussed. We concluded that both OpenPifPaf and OpenPose are viable options for human pose estimation, depending on whether accuracy or speed are more valued in the use case in question. However, for the purposes of safe industrial collaborative robotics, we selected OpenPose for its higher recall and 55% higher inference speed, in favour of OpenPifPaf's higher precision, which we believe compromises its robustness to occlusions and therefore, the operator's safety.

Additionally, we propose that future work should consider: training custom models to increase performance for each method, if possible to accomplish without overfitting; attempt to fuse 2D pose estimates from different views of the scene into a refined 3D pose estimate that is robust to occlusions; as well as further investigate the effect of image input resolution on the performance of both metrics, with emphasis on OpenPifPaf's speed, and the relationship between the accuracy metrics of AP, AR, and MPJPE to determine if achieving better performance in one might

compromise performance in the others.

Appendix A

Implementation Details and Examples of Obtained Files

This appendix provides some implementation details regarding the Python scripts that were created for this work - which are publicly available at https://github.com/eduardocaldasfonseca/cobot-monitor/ - as well as examples or excerpts of the files that are obtained throughout the methodology described in chapter 4.

A.1 Ground Truth Annotation - Supervisely and *JSON* files

Section 4.3 describes that, after annotation in the Supervisely web interface, the exported datasets include two important types of *JSON* files: 'meta.json'; and individual files for each annotated image (for example, 'left-small1.jpg.json'). JSON is a standardised data format for lightweight and human-readable information exchange. The Python programming language has built-in support for JSON encoding and decoding, in the form of the 'json' package[1], which leverages dictionaries (one of Python's data structures). These concepts were crucial for the development of this work, thus, we rely on the terminology of JSON and Python dictionaries throughout this appendix.

meta.json


```
{
    "classes": [
        {
            "title": "Skeleton",
            "shape": "graph",
            "color": "#EE2727",
            "geometry_config": {
...
                "nodes": {
                    "9b852c05-5e47-420a-9a46-86440bf0782c": {
```


[1] See https://docs.python.org/3/library/json.html

```
                "label": "1",
                "loc": [
                    135,
                    126
                ]
            },
            "b8737dd3-b95b-45be-8b6f-0133e238eb5b": {
                "label": "2",
...
```


Above is an excerpt of a '**meta.json**' file. This file uses the "classes" array to describe all the classes that were created for annotation in Supervisely. For our purposes, a single class, named 'Skeleton', was defined as the set of keypoints depicted in figure 4.7a. As can be seen in the excerpt, Supervisely refers to each 'Skeleton' keypoint as a node and attributes it a custom code, also indicating the joint label we defined.

left-small1.jpg.json


```
{
    "description": "",
    "tags": [],
    "size": {
        "height": 720,
        "width": 1280
    },
    "objects": [
        {
            "id": 731157765,
...
            "classTitle": "Skeleton",
            "nodes": {
                "9b852c05-5e47-420a-9a46-86440bf0782c": {
                    "loc": [
                        1010,
                        261
                    ]
                },
                "b8737dd3-b95b-45be-8b6f-0133e238eb5b": {
                    "loc": [
                        943.9473684210527,
                        239.24060150375936
                    ]
                },
...
```


This excerpt of file 'left-small1.jpg.json' exemplifies the information contained in the individual files obtained from Supervisely for each annotated image. The "objects" array contains the annotated "Skeleton" object and its respective nodes, the keypoints that represent the human operator's joints.

In our evaluation procedure, we use this file to extract the annotated skeleton of each image, if there is one, as well as the pixel location of each annotated keypoint. To identify which of the operator's joints each Supervisely node is referring to, the 'meta.json' file is used to map each of the Supervisely nodes of the annotated skeletons to the joint labels that we defined in table 4.2. For example, the node referred by the code "9b852c05-5e47-420a-9a46-86440bf0782c", present in this example, corresponds to the human joint label of "1", which is the 'Chest' keypoint.

