MEDIA INFORMATICS

RWTH AACHEN UNIVERSITY

Master Thesis

Bottom-Up Multiple Person Multi-Camera 3D Human Pose Estimation

Wei Wang

MEDIA INFORMATICS

RWTH AACHEN UNIVERSITY

Master Thesis

Bottom-Up Multiple Person Multi-Camera 3D Human Pose Estimation

Author:	Wei Wang
Supervisor:	Supervisor
Advisor:	Advisor
Submission Date:	Submission date

I confirm that this master thesis is my own work and I have documented all sources and material used.

Munich, Submission date Wei Wang

Acknowledgments

Abstract

Contents

1 Introduction

Computer vision has been an important role in the automation of processes in many different areas. A well-studied subject within the field of computer vision is humans. In detecting and tracking humans, humans are usually localized, by means of bounding boxes, within a single or multiple views. In pose estimation, a human body is usually represented, by a skeleton composed of a set of joints. The objective in pose estimation is to recover the joint locations of human bodies in 2D images or 3D space. While 2D single person detection and tracking of humans has been well addressed in real-world environments, 3D multi-person pose estimation remains an open problem, due to the complex scenarios that it has to consider. In this thesis, we study the problem from multiple views. We investigate the approach that aggregates multiple view information using epiporlar geometry. Our ultimate goal is to estimate 3D human pose from multiple 2D views.

1.1 Motivation

Estimation 3D multi-person human poses have been addressed, by using a single view [1, 2, 3] or multi-view camera [4, 5, 6, 7, 8, 9]. Regardless of single view or multi-view approaches, the first step is usually generating 2D likelihood maps of joint locations by a backbone network trained on a 2D pose estimation task. However, the accuracy of the 2D pose estimation backbone degrades significantly when joints being occluded [10], resulting single-view methods to produce unrealistic 3D poses. Since occlusions are a common issue in multi-person and in-the-wild environments, leveraging multi-view information becomes a natural choice in 3D multi-person pose estimation. Some previous studies adopt 3D pictorial structure (3DPS) [7, 4, 5], where the 3D poses are recovered from 2D joints in a discretized 3-space, with predefined geometric constraint of human bodies. However, a severe problem of 3DPS is the expensive computational cost due to the huge state space with multiple people in multiple views. Other approaches rely on deep networks that operate on volumetric grids [6, 3] to combine features coming from different views in accordance with epipolar geometry. Unfortunately, it requires a discretized space that covers the whole environment, which is memory demanding and, more importantly, impractical in outdoor environments.

1.2 Problem Definition and Challenges

In this work, we address the 3D multi-person pose estimation task from multiple views using epipolar geometry. Inspired by the lightweight approaches that utilize multiview geometry [4, 11, 12], we use a lightweight 2D pose estimator trained on the COCO dataset as backbone and aggregate multi-view information using epipolar geometry. Our goal is to automatically estimate the pose of multiple individuals in 3D space, given a set of images from a calibrated multi-view camera system. However, the transition from 2D to 3D space and from single to multiple human pose estimation is a challenging task.

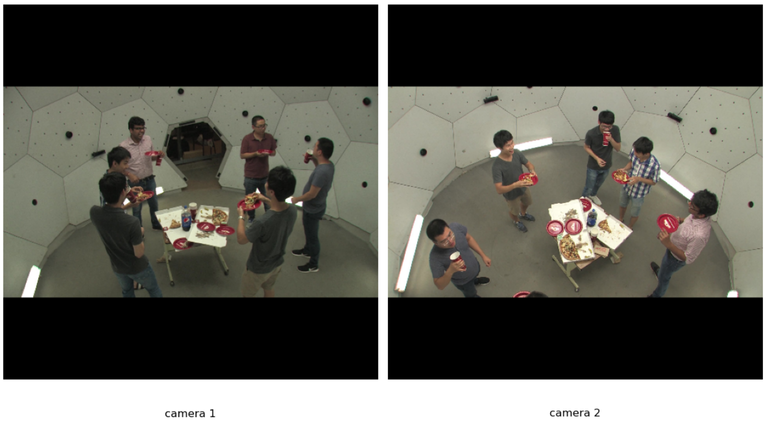

Figure 1.1: Estimating multi-person 3D human poses without using computational heavy operations is the task we address in this work.

First of all, learning 2D human bodies from image data is a hard task due to its articulation and large deformation it can go through. Thus, 2D human pose estimation researchers usually use a rigid skeleton compose of a various number of joints, such as elbow, pelvis, etc., shown in 1.2. Since the 2D models are built from images, we have to compensate the missing dimension when estimating 3D poses.

Secondly, the appearance of a joint varies significantly from one view to another. Most 2D human pose datasets that based on natural images provide only frontal and horizontal views. Yet it is common in multi-person environments that most of the joints are occluded in frontal view and can only be revealed at a novel view, see Fig.1.1. Occlusions cause a serious problem in 2D multi-person human pose estimation in single view images [10]; occlusions can not only caused by objects but by oneself and

Figure 1.2: A human pose is approximated by a rigid skeletons, composed by various joints. Performing human pose estimation, however, from images can be a difficult task due to the body pose and appearance high variation.

other humans as well, shown in Fig. 1.1.

Moreover, recent works [5, 4, 8, 13] cast multi-person 3D human poses estimation as associating 2D poses in all camera views. This is complicated problem in multiple human 3D pose estimation when the identity of individuals is unknown. An association between the individuals across all views is required to avoid mixing the body parts of different individuals. For instance, a left hand of one person in one view will have multiple left hand candidates in other camera views coming not only from the same person, but also from other individuals and potential false positive detections. In practice, this will create incorrect body part hypotheses that can lead to fake body poses in the 3D space. In addition, matching 2D poses between all pairs of views still makes the computational complexity explode as the number of cameras increases [9] .

To address these challenges, we propose to fuse features from multiple views using epipolar geometry and a deep convolution network. Our intuition is that a joint that being occluded in one view could be visible in other views and a deep convolution network can learn how to see through occlusions from the fused features.

1.3 Applications

A framework for multi-person human pose estimation from multiple views has a wide range applications, such as motion capture, surveillance, sports, and human activity recognition, etc.. We provide some examples that our framework could be applied on. Motion capture systems haven been beneficial in film industry, especially for CG characters. The current technology for estimating reliable 3D human poses is based on maker-based solutions, which work only in a studio environment. On the contrary, our approach is marker-less and can be adapted to any unconstrained environments. Another useful scenario for our framework is sport science or analysis, where athletes can not wear MOCAP suits. For example, estimating the poses of basketball players, captured from different view, supports game analysis. In addition, body pose estimation can be used to study the tactic of opponents or to improve the training of teams. Furthermore, our framework can be used in the public surveillance systems since there are usually multiple cameras installed in public areas. As this work was done during the COVID-19 pandemic period, a crowd control system, combining our work and activity recognition, for detecting citizen gathering in a large group would be beneficial to the health of public. Lastly, our frame work can be applied on multi-pedestrian path prediction for autonomous driving. Lastly, our frame can add to human action recognition, where human poses are one of the strong cues. In general, our frame benefits various real-world applications because it is less computation demanding and suits for unconstraint environments.

