Human detection and pose estimation using ceiling mounted cameras

Master's thesis in Electrical Engineering

HARISH RAVI

SANJEEV MADHAVAN

DEPARTMENT OF ELECTRICAL ENGINEERING- EENX30

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Human detection and pose estimation using ceiling mounted cameras

HARISH RAVI & SANJEEV MADHAVAN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2023

Human detection and pose estimation using
ceiling mounted cameras
HARISH RAVI & SANJEEV MADHAVAN

Academic supervisor: Emmanuel Dean, Associate Professor,
Department of Electrical Engineering, Chalmers University of Technology

Industrial supervisor: Erik Brorsson, Group Truck Operations, Volvo AB

Examiner: Knut Åkesson, Professor, Department of Electrical Engineering, Chalmers University of Technology

Master's Thesis 2023
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A snapshot of Human detection and pose estimation results using RGB ceiling-mounted cameras.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Human detection and pose estimation using ceiling-mounted cameras
HARISH RAVI
Department of Electrical Engineering
Chalmers University of Technology

SANJEEV MADHAVAN
Department of Applied Physics
Chalmers University of Technology

Abstract

This project is part of an idea for developing a futuristic factory, wherein a fleet of Autonomous Transport Robots(ATRs) and humans work assembling trucks at Volvo Group. This will be made possible with the help of an array of ceiling-mounted cameras to look over the entire factory floor. Volvo Group calls this a "Generic Photogrammetry-based Sensor System(GPSS)".
Our thesis project deals with the detection of humans and the estimation of their poses from ceiling-mounted cameras. Though there are many state-of-the-art (SOTA), deep learning models to detect human poses, they happen to perform poorly for this application. This is because these models are trained using standard datasets available in the wild which consist of front-facing camera views, whereas Volvo's application involves top-view images. The challenge in using top-view images was the scarcity of top-view based human pose datasets available for training these models and the extensive manual efforts needed for annotating these datasets. To address these concerns, we propose a semi-automatic image annotation pipeline using triangulation of multi-view cameras. The setup for this pipeline will have two side cameras in addition to the top camera from which hundreds of synchronized images are going to be collected. Google's open-to-use MoveNet model will then be applied on side camera images to predict human poses by body key points. These detected key points are triangulated to 3D space and then re-projected to top-view images resulting in the generation of annotated top-view images. Further, for implementing our human pose model, we will adopt a top-down approach, comprising a human detector and followed by a pose estimator network predicting the poses of detected humans. Our selected models will be fine-tuned with our generated training data. As a last step, we also introduce two custom metrics along with the standard metrics used in the literature to evaluate our fine-tuned model based on Volvo's requirements. Our improved fine-tuned model results in fewer false detections and also has higher accuracy.

Keywords: Computer Vision, Deep learning, automatic annotation, Human detection, Human Pose Estimation, ceiling mounted cameras, transfer learning, OpenMMLab

Acknowledgements

We would like to take this opportunity to thank our industrial supervisors Erik Brorsson and Per-Lage Götvall from R&D department, Volvo Group Truck Operations(GTO) and academic supervisor Emmanuel Dean whose support, guidance, and enthusiasm were instrumental in completing this project. Special thanks to our examiner Prof. Knut Åkesson for his valuable feedback and assistance.

Also, thanks for our thesis opponents Johan Brandby and Tobias Schwarz for their feedback on our report and presentation.

The gained results of our thesis wouldn't have been possible without the support from Volvo GTO who provided us with resources and opportunities to carry out our research.

Finally, we are grateful to our family members for their support and motivation.

Harish Ravi & Sanjeev Madhavan

Contents

List of Figures **xiii**

List of Tables **xv**

1 Introduction **1**
1.1 Company Background . . . 1
1.2 Project Background . . . 1
1.3 Project Description . . . 2
1.4 Objectives of the Project . . . 3
1.5 Challenges . . . 3

2 Theory **5**
2.1 Introduction to Deep Learning . . . 5
2.1.1 ANN architectures . . . 6
2.1.2 Multi-layer perceptrons and fully connected layers . . . 7
2.1.3 Activation Function . . . 7
2.1.4 Forward propagation . . . 8
2.1.5 Back-Propagation . . . 9
2.1.6 Convolutional Neural Network . . . 9
2.1.6.1 CNN structure . . . 10
2.1.7 Transfer Learning . . . 11
2.2 Human detection (Object detection using deep learning) . . . 12
2.2.1 Datasets for object detection . . . 13
2.2.1.1 ImageNet dataset . . . 13
2.2.1.2 PASCAL VOC dataset . . . 13
2.2.1.3 MS COCO dataset . . . 13
2.2.1.4 Open Images dataset . . . 13
2.2.2 Faster RCNN model architecture . . . 14
2.3 Human Pose Estimation using deep learning . . . 14
2.3.1 Datasets for human pose estimation . . . 15
2.3.1.1 MPII . . . 15
2.3.1.2 OCHuman . . . 15
2.3.1.3 Human 3.6M . . . 16
2.3.1.4 COCO (Common Objects in Context) . . . 16
2.3.2 Approaches and Networks for human pose estimation . . . 16
2.3.2.1 Top-Down . . . 16

2.3.2.2 Bottom-Up . . . 17
2.3.2.3 Deep High-Resolution Representation Learning for Human Pose-Estimation(HRNet) . . . 18
2.4 Performance Metrics . . . 19
2.4.1 Qualitative Analysis of Prediction . . . 19
2.4.1.1 Intersection Over Union (IOU) . . . 19
2.4.1.2 Object Key-point Similarity (OKS) . . . 20
2.4.2 Quantitative Analysis of Prediction . . . 20
2.5 Camera-based computer vision . . . 21
2.5.1 Extrinsic Parameters . . . 22
2.5.2 Intrinsics Parameters . . . 22
2.5.2.1 Distortion of Image . . . 22
2.5.3 Camera Matrix . . . 23
2.5.4 Triangulation . . . 23
2.5.5 (Re)projection . . . 25

3 Methodology **27**
3.1 Camera Setup . . . 27
3.2 Modules of Human Detection and Pose Estimation . . . 27
3.2.1 Module 1: Image Pre-Processing . . . 27
3.2.1.1 Undistortion of Images (Intrinsic Calibration) . . . 28
3.2.1.2 Extrinsic Calibration . . . 29
3.2.2 Module 2: Semi-Automatic Training Data Annotation . . . 31
3.2.2.1 Annotation of key points using triangulation and re-projection . . . 31
3.2.2.2 Bounding Box Annotation . . . 32
3.2.3 Module 3: Training of HDPE Model . . . 33
3.2.3.1 Dataset for training and testing . . . 34
3.2.3.2 Fine-Tuning Human Detector . . . 35
3.2.3.3 Fine-Tuning Pose Estimation Model . . . 36
3.2.4 Module 4: HDPE Model Evaluation . . . 36

4 Results **38**
4.1 Evaluation of Human Detector Model results . . . 38
4.1.1 Analysis of false negatives . . . 39
4.1.2 Analysis of occlusion handling . . . 40
4.1.3 Analysis of bounding box . . . 40
4.2 Evaluation of Human Pose Estimation Model results . . . 41
4.2.1 Evaluation of 17 key points using Standard COCO metrics . . . 41
4.2.2 Visual inspection of ankle key points . . . 42
4.2.3 Evaluation of ankle key points using custom metrics. . . . 43
4.2.4 Results on multi-human pose data . . . 44

5 Conclusion **46**
5.1 Discussions . . . 46
5.2 Limitations and Future work . . . 47
5.3 Conclusion . . . 48

Bibliography **49**

List of Figures

1.1 Generic Photogrammetry-based Sensor System (GPSS) at Volvo's Factory. 2
1.2 Human key points occlusions in the top-view image. 4

2.1 Illustration of an activation process of McCulloch-Pitts neuron. $w_{i1,i2,...,i5}$ are connection strengths(weights) of five other neurons connected to neuron i. The incoming signals $v_{1,2,...,5}$ are multiplied with the connection strengths and summed together. The activation function $g(.)$ is applied to this sum to turn it into the output s_i. This figure is based on Figure 2 in [1]. 6
2.2 An illustration of MLP with 3 layers; an input layer, a hidden layer, and an output layer. 7
2.3 Common activation functions used in ANNs. 8
2.4 Pre-training vs Transfer learning vs fine-tuning. 11
2.5 Example of human detection result with a bounding box. 12
2.6 Faster RCNN Architecture overview. 14
2.7 Human Keypoints in COCO Dataset [2]. 15
2.8 Sample of OCHuman Dataset [3]. 16
2.9 Human Pose Estimation algorithm. (a): The example of a bottom-up approach. (b): The example of a top-down approach [4]. 17
2.10 An example of an HRNet architecture: Here, there are four stages. The 1st stage consists of high-resolution convolutions. The 2nd (3rd, 4th) stage repeats two-resolution (three-resolution, four-resolution) blocks. More details can be found in [5]. 18
2.11 HRNet exchange unit: Illustrating how the fusion module aggregates the information for high, medium, and low resolutions from left to right, respectively. Source: [5] 19
2.12 Intersection Over Union (IOU). 20
2.13 World coordinate system with camera extrinsic parameters [6]. 21
2.14 Effect of radial distortion. Solid lines: no distortion, dashed lines: with radial distortion (a: pincushion b: barrel) [7]. 23
2.15 Triangulation: In practice, the image points x and y cannot be measured with arbitrary accuracy. Instead points $x\prime$ and $y\prime$ are detected and used for the triangulation. The corresponding projection lines (dotted) do not, in general, intersect in 3D space and may also not intersect with point X. 24

2.16 The different mappings from 3D to 2D from left to right. 25

3.1 Camera Setup with two Side-View cameras taken from Top-view camera. 28
3.2 Output after findChessboardCorners function. 29
3.3 Intrisic calibration mainly to estimate and remove distortion effects from raw camera images. 29
3.4 Process of extrinsic calibration. Computes the orientation (Rotation and Translation) of the camera with respect to each other in the scene. 30
3.5 Detected corners in an Aruco board. 31
3.6 Keypoint triangulation using side-view cameras. 32
3.7 Pictorial representation of bounding box. 33
3.8 The process flow in a top-down human detection and pose estimation(HDPE) network. 34
3.9 Dataset for training and testing the fine-tuning process. Dataset 1: In order to induce data variability, images from two cameras (CAM 1 and CAM 2) were used for the training (80%) and validation dataset (20%). All the required labelling was created using our semi-automatic annotation process. Dataset 2: The test dataset consisted of a total of 74 images from a different camera (CAM 3) feed to test generalization. 35
3.10 Custom test metrics for feet ankle key points. 37

4.1 Comparison of false negative detections between (a) Pre-trained and our (b) Fine-tuned detector models. 39
4.2 Human detector's occluded detection comparison between (a) Pre-trained and our (b) Fine-tuned models. 40
4.3 Human detector's bounding box regression comparison between (a) Pre-trained (b) Fine-tuned models. 40
4.4 Human pose estimation model with 17 key points (a) Pre-trained and (b) Fine-tuned models. 42
4.5 Visual comparison of ankle prediction between (a) Pre-trained and our (b) Fine-tuned model. 42
4.6 Comparison of ankle prediction for custom metric evaluation: Blue and Green - Pre-trained model, Black and White - Fine-tuned model. 43
4.7 Results on Multi-person human test dataset without re-training. . . . 45

List of Tables

2.1 Model complexity information and inference speed various top-down models in MMPose. mAP = Mean Average Precision trained with COCO dataset [8]. 17
2.2 Model complexity information and inference speed various bottom-up models in MMPose. mAP = Mean Average Precision trained with COCO dataset [8]. 18

4.1 Performance Comparison for human detector using COCO metric. AP = Average Precision, AR= Average Recall. 39
4.2 Performance comparison for human pose estimation model - Standard COCO metrics. AP = Average Precision, AR= Average Recall. 41
4.3 Performance comparison for human detector using custom Metric - mAP of 2 ankle points. 44
4.4 Performance comparison for human detector using custom Metric - Mean L2 Pixel error of each ankle point. 44

1
Introduction

This chapter gives an overview of the company's background, project background, project description, aim and limitations of the project.

1.1 Company Background

Volvo AB are pioneers in the design and manufacturing of buses, trucks, and construction equipment. Based in Gothenburg, the company was established in 1915 as a subsidiary of SKF, a ball-bearing manufacturer that is also based in the same city. Volvo has been at the forefront of innovation throughout its century-long history. Volvo Trucks were one of the first trucks to use exhaust gas recirculating (EGR) valves. Currently, their focus is on developing future transport solutions that include but are not restricted to automation and electromobility [9]. Hereafter, Volvo AB will be called Volvo.

1.2 Project Background

The advancements in software technology and data science are enabling the Fourth Industrial Revolution [10]. While the first three industrial revolutions have brought about immense change, the impact of Industry 4.0 will be much wider and far greater [11]. By enabling "smart factories", the fourth industrial revolution aims to create an environment where virtual and physical systems of manufacturing processes cooperate globally with each other in a flexible way. This gives a huge advantage in terms of customization of products and models [12].
Recent trends such as electrification and the development of autonomous technologies are impacting not only the engineering aspects of truck companies but also the manufacturing and production entities. Until now, truck manufacturing didn't require many assembly lines to cater to producing models. However, all this is expected to change when newer technologies are introduced such as hybrid, fully electric and fuel cell-based powertrains with varying levels of autonomous capabilities. [13]. As a result, the existing factory setups will get more crowded, and difficult to meet up with the product's diversity.
Volvo's solution to this challenge is to develop a futuristic factory, incorporating flexible manufacturing concepts where Collaborative Robots (Cobots) and Autonomous Transport Robots (ATR), humans and robots can work and collaborate on equal terms. For this Unified Environment [14], a Generic Photogrammetry-based Sensor System (GPSS) is being set up. This system depicted in Figure 1.1, consists of

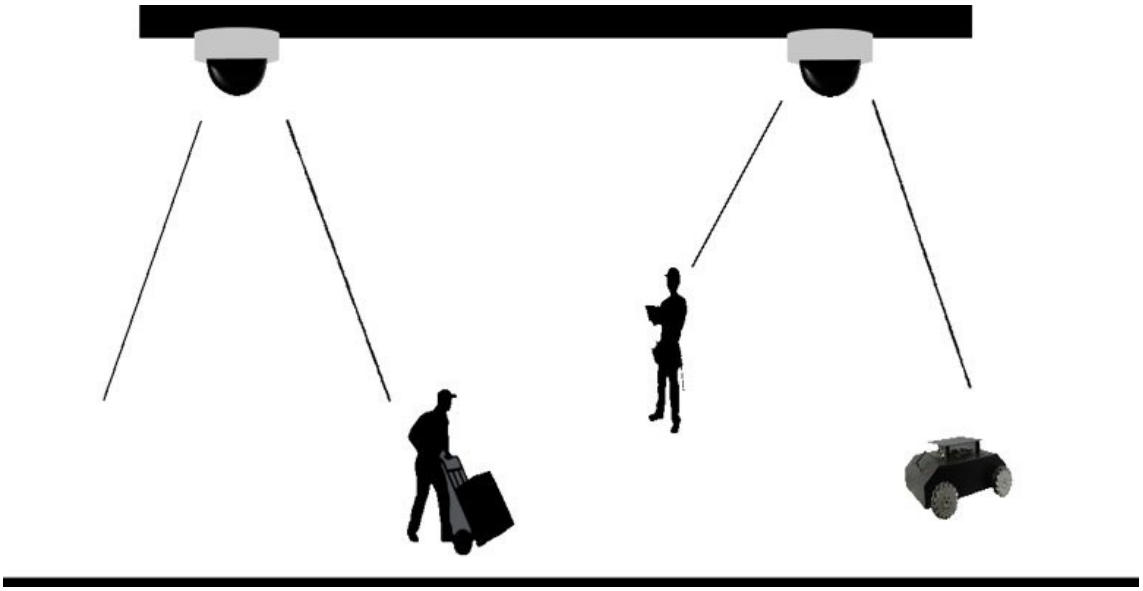


Figure 1.1: Generic Photogrammetry-based Sensor System (GPSS) at Volvo's Factory.

a series of ceiling-mounted Red-Green-Blue (RGB) cameras that can "see around corners" and have the ability to detect and track humans. For this factory for the future to become a possibility, we need an Artificial Intelligence (AI) system that can predict the poses of humans so that the robot can plan its trajectory safely, keeping a safe distance from the workers.