A.2 Operator Pose Estimation - Run Outputs

As described in section 4.4, two Python scripts, 'run_openpifpaf.py' and 'run_openpose.py', were created to standardise the outputs of the two tested human pose estimation methods, OpenPifPaf and OpenPose, using their respective Python APIs. These scripts can be used to obtain human pose estimates for all images in a dataset, while registering the inference time of each, to later obtain a measure of inference speed. Furthermore, the scripts allow the user to customise each 'run' (an execution of the script) by easily setting some attributes related to the features provided by each method. For example, which pre-trained model should be used, if images with rendered skeleton predictions should be created, or even additional flags like the ones provided by OpenPose - such as 'number_people_max', which limits the maximum number of people detected, as discussed in subsection 5.1.1. In this section, we provide examples of the JSON outputs from the first run of OpenPose estimation (openpose_run1), which estimated keypoints for all images in the left dataset with rendered image creation enabled. Besides these files, if the 'image_creation' flag is enabled, rendered images with an overlay of the skeleton predictions are created - some examples are used throughout chapters 4 and 5.

openpose_run1_meta.json


```
{
    "Run Number": 1,
    "OpenPose version": "check individual json files",
    "Model": "BODY_25",
    "Dataset": "left-small",
    "With image creation?": true,
    "Controlled setup?": false,
    "Number people max flag": -1,
    "Net resolution flag": "-1x368",
    "Times": {
        "/left-small1.jpg": "0.5361075660039205",
...
        "/left-small84.jpg": "0.14358337000157917"
    },
    "Total time": "13.57802638800058"
}
```

Each run has a 'meta.json' file that contains useful information about the run, such as: its assigned number; the version of the pose estimation method; which pre-trained model and which dataset were used; if image creation was enabled and if the run was done in a controlled setup (both for the purposes of ensuring accurate inference time measurements); as well as flags for the method's custom features. This file also contains both the individual and total inference time measurements that are later used to determine inference speed and frame rate.

left-small1.jpg_keypoints.json


```
{
    "version": 1.3,
    "people": [
        {
            "person_id": [
                -1
            ],
            "pose_keypoints_2d": [
                966.331,
                227.818,
                0.877919,
                1034.91,
                259.153,
                0.797156,
...
                0,
                0,
                0
            ],
            "face_keypoints_2d": [],
...
            "hand_right_keypoints_3d": []
        }
    ]
}
```


Above is an excerpt of the JSON file obtained from run 1 that contains OpenPose's joint location predictions for the 'left-small1.jpg' image (the first image of the left dataset), in sets of 3 values (x, y, c), as described in subsection 4.5.1.

To obtain the joint estimates of each method in an easily interpretable format, we used each method's API to output a 'keypoints.json' file. OpenPose does this directly when it is provided with an output path using the "write_json" flag. In contrast, OpenPifPaf provides an 'annotations' object by default that is typically used for image creation with its annotation painter. However, if its 'json_data' flag is enabled, this will instead be JSON data that we can output to a '[image-file-name]_pifpaf_run[number]_keypoints.json' file. It is important to note that, if this is done, these predictions cannot be used for the annotation painter and no rendered images can be created.

For the sake of comparison, below is an excerpt of the 'keypoints' file of an OpenPifPaf run for the same image - left-small1.jpg_pifpaf_run1_keypoints.json. The same sets of values for each keypoint are present in the same format, but there are additional indicators for bounding boxes and overall confidence scores for each detected skeleton, which could be useful for expanding the evaluation procedure of section 4.5.

A.3 Evaluation Procedure - Outputs

Since the evaluation procedure implemented in the 'eval.py' script is considerably extensive, several JSON files are outputted throughout to provide additional information and facilitate debugging. Most of the programming of this script involves processing Python dictionaries to extract information. Therefore, observing the intermediary/auxiliary dictionaries that are created can help to explain how the evaluation metrics are determined.

annotation_joint_dict.json


```
{
    "9b852c05-5e47-420a-9a46-86440bf0782c": "1",
    "b8737dd3-b95b-45be-8b6f-0133e238eb5b": "2",
    "8f1581c5-e158-4231-b399-e1fd05eed204": "3",
    "33c72455-9676-4c0c-ac55-8af1a2e5de1a": "4",
    "feef618f-6e72-4210-a49f-89bfca0b3af7": "5",
    "1dad6ff5-d7ea-44a2-bbd3-5934f16fbd43": "6",
    "4e1f1888-956e-421b-86c3-1d45cac3d9bc": "7",
    "d950f8af-a92b-4f28-a3d8-561add85fe73": "8",
    "c7cda86b-e36e-498b-91e1-8dbdb2b74dd8": "9",
    "8e77aa80-ff9a-4226-a0a8-5dd0836403aa": "10",
    "58d71231-c7db-4683-a62d-c5b87f2c550c": "11",
    "e65eb824-7d0f-4344-a35e-8fb8f9683409": "12",
    "46304555-805a-43bc-aaf5-ee87ad37da04": "13",
    "14c2d82c-5ccd-45d9-be81-5851ec216b2d": "14",
    "52fe0b68-0700-4bbb-a737-764deac8f816": "15",
    "bb9f521a-501b-43b8-9c5c-75d90c63b9c7": "16",
    "c0aba91a-5d7b-4d08-bf18-bd6968a1bf28": "17",
    "55479fb3-644e-4fc6-84d8-4bcb26d39790": "0"
}
```


