Learning Robust Features and Latent Representations for Single View 3D Pose Estimation of Humans and Objects

THÈSE N^{O} 8753 (2018)

PRÉSENTÉE LE 7 SEPTEMBRE 2018

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE DE VISION PAR ORDINATEUR

PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Bugra TEKIN

acceptée sur proposition du jury:

Dr R. Boulic, président du jury
Prof. P. Fua, Prof. V. Lepetit, directeurs de thèse
Prof. C. Theobalt, rapporteur
Dr S. N. Sinha, rapporteur
Prof. M. Pauly, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2018

The dude abides.
— The Dude
from *The Big Lebowski*

Sevgili aileme...
To my beloved family...

Acknowledgements

This thesis would not have been completed without the help and support of many people. I take this opportunity to express my gratitude to all of them.

First and foremost, I would like to thank my advisor Pascal Fua for giving me the opportunity to pursue my studies in his laboratory and for his continuous support during my PhD. His enthusiasm for research is a great source of inspiration for me. I would like to also thank my co-advisor Vincent Lepetit for his guidance, patience and helpfulness during all times. I greatly benefitted from his excellent advice and attempted to learn from his practical view on research. Furthermore, I am very thankful to Mathieu Salzmann for the helpful discussions that helped me very much during the last part of my PhD and Sudipta Sinha for his great guidance during my internship at Microsoft Research.

I would like to express my sincere gratitude to the members of my thesis committee, Christian Theobalt, Mark Pauly, and Ronan Boulic for kindly accepting to evaluate this work and providing valuable feedback.

During my years at CVLab, I met fantastic people who had a very positive influence on my life and I wish to say that I am grateful to all of them. In particular, to Timur, for being such a good friend and for all the fun conversations we had over countless *coffee* breaks. To Pierre and Edoardo, for making the office an enjoyable place and for all the amusing night outs. To Xiaolu, for teaching me many things about vision in my first few years in PhD. To Amos, for introducing me to CVLab and being a great mentor. To Pablo and Artem for their assistance and the good times we spent during preparation of our papers. To our secretary Ariane, for taking care of all of us and being such a nice and kind person. To all my colleagues and friends at CVLab, Xinchao, Horesh, Roberto, Alberto, Kwang, Eduard, Helge, Ksenia and many others for making CVLab an unforgettable place. I learned a lot from all of them both professionally and personally, and thanks to them, this PhD has been a great experience.

I owe my deepest gratitude to all my friends who made my life blissful and made me feel very much like at home. In particular, I would like to thank my friend and *zanza*, Can, for all the good times we have spent together for the past 7 years. I would like to also thank my long-lasting buddy Semih. Even though we are very far apart, I have the comfort of knowing that he is always going to be there for me. I am also thankful to my good friends in Lausanne, Kerem Seyid, Erhan,

Acknowledgements

Kerem Kapucu, Meric, Elsa, Nariye, Cem, Handan and Ece, among many others, for enriching my life outside of the lab. I am especially grateful to Gizem for all the wonderful times we spent together.

Last, and most importantly, I would like to express my sincere gratitude and appreciation to my beloved family, to whom this thesis is dedicated, my father Ahmet, my mother Nur, my sister Selin, and my little nieces Nil and Lina for their endless love, understanding and support during all times. Words simply cannot express how much I love them from the bottom of my heart.

Lausanne, 22 April 2018 Bugra Tekin

Abstract

Estimating the 3D poses of rigid and articulated bodies is one of the fundamental problems of Computer Vision. It has a broad range of applications including augmented reality, surveillance, animation and human-computer interaction. Despite the ever-growing demand driven by the applications, predicting 3D pose from a 2D image is a challenging and ill-posed problem due to the loss of depth information during projection from 3D to 2D. Although there have been years of research on 3D pose estimation problem, it still remains unsolved. In this thesis, we propose a variety of ways to tackle the 3D pose estimation problem both for articulated human bodies and rigid object bodies by learning robust features and latent representations.

First, we present a novel video-based approach that exploits spatiotemporal features for 3D human pose estimation in a discriminative regression scheme. While early approaches typically account for the motion information by temporally regularizing noisy pose estimates in individual frames, we demonstrate that taking into account motion information very early in the modeling process with spatiotemporal features yields significant performance improvements. We further propose a CNN-based motion compensation approach that stabilizes and centralizes the human body in the bounding boxes of consecutive frames to increase the reliability of spatiotemporal features. This then allows us to effectively overcome ambiguities and improve pose estimation accuracy.

Second, we develop a novel Deep Learning framework for structured prediction of 3D human pose. Our approach relies on an auto-encoder to learn a high-dimensional latent pose representation that accounts for joint dependencies. We combine traditional CNNs for supervised learning with auto-encoders for structured learning and demonstrate that our approach outperforms the existing ones both in terms of structure preservation and prediction accuracy.

Third, we propose a 3D human pose estimation approach that relies on a two-stream neural network architecture to simultaneously exploit 2D joint location heatmaps and image features. We show that 2D pose of a person, predicted in terms of heatmaps by a fully convolutional network, provides valuable cues to disambiguate challenging poses and results in increased pose estimation accuracy. We further introduce a novel and generic trainable fusion scheme, which automatically learns where and how to fuse the features extracted from two different input modalities that a two-stream neural network operates on. Our trainable fusion framework selects

the optimal network architecture on-the-fly and improves upon standard hard-coded network architectures.

Fourth, we propose an efficient approach to estimate 3D pose of object instances from a single RGB image. Existing methods typically detect 2D bounding boxes in the image and then predict the object pose using a pipelined approach. The redundancy in different parts of the architecture makes such methods computationally expensive. Moreover, the final pose estimation accuracy depends on the accuracy of the intermediate 2D object detection step. In our method, the object is classified and its pose is regressed in a single shot from the full image using a single, compact fully convolutional neural network. Our approach achieves the state-of-the-art pose estimation accuracy without requiring any costly pose refinement step and runs in real-time at 50 frames per second on a modern GPU, which is at least 5X faster than the state of the art.

Keywords: 3D human pose estimation, 3D object pose estimation, 6D pose estimation, 3D computer vision, motion compensation, deep learning, structured prediction

Résumé

L'estimation des poses 3D de corps rigides et articulés est l'un des problèmes fondamentaux de la vision par ordinateur. Les applications pratiques de ces méthodes sont la réalité augmentée, la surveillance, l'animation et l'interaction homme-machine. Malgré la pression exercée sur la recherche par les perspectives industrielles, prédire la pose 3D à partir d'une image 2D est un problème difficile et mal posé en raison de la perte d'informations de profondeur lors de la projection de la 3D à la 2D. En dépit d'années de recherche sur le problème de l'estimation des poses en 3D, le problème est loin d'être résolu. Dans cette thèse, nous proposons un ensemble de méthodes pour aborder le problème de l'estimation de pose 3D, à la fois pour les corps humains articulés et les corps d'objets rigides. Pour ce faire, nos algorithmes apprennent le plus souvent des caractéristiques robustes et des représentations latentes.

Premièrement, nous présentons une nouvelle approche basée sur la vidéo qui exploite les caractéristiques spatiotemporelles pour l'estimation de la posture humaine 3D dans un schéma de régression discriminatif. Alors que les approches antérieures prennent généralement en compte les informations de mouvement en régularisant temporellement les estimations de poses bruitées à partir d'images isolées, nous démontrons que la prise en compte des informations de mouvement, très tôt dans le processus de modélisation, et l'ajout de caractéristiques spatiotemporelles, améliore significativement les performances de ces algorithmes. Nous proposons en outre une approche de compensation de mouvement basée sur un CNN, qui stabilise et centre l'image du corps humain dans l'image et permet d'augmenter la fiabilité des caractéristiques spatio-temporelles. Cela nous permet alors de surmonter efficacement les ambiguïtés et d'améliorer la précision de l'estimation.