1.3 Project Description

In this project, we are using a series of RGB cameras mounted on the ceiling of a manufacturing plant to detect humans and predict their poses. The top view orientation of the camera gives a wider view of the factory floor with minimal obstruction. For the safety of workers, ATRs must always keep a safe distance from the workers. At the same time, as ATRs and humans will work closely together in the Unified Environment, the ATRs should also be allowed to come close to the human (approximately an arm's length distance), This requires that we should determine their 3D positions with high precision.

Assuming that the camera's orientation with respect to the factory floor is known, a 2D detection of the human feet can be transformed into a 3D estimate of the feet's location under the assumption that the feet are placed firmly on the floor and the human is standing. This constraint simplifies the problem and allows for quite accurate 2D / 3D bounding box predictions of (standing) humans from simple 2D detections of the human's feet in the image.

Though various deep learning approaches that can detect 2D human poses [5,15–17] exist, however just using these models without training on top-view data generates

below-par results. The reason is that these models were trained on many standard datasets consisting of almost no top-view images. In fact, we observed that at the time of this thesis, there were no datasets comprising top-view images of humans. Therefore, the difficulty here involves gathering ground truth data from Volvo's factories that can be used to train the chosen model. Since, we wanted to avoid extensive manual labour in annotating data, we developed a semi- automatic labelling or annotation approach based on utilizing multiple views from several calibrated monocular cameras including the top-view.

1.4 Objectives of the Project

The goal of our thesis is to develop and train a deep learning-based model to detect and localize humans from ceiling-mounted cameras with a focus on minimizing false negatives on human detection and maximizing feet key-point detection accuracy.

To achieve this goal, we aim to:

- Evaluate and select various SOTA human pose estimation models with emphasis on real-time implementation on edge cloud and inference speed.
- Develop a semi-automatic data annotation pipeline that can leverage views from multiple cameras to automatically generate the required annotated top-view dataset.
- Fine-tune the chosen model with the generated dataset, and compare the results with the original model based on both standard and custom metrics chosen based on the requirements from Volvo's use case.

1.5 Challenges

An important challenge in this project, when compared to other human pose estimation tasks was that we need to predict the pose of humans from images and videos of top-view ceiling-mounted cameras instead of the commonly used side-view. This can lead to complex scenarios where all body key points would be not visible and the key points might be subjected to varying degrees of occlusions compared to the usual case. Key points are a set of distinctive anatomical landmarks on the human body that are used in pose estimation algorithms to determine the position and orientation of the body in space. Some of body key points are head, shoulder joints, ankles, wrists etc. This problem is illustrated in Figure 1.2, where all body key points (marked as dots) of the human in the two side view images are all visible, whereas most of the key points are occluded in the top-view. This makes it harder to estimate human poses with accurate key points in top-view images.
Another challenge was explained in Section 1.3 which describes the general lack of available annotated top-view data sets in the wild which we can readily use for our application.

Figure 1.2: Human key points occlusions in the top-view image.

2

Theory

This chapter aims to provide an overview of the relevant theoretical frameworks used in the thesis. Since we use deep learning methods to solve the human detection and pose estimation tasks, this chapter has been divided into four main sections: Introduction to deep learning, Object detection using Deep learning, Human pose estimation using Deep Learning, and Camera-based computer vision.

2.1 Introduction to Deep Learning

Deep learning is a family of machine learning that is based on artificial neural networks. Artificial Neural networks (ANNs) are complex systems resembling the functions of how neurons work in a biological brain context. These biological neurons are wired together to form a complex network, where information is transmitted from one neuron to another using electrical signals as an intermediate connection. Each neuron processes the information it receives from its connected neurons and produces an output to be sent to other connected neurons. Artificial neural networks are mostly based on the neuron model proposed by McCulloch and Pitts [18]. Consider a neuron i - connected to N number of neurons, with neuron j in its vicinity. McCulloch and Pitts modeled the activation s_i of neuron i using equation 2.1. Here, v_j represents the incoming signals from the j connected neurons, w_{ij} represents the connection strengths (weights) between neuron i and j. The function $g(.)$ is called an activation function, and b_i represents the activation threshold or bias [19].

$$s_i = g\left(\sum_{j}^{N} w_{ij} v_j + b_i\right). \tag{2.1}$$

A simple illustration of this neuron model is shown in Figure 2.1

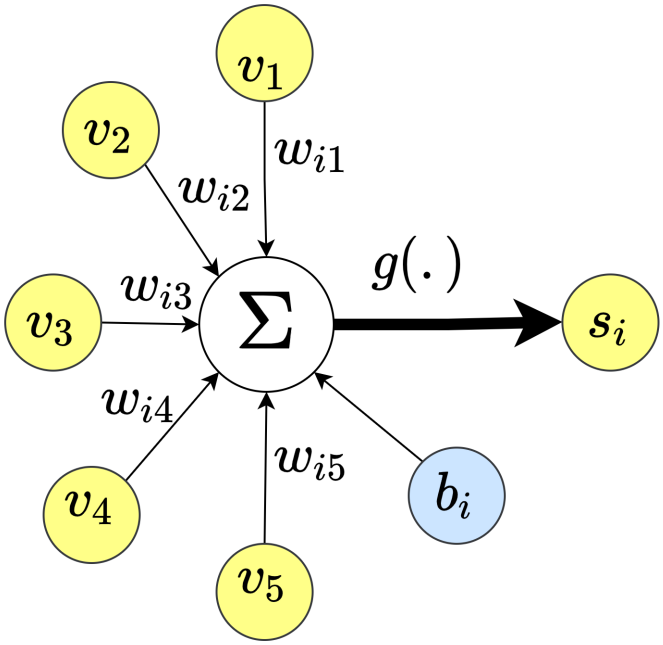


Figure 2.1: Illustration of an activation process of McCulloch-Pitts neuron. $w_{i1,i2,\ldots,i5}$ are connection strengths(weights) of five other neurons connected to neuron i. The incoming signals $v_{1,2,\ldots,5}$ are multiplied with the connection strengths and summed together. The activation function $g(.)$ is applied to this sum to turn it into the output s_i. This figure is based on Figure 2 in [1].

2.1.1 ANN architectures

We can visualize ANNs as weighted directed graphs with nodes representing the artificial neurons, and directed edges with weights are the connections between other neuron outputs and inputs. Hence, according to [1], ANNs can be grouped into two main categories based on their connection pattern:

- Feed-forward networks, in which graphs have no loops.
- Recurrent (or feedback) networks, in which loops, occur because of feedback connections.

Different connectivities in these two types, lead to different behaviours in the network. For instance, feed-forward networks are generally considered to be *static*, producing only one set of output values that is independent of the previous network state. Recurrent networks, on the other hand, are dynamic systems meaning that when a new input pattern is presented to it, due to its feedback paths, inputs to each neuron are then modified to enter a new state.
For this thesis scope, we would be looking deeper into a more common family of feed-forward networks called multi-layer perceptrons and Convolutional Neural Networks.

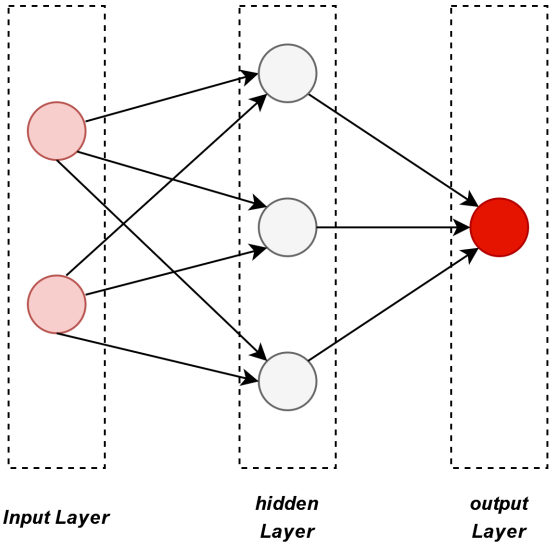


Figure 2.2: An illustration of MLP with 3 layers; an input layer, a hidden layer, and an output layer.

2.1.2 Multi-layer perceptrons and fully connected layers

Perceptrons are single-layered, simplest neural networks similar to McCulloch-Pitts model shown in Figure 2.1. However, it was realized that these single-layer perceptrons were limited in their capability to learn only linearly separable patterns as proved by Minsky and Papert in 1969 [20]. Multilayer perceptrons(MLPs) are the most common part of the family of feed-forward networks, where several of these Mcculloch-Pitts Neurons are coupled together in a layered structure. The left-most layer is the **input layer** and the right-most layer is the **output layer**. The intermediate layers are called **hidden layers**, where the states of the hidden layers cannot be read out. Input data fed to the neural network is captured by the input neurons. A single hidden layer MLP is illustrated in Figure 2.2.

Fully-connected layers, also known as linear layers, connect every input neuron to every output neuron and are commonly used in neural networks. In figure 2.2, the hidden layer shown is also a fully connected layer, as is the case in a traditional MLP. It is also usually the last layer that is used to perform the classification decision in classification tasks.

2.1.3 Activation Function

In ANNs, each layer's output is computed by first calculating the sum of products of inputs and their corresponding weights which is then passed over an activation function. This output is supplied as the input to the next subsequent layers. Therefore, activation functions basically transform the input signal into the output signal. The requirement of a neural network is to learn and represent any arbitrary, complex function to map the inputs to the outputs. Thus, if a neural network uses a linear function, it would lack the ability to extract complex information represented by

non-linearity in the real world. Therefore, non-linear functions are the most commonly used than linear functions in real-world applications. Some of the common activation functions [21] are illustrated in Figure 2.3.

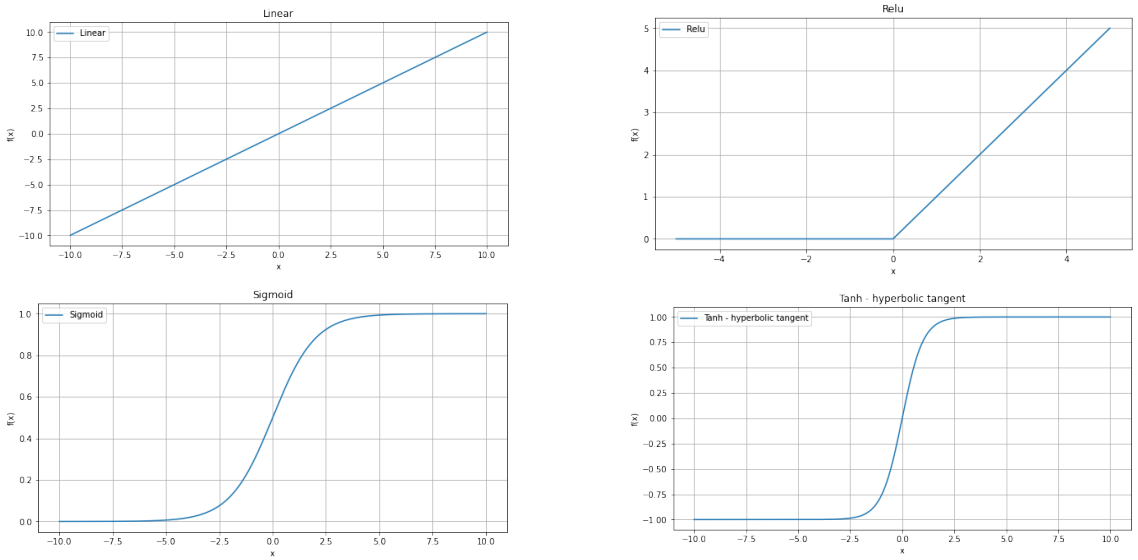


Figure 2.3: Common activation functions used in ANNs.

2.1.4 Forward propagation

Forward propagation refers to the steps performed to compute the output of the neural network given the input data. In other words, forward propagation refers to the calculation and storage of variables for a neural network from the input layer to the output layer in order.
Consider a simple neural network represented in Figure 2.2 which consists of only one input, hidden and output layer. Also, let's assume that the bias or activation threshold term is neglected or zero and the input is $x \in \mathbb{R}^d$.
The intermediate variable is:

$$z_2 = W_1^\top x \tag{2.2}$$

where W_1 is the weight parameter of the hidden layer and z_2 is the intermediate output of the hidden layer.
Next, applying the activation function θ_1 on the intermediate output of the hidden layer, we get the activation function output a_2 as follows in Equation 2.3

$$a_2 = \theta_1 (z_2) \tag{2.3}$$

Next, the overall output of the neural network, which is obtained by propagating the intermediate activation output from the hidden layer, can be represented as Equation 2.4.

$$\begin{aligned} z_3 &= {W_2}^\top a_2 \\ y &= \theta_2 (z_3) \end{aligned} \tag{2.4}$$

where W_2 is the weight parameter of the hidden layer and z_3 is the intermediate output of the output layer, the activation function of the output layer is represented as y. For larger networks with many layers, the output of each layer becomes the input for the next layer, and the above process of computing the weighted sum and applying the activation function is repeated for each subsequent layer until the final output is obtained. This final output is typically a vector of probabilities or a scalar value, depending on the task at hand. For instance, in a classification task, the output could be a vector of probabilities for each class, while in a regression task, the output could be a scalar value.

2.1.5 Back-Propagation

Back-propagation [22] is a neural network's training procedure that repeatedly adjusts the weight parameters of the network aiming to minimize a measure of the difference between the actual output and desired output, also known as the loss function L.
Essentially, the loss function is a mathematical function that quantifies the difference between the predicted output of the neural network and the ground truth output. The loss function is used to measure the performance of the network during training and to adjust the weights and biases of the network to improve its performance. There are many different types of loss functions used in deep learning, depending on the type of problem being solved. For example, for regression problems where the goal is to predict a continuous output, a common loss function is the mean squared error (MSE) loss. For classification problems, where the goal is to predict a discrete output, binary cross-entropy and categorical cross-entropy loss functions are commonly used.
For supervised learning, the actual outputs are essential for error calculations which are afterwards propagated to every node in the previous layers. This error e, Equation 2.6 is obtained as a gradient of the loss function L with respect to each layer's weights W_i given the input of the node x and the activation function θ.

$$a_i = \theta\left(W_i x\right) \tag{2.5}$$

$$e = \frac{\partial L}{\partial W_i} = \frac{\partial L}{\partial a_i}\frac{\partial a_i}{\partial W_i} \tag{2.6}$$

The gradient's computation requires the application of the chain rule in order to compute the partial derivative $\frac{\partial L}{\partial W_i}$. Using these gradients of the loss functions, weights are updated by an optimization algorithm such as gradient descent (or its variants) [23] to move the model towards the minimum of the loss function.

2.1.6 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a family of neural networks mainly used for processing images or visual data [24]. CNN-based architectures have become predominant in the field of computer vision. For instance, in the famous ImageNet challenge [25],use of CNNs boosted significant performance [24].

CNNs have become the go-to models because of their architecture. They are specifically designed to extract information from 2D and higher-order input spaces. Thus this process eliminates the need for various traditional and manual image processing methods.

CNNs achieve this function by incorporating a special mathematical operation called convolution (Equation 2.7).

$$y = x \circledast w \tag{2.7}$$

Here, the input data to the node is x and the output from the network is y. The important difference is on how the weights w, mostly referred to as *kernels* are applied to the input x. Instead of applying weights through a traditional multiplication operation as seen in the case of MLPs in Equation (2.1), they are applied through a convolution operation which is defined as *"computing the weighted average of a point of data by its adjacent points of data"* [19]. Thus, in other words, CNNs weights are specifically designed to form a convolution filter that is replicated over the whole visual field of the input. Another speciality of CNN is that all the weights of a convolution layer are the same thereby decreasing the number of parameters of the network and thus simplifying the training process. These filters convolves each pixel it covers and each output of these filter forms a feature map. Convolutional layers are usually defined by the number of feature maps, kernel size, and stride parameters (a parameter of the CNN's network filter that modifies the amount of movement over the image in turn affects the encoded output volume).