This dictionary is created from the Supervisely 'meta.json' and is used to map the Supervisely codes to their respective joint labels, as discussed previously.

left_annot_dict.json


```
{
    "left-small1": {
        "1": [
            1010,
            261
        ],
        "2": [
            943.95,
```

```
            239.24
        ],
        "3": [
            906.61,
            333.3
        ],
        "4": [
            891,
            414
        ],
        "5": [
            1113.88,
            257.56
        ],
        "6": [
            1162.56,
            430.84
        ],
        "7": [
            1012,
            467
        ],
        "8": [
            904.88,
            466.9
        ],
        "11": [
            1044,
            517
        ],
        "14": [
            958,
            208
        ],
        "15": [
            992.37,
            213.82
        ],
        "17": [
            1050.5,
            205.83
        ],
        "0": [
            972.5,
            230.91
        ]
    },
...
    "left-small84": {
        "1": [
            1105,
            290
        ],
...
    },
    "Total images": 84,
    "Empty images": 2
}
```


Two dictionaries are created from the steps of annotated skeleton extraction and keypoint mapping, one for the left dataset (whose excerpt is seen above) and another for the right dataset. These files contain the pixel location of each annotated operator joint in all images of that dataset. Additionally, they provide two counters: one for the amount of images in that dataset, and another for the amount of 'empty' images - those that have no annotated skeletons. Since these annotations are considered as our ground truth, if there is no annotated skeleton, the operator is not present in that image, which is useful for determining false positives and false negatives.

left_openpose_dict.json


```
{
    "left-small1": {
        "1": [
            1010,
            261
        ],
        "2": [
            943.95,
            239.24
        ],
        "3": [
            906.61,
            333.3
        ],
```

```
        "4": [
            891,
            414
        ],
        "5": [
            1113.88,
            257.56
        ],
        "6": [
            1162.56,
            430.84
        ],
        "7": [
            1012,
            467
        ],
        "8": [
            904.88,
            466.9
        ],
        "11": [
            1044,
            517
        ],
        "14": [
            958,
            208
        ],
        "15": [
            992.37,
            213.82
        ],
        "17": [
            1050.5,
            205.83
        ],
        "0": [
            972.5,
            230.91
        ]
    },
...
    "left-small84": {
        "1": [
            1105,
            290
        ],
...
    },
    "Total images": 84,
    "Empty images": 2
}
```


Similarly to how the annotated skeletons were extracted and saved in their own annotation dictionary, each method's predictions are extracted from the 'keypoints.json' files of the previous section, and saved into a corresponding dictionary for each dataset - for a total of four. Besides counting the number of total and empty images, we also register the number of 'extra' images - those that have more than one predicted skeleton and, therefore, have at least one false positive skeleton that does not match with the operator's annotated skeleton.