1.4 Thesis Outline

We provide an overview for each chapter of the thesis

- **Chapter 2** We present the theoretical background of our work.
- **Chapter 3** In this chapter, we provide an overview of the current researches in the field of 3D human pose estimation and compare the difference between our work and related researches.
- **Chapter 4** We explain the design of our feature fusion modules and the network architectures.
- **Chapter 5** In this chapter, we give an overlook of two different datasets we used in this work and explain metrics being used by the datasets to evaluate the performance of our model. Futher, we introduce the data augmentation methods.
- **Chapter 6** We evaluate the performance of our work and give an ablation study
- **Chapter 7** We conclude our work by presenting our findings, the limitations of the proposed methods and our directions for future work.

2 Theoretical Backgrounds

2.1 Camera Model

2.1.1 Camera Intrinsic Matrix

A camera can be consider as a linear mapping from 3D spaces to a 2D images when we ignore the lens distortion. Let us denote a point in 3D space as $\mathbf{X_{cam}} \in \mathbb{R}^{4\times1}$; the subscript *cam* mean the point is located in camera coordinate frame. The 3D point $\mathbf{X_{cam}}$ is imaged in a camera view, at $\mathbf{x} = \mathbf{PX_{cam}}$, where $\mathbf{x} \in \mathbb{R}^{3\times1}$ represent 2D points in images and $\mathbf{P} \in \mathbb{R}^{3\times4}$ is a *camera projection matrix*. Fig. 2.1 [14] illustrate the geometry relationship from 3D spaces to image spaces. Note that, for the sake of simplicity, we use homogeneous coordinate and column vector to represent a point.

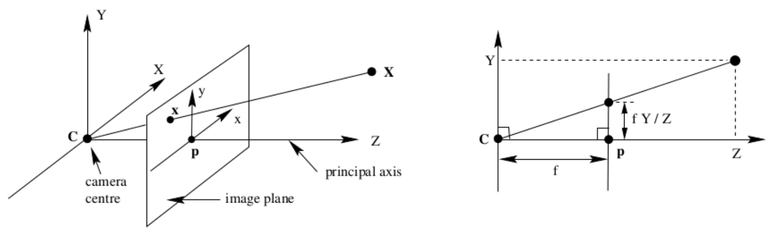

Figure 2.1: **C** is the camera centre and **p** is the center of the image plane or called *principal point*. The camera center is here placed at the origin of world coordinate frame. Note the image plane is placed in front of the camera center.

we can express a **basic pinhole camera model**, $\mathbf{x} = \mathbf{PX_{cam}}$, in the following equation

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.1}$$

Expression 2.1 assumes that the principle point is located at the center of image plane. Conventionally, we use the top-left or bottom-left corner of the image plane as origin.

If the principle point is located at (p_x, p_y) respective to a conventionally defined origin of image plane, this expression becomes

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.2}$$

Now, writing

$$\mathbf{K} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \tag{2.3}$$

then 2.2 has the concise form

$$\mathbf{x} = \mathbf{K}[\mathbf{I}|\mathbf{0}]\mathbf{X_{cam}} \tag{2.4}$$

The matrix **K** is called the *camera calibration matrix* or *camera intrinsic matrix* because it describe the projection from camera coordinate frame to image plane and independent from external coordinate frame, that is world coordinate frame.

2.1.2 Camera Extrinsic Matrix

In previous section, we describe a 3D point in camera coordinate frame. However, in 3D human pose datasets, 3D points **X** are mostly likely given in world coordinate frame. The two coordinate frames are related via a rotation and a translation. See Fig. 2.2.

If $\widetilde{\mathbf{X}}$ is an inhomogeneous 3-vector representing the coordinates of a point in the world, and $\widetilde{\mathbf{X}}_{cam}$ represents the same point in the camera coordinate frame, then we may write $\widetilde{\mathbf{X}}_{cam} = \mathbf{R}(\widetilde{\mathbf{X}} - \widetilde{\mathbf{C}})$, where $\widetilde{\mathbf{C}}$ represents the coordinates of the camera center in the world coordinate frame, and **R** is 3×3 rotation matrix representing the orientation of the camera coordinate frame. This equation may be written in homogeneous coordinates as

$$\mathbf{X}_{cam} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\widetilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\widetilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \mathbf{X} \tag{2.5}$$

Putting this together with (2.2) leads to the formula

$$\mathbf{x} = \mathbf{K}\mathbf{R}[\mathbf{I}| - \widetilde{\mathbf{C}}]\mathbf{X} \tag{2.6}$$

It is often convenient not to make the camera center explicit, and instead to represent he world to image transformation as $\widetilde{\mathbf{X}}_{cam} = \mathbf{R}\widetilde{\mathbf{X}} + \mathbf{t}$ In this case the camera project matrix is simply

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \tag{2.7}$$

, where from (2.6) $\mathbf{t} = -\mathbf{R}\widetilde{\mathbf{C}}$.

The concatenated matrix $[\mathbf{R}|\mathbf{t}]$ in (2.7) is called *camera extrinsic matrix*, respect to camera intrinsic matrix. It is transformation from world coordinate frame to camera coordinate frame or called the *pose* of a camera.

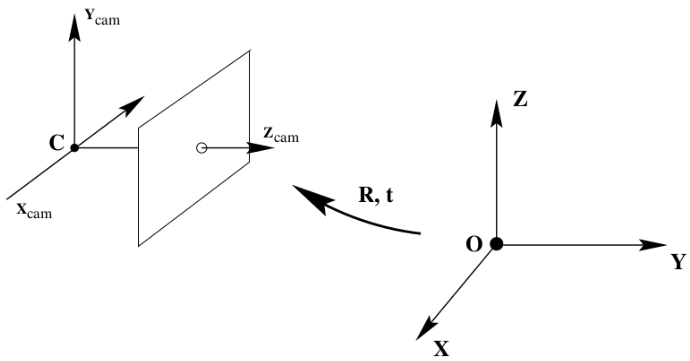

Figure 2.2: The Euclidean transformation between the world and camera coordinate frames.