2.1.6.1 CNN structure

The layers usually part of CNNs are listed below:

- **Convolutional layer or CONV** - This layer serves as a feature extractor. Here nodes of a CONV layer perform convolution on different parts of the image known as local regions each computing a dot product between their weights and this local region.

- **Pooling/Subsampling** - This layer performs a down-sampling operation along the spatial dimensions (width, height) to eliminate variance within local regions of the image by reducing the spatial size. This also reduces the number of parameters and the computational load in the network. A popular type of pooling is *max-pooling* which extracts the maximum value of the sub-region on which it is applied.

- **Fully connected** - This layer is the same as the one found in traditional MLP. Fully connected means that every neuron in the previous layer is connected to every neuron. This layer performs the reasoning of the network by processing all the features through the network.

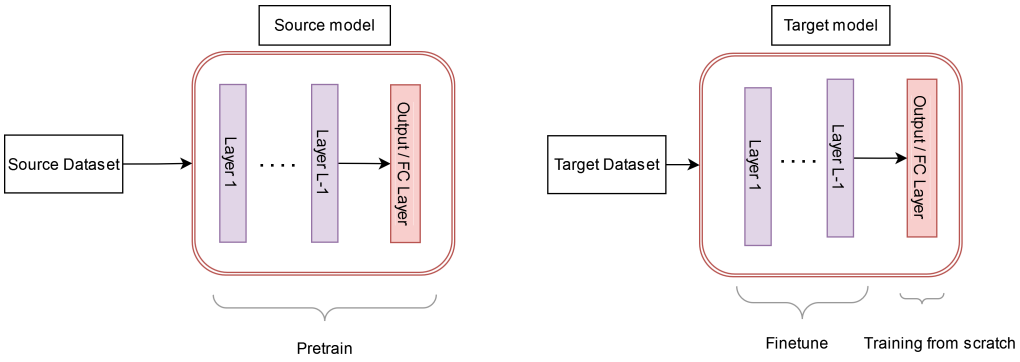


Figure 2.4: Pre-training vs Transfer learning vs fine-tuning.

- **Classifier** - Outputs posterior probabilities for each class, usually consisting of a soft-max function with a cross-entropy loss function. The soft-max function takes a vector of scores and squashes it to a vector of values between 0 and 1 that add up to 1.

Generally, a standard convolutional neural network consists of many pairs of convolutional layers and a subsequent max-pooling layer followed by one or more fully connected layers using a RELU activation function [26].

2.1.7 Transfer Learning

Although deep learning technology has achieved great success, it still has some limitations for certain real-world applications. In an ideal setting, we assume that there are abundant labelled training data and also that these would resemble similar characteristics of the application that we test the model on. However, collecting sufficient training labelled data is often expensive and time-consuming. Semi-supervised learning approach [27] can be considered as a solution for this problem but often these methods result in unsatisfactory results or add more complexity.
Transfer learning aims to improve the performance of a network in a new but similar domain as the network was trained on. In this way, the dependence on a large number of target domain data can be reduced [28]. For instance, a network trained to identify images of cats and dogs can be *transfer learned* to identify other animals. According to the literature, three major transfer learning scenarios exist for CNNs as explained in the following categories, refer Figure 2.4:

- **Pretrained CNN as fixed feature extractor**: Here, consider a pre-trained CNN on ImageNet for example, where the last fully connected layer (or classifier layer - which outputs the class scores for all the categories) is removed and retrained according to the number of classes based on the new dataset for the target task. This scenario can be used when the new dataset is small and very similar to the original dataset.

- **Fine-tuning CNNs**: This second strategy is when not only to replace and retrain the classifier layer as discussed above but also to fine-tune the weights

of the pre-trained network by back-propagation during training. It is often possible to only fine-tune some higher-level portions of the network as the features of earlier lower layers of a CNN contain more generic features, such as edge detectors that should be useful for many tasks and can be reused but whereas the later layers use larger filters with a larger receptive field to extract high-level features such as object parts, textures, and complex patterns specific to the input dataset.

- **Pretrained models**: Most often training CNNs from scratch can take a couple of weeks even on multiple GPUs. Hence, developers usually release checkpoints of CNNs for the benefit of others to use for fine-tuning. This scenario can also be used when the new dataset is large and very different from the original dataset. For example, the mmDetection [29] library has a Model Zoo [30] where contributors share their network weights.

2.2 Human detection (Object detection using deep learning)

Human detection is an application of the original object detection task. This detector is a computer vision problem dealing with localizing instances of semantic objects of a certain class (here humans) in digital images and videos [31]. The location of these object instances is commonly represented by a rectangular bounding box that is drawn around the instance and contains every part of it, as shown in Figure 2.5

Figure 2.5: Example of human detection result with a bounding box.

Object detection can be divided into 2D object detection where the bounding box is drawn on a 2D plane like on a camera image whereas, in 3D object detection, the bounding box is 3D. Here, we concentrate on 2D object detection and particularly on a specific two-stage detector known as Faster R-CNN which would be elaborated in detail in the following sections. Now, we will look into some of the standard datasets available for object detection tasks.

2.2.1 Datasets for object detection

Datasets have always played a key role throughout the history of object recognition research, not only as a benchmark for comparing the performance of competing algorithms but also as pushing the field toward increasingly complex scenarios. For generic object detection tasks, there are four famous datasets [32].

2.2.1.1 ImageNet dataset

An important large-scale benchmark dataset is ImageNet dataset [33]. It is a result of The ImageNet Large Scale Visual Recognition Challenge which is a benchmark in object category classification and detection on hundreds of object categories and millions of images. ILSVRC2014 has 200 object classes and nearly 450k training images, 20k validation images, and 40k test images.

2.2.1.2 PASCAL VOC dataset

This popular dataset was the result of a multi-year effort from 2005 to 2012. The PASCAL VOC dataset [34] contains 20 object categories spread over 11,000 images. The 20 categories can be considered as four main branches; vehicles, animals, household objects, people. However, these PASCAL datasets suffer from data class imbalance for instance in VOC2007 dataset the class person is nearly 20 times larger than the smallest class.

2.2.1.3 MS COCO dataset

MS COCO dataset [35] has come to be the de-facto standard for object detection just like ImageNet was in its time. MS COCO database was a response to the criticism that ImageNet was largely atypical of real-world scenarios. It contains over 330,000 images with more than 2.5 million object instances labelled across 80 object categories, making it one of the largest and most diverse datasets available.

2.2.1.4 Open Images dataset

Open Images [36] is a dataset of 9.2M images annotated with image-level labels, object bounding boxes, object segmentation masks, and visual relationships. Open Images V5 contains a total of 16M bounding boxes for 600 object classes on 1.9M images, which makes it the largest existing dataset to date.

For our work, we don't use any of these standard datasets in our models, mainly because these open datasets are trained only on side view images and using them would provide poor results. Still, we adopted the data format and the evaluation metrics from MS COCO dataset such as mean average precision (mAP) as they are widely used in the computer vision community allowing comparisons using a common standard benchmark.

2.2.2 Faster RCNN model architecture

Faster RCNN is a type of object detection model that is built on top of the popular R-CNN (Region-based Convolutional Neural Network) object detection model [37]. The main goal of Faster RCNN is to improve the speed of the R-CNN model while still maintaining a high level of accuracy.

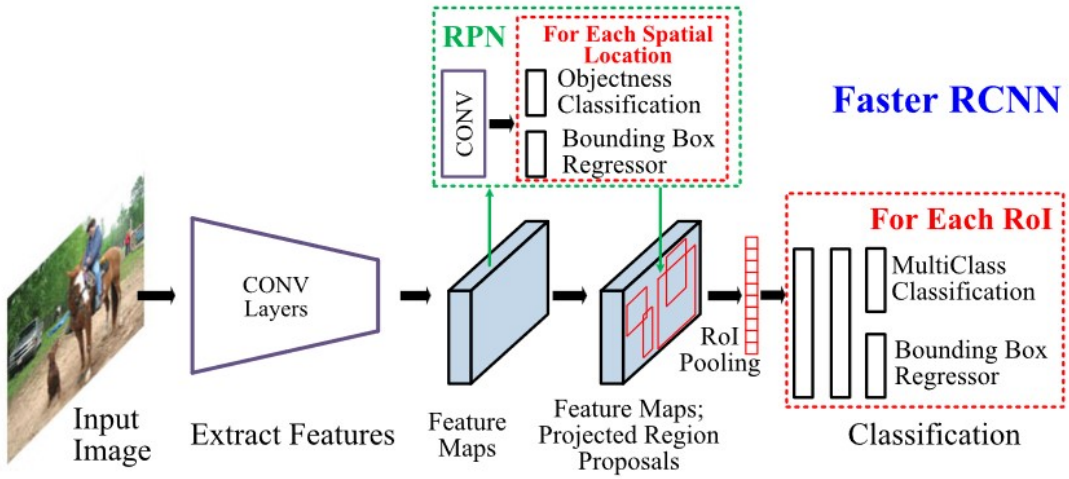


Figure 2.6: Faster RCNN Architecture overview.

Faster RCNN consists of three parts: a base network, an RPN (Region Proposal Network) [37] and a classification layer. The base network is typically a pre-trained CNN (Convolutional Neural Network) model, such as VGG or ResNet, that is used to extract features from the input image [38, 39]. The RPN is a fully convolutional network that takes the feature map produced by the base network as input and generates a set of object proposals, or potential regions in the image where an object might be present [37]. The RPN then passes these object proposals to the Fast R-CNN model, which classifies each proposal as either an object or background and refines the boundary box around the object [37].
In this thesis, we would be using the above Faster RCNN model for its efficiency and accuracy discussed briefly in the Methodology section 3.

2.3 Human Pose Estimation using deep learning

Human pose estimation is a common computer vision task with a focus on the detection of human body key joint points, such as ears, shoulders, elbows, wrists, waist, knees and ankles, etc., from a single image. There are several approaches to solving the problem of human pose estimation, ranging from traditional model-based methods to more recent deep learning-based approaches. Accurate human pose estimation is a challenging task especially due to the variability present in human appearance, pose, and occlusion. The deep learning model used in this project detects 17 key points of the human body in the image, similar to Figure 2.7.

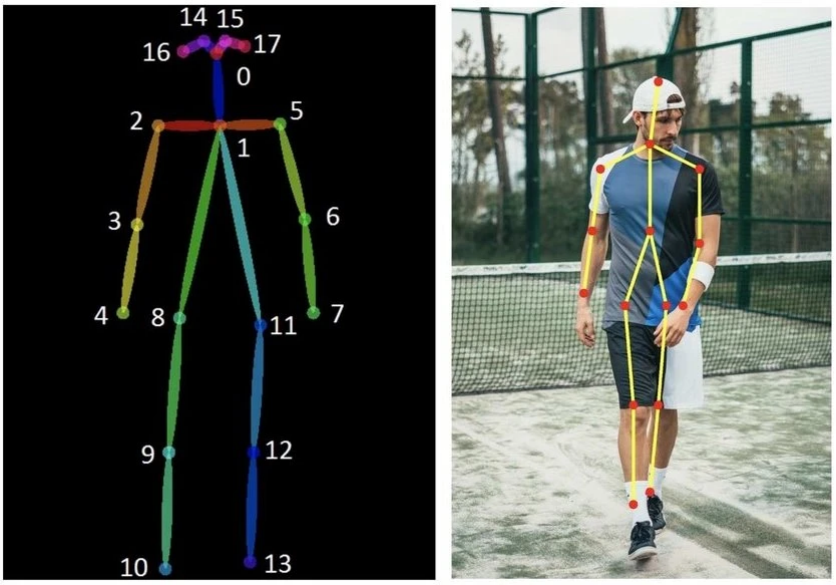


Figure 2.7: Human Keypoints in COCO Dataset [2].

2.3.1 Datasets for human pose estimation

A good human pose estimation model requires training on a suitable dataset. In this case, the dataset is a collection of videos or images with coordinates of key points of the humans in the image. There is no standard for the number of key points therefore the number of key points is one of the selection criteria of the training dataset. There are several public datasets available and there are features that differentiate one dataset from another. As the input feed in this project is from top-view cameras of a factory floor, occlusions are commonly present. Based on this assumption, the following are the datasets that were considered during the selection of suitable training datasets.

2.3.1.1 MPII

This dataset was created in 2014 and has close to 25K images with over 40K people. The data-set comprises human performing different activities and overall there are 410 labelled activities. The feature of this data set that would be useful to our project is that the test set contains body part occlusions and head orientations [40].

2.3.1.2 OCHuman

This dataset was created in 2019 and focuses on heavily occluded humans. This can be seen in Figure 2.8. The annotation provides bounding-box, human poses, and instances. Even though the size of the dataset is small with close to 5000 images, the average MaxIoU of each person is 0.573. Therefore, a pre-trained model trained on this dataset can solve challenging pose estimation tasks with respect to occlusions [3].

Figure 2.8: Sample of OCHuman Dataset [3].

2.3.1.3 Human 3.6M

This large-scale dataset contains 3.6M 3D human poses. This dataset was also created in 2014 with 11 professional actors. There were 17 scenarios in this dataset and each image has 24 keypoint annotations [41].

2.3.1.4 COCO (Common Objects in Context)

COCO dataset is the most popular dataset for object detection, instance segmentation and pose estimation tasks. For keypoints detection there are over 200K images and 250K subjects. This dataset has 17 keypoints. The annotation has both localization of humans using bounding box coordinates and their respective keypoints [35].

2.3.2 Approaches and Networks for human pose estimation

In real life conditions, an image can contain multiple humans and each person might have different poses. This situation increases the chances of occlusions and complexity in estimation of poses. To overcome this problem, human pose estimation can be categorized into two approaches [42].

2.3.2.1 Top-Down

In this two part approach, for each person, the human is localized using a human detector first, then for each segment within the bounding box regions of each detected human, keypoints are predicted. The advantage of top-down approach is the capability to split the task into multiple relatively easier tasks of object detection and single-person pose estimation. As the object detector performs well in detecting small candidates, the pose estimation model is also expected to perform well [43]. As the performance of this approach depends on the human detection result, images with tightly crowded scenes may lead to detection errors, resulting in poor pose estimation [42].

2.3.2.2 Bottom-Up

In this approach, first, all the key-points of the human are detected. Then in the optimization stage, the detected key-points are associated with the corresponding targets [44]. This approach is robust to occlusion and complex poses. However, since this method does not have the structural information of the human body, the chances of false positives,occurring are high. In human pose estimation tasks, false positives refer to cases where the pose estimation algorithm detects a body part or joint that is not actually present in the image. Reducing false positives in human pose estimation is critical to ensure that the algorithms produce reliable results and are useful for various applications.

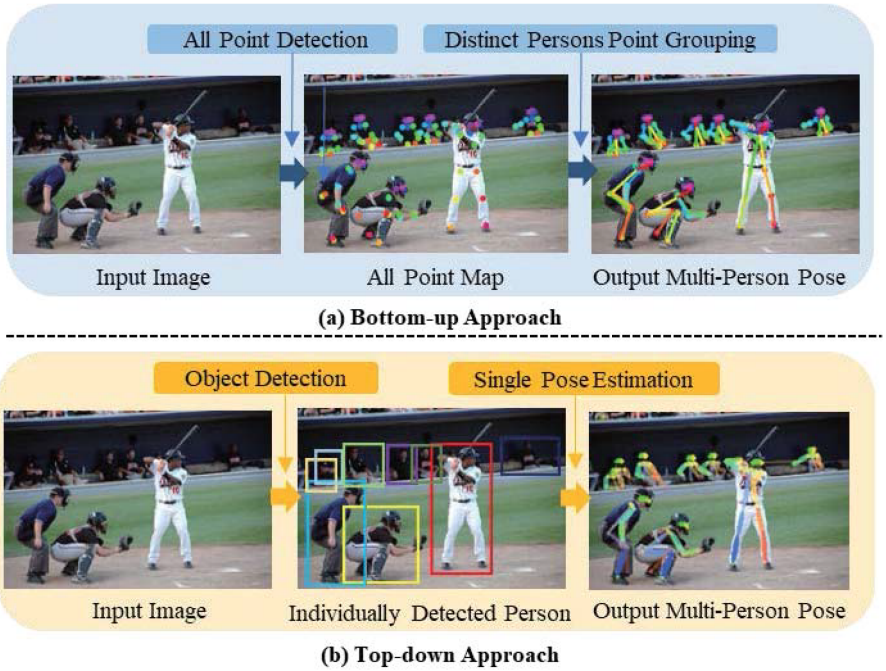


Figure 2.9: Human Pose Estimation algorithm.
(a): The example of a bottom-up approach. (b): The example of a top-down approach [4].