results_left_openpose.json


```
{
    "left-small1": {
        "annotated_skeleton_total": 1,
        "annotated_joint_total": 13,
        "openpose_skeleton_total": 1,
        "matching_skeleton_total": 1,
        "matching_skeletons": {
            "1": {
                "0": [
                    7.585,
                    0.877919
                ],
                "1": [
                    24.083,
                    0.797156
                ],
                "2": [
                    2.702,
                    0.787948
                ],
                "3": [
                    4.405,
```

```
        0.830124
    ],
    "4": [
        12.649,
        0.831493
    ],
    "5": [
        28.683,
        0.56679
    ],
    "6": [
        0.582,
        0.750388
    ],
    "7": [
        5.831,
        0.752779
    ],
    "8": [
        36.53,
        0.479506
    ],
    "11": [
        29.206,
        0.447732
    ],
    "14": [
        4.472,
        0.839959
    ],
    "15": [
        7.591,
        0.86033
    ],
    "17": [
        9.843,
        0.733433
    ],
    "joint_counter_total": 13, }
```

```
                "mpjpe": 13.3971,
                "fp_joint_total": 0,
                "fn_joint_total": 0
            }
        },
        "joint_image_total": 13,
        "mpjpe_image_average": 13.397,
        "fp_joint_image_total": 0,
        "fn_joint_image_total": 0
    },
    ...
    "left-small184": {
        "annotated_skeleton_total": 1,
        "annotated_joint_total": 11,
        "openpose_skeleton_total": 4,
        "matching_skeleton_total": 1,
        "matching_skeletons": {
            "1": {
                "0": [
                    7.471,
                    0.887916
                ],
    ...
                "joint_counter_total": 14,
                "mpjpe": 9.3214,
                "fp_joint_total": 3,
                "fn_joint_total": 0
            }
        },
        "joint_image_total": 14,
        "mpjpe_image_average": 9.321,
        "fp_joint_image_total": 3,
        "fn_joint_image_total": 3
    },
    "joint_total_average_per_image": 12.738,
    "mpjpe_average_per_image": 13.389,
    "fp_joint_total_average_per_image": 1.226,
    "fn_joint_total_average_per_image": 0.095
```


Finally, four 'results' files are obtained from the skeleton matching step, one for each method and dataset combination. These present all the relevant information regarding the comparison of the annotation and the predictions for each image, which was used to compile the evaluation metrics presented in chapter 5. These are self-explanatory, with the exception of the 'matching_skeletons' array, which contains all the predicted skeletons that were matched to the operator, including each of its predicted joints, as well as both its joint position error and the confidence score that the human pose estimation method attributed to it.

Additionally, we present an excerpt of the 'results_left_openpifpaf' file below where OpenPifPaf predicted more than one skeleton in that image and two of them were matched to the operator.


```
"left-small12": {
        "annotated_skeleton_total": 1,
        "annotated_joint_total": 11,
        "pifpaf_skeleton_total": 3,
        "matching_skeleton_total": 2,
        "matching_skeletons": {
            "2": {
                "5": [
                    18.064,
                    0.71
                ],
                "2": [
                    10.006,
                    0.81
                ],
                "6": [
                    "fp",
                    0.43
                ],
                "3": [
                    7.455,
                    0.95
                ],
                "7": [
                    3.7,
                    0.51
                ],
                "4": [
                    6.001,
                    0.89
                ],
                "11": [
                    59.55,
                    0.4
                ],
                "8": [
                    40.773,
                    0.6
                ],
                "1": [
                    "invalid",
                    0
                ],
                "15": [
                    "fn",
                    0
                ],
                "17": [
                    "fn",
                    0
                ],
                "0": [
                    "fn",
                    0
                ],
                "joint_counter_total": 8,
                "mpjpe": 18.1936,
                "fp_joint_total": 1,
                "fn_joint_total": 3
            },
            "3": {
                "0": [
                    5.404,
                    0.89
                ],
                "15": [
                    1.668,
                    0.83
                ],
                "14": [
                    "fp",
                    0.61
                ],
                "1": [
                    "invalid",
                    0
                ],
                "2": [
                    "fn",
                    0
                ],
                "3": [
                    "fn",
                    0
                ],
                "4": [
                    "fn",
                    0
                ],
                "5": [
                    "fn",
                    0
```

```
],
"7": [
    "fn",
    0
],
"8": [
    "fn",
    0
],
"11": [
    "fn",
    0
],
"17": [
    "fn",
    0
],
"joint_counter_total": 3,
"mpjpe": 2.3573,
"fp_joint_total": 1,
"fn_joint_total": 8
}
},
"joint_image_total": 11,
"mpjpe_image_average": 10.275,
"fp_joint_image_total": 2,
"fn_joint_image_total": 2
}
```

References

[1] Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 55:248–266, November 2018. URL: https://www.sciencedirect.com/science/article/pii/S0957415818300321, doi:10.1016/j.mechatronics.2018.02.009.

[2] Lihui Wang, Sichao Liu, Hongyi Liu, and X.V. Wang. Overview of Human-Robot Collaboration in Manufacturing. In *5th International Conference on the Industry 4.0 Model for Advanced Manufacturing (AMP 2020), 1-4 June 2020*, pages 15–58. Springer International Publishing, 2020. doi:10.1007/978-3-030-46212-3_2.