Deuxièmement, nous développons un nouveau cadre de Deep Learning pour la prédiction structurée de la pose humaine en 3D. Notre approche repose sur un auto-encodeur pour apprendre une représentation de pose latente en grande dimension qui tient compte des dépendances conjointes. Nous combinons les CNN traditionnels pour l'apprentissage supervisé avec des encodeurs automatiques pour un apprentissage structuré. Nous démon- trons que notre approche surpasse les méthodes existantes en termes de préservation de la structure et de précision des prédictions.

Troisièmement, nous proposons une approche d'estimation de pose humaine en 3D qui

repose sur une architecture de réseau de neurones à deux flux pour exploiter simultanément des cartes de probabilité de localisation et les caractéristiques d'image 2D. Nous montrons que la pose 2D d'une personne, prédite en termes de probabilités par un réseau entièrement convolutionnel, fournit des indices précieux pour désambiguer les poses difficiles et améliore l'estimation de pose. Nous introduisons en outre un schéma de fusion novateur et génératif, qui apprend automatiquement où et comment fusionner les caractéristiques extraites de deux modalités d'entrée différentes sur lesquelles un réseau de neurones à deux flux fonctionne. Notre cadre de fusion, basé sur l'apprentissage, sélectionne l'architecture optimale du réseau à la volée.

Quatrièmement, nous proposons une approche efficace pour estimer la pose 3D d'instances d'objets à partir d'une seule image couleur. Les méthodes existantes détectent généralement les cadres de délimitation 2D dans l'image, puis prédisent la pose de l'objet à l'aide d'une approche séquentielle. La redondance dans différentes parties de l'architecture rend ces méthodes coûteuses en termes de calcul. De plus, la précision de l'estimation de pose finale dépend de la précision de l'étape intermédiaire de détection d'objet 2D. Dans notre méthode, l'objet est classé et sa posture est prédite en une seule fois à partir de l'image complète en utilisant un seul réseau de neurones compact entièrement convolutionnel. Notre approche permet une précision d'estimation de pose de maximale à 50 images par seconde sur un GPU moderne, ce qui est au moins 5 fois plus rapide que l'état de l'art antérieur.

Mots clés : Estimation de pose humaine 3D, estimation de pose d'objet 3D, estimation de pose 6D, vision 3D par ordinateur, compensation de mouvement, apprentissage profond, prédiction structurée

Contents

List of Figures

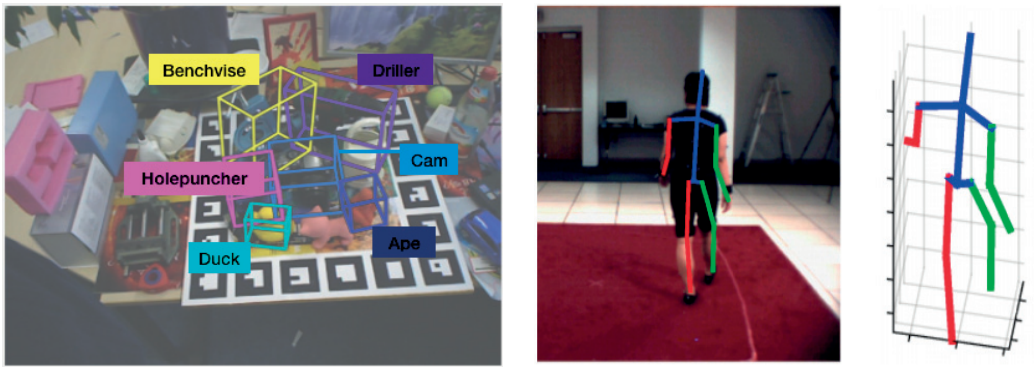
List of Tables

1 Introduction

Humans understand the world in 3 dimensions. Through the lenses of our stereo vision, we perceive the surrounding environment, humans and objects in 3D. This lets us navigate and interact within the 3D world. Much of the 3D perception has to do with understanding how the objects and people are positioned in the environment. While understanding the 3D positioning of an object is effortless for a human being, it is a challenging and ambiguous problem for a computer analyzing camera images. 3D pose estimation is the field of computer vision that addresses this problem. Its ultimate goal is to be able to understand the 3D pose of people and objects from 2D images as well as the human visual system.

3D pose estimation, as depicted by Fig. 1.1, refers to predicting from a 2D image the relative positioning of a rigid man-made object or an articulated object, such as human body, in the 3D space. Predicting the 3D pose from images or videos is a long-standing computer vision problem which has numerous applications including augmented reality, human-computer interaction, security and telepresence. However, it is a challenging and ill-posed problem. Projection from 3D onto a 2D image results in the loss of depth information and renders the problem of 3D pose estimation ambigous. A potential solution to resolve some of the ambiguities is to use multiple camera systems [139]. However, this becomes impractical due to the cost and effort in setting up a calibrated and synchrounous system of multiple cameras. Another solution to resolve the ambiguities of 3D pose estimation is to use depth sensors [173]. However, active RGB-D sensors are power hungry, which makes 3D pose estimation from RGB sensors more attractive. Monocular RGB cameras are becoming ever more prevelant in the form of mobile and web cameras and therefore there is a great interest in using them for general-purpose 3D computer vision applications, and in particular, for 3D pose estimation.

Despite many years of sustained effort, pose estimation remains a difficult problem because of the challenges due to the variability in visual appearance, variation of viewpoint, changes



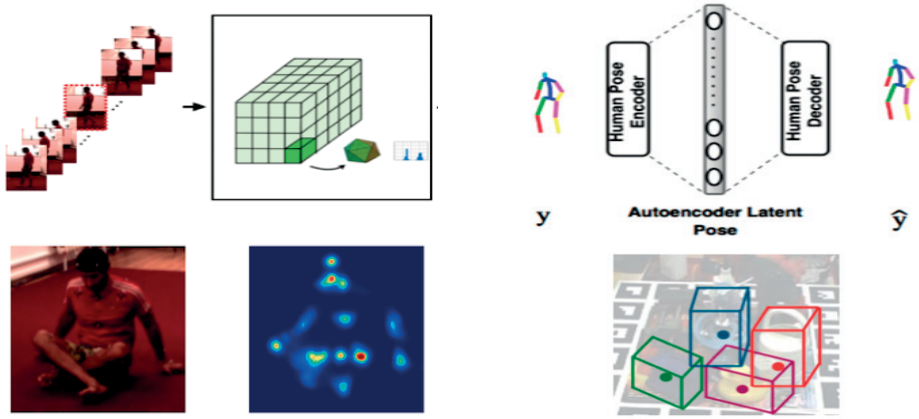
(a) 3D object pose estimation (b) 3D human pose estimation

Figure 1.1: **3D pose estimation of objects and humans.** (a) 3D object pose estimation example. The pose of an object is encoded with a rotation matrix and a translation vector that transorfms the object from its local coordinate system to the camera coordinate system. The rotation and translation both have 3 degrees-of-freedom. Therefore the problem is also commonly referred to as 6 degrees-of-freedom (DOF) object pose estimation. In the example, the 3D bounding box of objects are projected in the image plane with the estimated rotation matrix and the translation vector. (b) 3D human pose estimation example. The human pose is usually encoded with the relative 3D positions of the joints with respect to a root joint. We estimate the configuration of 3D body parts from a single RGB image in the camera coordinate system.

in illumination, occlusions and high dimensionality of the pose representations. In the face of these challenges, existing approaches are still fragile and error-prone in general unconstrained scenarios.

To tackle the challenges of 3D pose estimation, strong image features and representative prior information about the 3D pose play an essential role. In this thesis, we attempt to overcome the challenges of single-view 3D pose estimation by novel Deep Learning approaches that automatically learn reliable image features and representative latent pose representations. They typically involve finding a mapping function from robust image features to disentangled 3D pose parametrizations. As depicted by Fig. 1.2, we explore several features ranging from spatiotemporal ones to 2D joint location heatmaps and pose parametrizations ranging from automatically learned latent pose embeddings to projections of 3D virtual points. Although, the mapping between the image and the 3D pose is highly complex and entangled, we show that, in practice, we can reliably and accurately estimate the 3D pose of humans and objects in a broad range of datasets.