Even though the above approaches have their own features, the performance of a pose estimation model depends as well on the network used. The following table shows pre-trained models of various networks and their corresponding results.

Network	Input Size	mAP	Flops (GFLOPs)	GPU Inference Speed (FPS)
Hourglass 52	(3×256 × 256)	0.726	28.67	25.50 + 1.68
HRNet W48	(3×192 × 256)	0.756	15.77	15.03 + 1.03
LiteHRNet 30	(3×192 × 256)	0.675	0.42	11.86 + 0.38
ResNet-50	(3×192 × 256)	0.718	5.46	64.23 + 6.05

Table 2.1: Model complexity information and inference speed various top-down models in MMPose. mAP = Mean Average Precision trained with COCO dataset [8].

Network	Input Size	mAP	Flops (GFLOPs)	GPU Inference Speed (FPS)
Hourglass AE	(3×512 × 512)	0.613	221.58	3.55 + 0.24
HRNet W48	(3×512 × 512)	0.665	84.12	5.27 + 0.13
ResNet-50	(3×640 × 640)	0.479	45.62	8.20 + 0.58

Table 2.2: Model complexity information and inference speed various bottom-up models in MMPose. mAP = Mean Average Precision trained with COCO dataset [8].

From the above section, we can infer that even though bottom-up approach is robust to occlusion and complex poses, our current focus is more towards reliable results. Therefore, we have chosen the top-down approach. While selecting the pre-trained model, we wanted our model to have low GFLOPs (Giga Floating Point Operations per second) and high mAP (mean Average Precision). From 2.1, HRNet W48 pre-trained meets our criteria. The relatively low GPU inference speed was a trade-off we accepted.

2.3.2.3 Deep High-Resolution Representation Learning for Human Pose-Estimation(HRNet)

HRNet [5] is a state-of-the-art algorithm in the field of semantic segmentation, human pose estimation and facial landmark detection. This network takes an image of size W x H x 3 and aims to estimate K heatmaps, one for each key point, resulting in heatmaps $\{H_1, H_2, ..., H_k\}$ indicating the confidence of the k^{th} key point network that combines low and high-resolution convolutional networks.
The architecture of the algorithm has multiple sub-networks connected in parallel. Starting from a high-resolution sub-network at the top, high-to-low-resolution convolutional networks are gradually added one after one by downsampling from the previous high-resolution sub-network.

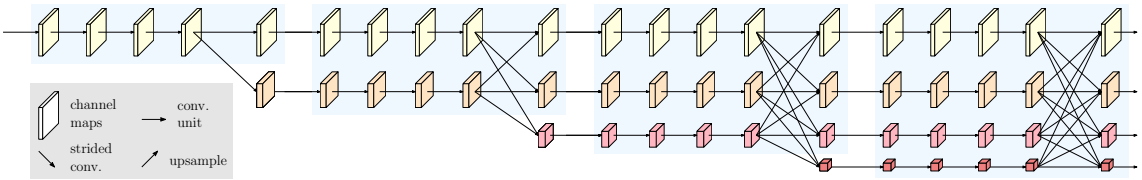


Figure 2.10: An example of an HRNet architecture: Here, there are four stages. The 1st stage consists of high-resolution convolutions. The 2nd (3rd, 4th) stage repeats two-resolution (three-resolution, four-resolution) blocks. More details can be found in [5].

From Figure 2.10 of the architecture, it can be seen that at the end of each stage, a full connection to the multi-resolution group of the next stage can be seen. Here, the information is transferred from high to low resolution and vice-versa through upsampling and downsampling respectively. High-to-low-resolution occurs via strided 3x3 convolutions and low-to-high resolution occurs via 1x1 nearest neighbour upsampling. An overview of this exchange unit can be seen in Figure 2.11.

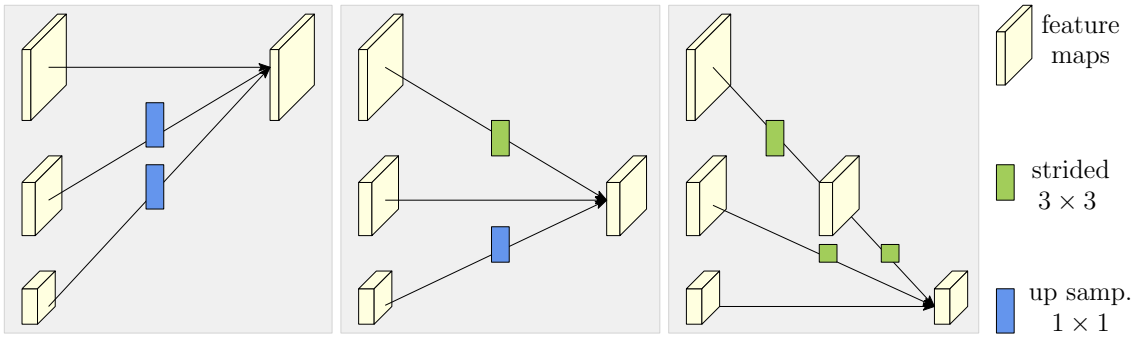


Figure 2.11: HRNet exchange unit: Illustrating how the fusion module aggregates the information for high, medium, and low resolutions from left to right, respectively. Source: [5]

The HRNet body consists of four parallel subnetworks over 4 stages, containing 1, 4, and 3 exchange units for the 2nd, 3rd, and 4th stages respectively. Two different versions of HRNet are used, HRNet-W32 and HRNet-W48, with different numbers of channels. In this way, the network maintains high-resolution representation throughout the whole process.

2.4 Performance Metrics

Evaluating the performance of human pose estimation results requires the evaluation of the human detector and pose estimation model. As the pre-trained models are trained with the COCO dataset, the evaluation metrics of the COCO dataset [35] is used. This section consists of two parts. First, the metric to quantify the prediction of human detection and estimation of the pose is discussed, and in the second part, based on the confusion matrix generated from the previous parts, we evaluate the model using the COCO metrics for precision and recall which are mean average precision (mAP) and average recall (AR) respectively.

2.4.1 Qualitative Analysis of Prediction

The first step in analyzing the performance of prediction requires finding out how much the predicted point overlaps with the actual ground truth. In our case, to find this overlap, we have the Intersection Over Union (IOU) metric from our human detector, and Object Keypoint Similarity (OKS) from our pose estimation model. Even though these evaluators assess in different ways, their results are used to generate a precision-recall curve which is required to calculate mean average precision (mAP) and average recall (AR). This topic will be discussed in the next section.

2.4.1.1 Intersection Over Union (IOU)

Intersection Over Union is used to describe the extent of overlap of two boxes. The value varies between 0 to 1 and the greater the overlap region, the greater the IOU metric value.

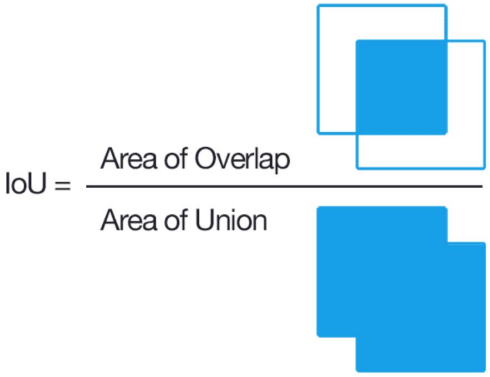


Figure 2.12: Intersection Over Union (IOU).

From the above figure 2.12, we can see that IOU is simply a ratio. The overlap area between the predicted bounding box and ground truth bounding box is the numerator and the denominator is the union of areas encompassed by both the predicted bounding box and the ground-truth bounding box.

2.4.1.2 Object Key-point Similarity (OKS)

This metric gives a measure of how close a predicted key point is close to the ground truth. Object key-point similarity is related to IoU in the key points prediction performance.
For each object, the ground truth key points are of the form $[x_1, y_1, v_1, ..., x_k, y_k, v_k]$ where x,y are the key-point locations and v is a visibility flag defined as v=0: not labelled, v=1: labelled but not visible, and v=2: labelled and visible. However, the predicted v_i is not used during evaluation. The mathematical equation for OKS is given below.

$$\frac{\Sigma_i[\exp(\frac{-d_i^2}{2s^2k_i^2})\delta(v_i > 0))]}{\Sigma_i[\delta(v_i > 0)]} \tag{2.8}$$

where

- d_i is the Euclidean distance between the predicted key point and corresponding ground truth
- s is the object's scale
- k is the per-keypoint constant that controls fall off.
- v_i is the visibility flag.

2.4.2 Quantitative Analysis of Prediction

The second part is to discuss how the results of IOU and OKS are quantified. To start with, let us discuss the four parts of the confusion matrix:

- True Positive: A correct detection (detection with IOU $\geq$ threshold)
- False Positive: A wrong detection (detection with IOU $\leq$ threshold)
- False Negative: A ground truth not detected
- True Negative: The part of the image that was not detected by the model.

Using the above confusion matrix, we find precision and recall.
Precision measures the proportion of predicted positives that are correct. In other words, it is the True Positives of total detections.
Recall measures the proportion of actual positives that were predicted correctly. In other words, it is the True Positives of all ground truths.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2.9}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2.10}$$

Therefore, if a model has high precision and recall, then the model correctly predicts the samples and also predicts most of the positive samples. But if a model has high precision and low recall, it correctly predicts samples as positive but has only a few positive samples. In our context, this means that the detected key points are accurately predicted but only a few key points have been predicted.
As high precision and high recall can be ideal, a Precision-Recall curve is formed to understand the trade-offs based on different thresholds.

2.5 Camera-based computer vision

The training of our pose-estimation model requires images taken from ceiling-mounted cameras. The images captured will be distorted and the parameters and characteristics of the camera have to be found to make the image useful. This section will explain the concepts of computer vision that have been applied to generate the training data for our pose-estimation model.

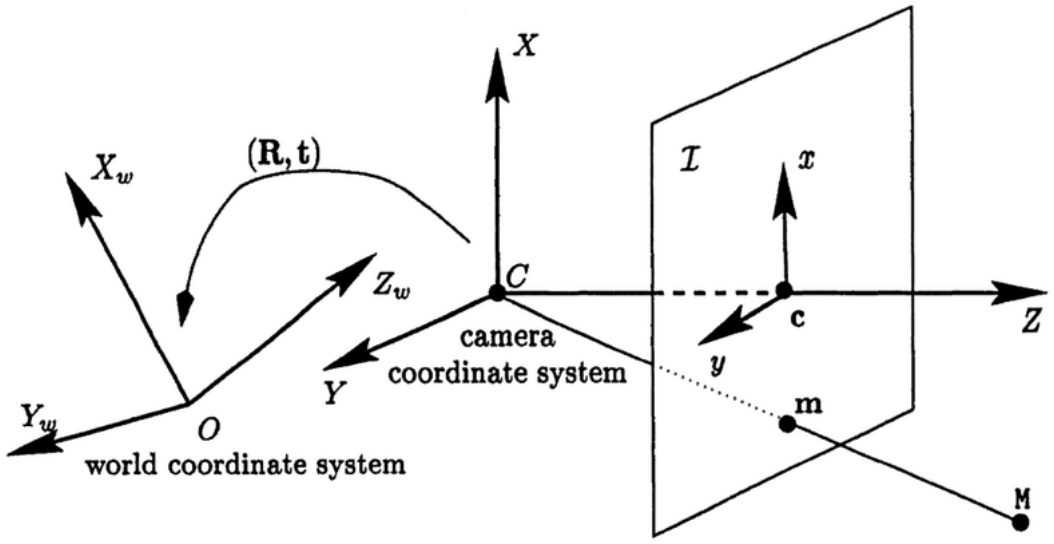


Figure 2.13: World coordinate system with camera extrinsic parameters [6].

2.5.1 Extrinsic Parameters

The extrinsic parameters are the parameters used to describe the transformation between the cameras and their external world. They are also called camera poses. In Figure 2.13, a single point M_w on a world coordinate system with optical center C can be transformed to a new camera coordinate system M_c with centre O by rotation R and translation t with the equation:

$$M_w = RM_c + t \tag{2.11}$$

Here R and t are extrinsic parameters [6] and this Euclidean transformation can be used to describe the extrinsics of a camera, in other words, how a certain camera is placed and rotated in a coordinate system.

2.5.2 Intrinsics Parameters

Each camera has different properties compared to another, hence these properties need to be modelled before using various computer vision algorithms. We call these properties inherent to a camera,the intrinsic parameters. A common way of referring to a certain camera's calibration settings is writing it as a matrix K. These settings are commonly known as camera intrinsics.

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

The variables f_x and f_y represent focal length with respect to the camera sensor for x and y dimensions. Variable s denotes skew to account for misalignment between the camera sensor axis and principal point (or image center). Lastly, the variables c_x and c_y denote the image center in pixel coordinates. The intrinsics for each camera need to be estimated and this process is commonly called as calibration process. These tend to differ for every camera even those of the same model and manufacturer. The calibration process used in this thesis will be explained in Section 3 in detail.

2.5.2.1 Distortion of Image

In an ideal camera, the linear projection model is obeyed, i.e, a straight line in the world would also, result in a straight line in the image. In the real world, on the other hand, many wide-angle lenses have noticeable radial distortions where the projected straight line has a visible curvature. The two common forms of radial distortions as seen in Figure 2.14 are barrel distortion and pincushion distortion [7]. Therefore, to have an accurate image for training and annotation, the image should be undistorted first.

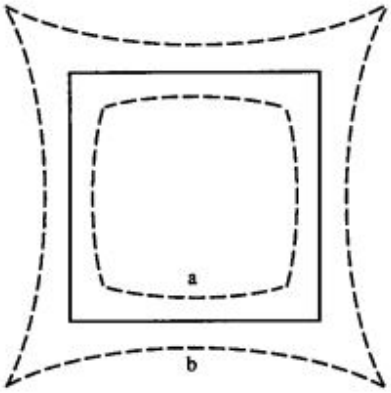


Figure 2.14: Effect of radial distortion. Solid lines: no distortion, dashed lines: with radial distortion (a: pincushion b: barrel) [7].

2.5.3 Camera Matrix

The camera matrix P, also known as the projection matrix, is a 3x4 matrix that represents the relationship between the 3D world coordinates of a point and its 2D image coordinates captured by a camera.The camera matrix P can be written as:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \tag{2.13}$$

$$\mathbf{X}_{cam} = -\mathbf{R}^{-1}\mathbf{t} \tag{2.14}$$

where,

K is a 3x3 matrix known as the camera intrinsic matrix that contains information about the camera's focal length, principal point, and skew coefficient.
R is a 3x3 rotation matrix that describes the orientation of the camera in the world coordinate system.
t is a 3x1 translation vector that represents the position of the camera center in the world coordinate system.

For a camera defined by (2.13), the center of the camera, X_{cam} in the world coordinate system is given by equation (2.14).

The camera matrix P can be used to project a 3D point X in the world coordinate system onto its corresponding 2D image point x in the camera coordinate system, which will be discussed in detail in the below section on 2.5.5.

2.5.4 Triangulation

Triangulation is the process of finding a 3D point given the pixel coordinates of the point in two or more different views as shown in Figure 2.15. To solve this problem, we need to know the projection camera matrices P for all the cameras involved.