[3] A survey of methods for safe human-robot interaction. *Foundations and Trends® in Robotics*, 5, 2017. URL: http://dx.doi.org/10.1561/2300000052, doi:10.1561/2300000052.

[4] ScalABLE 4.0 – Development and demonstration of an (OSPS). URL: https://www.scalable40.eu/.

[5] PSA UseCase – ScalABLE 4.0. URL: https://www.scalable40.eu/index.php/psa-usecase/.

[6] INESC-TEC Robotics. H2020 ScalABLE4.0 - Spatial augmented reality interface for human-robot collaboration, April 2020. URL: https://www.youtube.com/watch?v=kf4gimDdebE&ab_channel=INESC-TECRobotics.

[7] J.E. Colgate, W. Wannasuphoprasit, and M.A. Peshkin. Cobots: robots for collaboration with human operators. In *Proceedings of the ASME Dynamic Systems and Control Division, 17-22 Nov. 1996*, Proceedings of the ASME Dynamic Systems and Control Division, New York, NY, USA, 1996. ASME.

[8] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. 36(4), 2017. URL: https://doi.org/10.1145/3072959.3073596, doi:10.1145/3072959.3073596.

[9] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Mohamed Elgharib, Pascal Fua, Hans-Peter Seidel, Helge Rhodin, Gerard Pons-Moll, and Christian Theobalt. Xnect: Real-time multi-person 3d motion capture with a single rgb camera. 39(4), 2020. URL: https://doi.org/10.1145/3386569.3392410, doi:10.1145/3386569.3392410.

[10] Z.R. Khavas, S.R. Ahmadzadeh, and P. Robinette. Modeling Trust in Human-Robot Interaction: A Survey. In *Social Robotics. 12th International Conference, ICSR 2020, 14-18 Nov. 2020*, pages 529–41, Cham, Switzerland, 2020. Springer International Publishing. doi:10.1007/978-3-030-62056-1_44.

[11] C. Faria, A. Colim, J. Cunha, J. Oliveira, N. Costa, P. Carneiro, S. Monteiro, E. Bicho, L.A. Rocha, and P. Arezes. Safety Requirements for the Design of Collaborative Robotic Workstations in Europe - A Review. In *Advances in Safety Management and Human Performance. AHFE 2020*, pages 225–32, Cham, Switzerland, 2020. Springer International Publishing. doi:10.1007/978-3-030-50946-0_31.

[12] 14:00-17:00. ISO 10218-1:2011. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/05/13/51330.html.

[13] 14:00-17:00. ISO 10218-2:2011. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/04/15/41571.html.

[14] Yujiao Cheng, Liting Sun, Changliu Liu, and M. Tomizuka. Towards Efficient Human-Robot Collaboration With Robust Plan Recognition and Trajectory Prediction. *IEEE Robotics and Automation Letters*, 5(2), April 2020. doi:10.1109/LRA.2020.2972874.

[15] B. Sadrfaridpour and Yue Wang. Collaborative assembly in hybrid manufacturing cells: an integrated framework for human-robot interaction. *IEEE Transactions on Automation Science and Engineering*, 15(3):1178–92, July 2018. Place: USA Publisher: IEEE.

[16] C. Byner, B. Matthias, and Hao Ding. Dynamic speed and separation monitoring for collaborative robot applications - Concepts and performance. *Robotics and Computer-Integrated Manufacturing*, 58, August 2019. doi:10.1016/j.rcim.2018.11.002.

[17] H. Nascimento, M. Mujica, and M. Benoussaad. Collision Avoidance Interaction Between Human and a Hidden Robot Based on Kinect and Robot Data Fusion. *IEEE Robotics and Automation Letters*, 6(1):88–94, January 2021. Place: USA Publisher: IEEE. doi:10.1109/LRA.2020.3032104.

[18] F. Flacco, T. Kröger, A. De Luca, and O. Khatib. A depth space approach to human-robot collision avoidance. In *2012 IEEE International Conference on Robotics and Automation*, May 2012. ISSN: 1050-4729. doi:10.1109/ICRA.2012.6225245.

[19] Fang Wang and Yi Li. Beyond physical connections: Tree models in human pose estimation. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 596–603, 2013. doi:10.1109/CVPR.2013.83.

[20] Matthias Dantone, Juergen Gall, Christian Leistner, and Luc Van Gool. Human pose estimation using body parts dependent joint regressors. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3041–3048, 2013. doi:10.1109/CVPR.2013.391.