In the remainder of this chapter, we first define the 3D pose estimation problem and then briefly discuss a few practical applications. We then present several key challenges and summarize our main contributions. Finally, we give an outline of the thesis.



(a) Features (b) Pose parametrizations

Figure 1.2: **Features and pose parametrizations for 3D pose estimation of humans and objects. (a)** The features include spatiotemporal features extracted from short image sequences (top), and 2D joint location heatmaps (bottom). 2D joint location heatmaps are directly predicted from images by a model trained on a dataset of images of people with their corresponding 2D pose annotations. The color code denotes the confidence of a 2D joint being at a specific image location. **(b)** Representative pose parametrizations range from learned latent pose embeddings (top), to projections of the 3D bounding box of an object (bottom). Latent pose embeddings are learned from a training set of 3D motion capture data and implicitly encode prior information about the 3D pose.

1.1 Problem Definition

Our goal is to reliably and accurately predict the 3D pose of an object from single-view color images. The object could be a rigid man-made one or an articulated one, such as human body. In this thesis, we consider both the 3D object pose estimation and 3D human pose estimation problems.

3D pose of a rigid object is defined with a rotation matrix and a translation vector that transforms an object from its local coordinate system to the camera coordinate system. Rotation and translation of the object both have 3 degrees-of-freedom (DOF), therefore the rigid object pose estimation problem is also commonly referred to as 6 DOF object pose estimation.

3D human pose can be represented in a variety of ways, including kinematic trees, pictorial structures, or a set of 3D human body joint locations. We adopt the latter one, in which we represent the 3D pose of a person in terms of a skeleton, such as the one shown in Fig. 1.1. We use a 17-joint skeleton representation, therefore our output pose is a 51-dimensional vector. The output pose is defined in the camera coordinate system and consists of 3D joint locations relative to that of a root joint, e.g. pelvis. We predict the relative configuration of the 3D joint locations with respect to a root joint location, and do not consider the absolute 3D joint locations in the camera coordinate system. Therefore, our pose predictions are *person-centric* and are not with

respect to the camera center.

We formulate 3D pose estimation of both rigid and articulated bodies as a regression problem. Given an image or an image sequence, we aim to find a mapping function from robust image features to the 3D pose of an object or a human. 3D pose could either be encoded with the raw representations, as described above, or with latent pose embeddings that disentangle the complex 3D pose space. In our frameworks, we rely on deep networks to learn the regression function between the images and and their corresponding 3D pose representations.

1.2 Motivation and Applications

Computer Vision's main subjects of interest are objects and humans. To have a seamless understanding of a 3D scene, one would need to understand how the objects are situated in the 3D space and how the people move and pose. 3D pose estimation is an indispensable step towards bridging the gap between 3D human visual understanding and computer-based image analysis.

Manual acquisition of 3D poses of objects and humans requires painstaking annotation effort or expensive marker based systems. The annotation of 3D pose data involves clicking on robust 2D correspondence points on images acquired from a multi-view camera setup. These 2D annotations in multiple views can then be used for triangulation to estimate 3D coordinates or for predicting the rotation and translation with respect to a reference frame. The overall approach would typically require days to weeks of tedious work for a large set of images. Another alternative for obtaining 3D position information would be to use marker-based motion capture systems. However, expensive and invasive setup of these systems renders this approach impractical. Automatic and reliable estimation of 3D human pose has therefore emerged as a pressing need for a variety of industries and finds numerous applications ranging from augmented reality to surveillance. In the following, we briefly discuss a few prominent applications for 3D human and object pose estimation.

1.2.1 Applications of 3D Human Pose Estimation

3D human pose estimation has a wide application area in numerous industries and disciplines. A few examples are described below:

Human-Computer Interaction. 3D pose information is a reliable cue to estimate the activity and gesture of a person, and ultimately provides a natural computer interface by which computers can be controlled by human gestures. This allows for a more natural and seamless interaction

between humans and computers than the traditional means including keyboards and touchscreens.

Animation and Games. Character animation is crucial for realistic animated movies and online games. Conventional approaches rely on motion capture systems to recover the 3D pose of an actor and transfer the pose to an avatar. 3D human pose and shape estimation offers a convenient means to replace this approach, which requires a significant budget and effort.

Augmented Reality and Telepresence. Estimating the 3D human pose and shape is an initial step towards creating digital 3D avatars in the scene. This allows people to interact virtually in an augmented reality setting even when they are far apart from each other.

Security and Surveillance. Human pose and motion provides valuable information about the action and intent of a person in a video-based smart surveillance system. Since manual monitoring of the video footage in a large network of cameras is impractical and prone to human errors, such a system can assist a security personnel to detect unusual and anomalous activities.

Gait Analysis and Athletic Training. 3D human pose estimation can be used to measure the changes in physical activities of people with movement disorders, such as Parkinson's disease and Tourette syndrome, or analysing the performance of sport players for athletic training.

1.2.2 Applications of 3D Object Pose Estimation

3D object pose estimation is mainly used for augmented reality, virtual reality and robotics purposes. We briefly discuss them below.

Augmented Reality. Accurate localization of objects in the 3D space is crucial for augmented reality purposes. Augmented reality is the technology that allows to superimpose a computer-generated artifical image on an object in the scene. This is only possible when object's position in the world is known. 3D object pose estimation techniques find the rotation and translation of the object with respect to the camera, thus, allow to localize and composite the objects in the scene.

Camera Localization and Virtual Reality. Localizing the camera in the 3D space is a key task for virtual reality. The problem of camera localization, also known as camera pose estimation, is highly intertwined with object pose estimation. Predicting the pose of an object means to

localize the object in the 3D scene by finding its rotation and translation that transforms it from the local object coordinate system to the camera coordinate system. When capturing a stable object with a moving camera, this transformation also gives the rotation and translation of the camera in the 3D space.

Camera localization methods usually rely on robust image features, such as distinct corners and edges in the scene. However, when it is difficult to extract reliable features from corners and edges in the environment, the pose of stable objects could be used to localize the camera in the 3D space.

Robotics. Robot grasping and manipulation require accurate localization of an object in the 3D space. Once the object is localized in the scene, robot could navigate to the object's position in the 3D space by utilizing the translation estimate and rotate its arms in accordance with the rotation estimate.

1.3 Challenges

3D pose estimation of humans and objects is an ambiguous task due to the loss of depth information resulting from projection from 3D to 2D. We depict in Fig. 1.3 additional factors that make the problem even more challenging. We discuss these challenges below and describe common ways to address them.

Variation in Illumination. Illumination conditions play an important role in the quality of 3D object and human pose estimation. Extracting reliable cues become challenging in dim light conditions as demonstrated by example images in Fig. 1.3(a). Ultimately, pose estimation algorithms should generalize to objects captured in arbitrary illumination conditions. The common approach to gain robustness against different lighting and capturing conditions is to augment the training set with images whose hue and saturation values are synthetically changed from those of original images.

Occlusion and Clutter. As demonstrated by the examples in Fig. 1.3(b), when humans or objects are not fully visible because of occlusions or hard to distinguish from the background because of clutter, 3D pose estimation algorithms tend to become fragile. To gain robustness against occlusions and clutter, the training set could be augmented with images containing sythetically occluded objects and random backgrounds. Furthermore, methods that treat humans and objects as a combination of several parts and aggregate local part predictions to estimate the

(a) Variation in illumination (b) Occlusion & Clutter (c) Changes in viewpoint

Figure 1.3: **Challenges of 3D pose estimation for objects (top) and humans (bottom).** 3D pose estimation is a challenging problem to solve. The challenges include variation in illumination, occlusion and clutter, variation of viewpoint and more generally the loss of depth information resulting from projection from 3D to 2D. Top row: (a) an image showing objects captured at different illumination conditions from the Phos dataset [212], (b) an image showing objects on a cluttered office desk from the LINEMOD dataset [69], (c) a 3D object viewed from different angles (adapted from the slides of [73]). Bottom row: (a) a group of people under low illumination conditions (retrieved from Flickr, photo credit: Nattu Adnan) (b) an image showing occluded ultimate frisbee players on a cluttered background (photo credit: Jon Hope), (c) a football player captured from different viewpoints and its 3D pose estimation result by [197].

global pose are known to be more robust against occlusions [25, 152].