2.2 Two-view Geometry

2.2.1 Epipolar Geometry

Let us denote a 3D point space as $\mathbf{X} \in \mathbb{R}^{4\times1}$ as shown in Fig. 2.3. The 3D point is imaged in two camera view, at $\mathbf{x} = \mathbf{PX}$ by the first camera, and $\mathbf{x}' = \mathbf{P}'\mathbf{X}$ by the second camera, where $\mathbf{x}$ and $\mathbf{x}' \in \mathbb{R}^{3\times1}$ represent 2D points in images, $\mathbf{P}$ and $\mathbf{P}' \in \mathbb{R}^{3\times4}$ are the camera intrinsic matrix (2.3). Since both points, $\mathbf{x}$ and $\mathbf{x}'$, have the same semantic meanings because they are the same point in 3D, we can, for example, fuse their features such that each view benefits from the other view.

The epipolar geometry [14] between two views is essentially the geometry of the intersection of the image planes with the pencil of planes having the baseline as axis. The baseline is the line joining the camera centers $\mathtt{C}_1$ and $\mathtt{C}_2$. In particular, for each location **x** in the first view, it helps us to determine the location of the corresponding point $\mathbf{x}'$ in the second view without having to know X.

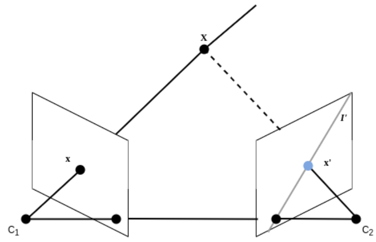

Figure 2.3: Illustration of the point-line correspondence in two views. For an arbitrary point x in one view, the corresponding point **x** in another view has to lie on the epipolar line $\mathbf{I}'$

2.2.2 Fundamental Matrix

In most cases, the 3D point **X** is unknown and, yet, we want to know where is the correspond point of **x** in the second view. We can naively search all locations in the second view, but this approach does not scale when we query more correspondent points from view 1 to view 2. Fortunately, we can compute a point-to-line relationship between two views to constrain the searching range.

The fundamental matrix is the algebraic representation of epipolar geometry. In this section, we'll only consider the case of two **calibrated** cameras. That is, the intrinsic matrices (2.3) and the poses (2.7) for both cameras are known. As illustrated in Fig. 2.3, assuming the camera centers do not overlap, the epipolar line $\mathbf{I}'$ corresponding to a given query pixel **x** = (x, y, 1) located on first view can be deterministically located on the other view as follows [14]

$$\mathbf{I}' = [\mathbf{P}'\mathtt{C}_1]_\times \mathbf{P}'\mathbf{P}^{-1}\mathbf{x} = \mathbf{F}\mathbf{x} \tag{2.8}$$

,where $[\cdot]_\times$ represents the skew symmetric matrix and **F** is the fundamental matrix.

x's corresponding point in the second view: $\mathbf{x}'$, should lie on the epipolar line: ${\mathbf{I}'}^T\mathbf{x}' = 0$.

2.3 Supervised Learning with Deep Convolutional Networks

2.3.1 Deep Convolutional Networks

Deep convolutional neural networks (DCNNs) have achieved state-of-the-art results in many computer vision tasks, such as image classification, object detection, semantic segmentation, human pose estimation, and so on. The strength is that DCNNs are able to learn richer representations than conventional hand-crafted representations.

Most recently-developed 2D pose estimation networks that able to reach state-of-art performance, including Convolutional Pose Machine [15], Stacked Hourglass Network [16], High-Resolution Network [17], adopt repeative convolutional modules in their network architectures. Convolution leverages three important ideas that can help improve a machinelearning system: **sparse interactions**, **parameter sharing** and **equivariant representations** [18].

sparse interactions

A conventional 2D covolution operation is define as

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{2.9}$$

, where K refer to an 2D kernel and I refer to an 2D tensor. One can imaging applying 2D convolution operation is like sliding a fixed window (refer to the 2D kernel) through the 2D input tensor. See Fig. 2.4.

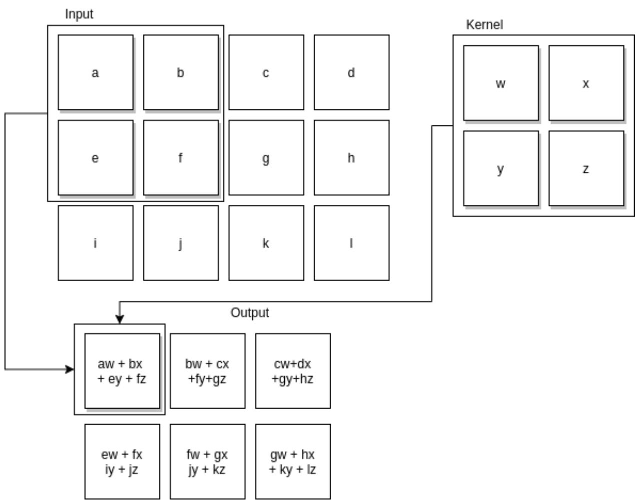

Figure 2.4: A 2D convolution. We draw boxes with arrows to indicate how the upper-left element ofthe output tensor is formed by applying the kernel to the corresponding upper-left region of the input tensor [18]

Convolutional networks typically have sparse interactions. This is accomplished by making the kernel smaller than the input. For example,when processing an image, the input image might have thousands or millions of pixels, but we can detect small,

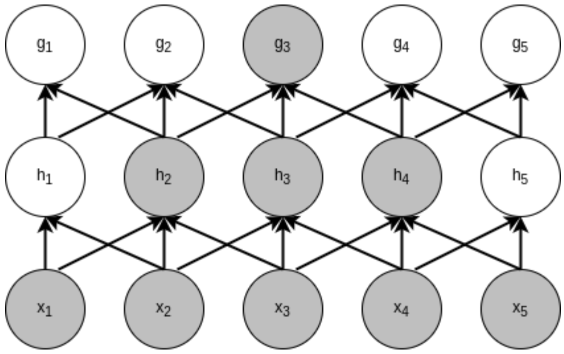

Figure 2.5: A griphical demotration of a simple 3-layer CNN with only convolutional layer. Layer **g** and **h** are formed by kernel size with width 3. The receptive field of the units in the deeper layers of a convolutional network is larger than the receptive field of the units in the shallow layers. This means that even though direct connections in a convolutional net are very sparse, units in the deeper layers can be indirectly connected to all or most of the input image [18]

.

meaningful features such as edges with kernels that occupy only tens or hundreds of pixels. This means that we need to store fewer parameters, which both reduces the memory requirements of the model and improves its statistical efficiency. In a deep convolutional network, units in the deeper layers may indirectly interact with a larger portion of the input, as shown in Fig. 2.5. This allows then etwork to efficiently describe complicated interactions between many variables byc onstructing such interactions from simple building blocks that each describe only sparse interactions.