[21] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial Structures for Object Recognition. *International Journal of Computer Vision*, 61(1):55–79, January 2005. URL: https://doi.org/10.1023/B:VISI.0000042934.15159.49, doi:10.1023/B:VISI.0000042934.15159.49.

[22] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2878–2890, 2013. doi:10.1109/TPAMI.2012.261.

[23] Chi Qin Lai and Soo Siang Teoh. A review on pedestrian detection techniques based on histogram of oriented gradient feature. In *2014 IEEE Student Conference on Research and Development*, pages 1–6, 2014. doi:10.1109/SCORED.2014.7072948.

[24] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005. doi:10.1109/CVPR.2005.177.

[25] Qi Dang, Jianqin Yin, Bin Wang, and Wenqing Zheng. Deep learning based 2d human pose estimation: A survey. *Tsinghua Science and Technology*, 24(6):663–676, 2019. doi:10.26599/TST.2018.9010100.

[26] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014. doi:10.1109/CVPR.2014.214.

[27] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016. Recent Developments on Deep Big Vision. URL: https://www.sciencedirect.com/science/article/pii/S0925231215017634, doi:https://doi.org/10.1016/j.neucom.2015.09.116.

[28] Min Sun and Silvio Savarese. Articulated part-based model for joint object detection and pose estimation. In *2011 International Conference on Computer Vision*, pages 723–730, 2011. doi:10.1109/ICCV.2011.6126309.

[29] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. Towards accurate multi-person pose estimation in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3711–3719, 2017. doi:10.1109/CVPR.2017.395.

[30] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. Rmpe: Regional multi-person pose estimation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2353–2362, 2017. doi:10.1109/ICCV.2017.256.

[31] Shaoli Huang, Mingming Gong, and Dacheng Tao. A coarse-fine network for keypoint localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3047–3056, 2017. doi:10.1109/ICCV.2017.329.

[32] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. doi:10.1109/ICCV.2017.322.

[33] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7103–7112, 2018. doi:10.1109/CVPR.2018.00742.

[34] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[35] Georgia Gkioxari, Justin Johnson, and Jitendra Malik. Mesh r-cnn. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9784–9794, 2019. doi:10.1109/ICCV.2019.00988.

[36] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4929–4937, 2016. doi:10.1109/CVPR.2016.533.

[37] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. DeeperCut: A Deeper, Stronger, and Faster Multi-person Pose Estimation Model. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 34–50. Springer International Publishing, 2016. doi:10.1007/978-3-319-46466-4_3.

[38] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1302–1310, 2017. doi:10.1109/CVPR.2017.143.

[39] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186, 2021. doi:10.1109/TPAMI.2019.2929257.

[40] Muhammed Kocabas, Salih Karagoz, and Emre Akbas. Multiposenet: Fast multi-person pose estimation using pose residual network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[41] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. Pifpaf: Composite fields for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[42] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. Openpifpaf: Composite fields for semantic keypoint detection and spatio-temporal association, 2021. arXiv:2103.02440.

[43] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[44] Tsung-Yi Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C.L. Zitnick. Microsoft coco: Common objects in context. volume pt.V, pages 740 – 55, Cham, Switzerland, 2014//. URL: http://dx.doi.org/10.1007/978-3-319-10602-1_48.

[45] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3686–3693, 2014. doi:10.1109/CVPR.2014.471.

[46] David Powers. Evaluation: From precision, recall and f-factor to roc, informedness, markedness correlation. *Mach. Learn. Technol.*, 2, 01 2008.

[47] Mykhaylo Andriluka, Umar Iqbal, Eldar Insafutdinov, Leonid Pishchulin, Anton Milan, Juergen Gall, and Bernt Schiele. Posetrack: A benchmark for human pose estimation and tracking. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5167–5176, 2018. doi:10.1109/CVPR.2018.00542.

[48] Vittorio Ferrari, Manuel Marin-Jimenez, and Andrew Zisserman. Progressive search space reduction for human pose estimation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. doi:10.1109/CVPR.2008.4587468.

[49] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2878–2890, 2013. doi:10.1109/TPAMI.2012.261.

[50] Coco 2017 keypoint evaluation metrics website. URL: https://cocodataset.org/#keypoints-eval.

[51] Coco 2017 keypoint detection task website. URL: https://cocodataset.org/#keypoints-2017.