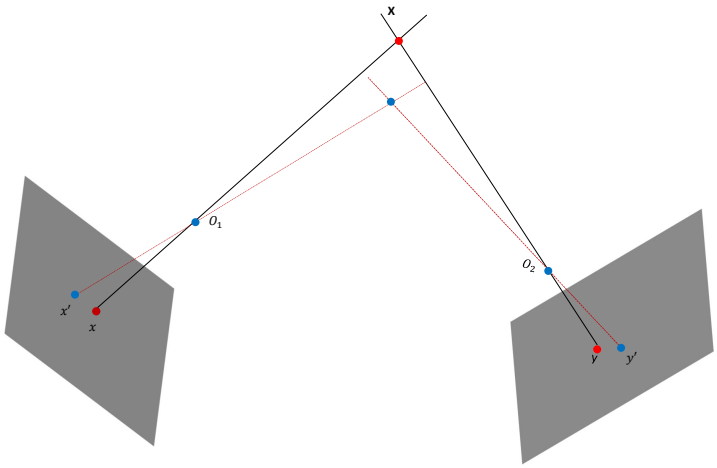


Figure 2.15: Triangulation: In practice, the image points x and y cannot be measured with arbitrary accuracy. Instead points $x\prime$ and $y\prime$ are detected and used for the triangulation. The corresponding projection lines (dotted) do not, in general, intersect in 3D space and may also not intersect with point X.

There are many advanced algorithms for triangulation [45], however the most common and the simplest is the solution utilizing the Direct Linear Transformation (DLT).

For each input image, we have a measurement,

$$\begin{aligned} \lambda_1 \mathbf{x}' &= \boldsymbol{P}\boldsymbol{X}, \\ \lambda_2 \mathbf{y}' &= \boldsymbol{P}'\boldsymbol{X} \end{aligned} \tag{2.15}$$

where,
$\mathbf{x}', \mathbf{y}'$ are the 2D camera-space coordinates of a world point in the first and second cameras.
$\lambda_{1,2}$ are unknown non-zero scale factors.
X represents the 3D world-space coordinate that we intend to recover.
The two equalities (Equation 2.15) mathematically denote that the vectors $\lambda_1 x'$ ($\lambda_2 y'$) and PX ($P'X$) are parallel and therefore the cross products are zero, $x' \times PX = 0$ ($y' \times P'X = 0$). Scale factors were eliminated by cross-products.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \times \begin{bmatrix} {p_1}^T X \\ {p_2}^T X \\ {p_3}^T X \end{bmatrix} = \begin{bmatrix} v p_3^T X - p_2^T X \\ p_1^T X - u p_3^T X \\ u p_2^T X - v p_1^T X \end{bmatrix} = \underbrace{\begin{bmatrix} v p_3^T - p_2^T \\ p_1^T - u p_3^T \\ u p_2^T - v p_1^T \end{bmatrix}}_{A} X = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

where, u,v, are the pixel coordinates of x' and three-row vectors of $\mathbf{P}$ represented

as p_1,p_2 and p_3. But the third row is a linear combination of the first two rows which then only gives 2 systems of equations, which is not enough to solve the 3 unknowns in $\mathbf{X}$. This is expected since we can't determine a 3D coordinate from a single camera view. Since we have two cameras, we can extend the matrix to have more rows.

$$\mathbf{AX} = \begin{bmatrix} vp_3^T - p_2^T \\ p_1^T - up_3^T \\ v'p_3'^T - p_2'^T \\ p_1'^T - u'p_3'^T \\ \vdots \end{bmatrix} \mathbf{X} = \mathbf{0}$$

Since there are more equations than unknowns, we cannot find exact solution, we solve the least squares minimization problem using SVD [46] which allows us to estimate the value of $\mathbf{X}$ and thus the 3D coordinate of any point for which we know the camera-space coordinates from two cameras for the projection matrices P, P' has already been determined. The advantage of this method is that it can be easily extended to any number of two or more views. For our thesis work, we take use two views.

2.5.5 (Re)projection

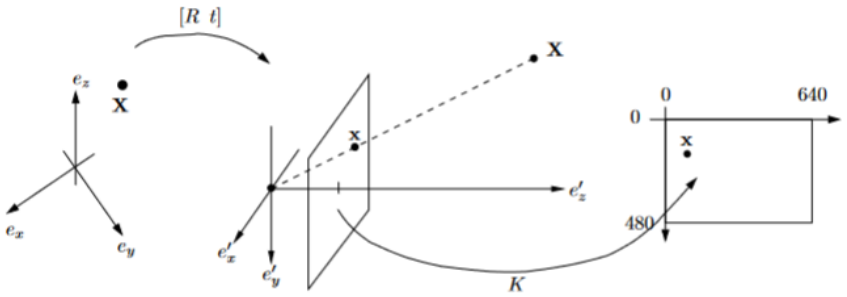


Figure 2.16: The different mappings from 3D to 2D from left to right.

Here, in this thesis, we define re-projection as mapping 3D points to image points in 2D, given that the camera matrix P is known. With this information, one would be able to find the projection of any 3D point of the scene, say a new third camera image point in 2D (up to a scale factor λ). An illustration of the different mappings is shown in Figure 2.16
The equation for this projection is given by,

$$\begin{aligned} \lambda x &= \mathbf{PX} \\ &= \underbrace{\mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}}_{\mathbf{P}} \underbrace{\begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}}_{\mathbf{X}} \end{aligned} \tag{2.16}$$

where Equation 2.16 is usually called the camera equation and this process is illustrated in Figure 2.16.

3
Methodology

In this chapter, the methods used to perform our task of human detection and pose estimation will be discussed as modules. We start by explaining the camera setup which is crucial for our data collection and the semi-automatic annotation pipeline.

3.1 Camera Setup

In our case, the collection and annotation of data require at least three cameras: a ceiling-mounted camera and two front-facing cameras. The images of the ceiling-mounted camera will be used for training and testing the human detector and pose estimation model. The significance of the two front-facing cameras will be discussed in detail in the next section when we discuss the training data annotation, but in brief, the side cameras will be used to predict the human key points from the front view which will be later triangulated and re-projected to the ceiling-mounted camera images to get the required key point annotations for the training data.

Figure 3.1, an image taken from the top-view camera, depicts the setup of the cameras. The two side-view (front-facing) cameras are placed perpendicular to each other, right below the top-view camera. As a result of this arrangement, an object at the center of the three cameras can be captured with minimal overlap.
After the cameras are set up, a continuous synchronized image feed is collected from all three cameras. This data was collected for different processes such as calibration, pose estimation, etc., which will be discussed in the following sections.

3.2 Modules of Human Detection and Pose Estimation

The project has been divided into four modules based on the tasks and their significance. The choices and motivations taken for each module will also be discussed.

3.2.1 Module 1: Image Pre-Processing

As the data for our project are images taken from a ceiling-mounted camera, the first stage was calibrating the camera for the collection of synchronized images.

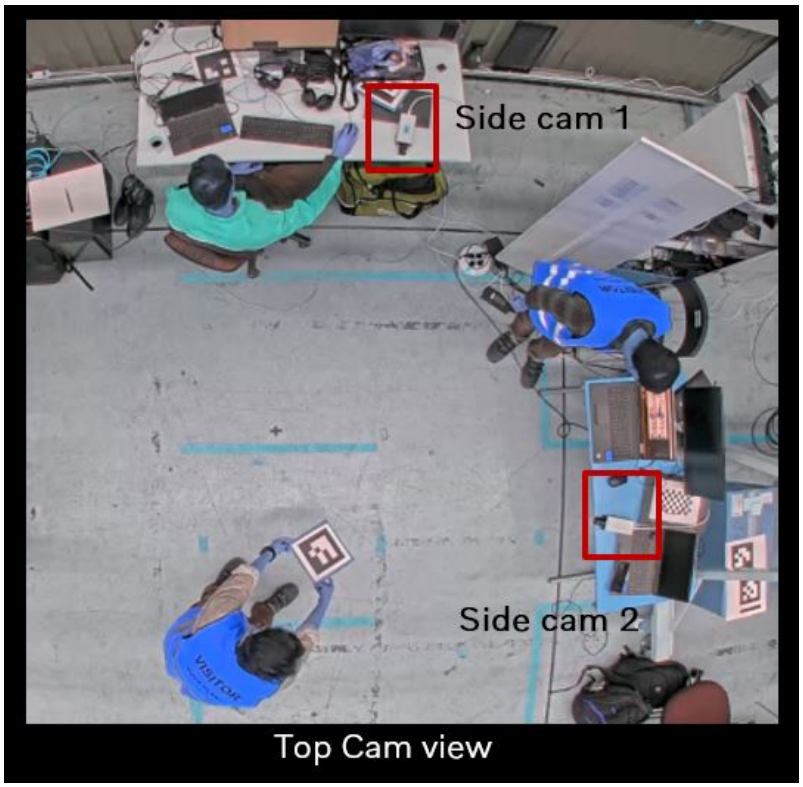


Figure 3.1: Camera Setup with two Side-View cameras taken from Top-view camera.

3.2.1.1 Undistortion of Images (Intrinsic Calibration)

The purpose of intrinsic calibration is primarly to estimate and correct the distortions in images taken directly from the camera. To do this, a set of images with checkerboards were taken. An example of the checkerboard image can be seen in figure 3.3. The undistortion process involves the following steps which were performed using the OpenCV library functions in python.

- cv2.findChessboardCorners: For each checkerboard image, the corners were found. These corner points were then refined using cv2.cornerSubPix and then added to image points. This function also returns if the checkerboard was detected successfully. The object points were manually set based on the checkerboard dimensions.

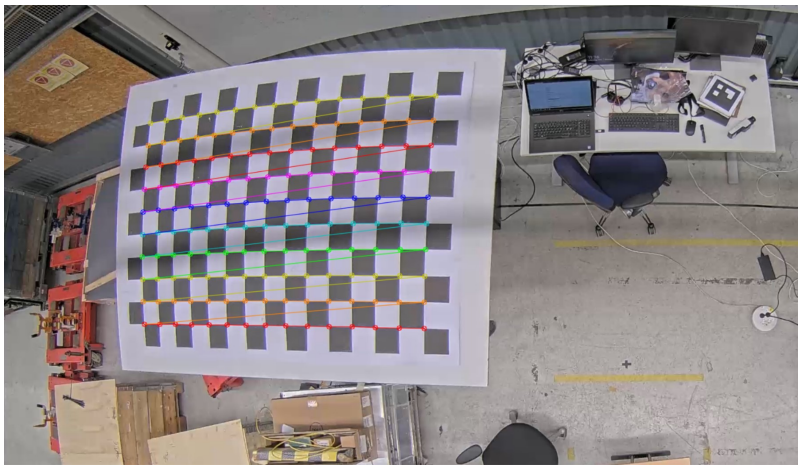
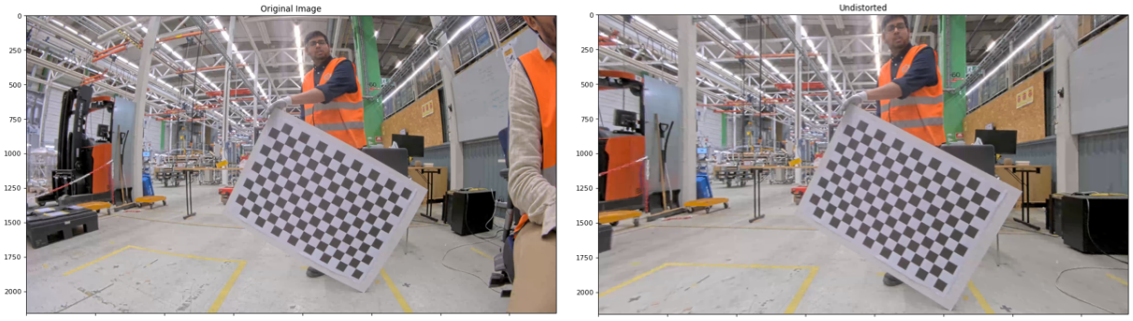


Figure 3.2: Output after findChessboardCorners function.

- cv2.calibrateCamera: With the collection of image and object points, the calibration matrix, distortion coefficients, rotation, and translation vector could be found using this function.
- cv2.getOptimalNewCameraMatrix: This function is used to refine the calibration matrix based on a scaling parameter alpha, which has values from 0 to 1. With alpha as 0, only the sensible pixels or minimum unwanted pixels are retrieved from the calibration matrix. The ROI (region Of interest) is also obtained from this function. This will be helpful for cropping the distorted image.
- cv2.undistort: After the calculation of the optimized calibration matrix, calibration matrix, distortion coefficients, ROI, and shape of the image, we would be able to get the undistorted image from this function. The image that outputs from this function is then cropped based on the shape of the image from ROI. The final cropped image would be our undistortedimage.



(a) Original distorted image. **(b)** Undistorted image.

Figure 3.3: Intrisic calibration mainly to estimate and remove distortion effects from raw camera images.

3.2.1.2 Extrinsic Calibration

Extrinsic calibration is performed to map points from 3D coordinates to 2D coordinates and vice-versa. For this, the orientation of the cameras i.e., rotation and

translation with respect to the scene are computed. The calibration process is similar to intrinsic calibration. Here, Aruco markers are used in place of the checkerboard. An Aruco marker [47] is a binary square marker with wide black border and inner binary matrix. This 4×4 inner matrix determines the ID of the marker. As the field of view for the camera was wide, an Aruco board was used instead of a single Aruco marker. The markers were generated using Aruco marker generator and the markers were printed and stuck in a long vertical board.

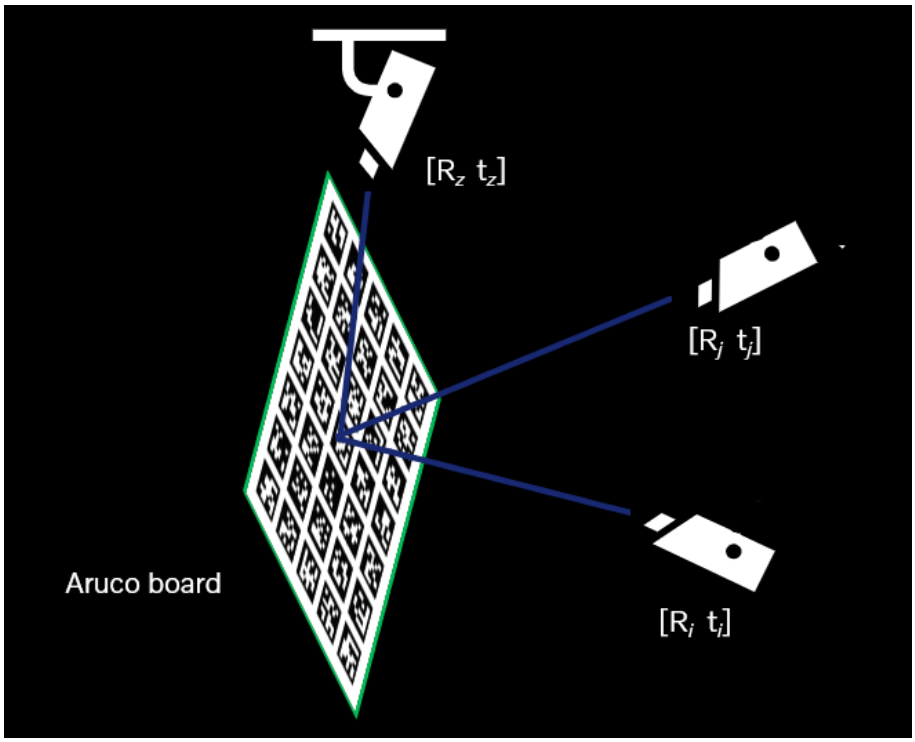


Figure 3.4: Process of extrinsic calibration. Computes the orientation (Rotation and Translation) of the camera with respect to each other in the scene.

Unlike intrinsic calibration, where lots of checkerboard images were needed, a single clearly visible Aruco board on all the cameras was enough to obtain robust extrinsic calibration. The captured image is first undistorted using the steps discussed above in intrinsic calibration. Then, the following steps using the following functions of OpenCV are performed to finally obtain the poses of the cameras in the scene.

- cv2.aruco.detectMarkers: We define Aruco board parameters in openCV and then use this function to detector corners of Aruco markers as shown in Figure 3.5.