parameter sharing

Parameter sharing refers to using the same parameter for more than onefunction in a model. This is a direct effect of the convolution, since we apply the same kernel at every location of the input tensor, shown in Fig. 2.4.

equivariant representations

In the case of convolution, the particular form of parameter sharing causes thelayer to have a property called equivariance to translation. To say a function is equivariant means that if the input changes, the output changes in the same way. Specifically, a function $f(x)$ is equivariant to a function g if $f(g(x)) = g(f(x))$. In the case of

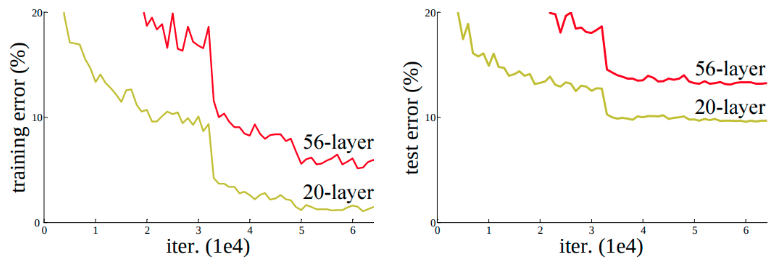

Figure 2.6: Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks (that simply stack layers). The deeper network has higher training error, and thus test error [19].

convolution, if we let g be any function that translates the input, that is, shifts it, then the convolution function is equivariant to g. In images, convolution creates a 2D map of where certain features appear in the input. If we move the object in the input, its representation will move the same amount in the output. This is useful for when we know that some function of a small number of neighboring pixels is useful when applied to multiple input locations.

2.3.2 Residual Learning and ResNets

As deeper CNNs achieve better performance on large scale datasets, researchers observe that when deeper networks are able to start converging, a degradation problem shown in has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Unexpectedly, such degradation is *not caused by overfitting*, and adding more plain layers to a suitably deep model leads to higher training error, shown in Fig. 2.6.

Let us $\mathbf{H}(\mathbf{x})$ as an underlying mapping to be fit by a few stacked layer and $\mathbf{H}(\mathbf{x})$ can approximate the desired underlying function. *He et al.* [19] propose that rather than expect stacked layers to approximate $\mathbf{H}(\mathbf{x})$, they explicitly let these layers approximate a residual function $\mathbf{F} := \mathbf{H}(\mathbf{x}) - \mathbf{x}$. The layer that learns such function $\mathbf{F}$ becomes a residual learning block Fig. 2.7.The original function thus becomes $\mathbf{F}(\mathbf{x}) + \mathbf{x}$. They argue that although both forms should be able to asymptotically approximate the desired functions, but the form of residual learning might be easier and conduct empirical experiments supporting the argument.

Residual Networks or ResNets, shown in Fig. 2.8, are able to reach a deeper depth and continuously improve their accuracy compare to plain networks without residual learning. Moreover, deeper ResNets outperform the previously state-of-art VGG nets [20] with significantly less parameters [19]. One reason that explain the improved

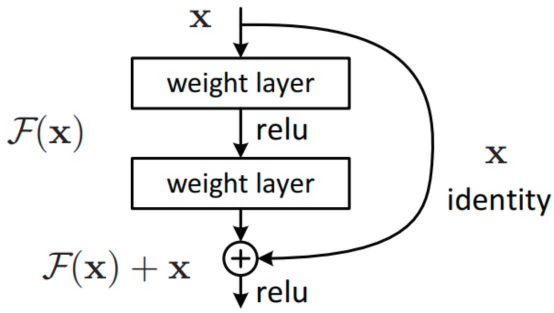

Figure 2.7: An illustration of residual block: the task for the stacked layer is to learn the residual representation $\mathbf{F}(\mathbf{x})$ [19]. The curved connection is commonly called *residual connections* or *skip connections* interchangeably nowadays.

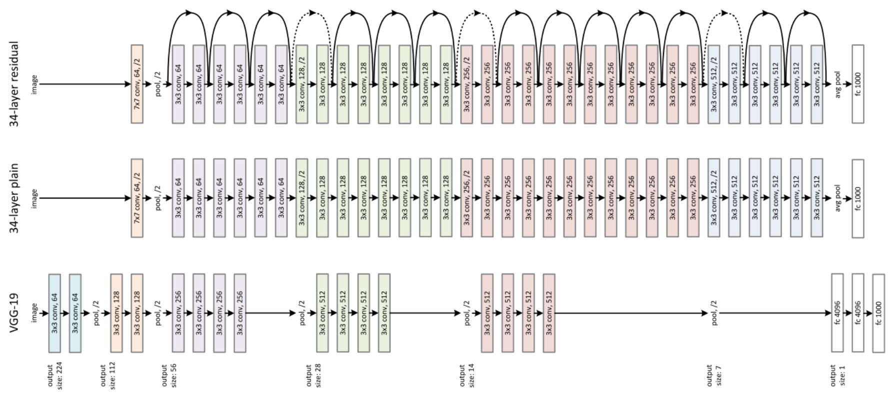

Figure 2.8: Example network architectures for ImageNet. Top: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. Middle: a plain network with 34 parameter layers (3.6 billion FLOPs). Bottom: the VGG-19 model [20] (19.6 billion FLOPs) as a reference. [19]

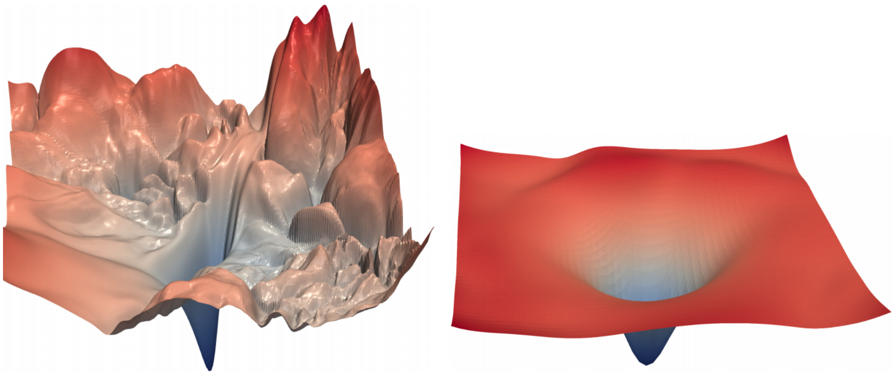

Figure 2.9: The loss surfaces of ResNet-56 (left) without residual connections (right) with residual connections [21].

learning capabilities of ResNet might be given by a research [21], that observes residual connections promote flat minimizers, shown in Fig. 2.9, and prevent the transition to chaotic behavior, which helps explain why skip connections are necessary for training extremely deep networks.