Changes in the Viewpoint. Appearance based cues tend to vary significantly when the objects or humans are viewed from different angles as in example images shown in Fig. 1.3(c). This, in turn, negatively impacts the accuracy and robustness of pose estimation algorithms. Therefore, it is essential to gain invariance against the changes in viewpoint either by collecting a large dataset of images from different viewing angles or extracting viewpoint-invariant features.

Variability in Appearance. People vary in shape, size and clothing. Therefore, to robustly predict the 3D body pose, the features extracted from images of people and the pose representations should be invariant to these factors that change across different people. Model-based 3D

object pose estimation approaches consider only a fixed known object with its corresponding 3D model, therefore such approaches do not have to account for the variability in appearance. However, model-free object pose estimation approaches have to categorize each object and be robust against the varying appearances of the same object. A large dataset covering a wide range of appearances and statistical techniques that learn appearance-invariant features are key to address this challenge.

Collecting Ground-Truth Data. As also explained in Sec. 1.2, collecting ground-truth 3D human or object pose data requires arduous annotation effort or expensive marker-based optical motion capture systems. The lack of ground-truth annotations necessitates to make the best of limited data. This would be possible by learning more robust features and representative pose priors from the already available dataset.

1.4 Contributions

The main goal of this thesis is to develop efficient and accurate methods for 3D pose estimation of humans and objects by learning robust features and latent pose representations. We demonstrate the effectiveness and versatility of our approaches in a wide range of datasets while addressing the aforementioned challenges of 3D pose estimation. We describe below the main contributions of this thesis.

3D Human Pose Estimation from Spatiotemporal Features. We propose an efficient approach to exploiting motion information from consecutive frames of a video sequence to recover the 3D pose of people. Previous approaches that rely on temporal information typically compute candidate poses in individual frames and then link them in a post-processing step to resolve ambiguities. By contrast, we directly regress from a spatio-temporal volume of bounding boxes to a 3D pose in the central frame. We further show that, for this approach to achieve its full potential, it is essential to compensate for the motion in consecutive frames so that the subject remains centered. To this end, we propose a CNN-based motion compensation approach that factors out the global body and camera motion, while preserving nonrigid body motions that serve as useful cues for 3D pose estimation. This then allows us to effectively overcome ambiguities and improve upon the state-of-the-art by a large margin on the standard 3D human pose estimation benchmarks.

Structured Predicton of 3D Human Pose with Deep Neural Networks. We develop a new deep structured learning architecture for 3D human pose estimation. Majority of existing approaches regress to individual joint coordinates independently without considering the structural dependencies in the human skeleton. By contrast, we propose to learn a structured latent space for 3D human pose by training an autoencoder on motion capture data. Instead of regressing to the individual coordinates, we regress to the latent space that accounts for the dependencies among different body parts. We show experimentally that using our algorithm that combines bottom-up and top-down reasoning yields improved performance in comparison to direct regression and existing pose estimation approaches.

Learning to Fuse 2D and 3D Image Cues for 3D Human Pose Estimation. State-of-the-art approaches to monocular 3D human pose estimation rely on Deep Learning. They typically involve regressing from an image to either 3D joint coordinates directly or 2D joint locations from which 3D coordinates are inferred. Both approaches have their strengths and weaknesses and we therefore propose a novel architecture designed to deliver the best of both worlds by performing both simultaneously and fusing the information along the way. At the heart of our framework is a trainable fusion scheme that learns how to fuse the information optimally instead of being hand-designed. This yields significant improvements upon the state-of-the-art on standard 3D human pose estimation benchmarks.

Real-Time Seamless Single Shot 6D Object Pose Prediction. We propose a single-shot approach for simultaneously detecting an object in an RGB image and predicting its 6D pose without requiring multiple stages or having to examine multiple hypotheses. Unlike existing approaches that only predict an approximate 6D pose that must then be refined, ours is accurate enough not to require additional post-processing. As a result, it is much faster (50 fps) and more suitable for real-time processing. The key component of our method is a new single-shot CNN architecture that directly predicts the 2D image locations of the projected vertices of the object's 3D bounding box. The object's 6D pose is then estimated using a PnP algorithm. For single object and multiple object pose estimation, our approach substantially outperforms other recent CNN-based approaches.

1.5 Outline

The remainder of this thesis is organized as follows. We begin Chapter 2 with an overview of the relevant literature on 3D human pose estimation of humans and objects. Chapter 3 presents our 3D human pose estimation approach from motion-compensated spatiotemporal features.

We demonstrate that using temporal information along with motion stabilization disambiguates challenging human poses with self-occlusions and depth ambiguities. Chapter 4 introduces our structured prediction approach to 3D body pose recovery using deep neural networks. We demonstrate that our approach outperforms state-of-the-art ones both in terms of structure preservation and prediction accuracy. In Chapter 5, we present a two-stream convolutional neural network architecture that learns to fuse image cues with reliable 2D joint location heatmaps. We show that 2D pose of the person, encoded as joint location heatmaps, provide valuable cues for 3D human pose estimation and guide the 3D pose estimation process. Chapter 6 introduces our real-time single shot 3D object pose estimation approach. We demonstrate that while providing state-of-the-art accuracy, our approach is at least 5 times faster than existing approaches. Finally, Chapter 7 concludes the thesis with a short summary and brief discussion of future research directions.

2 Related Work

We start this chapter by reviewing existing approaches to 3D human pose estimation, which has numerous applications ranging from surveillance to augmented reality. We then give a brief overview of state-of-the-art 3D object pose estimation techniques.

2.1 3D Human Pose Estimation

3D human pose estimation is one of the key problems in Computer Vision that has been studied for well over 20 years [4]. Existing approaches to 3D body pose recovery can be classified into two categories, traditional marker-based motion capture systems and image-based 3D human pose estimation techniques, which we describe below.

2.1.1 Motion Capture Systems

A motion capture system exploits markers near each body joint to identify the motion by the positions or angles between them. Inertial, magnetic, acoustic or optical markers, or combinations of any of these, are tracked to recover the 3D human pose. We describe below 4 different types of commercially available motion capture systems.

- *Inertial motion capture systems* rely on specialized suits onto which small gyroscopes are attached at different body locations. The angular data collected from the gyroscopes are transmitted to a computer where the 3D positions of the joints are reconstructed. Examples of such motion capture systems include Meta Motion Gypsy™ [125] and Xsens MVN™ [220].
- *Electromagnetic motion capture systems* use an array of receivers placed on different

body parts. A nearby transmitter generates a low-frequency electromagnetic signal that is detected by the receiver. This signal is then transmitted to an electronic control unit, where it is filtered and amplified. Finally it is sent to a central computer where a specialized software resolves the position and orientation of each receiver. Polhemus Liberty™ [145] and Ascension trakStar™ [10] are examples of such motion capture systems.

- *Acoustic motion capture systems* rely on transmitters attached at body joint locations that send synchronized acoustic signals. A recevier retrieve timing data from these transmitters and determines absolute body joint distances by using the differences in time of arrival of the signals [53].
- *Optical motion capture systems* use either reflective balls or pulsed LEDs attached to a specialized suit near body joint locations. Reflective markers reflect Infrared (IR) signals received from an IR transmitter, whereas pulsed LEDs directly emit light. The optical signals collected from them are used to track the 3D body joint positions. Such systems [209] are more popular because of their practical use without the need for cabling a mocap suit.

2.1.2 Image-Based 3D Human Pose Estimation

The traditional motion capture systems described above require costly hardware setups and dedicated controlled studio environments. Furthermore, most of the time, they are prone to errors due to interference from other sensors and marker occlusions, and hence require further manual post-processing. Their use is cumbersome, restrictive and invasive.