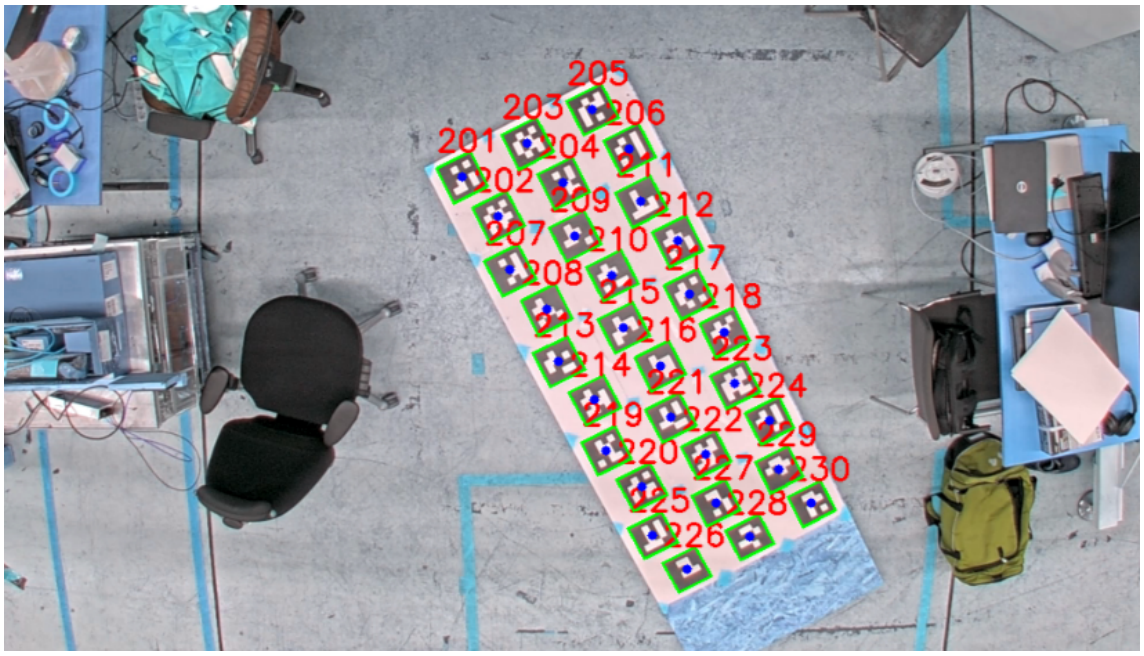


Figure 3.5: Detected corners in an Aruco board.

- cv2.aruco.estimatePoseBoard: Once the corners are detectors, we can estimate the pose ($\boldsymbol{R}$ and $\boldsymbol{t}$) of the camera using this function. Having known all the parameters needed for camera matrix $\boldsymbol{P}$ can then be computed.

3.2.2 Module 2: Semi-Automatic Training Data Annotation

3.2.2.1 Annotation of key points using triangulation and re-projection

As discussed in the camera setup section and from Figure 3.1, in addition to the ceiling-mounted camera, there exists two side cameras. These side cameras were primarily used to aid the annotation of the human key points for the images taken on the top-view camera. To begin with, time-synchronized images from all three cameras were captured. Care was taken to ensure that the person or human was visible on all three cameras. Thereafter, for each image captured on the top-view camera, there are two side-view images that capture the same person from their respective angles. After the images were captured, the human pose key points annotation for the top-view cameras was created with the following steps.

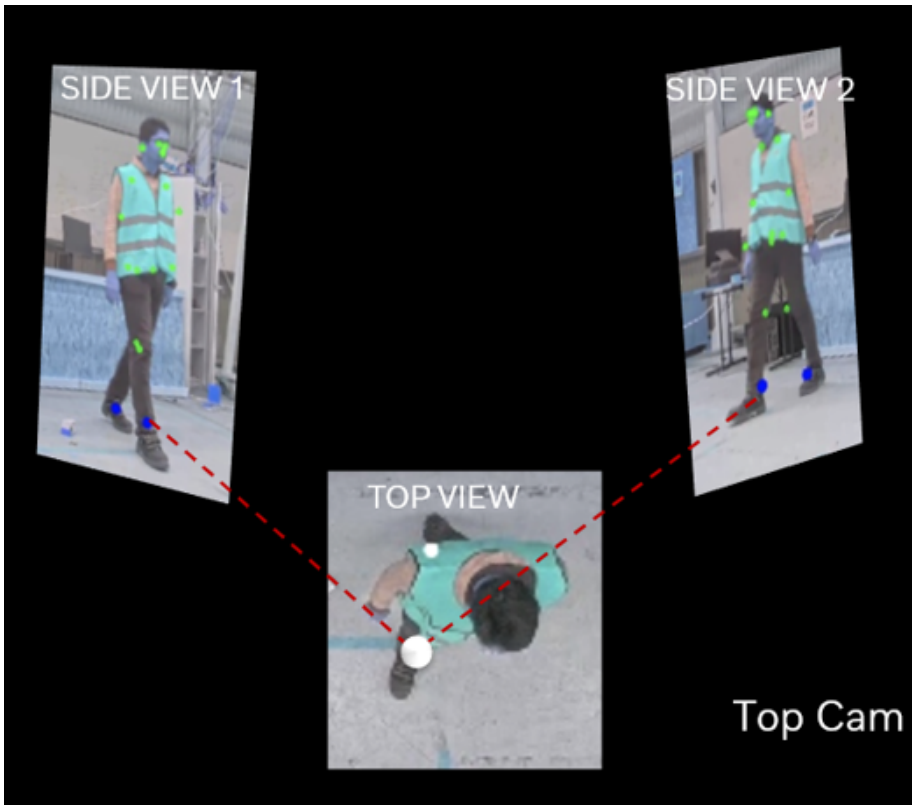


Figure 3.6: Keypoint triangulation using side-view cameras.

For each image captured on the three cameras:

1. First, the human key points were detected on each of the two side-view images using a pre-trained MoveNet model [48]. As the pose of the human on the side-view cameras was completely visible, the MoveNet model detects the 17 body key points fairly well.
2. Next, from the already computed respective camera pose and orientation from the external calibration process, the 17 key points are triangulated into 3D key points using the technique described in Section 2.5.4.
3. Finally, the obtained 3D points are re-projected to the 2D camera coordinates of the top-view camera using the projection matrix of the top-view camera. In this way, each of the predicted human key points of the side view camera images was re-projected to the top-view camera image obtaining the annotated key points.

3.2.2.2 Bounding Box Annotation

The dataset for training our human detector and pose estimator models requires key points coordinates and the bounding box coordinates used to localize the human. A simple method to do this labeling is to draw a bounding box on the top-view images. In order to avoid a manual process, we wanted to develop an automatic process of annotating the bounding boxes. Below, we explain our process of obtaining bounding box annotations for the top-view camera images from the human ankle key points

annotations. Here, our main assumption that simplifies this process is that both ankle points lie on the ground. This assumption is realistic as all factory floors are mostly likely level ground and humans detected at factory floor will be only standing.

1. We compute the camera matrix for the top-view camera by using an Aruco board placed on the ground (same process as external calibration).
2. The left and right ankle 2D key points of the top-view camera obtained in the previous section are transformed into 3D world coordinates, using the camera matrix of the top-view camera.
3. Based on the three-dimensional left and ankle key points, a 3D bounding box structure of an arbitrary dimensions (1.8 m height and 0.5m width) is centered on the midpoint of the left and right ankle points. This assumed dimensions is chosen such that it fits well with the average human's dimensions. By this way, we ensure that the person is covered from head to ankle and shoulders.
4. The points of the outer edges of the 3D bounding box are then transformed back to the image coordinates of the top-view camera. These points, after re-projection, cover the human on the top-view camera. As the positions of the points changes during transformation, regardless of the orientation of the person on the top-view camera, the 2D bounding box covers the person completely.

These steps are represented in the below figure.

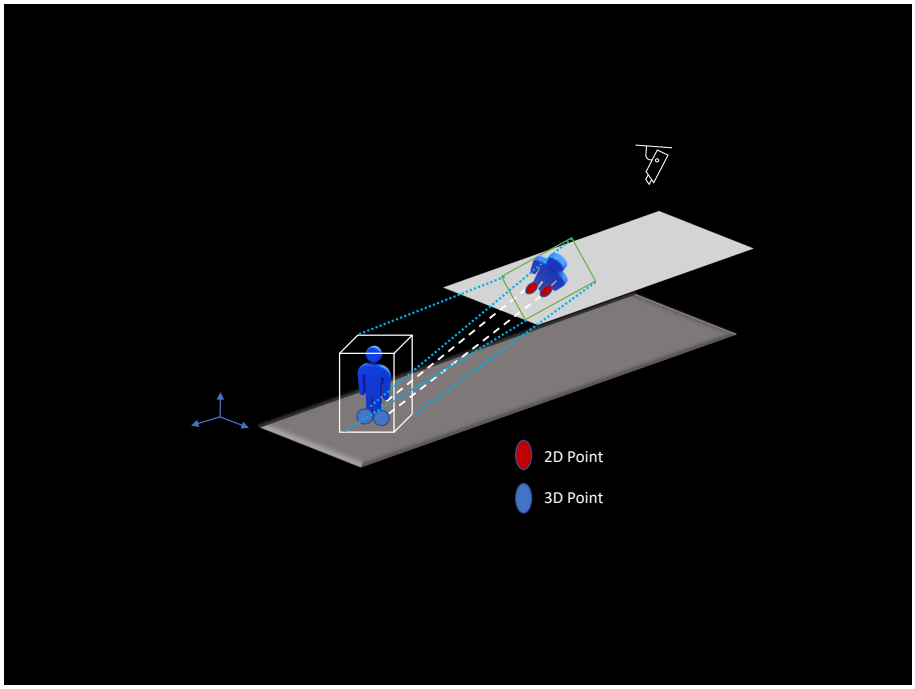


Figure 3.7: Pictorial representation of bounding box.

3.2.3 Module 3: Training of HDPE Model

In this module, we look into the training details involved in the transfer learning of the HDPE models. Information about the dataset used is also explained before

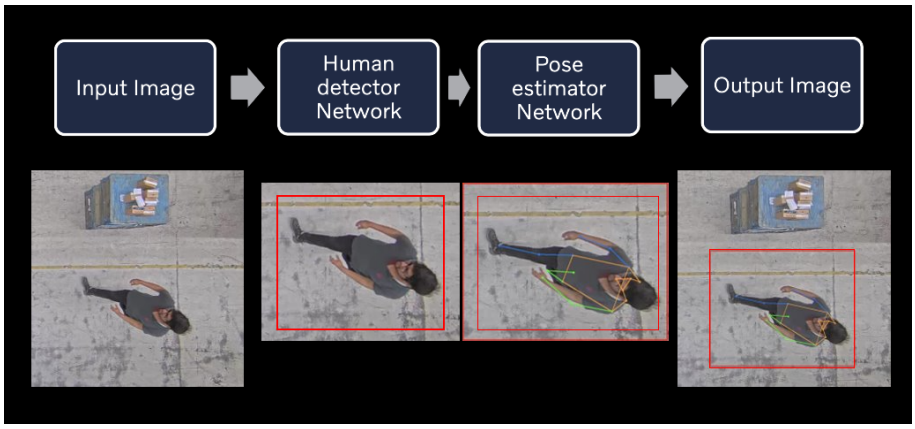


Figure 3.8: The process flow in a top-down human detection and pose estimation(HDPE) network.

we look into transfer learning (or fine-tuning) details for the human detector model and pose estimation model. In the fine-tuning process, a model can be trained several times with the use of checkpoints, and not trained from scratch to learn new things repeatedly. In our work, we used MMDetection [29] and MMPose [49] toolboxes both from OpenMMLab project for the human detector and pose estimator models, respectively. OpenMMLab has a wide collection of object detector and pose estimation models that are pre-trained on a variety of widely used datasets. For our task, we have picked the Faster R-CNN model for the human detector network and the HRNet-W48 model for pose estimation. Both models were pre-trained with the COCO dataset [35]. For visualizing and keeping track of the training and evaluation progress, a sub-tool of Tensorflow was used called Tensorboard [50].

3.2.3.1 Dataset for training and testing

In this section, we look into the train-val-test data split in detail.

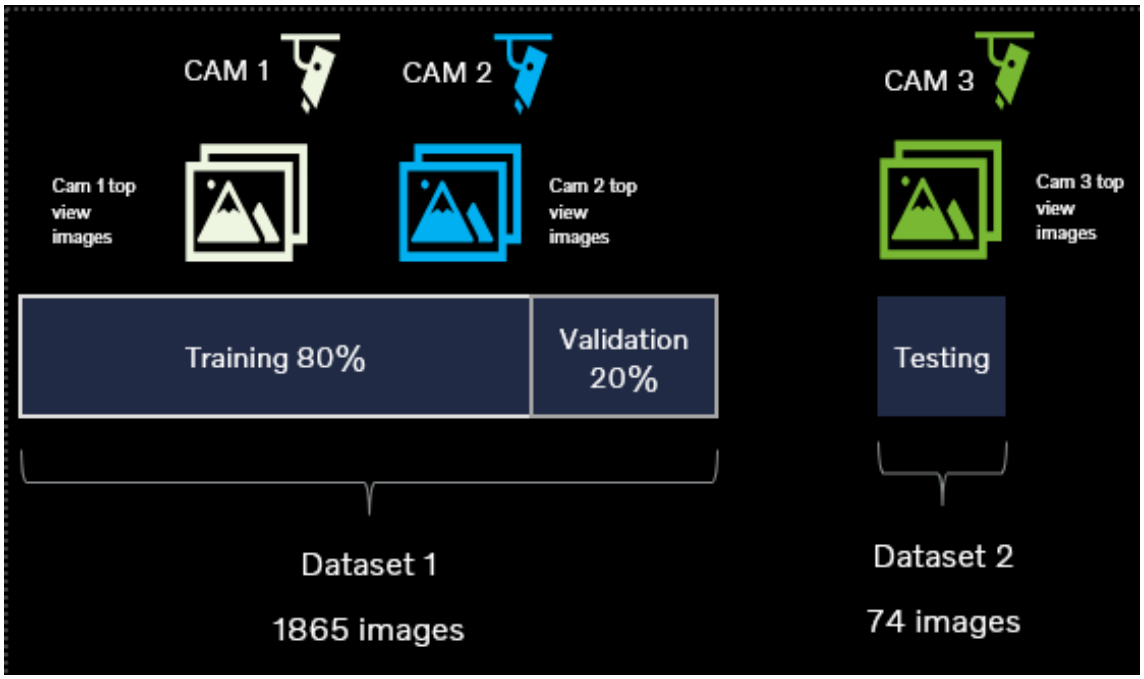


Figure 3.9: Dataset for training and testing the fine-tuning process.
Dataset 1: In order to induce data variability, images from two cameras (CAM 1 and CAM 2) were used for the training (80%) and validation dataset (20%). All the required labelling was created using our semi-automatic annotation process.
Dataset 2: The test dataset consisted of a total of 74 images from a different camera (CAM 3) feed to test generalization.

The number of annotated images generated from our semi-automatic annotation pipeline was 1895 images. These images consisted of one object class for all the images which were termed as "person", bounding box coordinates, and human key points annotations as per COCO format [35]. In order to generalize the model, this data consisted of images compared from two ceiling cameras. Out of the 1865 images, the training-validation split was decided as 80/20% in order for the validation set to have a lesser variance in the metrics.
As for the test dataset, a total of 74 images, amounted to roughly 3% of train-val dataset was chosen. These test images were generated from a third camera, different from the camera used for the training/validation sample dataset. Also, the labeling was done manually using an open-source labeling tool CVAT. [51]

3.2.3.2 Fine-Tuning Human Detector

There are two steps for fine-tuning the Faster R-CNN model. They are:

1. Creating Train-val-test annotation file: Fine-tuning requires training the model with a new set of training data. Therefore, the initial step is to create an annotation JSON file that is in the COCO format. This annotated top view dataset was created by following the steps in Module 3.2.2

2. Modifying config file: To perform transfer learning, the base config file from the MMdetection toolbox was inherited and only the following changes were made:

    - "num-of-classes" parameter: As we want the detector model to detect just human or person class, the number of output classes were changed to 1 to include "person" class. By changing this in the classification head, the weights of the pre-trained models are mostly reused except for the final prediction head.
    - Training hyperparameters: the learning rate and the number of epochs were modified to 0.0025 and 40 respectively to facilitate fine tuning. The optimizer used was the Stochastic gradient descent (SGD).