2.3.3 3D U-Net

3D U-Net [22] is a network for volumetric segmentation that learns from sparsely annotated volumetric images. See Fig. 2.10 The network is based on its predecessor that takes 2D images as input, U-Net [23], but takes 3D volumes as input and processes them with corresponding 3D operations, in particular, 3D convolutions, 3D max pooling, and 3D up-convolutional layers. Using 3D kernels allow the network to learn dependecies across the volumetric structures in data.

The network is composed by two stages: contracting stage and a expansive stage (or an analysis and a synthesis path in [22]). In contracting stage, channel size of the input is double, but the width and height are shrink to half by mas pooling, leading the receptive field of output unit to grow fast. In expansive stage, $2 \times 2 \times 2$ transpose convolution kernels with stride two are used to double the dimiension before proceeding to the next layer. In addition, shortcut connections from layers of equal resolution in the contracting stage provide the essential high-resolution features to the synthesis stage.

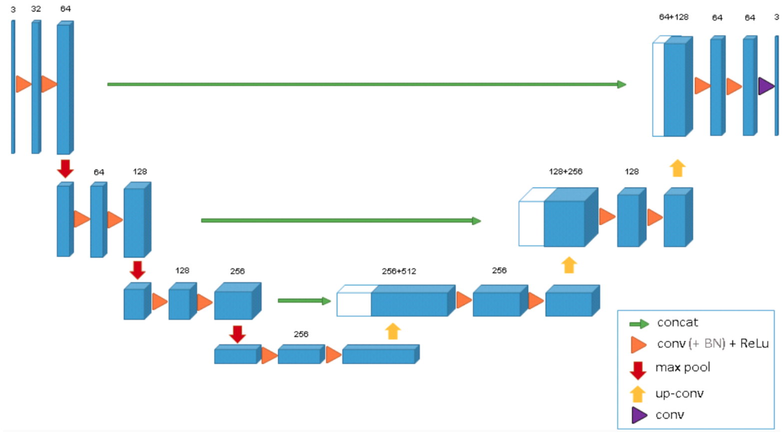

Figure 2.10: 3D U-Net architecture

2.3.4 Transfer Learning

DCNNs proof to be successful in demonstrating high accuracy in various tasks. In practice, however, very few people train an entire DCNN from scratch (with random initialization) using the specific dataset for the task, because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a DCNN on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the DCNN either as an initialization or a fixed feature extractor for the task of interest. This is called *transfer learning*

There several tricks in conducting transfer learning:

- **DCNN as fixed feature extractor** Take a DCNN pretrained on ImageNet, remove the last fully-connected layer (this layer's outputs are the 1000 class scores for a different task like ImageNet), then treat the rest of the DCNN as a fixed feature extractor for the new dataset. In an AlexNet, this would compute a 4096-D vector for every image that contains the activations of the hidden layer immediately before the classifier. We call these features CNN codes. It is important for performance that these codes are ReLUd (i.e. thresholded at zero) if they were also thresholded during the training of the DCNN on ImageNet (as is usually the case). Once you extract the 4096-D codes for all images, train a linear classifier (e.g. Linear SVM or Softmax classifier) for the new dataset.

- **Fine-tuning the DCNN** The second strategy is to not only replace and retrain the classifier on top of the DCNN on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation. It is possible to fine-tune all the layers of the DCNN, or it's possible to keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion of the network. This is motivated by the observation that the earlier features of a DCNN contain more generic features (e.g. edge detectors or color blob detectors) that should be useful to many tasks, but later layers of the DCNN becomes progressively more specific to the details of the classes contained in the original dataset. In case of ImageNet for example, which contains many dog breeds, a significant portion of the representational power of the DCNN may be devoted to features that are specific to differentiating between dog breeds.

• **Pretrained models** Since modern DCNNs take 2-3 weeks to train across multiple GPUs on ImageNet, it is common to see people release their final DCNN checkpoints for the benefit of others who can use the networks for fine-tuning. For example, the Caffe library has a Model Zoo where people share their network weights.

3 Related Works

In 3D multi-person human pose estimation, there are two challenging association problems in this task. First, it needs to associate the joints of the same person by either top-down or bottom-up strategies. Second, it needs to associate the 2D poses of the same person in different views based on appearance features which are unstable when people are occluded. We will show the pros and cons of both top-down and bottom-up approaches that tackle the mentioned association problems.

3.1 Top-down Multi-person 3D Pose Estimation

In top-down methods, humans are first detected, through bounding boxes, in each view and 2D poses are estimated for each bounding box. Then, bounding boxes are associated with individuals across multiple views. Next, estimated 2D poses are lifted to hypothesized 3D poses, where each individual might have more than one hypothesized 3D poses that need to be refined later. Finally, hypothesized 3D poses are refine, based on heuristic, 3DPS etc., to a single 3D pose for each individual.

3.1.1 Single View

Rogez *et al.* [2] estimate multiple 3D humans poses with single-view natural images using the learning-based model. They cast the task of estimating multiple 3D human poses to a combination of classification and a regression problem. In their work, they store a set of anchor 2D-3D poses, denoted by $(p, \mathbf{P})$, that collected from the MoCap dataset. In their framework, an image is first fed into a localization network that outputs a set of bounding boxes, where each box B contain a fixed set of predefined anchor poses with labels $c_B \in \{0 \dots K\}$ (0 is the background class). The scale of anchor poses is normalize according to the size of bounding boxes, making the regression independent of the scale and position of the person in the image. After the localization network is an RoI pooling layer and it is connected by two consecutive fully-connected networks: (1) *pose classification network* and (2) *pose regression network*. *The classification network* aims at predicting the closest anchor-pose, i.e., the correct label, for each bounding box B. In other words, each bounding box B is assigned with distribution u among K anchor poses plus background. *The regression network* contains $K + 1$ regressors, learned