Automated Computer Vision and Machine Learning approaches to 3D human pose estimation, on the other hand, provide convenient solutions to the challenges of marker-based motion capture systems. They dispense with the need for controlled studio settings, specialized suits and expensive hardware setups.

Approaches to estimating the 3D pose of humans and objects from images can be clasiffied into different taxonomic categories, depending on their modelling choices or their inputs. Following the taxonomy depicted in Fig. 2.1, we present below a detailed analysis of the existing approaches to image-based 3D human pose estimation.

2.1.2.1 Modelling Choices

Existing 3D human pose estimation approaches have adapted a wide range of different modelling choices in terms of the pose representations, features and inference frameworks they use. In what follows, we present an analysis of these modelling choices.

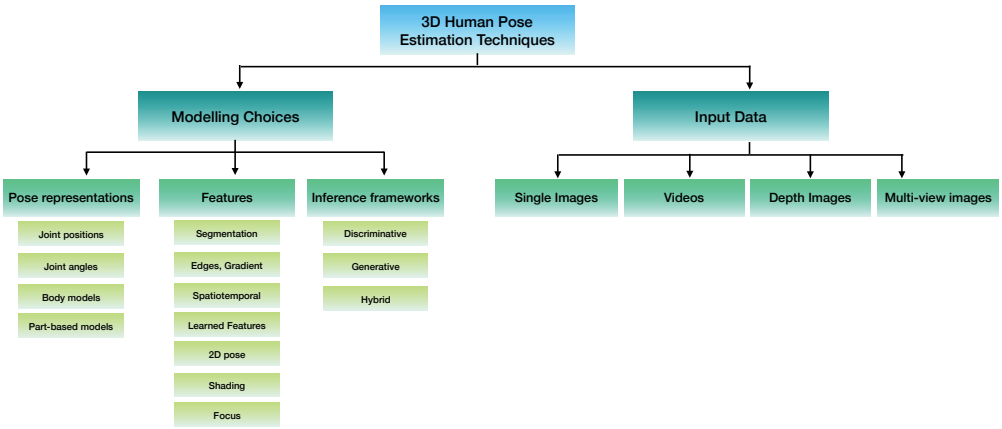


Figure 2.1: **Taxonomy of 3D human pose estimation methods.** Image-based 3D human pose estimation approaches can be classified into different categories according to their modelling choices and input data. Existing techniques employed different modelling choices depending on their pose representations, features and inference frameworks, and operated on either single images, image sequences, depth images or multi-view images.

Pose Representations. The configuration of the human body can be represented in a variety of ways. The most direct and common representation is obtained by parameterizing the pose by **3D joint positions**. The 3D locations can be defined with respect to a root joint (e.g. pelvis) [139, 195, 197, 231], with respect to the camera center [214, 233], or with respect to the parent joint [192]. Pose can also be represented as a linear combination of a set of basis poses [214, 233], or, as will be discussed in more detail in Chapter 4, can be embedded into structured latent spaces [195].

Alternatively, one can parametrize the pose as a set of **joint angles**, $\mathbf{x} = \{\tau, \theta_\tau, \theta_1, \theta_2, \ldots, \theta_N\}$. Here, the pose is encoded with the translation (τ) and orientation (θ_τ) of the root segment, and a set of relative joint angles ($\{\theta_i\}_{i=1}^{N}$). The relative angles of the joints are defined with respect to their parents. Directly predicting a pose parametrized by joint angles by defining a cost function directly on the rotations involve a complex optimization process [232]. Furthermore, with such a parametrization, the error in parent joints propagates to the other joints and results in inaccurate pose estimates at skeleton extremities (e.g. lower arms and legs). This representation, therefore, has been rarely used in direct prediction studies with few exceptions [218, 232].

By contrast, **parametrized body shape models** are more commonly used to represent 3D human poses. These detailed models can then be fitted to either 2D image evidence [17, 32, 63, 103, 179] or a rough 3D shape [208] to estimate the pose parameters. Although joint angles are seldom used for direct prediction purposes, they have been extensively used to parametrize a body shape [9, 18, 116]. Recently proposed *dynamic* human shape models [19, 147] also rely on joint angles to represent the articulation of 3D body parts.

A typical alternative to joint position and parametrized body model representations is to model the body as a set of parts, each with its own position and orientation in space that are connected by a set of physical constraints [178]. This parametrization is often called a **part-based model** and has been used by a large number of studies [6, 13, 14, 15, 26, 139].

Features. Accuracy of any 3D human pose estimation method depends heavily on the image features that are chosen to represent salient parts of the image related to the body pose. Early approaches on 3D human pose estimation have employed handcrafted image feaures. The most common ones include silhouettes [12, 32, 54, 56, 63, 82, 135, 146, 177] to separate the person from the background, edges [3, 55] to model contours of the body, and spatio-temporal features [197, 229, 230] to model the motion of body parts. [45] have further exploited shading and focus features for 3D human pose estimation. These raw feautures have been encoded by various authors using different feature descriptors, such as SIFT [2, 187], HoG [80, 81], HMAX [16], HoG3D [197], dense trajectories [229] or shape context [130], to increase robustness to noise. These features can be further processed to reduce the dimensionality by vector quantization [3] or bag-of-words [133].

Recent approaches mostly employ convolutional neural networks to automatically learn the features relevant to 3D human pose estimation [108, 109, 138, 195, 196, 199]. Early deep learning techniques have directly operated on input RGB images and employed rather shallow networks [108]. As with many other Computer Vision techniques, deeper networks have proven more useful to encode high-level features and have significantly increased the pose estimation accuracy [124, 192, 232]. Current techniques also frequently employ intermediate features, such as body part segmentations [150], 2D human pose estimates [138, 196, 199] or depth of body parts [208], and fuse them with image features to get a richer description of the body pose.

Inference Framework. Existing 3D human pose estimation approaches can be roughly categorized into discriminative and generative methods in terms of their inference frameworks.

Discriminative methods aim at predicting 3D pose directly from the input data, may it be single images [79, 80, 99, 108, 109, 129, 138, 162, 165, 195, 226], short image sequences [197], depth images [58, 149, 174] or multi-view images [16]. Early approaches falling into this category typically worked by extracting hand-crafted features and learning a mapping from these features to 3D poses [1, 16, 79, 80, 99, 166, 203], or by retrieving 3D poses from a database based on similarity with the 2D image evidence [45, 76, 109, 130, 131]. The more recent methods tend to rely on Deep Networks [108, 195, 197, 232] and reliable 2D joint location estimates obtained with them. In particular, [108, 195] rely on 2D poses to pretrain the network, thus exploiting the commonalities between 2D and 3D pose estimation. [137] introduces a network that uses initial

2D pose estimates for 3D pose estimation in a regression scheme. More recently, [138] and [199] also used 2D joint location confidence maps as an intermediate representation and combined them with the image features at certain layers of the network to guide the pose estimation process in a discriminative framework.

Another popular way to infer joint positions is to use a generative model to find a 3D pose whose projection aligns well with the image data. In the past, this usually involved inferring a 3D human pose by optimizing an energy function derived from image information, such as silhouettes [12,32,54,56,63,82,135,146,177], trajectories [229], feature descriptors [171,184,185] and 2D joint locations [5, 6, 8, 49, 95, 153, 169, 204, 207]. With the growing availability of large datasets and the advent of Deep Learning, the emphasis has focused on using discriminative 2D pose regressors [28, 31, 35, 43, 59, 78, 83, 132, 140, 142, 201, 215, 221] to extract the 2D pose and infer a 3D one from it [17, 46, 223, 233]. The 2D joint locations are usually represented by heatmaps that encode the confidence of observing a particular joint at any given image location. A human body representation, such as a skeleton [233], or a more detailed model [17] can then be fitted to these predictions. Although this takes 2D joint positions and their corresponding uncertainties into account, it ignores image information during the fitting process. Therefore, they discard potentially important 3D cues, such as shadow, texture, and color, that could help resolve ambiguities.