3.2.3.3 Fine-Tuning Pose Estimation Model

Similar to the human detection model, fine-tuning the pose estimation model also requires modification of the annotation file and the config file.

1. Creating Train-val-test annotation file: As the training will be performed on data annotated using the automatic annotation pipeline, the 17 key points generated using this pipeline should be converted to COCO format JSON file. As the human detector and pose estimation model are given the same JSON file, the key points annotation should be clearly distinguished from the detector points.

2. Modifying the config file: Here, since the pre-trained pose estimation model was also trained on COCO dataset, we didn't require to change the number of joints in "inference channel" as there are 17 key points in our training data also. With regard to the learning rate and a number of epochs, they were changed to 0.0005 and 100. The optimizer used was ADAM.

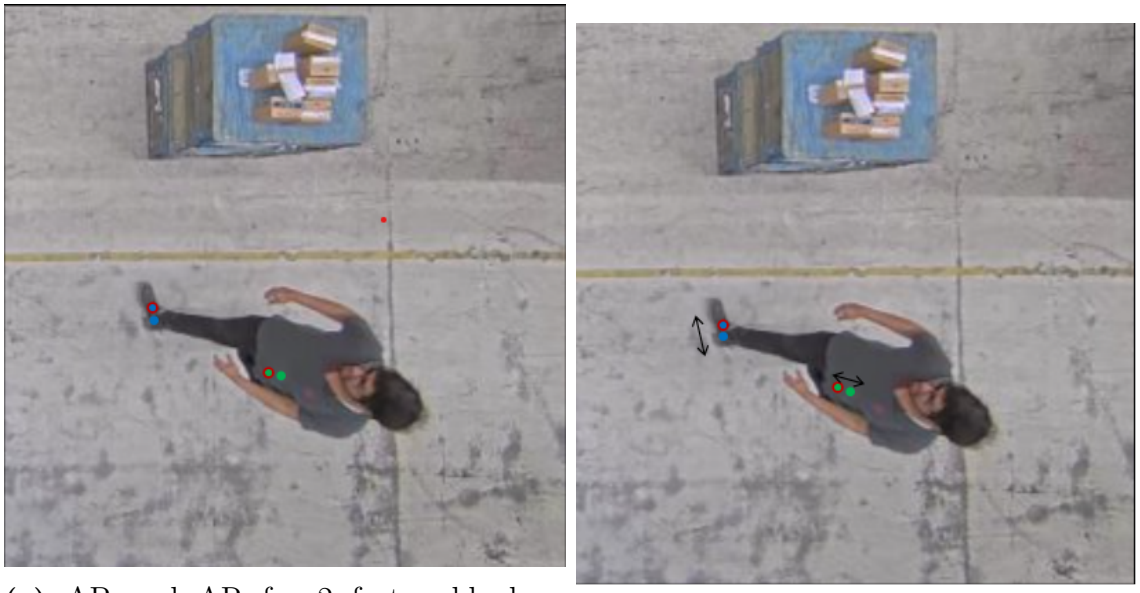
3.2.4 Module 4: HDPE Model Evaluation

In the final module, after training the human pose model, the results are evaluated by various metrics. The metrics consists of both the standard COCO evaluation metrics as discussed in Chapter 2 and custom test metrics which will be discussed further below.
The COCO metrics for key-point evaluation computes the average precision and average recall based on all the 17 key points of a person with various thresholds. This can be used to compare our model with the pre-trained model. In our project, however, upon these basic metrics, there are two additional important metrics for better understanding the performance during the implementation of the model.

- Average Precision and Average Recall for 2 key points: This metric is similar to COCO metrics but instead of 17 key points, only the left and right feet ankle key points are considered. The detection of humans and the key points are preliminarily used by the ATRs to maneuver the factory floor safely. The most important key points that help ATRs in this task will be the left and

right ankle points, as these are the points that are planted on the ground most of the time and even if predictions of other key points are not perfect, based on these 2 key-points, the ATRs can make safe path calculation. These two key points are also important to track as they are prone to occlusion when viewed from a ceiling-mounted camera.

- Mean L2 Pixel Error of Ankle Points: This metric is used to find the distance between predicted and target points in pixels. In the real world when evaluating the performance of the model to analyze safe contact between the human operator and ATRs, this metric will be informative as the distance between target ground truth and prediction can be found in pixels which can be converted to a measurement system.



(a) AP and AR for 2 feet ankle key-points.

(b) L2 pixel error of ankle key points.

Figure 3.10: Custom test metrics for feet ankle key points.

4
Results

This chapter will provide the results of the aforementioned methods to perform human detection and pose estimation. The test set was not used during the training and validation process and hence the findings reported in this section are on the test set. In this chapter, we divide it into human detection tasks and human pose estimation tasks, and finally, we also look into the multi-human pose estimation results.

4.1 Evaluation of Human Detector Model results

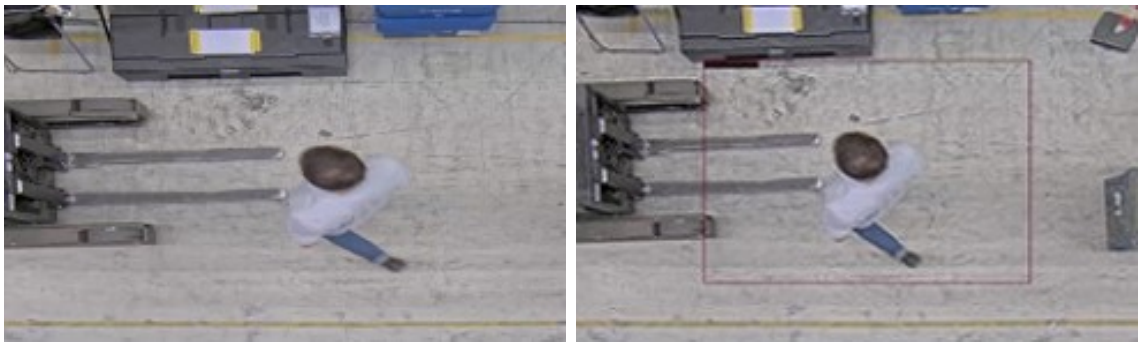
The primary objective of our human detector model, as the name suggested is to localize the human's position in 2D coordinates in the image taken from the top view with the help of bounding box coordinates. These detections are crucial because the human pose estimator model, discussed later, needs these bounding boxes to predict the poses of the human.
The pre-trained deep learning model used for the human detection task in our thesis is Faster-RNN model which was trained in the original COCO dataset but tuned further to detect the "human" class. The weights and the checkpoints of this model already exist as part of the OpenMMLab framework [30] which could be downloaded. Though this model is one of the SOTA models that exist for object detection, we observe that it performs poorly for top-view images, which is an important feature of our application. Hence, we fine-tuned the above model using our own annotated training set explained in Chapters 3.2.2,3.2.3 This model is referred as fine-tuned model.
The side-by-side visual comparisons can be seen from the following figures 4.1,4.2, and 4.3.
The comparison of performance of pre-trained Faster RCNN model and fine-tuned model can be seen in the table below. From the table, it is clear that the fine-tuned model performs better than the pre-trained model in all metrics for top-view images. Particularly, the improvement in AR which in turn demonstrates the number of missed detections is important for this application.

Metric	Pre-trained model	Fine-tuned model
(AP) @[IoU=0.50:0.95]	0.009	**0.298**
(AP) @[IoU=0.50]	0.036	**0.851**
(AR) @[IoU=0.50:0.95]	0.048	**0.399**
(AR) @[IoU=0.50]	0.137	**0.918**

Table 4.1: Performance Comparison for human detector using COCO metric. AP = Average Precision, AR= Average Recall.

To further understand the above table and the performance of the fine-tuned and pre-trained models, the models are compared in the following scenarios.

4.1.1 Analysis of false negatives



(a) Pre-trained(off-the-shelf) model **(b)** Fine-tuned model

Figure 4.1: Comparison of false negative detections between (a) Pre-trained and our (b) Fine-tuned detector models.

A high average recall indicates that the detector is able to identify most instances, while a low average recall indicates that many instances are being missed. Missed detections, also known as false negatives, are instances that should have been identified as a certain class but were not. Here, it is important that the detector exhibits high recall than high precision, as it's far more serious if it misses detecting a human on the scene rather than being correct most of the time.
Figure 4.1 shows a sample result where the pre-trained model has missed detecting the human in the image, whereas the fine-tuned model accurately detects it.

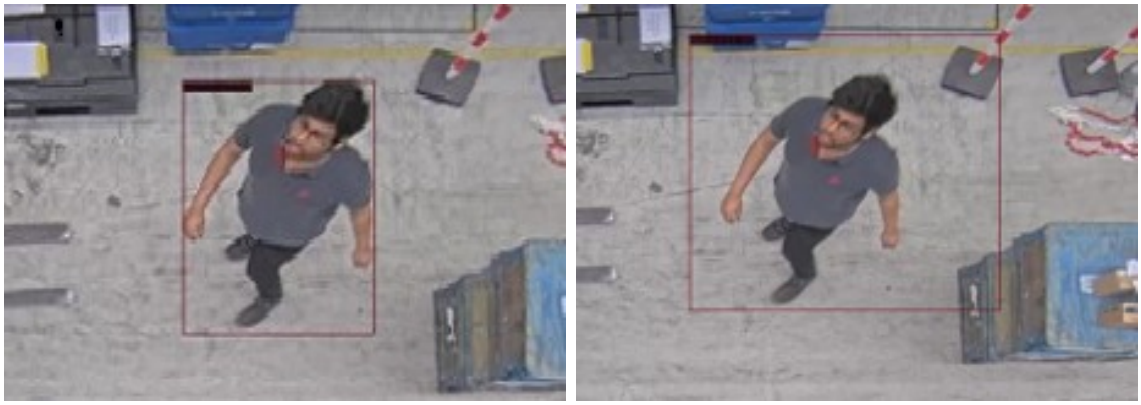
4.1.2 Analysis of occlusion handling

(a) Pre-trained model **(b)** Fine-tuned model

Figure 4.2: Human detector's occluded detection comparison between (a) Pre-trained and our (b) Fine-tuned models.

To further evaluate false negative detection performance, in Figure 4.2 we notice an improvement in the occlusion handling capability in our fine-tuned model compared to the pre-trained. Occlusion handling is important for a human detector model because it allows the model to accurately detect and localize humans even when they are partially occluded by objects or other people in the scene. This evaluation was also done to understand the performance of our human detector model in cases where the humans are partially visible in the edges of the area covered by the camera.

4.1.3 Analysis of bounding box



(a) Pre-trained model **(b)** Fine-tuned model

Figure 4.3: Human detector's bounding box regression comparison between (a) Pre-trained (b) Fine-tuned models.

But a downside of this approach can be seen in Figure 4.3. In this figure, the pre-trained model has a tighter bounding box than the fine-tuned model. The difference in dimensions can be explained by understanding the datasets the models are trained with. In the case of the pre-trained model, the model is trained with COCO dataset,

which is a huge dataset consisting of tightly annotated bounding boxes. In the case of fine-tuned model, however, the pre-trained model is re-trained with a relatively small dataset generated using the semi-automatic data annotation pipeline. In this pipeline to cover a wide range of human angles, a wide fixed bounding box is used. As we re-train with this dataset, the quality of bounding boxes decreases, leading to less tightness.

4.2 Evaluation of Human Pose Estimation Model results

In this subsection, the results comprising the performance of the human pose estimation model are presented. The used human pose estimation model follows the top-down approach to predict the key points of the human given the bounding box detections from the earlier human detector model.
We observed the standard COCO metrics alone were not enough to capture the important aspects of our application on the factory floor. Hence, we also present the results evaluated by our custom metrics in this section.
The pre-trained model used for human pose estimation model used in our work is HRNet W48 model pre-trained on COCO Dataset. The checkpoint model consisting of pre-trained weights is available for download, as part of (MMPose) OpenMMLab framework [49]. Just like in the human detector case, the pre-trained human pose estimation model is also re-trained on our semi-automatically annotated top-view dataset. This re-trained model is referred to as the "Fine-tuned" model in the previous chapter. Now, having established the basic understanding of both the pose estimation models presented in our results section, the discussion continues to the findings from the experiments listed below.

4.2.1 Evaluation of 17 key points using Standard COCO metrics

Here, in this experiment, standard COCO metrics AP and AR for all 17 key points for both the pre-trained and our fine-tuned model are presented in Table 4.2.

Metric	Pre-trained model	Fine-tuned model
(AP) @[IoU=0.50:0.95]	**0.628**	0.449
(AP) @[IoU=0.50]	**0.933**	0.828
(AR) @[IoU=0.50:0.95]	**0.697**	0.504
(AR) @[IoU=0.50]	**0.959**	0.849

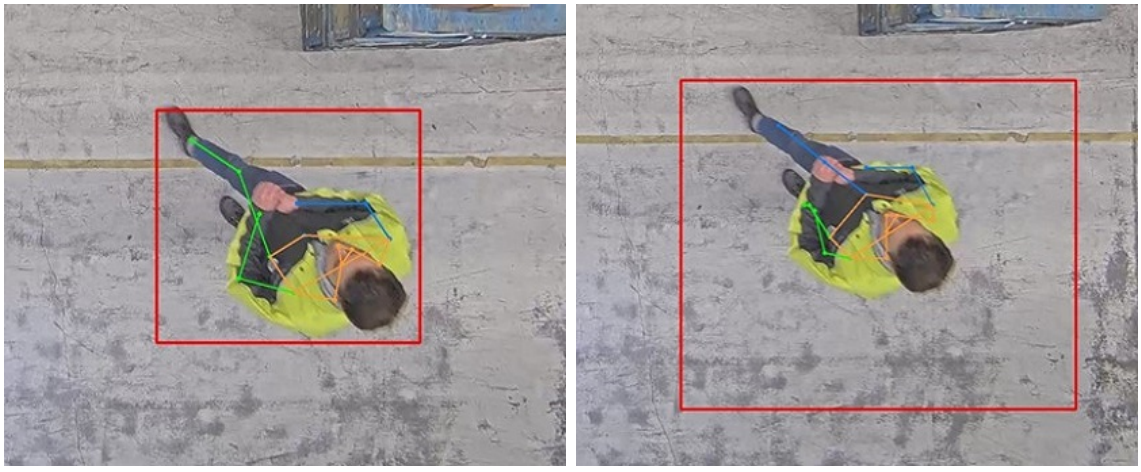
Table 4.2: Performance comparison for human pose estimation model - Standard COCO metrics. AP = Average Precision, AR= Average Recall.

(a) Pre-trained model. **(b)** Fine-tuned model.

Figure 4.4: Human pose estimation model with 17 key points (a) Pre-trained and (b) Fine-tuned models.

Table 4.2 shows the comparison between the pre-trained model and our fine-tuned model when evaluated using Standard COCO metrics AP and AR. The values obtained indicate that our fine-tuned model does not manage to outperform the pre-trained model.
Though this is true when looking at just the metric values, the visual result sample images tell a different point. This is more clear when one refers to figure 4.5b, where the fine-tuned model seems to be able to predict the foot or the ankle positions more accurately than the pre-trained model.

4.2.2 Visual inspection of ankle key points



(a) Pre-trained model **(b)** Fine-tuned model

Figure 4.5: Visual comparison of ankle prediction between (a) Pre-trained and our (b) Fine-tuned model.

In this test, we look at how accurately our fine-tuned model predicts the two ankle points. These ankle predictions are important especially for our application, as we intend to accurately localize humans in 3D using just 2D detections of ankle or feet positions with the help of the assumption explained earlier in Section 1.3 that human feet are always placed firmly on the floor and the human is standing. In Figure 4.5a and Figure 4.5b, the blue and green lines represent the right and left side of the human body, respectively. In the pre-trained model, the right leg is predicted as left leg as it is in green whereas, in the fine-tuned model, the right leg is represented correctly in blue colour.