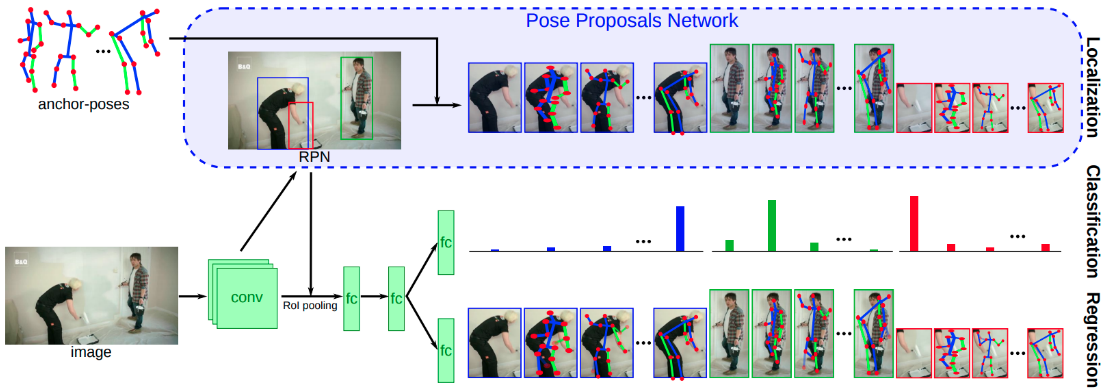

Figure 3.1: Overview of the [2] approach. It first extract candidate regions using a Region Proposal Network (RPN) and obtain pose proposals by placing a fixed set of anchor-poses into these boxes (top). These pose proposals are then scored by a classification branch and refined using class-specific regressors, learned independently for each anchor-pose.

independently for each anchor pose. The regression outputs v, that has a dimension of $5 \times J \times (K+1)$ (5 because 2D + 3D coordinates) for each B in one forward pass. See Fig. 3.1 for the pipeline. The final 2D and 3D poses are the weighted averages on v using the probability of anchor poses u.

The advantage of this work is the network can outputs full 2D and 3D poses even when the humans are partially occluded. However, the diversity of the poses depending on the number of anchor poses. In addition, the optimization for reliable 2D and 3D poses from anchor poses is not convex.

3.1.2 Multiple Views

Dong *et al.* [8] first detects p_i number of bounding boxes in *i-th* view from an off-the-shelf 2D human pose detector [24]. To associate bounding boxes across views, they formulate the task as an optimization problem, that looks for an optimal permutation matrix $\mathbf{P}$ maximizes the corresponding affinities $\mathbf{A}$. Entries of the affinity matrix $\mathbf{A_{ij}} \in \mathbb{R}^{p_i \times p_j}$ are scores that composed by the appearance similarity and the geometric compatibility between bounding boxes. Specifically, the appearance scores are calculated from the euclidean distance between a pair of feature vectors, that generated by feeding the bounding boxes into a pre-trained re-ID model [25]. Besides appearance, another important cue to associate two bounding boxes is that their associated 2D poses should be geometric consistent. More specifically, the corresponding 2D joint locations should satisfy the epiplolar constraint (2.8), i.e. a pair of corresponding joints should lie on the epipiolar lines associated with one and the other. Thus, the distance of joint 2D location $\mathbf{x_j}$, from *j-th* view, should be as close as possible to the epipolar line L_{ij} associated with $\mathbf{x_i}$ from the *i-th* view and vice versa. While solving the optimal permutation matrix $\mathbf{P_{ij}}$ for maximizing $\langle \mathbf{P_{ij}}, \mathbf{A_{ij}} \rangle$ between a pair of view can be done by Hungarian algorithm, when there are multiple views, solving the matching problem separately for each pair of views ignores the cycle-consistency constraint and may lead to inconsistent results. The all-view permutation matrix $\mathbf{P}$, composed by $\mathbf{P_{ij}}$, is an optimal solution of $\langle P_{ij}, A_{ij} \rangle$ with cycle consistency condition [26]. Finally, they solved 3D poses, from multiple hypothesized 3D poses estimated from paired 2D poses in different pairs of views, with 3DPS. As the number of views grows, however, the time complexity of using 3DPS to estimate 3D pose increases exponentially [9].

Chen *et al.* [9] formulate 3D human poses estimation as a tracking task across multiple views and timestamps. The approach has two stages: initialization and tracking stage. In tracking stage, given a pair of target (associated 2D and 3D poses in previous timestamps) and detection at the different time frames, they combine 2D and 3D geometric correspondences. For 2D correspondences, they use the euclidean distance between the 2D pose of target and detection as affinity measurements. For 3D correspondences, they back-project detection's 2D joint location to rays in 3-space and measures point-to-ray euclidean distance between target and detection. See Fig. 3.3. When initialization, they use [8] approach. To estimate 3D poses, they use a weighted triangulation with a penalty rate depending on timestamps. Since they use only triangulation for estimating 3D poses, the approach can reach realtime and time complexity only grow linearly with the number of cameras.

3.2 Bottom-up Multi-person 3D Pose Estimation

In bottom-up methods, joints are estimated first and ,then, individuals are associated through some graph algorithms.

3.2.1 Single View

Metha *et al.* [1] regress the 3D coordinates of joints directly from a single-view image with DCNN. To tackle occlusion, they treat the rigid skeleton of the human body as a kinematic tree and embed 2D locations of sibling joints and root joint to the entries of feature maps called OPRM. Thus, joints are implicitly chained with siblings and the root joint. With the embedding, the DCNN learns not only how to predict 3D coordinate of joint but also the relation between joints while training on OPRM. To associate individuals, the DCNN predict part affinity field and joint heatmaps as well.

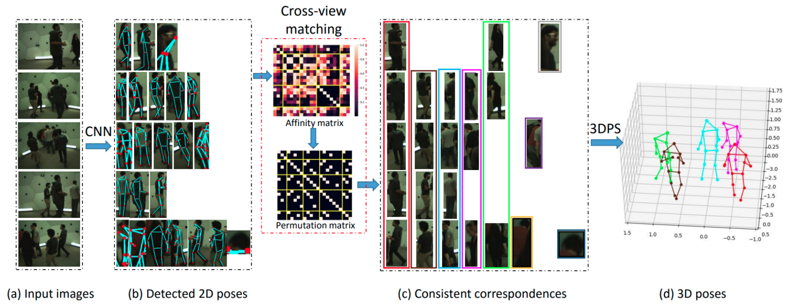

Figure 3.2: Overview of the [8] approach. (a), an off-the-shelf human pose detector is used to produce 2D bounding boxes and associated 2D poses in each view, which may be inaccurate and incomplete (b). Then, the detected bounding boxes are clustered. Each resulting cluster includes the bounding boxes of the same person in different views (c). The isolated bounding boxes that have no matches in other views are regarded as false detections and discarded. Finally, the 3D pose of each person is reconstructed from the corresponding bounding boxes and associated 2D poses (d).