Hybrid approaches [169, 179, 195], on the other hand, combine the advantages of generative and dicriminative modelling by learning efficient mapping functions from images to 3D poses, while also accounting for the structural dependencies of the 3D human pose.

2.1.2.2 Input Data

Image-based 3D human pose estimation methods could be classified into 4 different categories depending on their input types: Single-Image Methods, Video-Based Methods, Depth-Based Methods and Multi-View Methods. We describe them in detail below.

Single-Image Methods. Although 3D human pose estimation from single images is fraught with ambiguities due the challenges explained in Section 1.3, in many practical applicatons we only have a single image to estimate the 3D pose as the vast amount of existing media content is still just single images. There has been, therefore, extensive research on single image 3D human pose estimation methodologies.

Early single-image 3D human pose estimation approaches tended to rely on model-fitting approaches to search the state space for a plausible configuration of the skeleton that would align

with the image evidence [56, 135, 177]. These methods remain competitive but require a plausible pose initialization and are highly sensitive to model parameters. More recent ones [13, 26] extend 2D pictorial structure approaches [50] to the 3D domain. However, in addition to their high computational cost, they tend to have difficulty localizing people's arms accurately because the corresponding appearance cues are weak and can be easily confused with the background [172].

By contrast, regression-based approaches [1, 16, 79, 138, 150, 186, 196, 199] build a direct mapping from the features extracted from a single image to the corresponding 3D pose. Such methods have been shown to be effective, especially if a large training dataset, such as [81] is available. Within this context, rich features encoding body part information [112] have been shown to be effective at increasing the estimation accuracy. More recent approaches to single-image 3D human pose estimation learn robust features with deep neural networks to resolve ambiguities caused by weak appearance cues. In particular, [108] trains a convolutional neural network between RGB images and their corresponding 3D poses. [109] extends the structured SVM model to the Deep Learning setting by learning a similarity score between feature embeddings of the input image and the 3D pose. [231] introduces a weakly-supervised transfer learning method that uses mixed 2D and 3D labels in a unified convolutional neural network. [123] addresses the problem of lifting a 2D pose to a 3D one within a deep learning context and demonstrate that 2D pose of the person provides sufficient cues for compelling pose estimation accuracy. Within a deep learning framework, [192] introduces a different pose parametrization such that the 3D position of each joint is defined with respect to its parent in terms of directional bone vectors and achieves impressive pose estimation accuracy.

All these approaches rely on direct regression of individual joint locations and do not account for the body part dependencies in human skeleton, which, most of the time, results in physically invalid poses. By contrast to direct regression approaches, in Chapter 4, we propose a new Deep Learning regression architecture for single-image 3D human pose estimation that combines traditional CNNs for supervised learning with autoencoders for unsupervised structure learning. This approach accounts for the 3D joint dependencies, preserves the body statistics and results in increased pose estimation accuracy [195]. In Chapter 5, we further introduce a discriminative fusion framework to simultaneously use 2D joint location heatmaps and 3D image cues for single image 3D human pose estimation. Within this framework, we also introduce a novel trainable fusion scheme for deep networks, which automatically learns where and how to fuse two different input modalites (e.g. image features and 2D pose heatmaps) [196].

Video-Based Methods. Human pose estimation from image sequences can be categorized into two main classes.

The first class involves frame-to-frame tracking and dynamical models [205] that rely on Markov dependencies on previous frames. Their main weakness is that they require initialization and cannot recover from tracking failures.

To address these shortcomings, the second class focuses on detecting candidate poses in individual frames followed by linking them across frames in a temporally consistent manner. For example, in [8], initial pose estimates are refined using 2D tracklet-based estimates. In [235], dense optical flow is used to link articulated shape models in adjacent frames. Non-maxima suppression is then employed to merge pose estimates across frames in [27]. Another approach [22] estimates a mapping from consecutive ground-truth 2D poses to a central 3D pose.

While spatiotemporal features have long been used for action recognition [102, 217], person detection [136], and 2D pose estimation [51], they have been underused for 3D body pose estimation purposes. The only recent approach is that of [229] that involves a computationally costly procedure of building a set of point trajectories corresponding to high joint responses and aligning them with the motion capture data.

By contrast, in Chapter 3, we propose an efficient 3D human pose estimation approach that leverages spatiotemporal features in a discriminative framework [197]. We first compensate for the residual frame-to-frame global body motion and then directly regress from a spatio-temporal volume formed by consecutive frames to a 3D pose in the central frame [197].

Depth-Based Methods. Commodity depth sensors, such as Microsoft Kinect™ [176] offer several advantages over traditional intensity cameras to overcome the ambiguities caused by projection from 3D to 2D. The advent of such sensors has generated a vast literature [11, 29, 57, 58, 62, 67, 87, 88, 143, 148, 149, 173, 175, 176, 191, 216, 224, 225]. Similarly to RGB-based techniques, existing approaches can be classified into generative [57, 62, 216, 224, 225], and discriminative [29, 58, 87, 88, 143, 148, 149, 173, 175, 176] ones.

Generative methods aim to find correspondences between a parametrized body model and input depth map. In particular, [62] fits a body model consisting of a combination of kinematic chains to the input depth image. [57] parametrizes the human body by the deformations of a flexible capsule model and aligns it to the depth map by a constrained articulated ICP algorithm. [224] retrieves 3D body configurations from a database of exemplars and match them to the input observed data. [216] formulates the 3D-to-2D registration problem in a Maximum A Posteriori framework. [225] relates the depth observation with a body model represented as a Gaussian mixture model.

Depth-based discriminative approaches find a mapping function from the depth data to the 3D

pose of the person. Random Forest [38] based methods have achieved impressive accuracy and efficiency by modelling the 3D human pose estimation problem with body part classification [29, 143, 173, 175, 176], offset regression [58, 88], or random tree walks [87]. [148, 149] further propose to account for the dependencies between different body joints within a Random Forest framework. More recently, [65] and [128] have learned the mapping function between depth maps and 3D poses with convolutional neural networks.

Multi-View Images. While sensors such as Microsot Kinect™ [176] provide valuable cues to resolve ambiguties caused by the loss of depth information, they can only work within a specific distance range in indoor environments. RGB sensors, on the other hand, do not have such range and environment limitations. To overcome depth ambiguities, multiple view imagery have been extensively used by earlier studies [6, 13, 14, 26, 41, 46, 47, 181, 183, 189]. The data collected from multiple calibrated RGB sensors could be used to alleviate depth ambiguties by making use of the relative geometrical positioning of the cameras. Furthermore, if a part of the body is not visible in some views, but could be distinguished in other views, the images from the occlusion-free viewpoints provide helpful features to disambiguate the 3D pose. Following these insights, [41, 181, 183] propose to align a graphical body model to the low-level image evidence collected from multiple views. [189] also solves the model-to-image alignment problem, however uses a body model based on mixture of Gaussians. [6, 13, 14, 26] models the human body with 3D pictorial structure approaches. However, such approaches have difficulty in localizing people's arms accurately because the corresponding appearance cues are weak and easily confused with the background [172]. More recently, [46, 47, 139] have used Deep Learning techniques to increase the robustness of appearance cues. In particular, they have proposed to fit a 3D body model onto 2D pose predictions obtained by a CNN. Although multi-view approaches yield significantly higher accuracy than their single-view counterparts [139], they require an impractical and costly setup consisting of a set of calibrated and synchronous cameras, which heavily limits their use in unconstrained outdoor settings.

2.2 3D Object Pose Estimation

Estimating the 3D pose of an object in terms of its rotation and translation is key to many augmented reality and robotics applications and has been extensively studied over many years [104]. Directly measuring the pose of an object is possible through magnetic or electromagnetic sensors that transmit positional and rotational data to a processing unit [10]. However, such methods require costly and invasive setups. Image-based 3D object pose estimation methods on the other hand have been more commonly used in industrial applications [39, 40, 127] due to their accessible

and practical nature. In what follows, we give an overview of existing work on image-based 3D object pose estimation ranging from classical feature and template matching methods to newer end-to-end trainable CNN-based methods.