4.2.3 Evaluation of ankle key points using custom metrics.

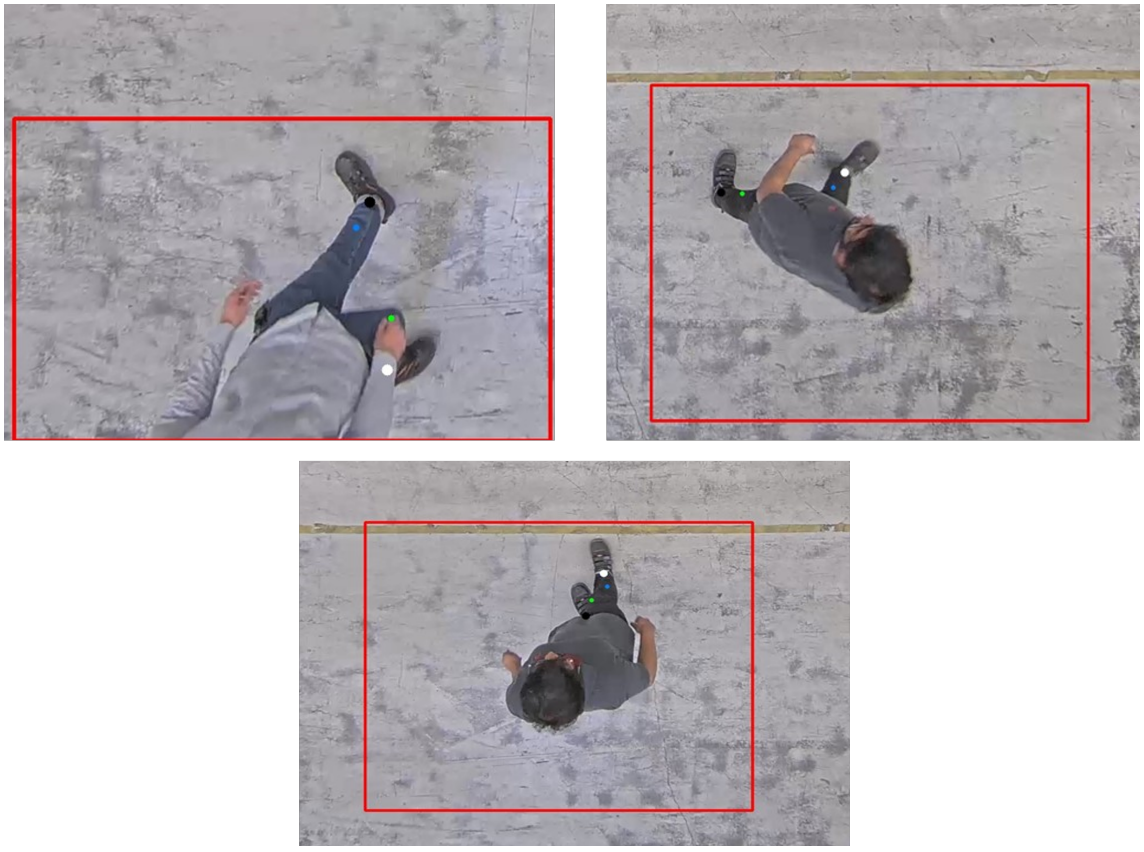


Figure 4.6: Comparison of ankle prediction for custom metric evaluation: Blue and Green - Pre-trained model, Black and White - Fine-tuned model.

To further experimentally measure and verify our above results on ankle key points prediction, we introduced our custom metrics which focus on the two ankle key points instead of all 17 key points used as part of standard COCO metrics.

Metric	Pre-trained model	Fine-tuned model
(AP) @[OKS=0.50:0.95]	0.451	**0.548**
(AP) @[OKS=0.50]	0.790	**0.856**
(AR) @[OKS=0.50:0.95]	0.608	**0.641**
(AR) @[OKS=0.50]	0.876	0.877

Table 4.3: Performance comparison for human detector using custom Metric - mAP of 2 ankle points.

L2 pixel distance	Pre-trained model	Fine-tuned model
Left ankle	96.64523	**78.95414**
Right ankle	100.49078	**85.60831**

Table 4.4: Performance comparison for human detector using custom Metric - Mean L2 Pixel error of each ankle point.

The evaluated results using two custom metrics can be summarized from the tables 4.3 and 4.4 respectively. A short description of each of these two metrics is needed before we look into the results demonstrated. The first custom metric used in this work is the Average Precision (AP) and Average Recall (AR) computed just for the two ankle key points instead of all the predicted key points as in COCO metrics. The second one is the L2 pixel distance metric, which is basically the L2 norm difference between two-pixel points. This difference is helpful in quantifying the difference between the pre-trained and fine-tuned compared to the ground truth key point location in pixels.
The results from table 4.3 show quite clearly that the fine-tuned model is better at predicting the ankle points than the pre-trained model. The L2 pixel difference table 4.4 results look promising as it shows that the fine-tuned model is able to reduce the L2 norm pixel difference in both ankle positions. A lower value here is better indicating the lower difference between prediction and ground truth positions. A visual comparison of the same can be also seen in Figure 4.6.

4.2.4 Results on multi-human pose data

Here, we apply our human detection and pose estimation model on multi-human test images without any retraining on multi-human datasets to check the adaptability of the model. We observed that the model performed quite well for multi-person images as shown in Figure 4.7.

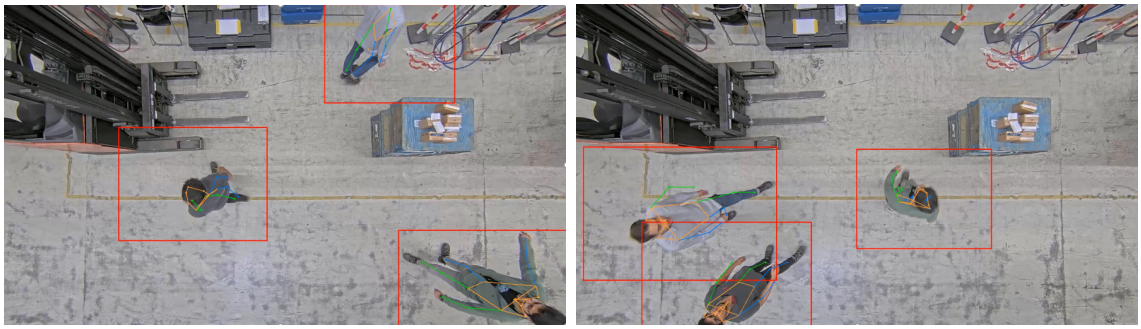


Figure 4.7: Results on Multi-person human test dataset without re-training.

5
Conclusion

This section provides an explanation and possible interpretations for the findings in the previous chapter's results involving human detectors and human pose estimation models. This discussion is then followed by an overall conclusion of our findings in relation to the research questions to be addressed in this thesis work.

5.1 Discussions

As seen in the previous chapter, the fine-tuned model performs better than the pre-trained model for both human detector and pose estimation models. This improvement can be due to re-training the pre-trained model on our annotated top-view dataset which was created using the semi-automatic annotation method 3.2.2.

We observed from Table 4.1 that the fine-tuned human detector has lesser false negatives detections and better occlusion handling compared to the pre-trained model. We speculate that these improvements are primarily due to re-training the pre-trained model on our annotated top-view dataset. For our particular use cases involving robots and humans, we believe that this trade-off in False Positives is definitely worth considering, ensuring safer scenarios.

In the case of the human detector, however, the pre-trained model has a tighter bounding box than our model even though the recall is lower than the fine-tuned model. The fit of the bounding box is an important aspect of a human detector, as it directly affects the accuracy and precision of human detection. A bounding box that is too small may exclude parts of the human object, making it difficult to accurately detect and classify the object. On the other hand, a bounding box that is too large may include parts of the background or other objects, which can lead to false positives and reduce the overall accuracy of the detector. Since the poorly fitted bounding boxes predictions from our fine-tuned model are directly a result of our semi-automatic annotation method, we think that more advanced techniques to create more accurate bounding boxes must be introduced.

When it comes to the multi-human pose estimation results depicted in Figure 4.7, We believe that the combination of top-down model architecture selected made it automatically suitable for multi-person pose tasks without the need for separate retraining. This can be due to the Faster R-CNN model, which detects the individuals and passes each individual detection through the single-person pose estimation

model. Though this may be true, we presume that they might not produce the same level of accuracy as a model that has been specifically trained on multi-person pose estimation. Therefore, if we require high accuracy for multi-person pose estimation, it is recommended to use a model that has been trained on a multi-person dataset.

5.2 Limitations and Future work

Due to the limited time for completion of this thesis, the scope had to be reduced, which had led to few limitations. These limitations also illustrate improvement opportunities that can be done as future work.

1. **Semi-automatic annotation method:**
   We use the MoveNet model for our annotation pipeline, hence the quality of the annotated dataset is limited by this model's accuracy and performance. Further ways to improve the quality of annotation could be part of future work. Similarly, the triangulation method used in this thesis is a basic approach and there are advanced methods in the literature that could give better-triangulated results [52], [53]. Another limitation is that the automatic annotation pipeline was currently developed for single-person data only and the extension to multi-human data would require more time. Hence, this topic is left for future work. Another future scope could be to study if adding more than two side-view cameras can improve triangulation without adding to the complexity.

2. **2D vs 3D pose estimation:**
   The estimated pose results part of this work is in 2D space as the focus was on the foot key-point estimation. However, there could be many Direct 3D pose estimation models that could work well [54].

3. **Hyper-parameter Tuning:**
   Due to lack of time, this thesis did not focus on Hyper-parameter tuning of deep learning models which could help to find the optimal parameters to extract the best performance from the model. We believe it could bring immense benefit when deploying as production systems.

4. **Unsupervised learning framework:**
   Weakly supervised and unsupervised methods have grown in popularity due to their efficiency in dealing with limited ground truth data. This method can be especially important here as there are no large publicly available datasets for top-view images.

5.3 Conclusion

This thesis focuses on developing a deep-learning model to detect and estimate human pose from ceiling-mounted cameras. It began by discussing the challenges in the availability of training data for top-view images and the poor performance resulting in using pre-existing and pre-trained models for this task. To overcome these issues, we used a semi-automatic annotation method using multi-views (a combination of side-view cameras and the intended top-view camera) to generate a labelled or annotated top-view dataset of humans and their poses. This dataset was then used to fine-tune and train a deep-learning model. Therefore, the core of this thesis is the development of a semi-automation annotated method which solved the problem of extensive manual annotation efforts and demonstrated that automatic annotation methods can substitute the lack of manual annotation to achieve satisfactory results. We also improved performance by fine-tuning the pre-trained human detector and pose estimation models for Volvo's use cases involving top-view images. This proved that re-training on top-view images was necessary to achieve accurate level predictions. Our fine-tuned human detector was able to deal better with occluded body parts and also reduced false positives. Two custom metrics for pose estimation were formulated which would better help to evaluate the requirements for Volvo Group's use case.

Bibliography

[1] A. Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks: a tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[2] V. Meel, "What is the coco dataset? what you need to know in 2023," Jan 2023. [Online]. Available: https://viso.ai/computer-vision/coco-dataset/

[3] S.-H. Zhang, R. Li, X. Dong, P. L. Rosin, Z. Cai, H. Xi, D. Yang, H.-Z. Huang, and S.-M. Hu, "Pose2seg: Detection free human instance segmentation," 2018. [Online]. Available: https://arxiv.org/abs/1803.10683

[4] C. Park, H. Lee, W. Kim, H. Bae, J. Lee, and S. Lee, "An efficient approach using knowledge distillation methods to stabilize performance in a lightweight top-down posture estimation network," *Sensors*, vol. 21, p. 7640, 11 2021.

[5] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[6] Z. Zhang, *Camera Parameters (Intrinsic, Extrinsic).* Boston, MA: Springer US, 2014, pp. 81–85. [Online]. Available: https://doi.org/10.1007/978-0-387-31439-6_152

[7] J. Weng, P. Cohen, and M. Herniou, "Camera calibration with distortion models and accuracy evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 10, pp. 965–980, 1992.

[8] "Welcome to mmpose's documentation!" [Online]. Available: https://mmpose.readthedocs.io/en/latest/

[9] "Electromobility is about to take off," Feb 2021. [Online]. Available: https://www.volvogroup.com/en/news-and-media/portraits/portraits/electromobility-is-about-to-take-off.html

[10] C. Bai, P. Dallasega, G. Orzes, and J. Sarkis, "Industry 4.0 technologies assessment: A sustainability perspective," *International Journal of Production Economics*, vol. 229, p. 107776, 2020.

[11] B. Tjahjono, C. Esplugues, E. Ares, and G. Pelaez, "What does industry 4.0 mean to supply chain?" *Procedia manufacturing*, vol. 13, pp. 1175–1182, 2017.

[12] K. Schwab, *The fourth industrial revolution.* Currency, 2017.

[13] "The factory of tomorrow is already here," Sep 2019. [Online]. Available: https://www.volvogroup.com/en/news-and-media/news/2019/sep/the-factory-of-tomorrow-is-already-here.html

[14] "Industry 4.0 people and robots work together on equal terms." [Online]. Available: https://www.chalmers.se/en/centres/chair/news/Pages/Industry-4-0-people-and-robots-work-together-on-equal-terms.aspx

[15] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1653–1660.

[16] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 648–656.

[17] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 466–481.

[18] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity - bulletin of mathematical biology." [Online]. Available: https://link.springer.com/article/10.1007/BF02478259

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[20] M. Minsky and S. Papert, *Perceptrons*, 1969.

[21] T. Szandała, "Review and comparison of commonly used activation functions for deep neural networks," p. 3, 10 2020.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[23] S. Ruder, "An overview of gradient descent optimization algorithms," 2016. [Online]. Available: https://arxiv.org/abs/1609.04747

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[26] A. F. Agarap, "Deep learning using rectified linear units (relu)," 2018. [Online]. Available: https://arxiv.org/abs/1803.08375

[27] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling, "Semi-supervised learning with deep generative models," *Advances in neural information processing systems*, vol. 27, 2014.

[28] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[29] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu *et al.*, "Mmdetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.

[30] "Benchmark and model zoo - mmdetection." [Online]. Available: https://mmdetection.readthedocs.io/en/stable/model_zoo.html

[31] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, 2023.

[32] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[34] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[35] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2014. [Online]. Available: https://arxiv.org/abs/1405.0312

[36] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov *et al.*, "The open images dataset v4," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.

[37] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[40] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[41] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, jul 2014.

[42] S. Jin, X. Ma, Z. Han, Y. Wu, W. Yang, W. Liu, C. Qian, and W. Ouyang, "Towards multi-person pose tracking : Bottom-up and top-down methods," 2017.

[43] G. Ning, P. Liu, X. Fan, and C. Zhang, "A top-down approach to articulated human pose estimation and tracking," 2019. [Online]. Available: https://arxiv.org/abs/1901.07680

[44] M. Kresović and T. D. Nguyen, "Bottom-up approaches for multi-person pose estimation and it's applications: A brief review," 2021. [Online]. Available: https://arxiv.org/abs/2112.11834

[45] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[46] M. E. Wall, A. Rechtsteiner, and L. M. Rocha, *Singular Value Decomposition and Principal Component Analysis*. Boston, MA: Springer US, 2003, pp. 91–109. [Online]. Available: https://doi.org/10.1007/0-306-47815-3_5

[47] Aruco. [Online]. Available: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html
[48] "Next-generation pose detection with movenet and tensorflow.js." [Online]. Available: https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html
[49] "Openmmlab pose estimation toolbox and benchmark," 2020.
[50] "Tensorboard nbsp;: nbsp; tensorflow." [Online]. Available: https://www.tensorflow.org/tensorboard
[51] "Cvat." [Online]. Available: https://www.cvat.ai/
[52] K. Iskakov, E. Burkov, V. Lempitsky, and Y. Malkov, "Learnable triangulation of human pose," 2019. [Online]. Available: https://arxiv.org/abs/1905.05754
[53] H. Rhodin, J. Spörri, I. Katircioglu, V. Constantin, F. Meyer, E. Müller, M. Salzmann, and P. Fua, "Learning monocular 3d human pose estimation from multi-view images," 2018. [Online]. Available: https://arxiv.org/abs/1803.04775
[54] T. Wang, J. Zhang, Y. Cai, S. Yan, and J. Feng, "Direct multi-view multi-person 3d pose estimation," 2021. [Online]. Available: https://arxiv.org/abs/2111.04076