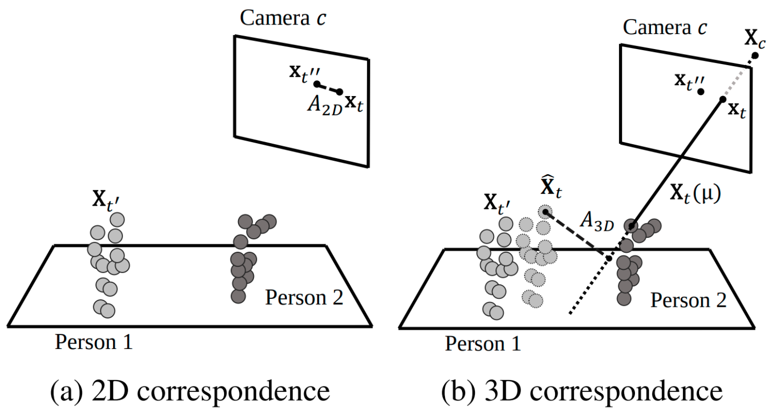

Figure 3.3: 2D and 3D correspondence affinity measurments in [9]. Geometric affinity measurement. (a) 2D correspondence is computed within the same camera. (b) 3D correspondence is measured between the predicted location and the projected line in 3-space.

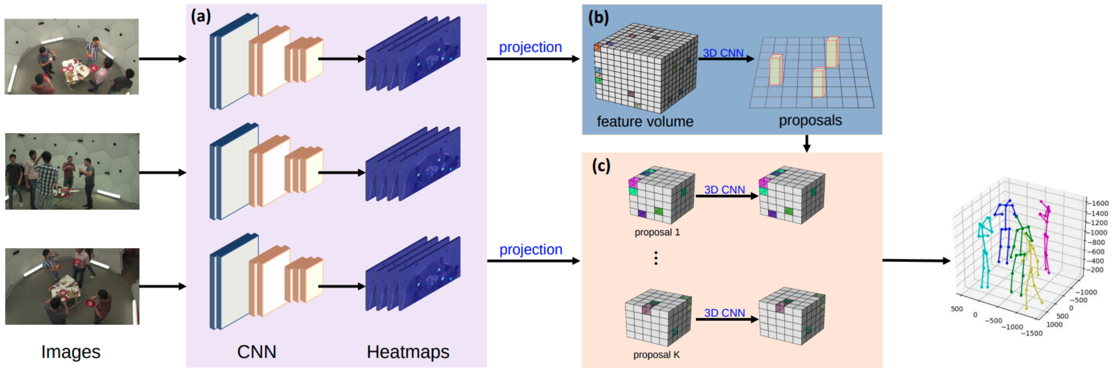

Figure 3.4: Overview of the [6] approach. : (a) it first estimate 2D pose heatmaps for all views; then, (b) it warps the heatmaps to a common 3D space and construct a feature volume which is fed into a Cuboid Proposal Network to localize all people instances; (c) for each proposal, we construct a finer-grained feature volume and estimate a 3D pose.

To predict 3D coordinate of joints, they traverse the kinematic tree from root node, to get the base 3D pose, and from edge node, to get a more versatile 3D poses. The approach is able to reach real-time performance but the 3D poses suffers from error accumulation. In addition, the approach is not robust to occlusions , predicting unreliable 3D poses when two edge joints from different individuals are in proximity or occluded.

3.2.2 Multiple Views

[3, 6] project 2D joint locations in all camera view into a common voxelized 3-space, where a projected 2D point become a ray in the voxelized 3-space. [3, 6] use 3D CNN to aggregate rays projected by same type of joints but from different views. Further, Tu *et al.* [6] use 3D CNN to predict cuboids that coarsely localize people, which can be consider as associating joints with individuals in 3-space with a learning-based method. See Fig. 3.4. The voxel-based approaches reach state-of-the-art, but these approaches require a fixed 3D grid that covers the whole environment. Thus, it only suits studio environments.

On one hand, approaches [4, 11] propose to fuse joint heatmaps from multiple source views to a single reference view, solving occlusions issue by aggregate 2D joint predictions from other views and localize 2D joints as accurate as possible. Comparing with [3, 6], these approaches does not work as good as voxel-based approach but are less computational demanding.

3.3 Temporal Fusion

[27, 28] has shown that fusing 3D joints from multiple timestamps can help learning-based methods achieve better results. Since human motion is the result of both physical limitations (torque exerted by muscles, gravity, moment preservation)) and the intentions of subjects, a learning-based method is able to pick these human motion dynamics to improve the smoothness of pose prediction. Pavllo *et al.* [28] use a ResNet-like architecture to estimate single-person 3D pose in YouTube video. The convolutional operation at each convolutional layer operate at temporal domain of the input. Moreover, the convolution kernel has dilation that increase with the depth of layer, drastically increasing the receptive field of hidden units at each layer. The residual connection connects neighboring layers between their inputs, so later layers have access to a higher temporal resolution from earlier layers. With only 4 layers, the hidden unit at output has a receptive field of 273 frames, generating temporal coherent single-person 3D pose from videos.

3.4 Brief Conclusion

In top-down approaches, the time complexity grows as number of people and view increased, since, in each view, human bounding boxes generates at least one 2D pose and bounding boxes needs to be associate with individuals across several views. On the contrary, ,in each view, bottom-up approaches estimate 2D locations of joints, by means of feature maps that has a fixed dimension regardless number of people in the view. Time complexity lies in the graph-based method or voxel-based method, which associating joints with individual across views. The voxel-based approaches suits only for constraint environments because it needs a dense 3D grid that cover the environments, refraining it from general applications.

To this end, we briefly introduce our framework for multi-person 3D pose estimation can be divided into two parts: in the first stage, we trained our own 2D joint heatmap estimator that composed of several modules inspired by [11, 4] multi-view fusion, [28] for temporal fusion. All the modules are fully convolutional and can be trained jointly. In the second stage, we associate 2D pose using [29], a graph-based greedy method in associating 3D poses. Our framework is lightweight since it operate in 2D space and can be extended into an end-to-end training scheme.

4 Multi-view Fusion Module and Fusion Network

4.1 Backbone Module: ResNets

4.2 Fusion Module: Epipolar Sampler

4.3 Fusion Network: U-Nets

5 Datasets

COCO Dataset

5.1 COCO 2017 Keypoint Dataset

5.2 CMU Panoptic Dataset

5.3 Data Augmentation

6 Experiments

6.1 Section

Citation test [**latex**].

6.1.1 Subsection

See Table 7.1, Figure 7.1, Figure 7.2, Figure 7.3.

Table 6.1: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

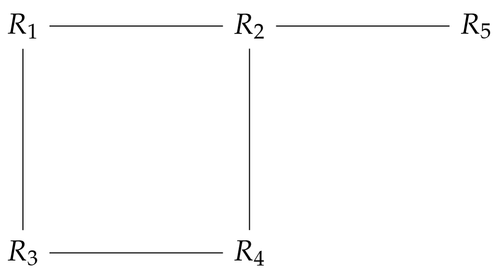

Figure 6.1: An example for a simple drawing.