Classical methods. Traditional RGB object instance recognition and pose estimation works used local keypoints and feature matching. Local descriptors needed by such methods were designed to be invariant to changes in scale, rotation, illumination and viewpoints [118, 167, 213]. Such methods are often fast and robust to occlusion and scene clutter. However, they only reliably handle textured objects in high resolution images [105]. Other related methods include 3D model-based registration [111, 117, 206], Hausdorff matching [77], oriented Chamfer matching for edges [114] and 3D chamfer matching for aligning 3D curve-based models to images [156].

RGB-D methods. The advent of commodity depth cameras has spawned many RGB-D object pose estimation methods [20, 33, 34, 92, 101, 126, 188, 228]. For example, Hinterstoisser et al. proposed template matching algorithms suitable for both color and depth images [68, 70]. Rios et al. [161] extended their work using discriminative learning and cascaded detections for higher accuracy and efficiency respectively. RGB-D methods were used on indoor robots for 3D object recognition, pose estimation, grasping and manipulation [33, 34, 36, 100, 101, 234]. Brachmann et al. [20] proposed using regression forests to predict dense object coordinates, to segment the object and recover its pose from dense correspondences. They also extended their method to handle uncertainty during inference and deal with RGB images [21]. Zach et al. [227] explored fast dynamic programming based algorithms for RGB-D images.

CNN-based methods. In recent years, research in most pose estimation tasks has been dominated by CNNs. Techniques such as Viewpoints and Keypoints [202] and Render for CNN [190] cast object categorization and 3D pose estimation into classification tasks, specifically by discretizing the pose space. In contrast, PoseNet [93] proposes using a CNN to directly regress from an RGB image to a 6D pose, albeit for camera pose estimation, a slightly different task. Since PoseNet outputs a translational and a rotational component, the two associated loss terms have to be balanced carefully by tuning a hyper-parameter during training.

To avoid this problem, the newer PoseCNN architecture [219] is trained to predict 6D object pose from a single RGB image in multiple stages, by decoupling the translation and rotation predictors. A geodesic loss function more suitable for optimizing over 3D rotations have been suggested in [121]. Another way to address this issue has recently emerged. In [91, 152], the CNNs do not directly predict object pose. Instead, they output 2D coordinates, 2D masks,

or discrete orientation predictions from which the 6D pose can be inferred. Because all the predictions are in the 2D image, the problem of weighting different loss terms for rotation and translation goes away. As a result, training becomes numerically more stable, resulting in better object pose estimation performance.

In parallel to these developments, on the 2D object detection task, there has been a progressive trend towards single shot CNN frameworks as an alternative to two-staged methods such as Faster-RCNN [159] that first find a few candidate locations in the image and then classify them as objects or background. Recently, single shot architectures such as YOLO [157, 158] and SSD [115] have been shown to be fast and accurate. SSD has been extended to predict the object's identity, its 2D bounding box in the image and a discrete estimate of the object's orientation [91, 144]. In Chapter 6, we go beyond such methods by extending a YOLO-like architecture [158] to directly predict a few 2D coordinates from which the full 6D object pose can be accurately recovered [198].

3 Direct Prediction of 3D Body Poses from Motion Compensated Sequences

In recent years, impressive motion capture results have been demonstrated using depth cameras, but 3D body pose recovery from ordinary monocular video sequences remains extremely challenging. Nevertheless, there is great interest in doing so, both because cameras are becoming ever cheaper and more prevalent and because there are many potential applications. These include athletic training, surveillance, and entertainment.

Early approaches to monocular 3D pose tracking involved recursive frame-to-frame tracking and were found to be brittle, due to distractions and occlusions from other people or objects in the scene [205]. Since then, the focus has shifted to "tracking by detection," which involves detecting human pose more or less independently in every frame followed by linking the poses across the frames [8, 155], which is much more robust to algorithmic failures in isolated frames. More recently, an effective single-frame approach to learning a regressor from a kernel embedding of 2D HOG features to 3D poses has been proposed [81]. Excellent results have also been reported using a Convolutional Neural Net [112].

However, inherent ambiguities of the projection from 3D to 2D, including self-occlusion and mirroring, can still confuse these state-of-the-art approaches. A linking procedure can correct for these ambiguities to a limited extent by exploiting motion information *a posteriori* to eliminate erroneous poses by selecting compatible candidates over consecutive frames. However, when such errors happen frequently for several frames in a row, enforcing temporal consistency afterwards is not enough.

In this chapter, we therefore propose to exploit motion information from the start. To this end, we learn a regression function that directly predicts the 3D pose in a given frame of a sequence from a spatio-temporal volume centered on it. This volume comprises bounding boxes

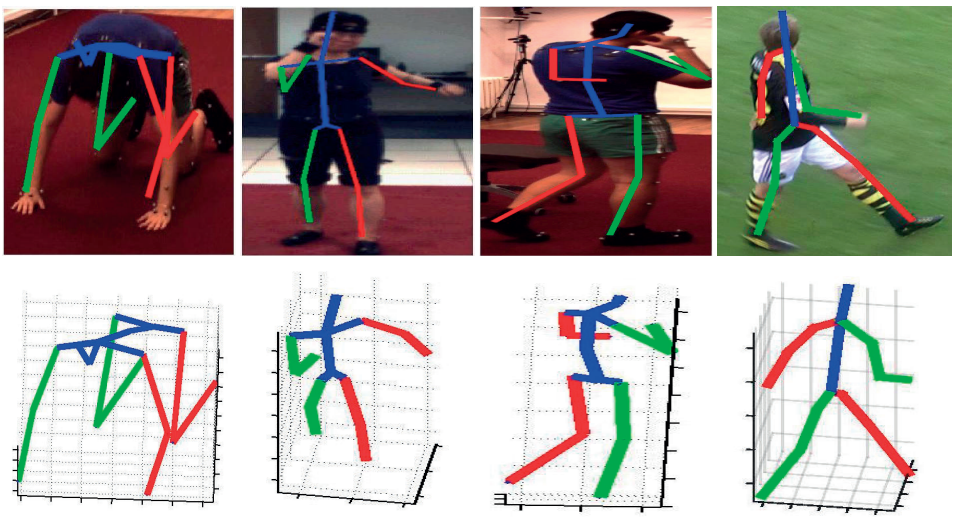
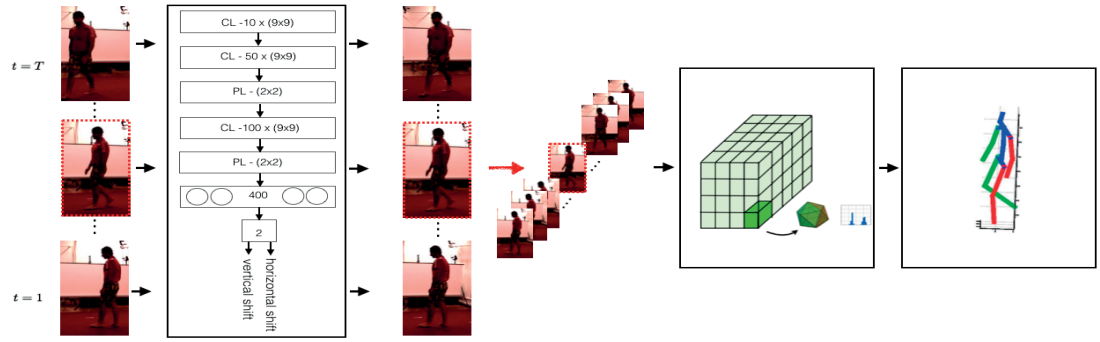


Figure 3.1: **3D human pose estimation with motion-compensated spatiotemporal features in Human3.6m, HumanEva and KTH Multiview Football datasets.** The recovered 3D skeletons are reprojected into the images in the top row and shown by themselves in the bottom row. Our approach can reliably recover 3D poses in complex scenarios by collecting appearance and motion evidence simultaneously from motion compensated sequences. All the figures in this chapter are best viewed in color.

surrounding the person in consecutive frames coming before and after the central one. We will show that this approach is more effective than relying on regularizing initial estimates *a posteriori*. We evaluated different regression schemes and obtained the best results by applying a Deep Network to the spatiotemporal features [96, 217] extracted from the image volume. Furthermore, we show that, for this approach to perform to its best, it is essential to align the successive bounding boxes of the spatio-temporal volume so that the person inside them remains centered. To this end, we trained two Convolutional Neural Networks to first predict large body shifts between consecutive frames and then refine them. This approach to motion compensation outperforms other more standard ones [136] and improves 3D human pose estimation accuracy significantly. Fig. 3.1 depicts sample results of our approach.