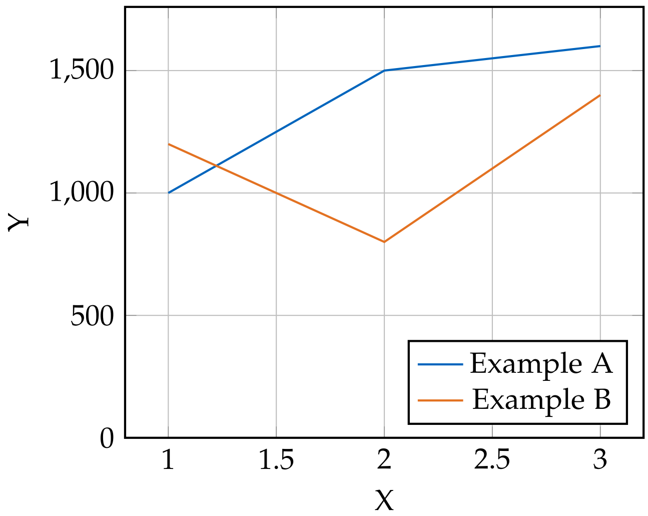

Figure 6.2: An example for a simple plot.

```
SELECT * FROM tbl WHERE tbl.str = "str"
```

Figure 6.3: An example for a source code listing.

7 Conclusion

7.1 Section

Citation test [**latex**].

7.1.1 Subsection

See Table 7.1, Figure 7.1, Figure 7.2, Figure 7.3.

Table 7.1: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

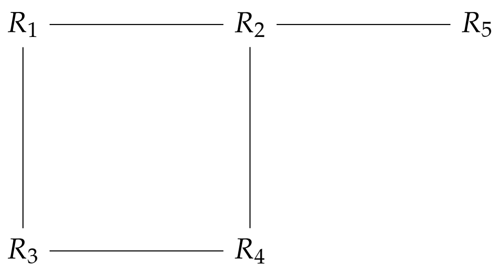

Figure 7.1: An example for a simple drawing.

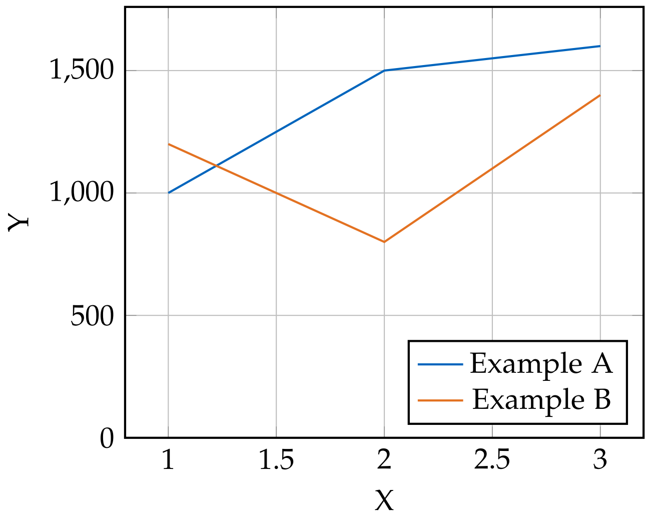

Figure 7.2: An example for a simple plot.

```
SELECT * FROM tbl WHERE tbl.str = "str"
```

Figure 7.3: An example for a source code listing.

List of Figures

List of Tables

Bibliography

[1] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, S. Sridhar, G. Pons-Moll, and C. Theobalt, “Single-shot multi-person 3d pose estimation from monocular rgb,” in *3DV*, 2018.

[2] G. Rogez, P. Weinzaepfel, and C. Schmid, “Lcr-net++: Multi-person 2d and 3d pose detection in natural images,” in *TPAMI*, 2019.

[3] K. Iskakov, E. Burkov, V. Lempitsky, and Y. Malkov, “Learnable triangulation of human pose,” in *ICCV*, 2019.

[4] H. Qiu, C. Wang, J. Wang, N. Wang, and W. Zeng, “Cross view fusion for 3d human pose estimation,” in *ICCV*, 2019.

[5] Y. Zhang, L. An, T. Yu, x. Li, K. Li, and Y. Liu, “4d association graph for realtime multi-person motion capture using multiple video cameras,” in *CVPR*, 2020.

[6] H. Tu, C. Wang, and W. Zeng, “Voxelpose: Towards multi-camera 3d human pose estimation in wild environment,” in *ECCV*, 2020.

[7] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic, “3d pictorial structures revisited: Multiple human pose estimation,” *TPAMI*, In Press.

[8] J. Dong, W. Jiang, Q. Huang, H. Bao, and X. Zhou, “Fast and robust multi-person 3d pose estimation from multiple views,” *CVPR*, 2019.

[9] L. Chen, H. Ai, R. Chen, Z. Zhuang, and S. Liu, “Cross-view tracking for multi-human 3d pose estimation at over 100 fps,” in *CVPR*, Jun. 2020.

[10] I. Sárándi, T. Linder, K. O. Arras, and B. Leibe, “How robust is 3d human pose estimation to occlusion?” In *IROS Workshop - Robotic Co-workers 4.0*, 2018.

[11] Y. He, R. Yan, K. Fragkiadaki, and S.-I. Yu, “Epipolar transformers,” in *CVPR*, 2020.

[12] Z. Zhang, C. Wang, W. Qiu, W. Qin, and W. Zeng, “Adafuse: Adaptive multiview fusion for accurate human pose estimation in the wild,” in *IJCV*, 2020.

[13] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic, “3d pictorial structures for multiple human pose estimation,” in *CVPR*, 2014.

[14] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. USA: Cambridge University Press, 2003, ISBN: 0521540518.

[15] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.

[16] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*, 2016.

[17] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *CVPR*, 2019.

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

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[21] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *NIPS*, 2018.

[22] Ö. Çiçek, A. Abdulkadir, S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: Learning dense volumetric segmentation from sparse annotation," in *MICCAI*, 2016.

[23] O. Ronneberger, P.Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[24] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Cascaded Pyramid Network for Multi-Person Pose Estimation," 2018.

[25] Z. Zhong, L. Zheng, Z. Zheng, S. Li, and Y. Yang, "Camera style adaptation for person re-identification," in *CVPR*, 2018.

[26] Q.-X. Huang and L. Guibas, "Consistent shape maps via semidefinite programming," 2013.

[27] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," in *CVPR*, 2017.

[28] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, "3d human pose estimation in video with temporal convolutions and semi-supervised training," 2019.

[29] J. Tanke and J. Gall, "Iterative greedy matching for 3d human pose tracking from multiple views," in *German Conference on Pattern Recognition*, 2019.