The novel contribution of this study is therefore a principled approach to combining appearance and motion cues to predict 3D body pose in a discriminative manner. Furthermore, we demonstrate that what makes this approach both practical and effective is the compensation for the body motion in consecutive frames of the spatiotemporal volume. We show that the proposed framework improves upon the state-of-the-art [8, 13, 16, 81, 108] by a large margin on Human3.6m [81], HumanEva [180], and KTH Multiview Football [26] 3D human pose estimation benchmarks.



(a) Image stack (b) Motion compensation (c) RSTV (d) 3D HoG (e) 3D pose regression

Figure 3.2: **Overview of our approach to 3D pose estimation with spatiotemporal features. (a)** A person is detected in several consecutive frames. **(b)** Using a CNN, the corresponding image windows are shifted so that the subject remains centered. **(c)** A *rectified spatiotemporal volume* (RSTV) is formed by concatenating the aligned windows. **(d)** A pyramid of 3D HOG features are extracted densely over the volume. **(e)** The 3D pose in the central frame is obtained by regression.

3.1 Approach

Our approach involves finding bounding boxes around people in consecutive frames, compensating for the motion to form spatiotemporal volumes, and learning a mapping from these volumes to a 3D pose in their central frame.

In the remainder of this section, we first introduce our formalism and then describe each individual step, depicted by Fig. 3.2.

3.1.1 Formalism

In this work, we represent 3D body poses in terms of skeletons, such as those shown in Fig. 3.1, and the 3D locations of their D joints relative to that of a root node. As several authors before us [16, 81], we chose this representation because it is well adapted to regression and does not require us to know *a priori* the exact body proportions of our subjects. It suffers from not being orientation invariant but using temporal information provides enough evidence to overcome this difficulty.

Let $\mathbf{I}_i$ be the i-th image of a sequence containing a subject and $\mathbf{Y}_i \in \mathbb{R}^{3 \cdot D}$ be a vector that encodes the corresponding 3D joint locations. Typically, regression-based discriminative approaches to inferring $\mathbf{Y}_i$ involve learning a parametric [1, 89] or non-parametric [203] model of the mapping function, $\mathbf{X}_i \rightarrow \mathbf{Y}_i \approx \mathbf{f}(\mathbf{X}_i)$ over training examples, where $\mathbf{X}_i = \Omega(\mathbf{I}_i; \mathbf{m}_i)$ is a feature vector computed over the bounding box or the foreground mask, $\mathbf{m}_i$, of the person in $\mathbf{I}_i$. The

model parameters are usually learned from a labeled set of N training examples, $\mathcal{T} = \{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1}^{N}$. As discussed in Section 1.3, in such a setting, reliably estimating the 3D pose is hard to do due to the inherent ambiguities of 3D human pose estimation such as self-occlusion and mirror ambiguity.

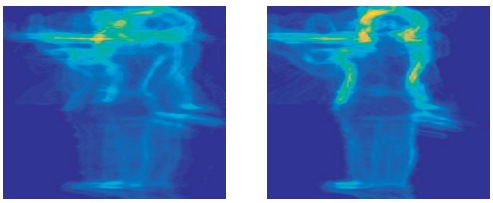
Instead, we model the mapping function $\mathbf{f}$ conditioned on a spatiotemporal 3D data volume consisting of a sequence of T frames centered at image i, $\mathbf{V}_i = [\mathbf{I}_{i-T/2+1}, \ldots, \mathbf{I}_i, \ldots, \mathbf{I}_{i+T/2}]$, that is, $\mathbf{Z}_i \rightarrow \mathbf{Y}_i \approx \mathbf{f}(\mathbf{Z}_i)$ where $\mathbf{Z}_i = \xi(\mathbf{V}_i; \mathbf{m}_{i-T/2+1}, \ldots, \mathbf{m}_i, \ldots, \mathbf{m}_{i+T/2})$ is a feature vector computed over the data volume, $\mathbf{V}_i$. The training set, in this case, is $\mathcal{T} = \{(\mathbf{Z}_i, \mathbf{Y}_i)\}_{i=1}^{N}$, where $\mathbf{Y}_i$ is the pose in the central frame of the image stack. In practice, we collect every block of consecutive T frames across all training videos to obtain data volumes. We will show in the results section that this significantly improves performance and that the best results are obtained for volumes of $T = 24$ to 48 images, that is 0.5 to 1 second given the 50 fps of the sequences of the Human3.6m [81] dataset.

3.1.2 Spatiotemporal Features

Our feature vector $\mathbf{Z}$ is based on the 3D HOG descriptor [217], which *simultaneously* encodes appearance and motion information. It is computed by first subdividing a data volume such as the one depicted by Fig. 3.2(c) into equally-spaced cells. For each one, the histogram of oriented 3D spatio-temporal gradients [96] is then computed. To increase the descriptive power, we use a multi-scale approach. We compute several 3D HOG descriptors using different cell sizes. In practice, we use 3 levels in the spatial dimensions—2×2, 4×4 and 8×8—and we set the temporal cell size to a small value—4 frames for 50 fps videos—to capture fine temporal details. Our final feature vector $\mathbf{Z}$ is obtained by concatenating the descriptors at multiple resolutions into a single vector.

An alternative to encoding motion information in this way would have been to explicitly track body pose in the spatiotemporal volume, as done in [8]. However, this involves detection of the body pose in individual frames which is subject to ambiguities caused by the projection from 3D to 2D and not having to do this is a contributing factor to the good results we will show in Section 3.2.

Another approach for spatiotemporal feature extraction could be to use 3D CNNs directly operating on the pixel intensities of the spatiotemporal volume. However, in our experiments, we have observed that, 3D CNNs did not achieve any notable improvement in performance compared to spatial CNNs. This is likely due to the fact that 3D CNNs remain stuck in local minima due to the complexity of the model and the large input dimensionality. This is also observed in [90, 122].



(a) No compensation (b) Motion compensation

Figure 3.3: **Heatmaps of the gradients across all frames for *Greeting* action (a) without and (b) with motion compensation.** When motion compensation is applied, body parts become covariant with the 3D HOG cells across frames and thus the extracted spatiotemporal features become more part-centric and stable.

3.1.3 Motion Compensation with CNNs

For the 3D HOG descriptors introduced above to be representative of the person's pose, the temporal bins must correspond to specific body parts, which implies that the person should remain centered from frame to frame in the bounding boxes used to build the image volume. We use the Deformable Part Model detector (DPM) [50] to obtain these bounding boxes, as it proved to be effective in various applications. However, in practice, these bounding boxes may not be well-aligned on the person. Therefore, we need to first shift these boxes as shown in Fig. 3.2(c) before creating a spatiotemporal volume. In Fig. 3.3, we illustrate this requirement by showing heatmaps of the gradients across a sequence without and with motion compensation. Without it, the gradients are dispersed across the region of interest, which reduces feature stability.

We therefore implemented an object-centric motion compensation scheme inspired by the one proposed in [168] for drone detection purposes, which was shown to perform better than optical-flow based alignment [136]. To this end, we train regressors to estimate the shift of the person from the center of the bounding box. We apply these shifts to the frames of the image stack so that the subject remains centered, and obtain what we call a rectified spatio-temporal volume (RSTV), as depicted in Fig. 3.2(c). We have chosen CNNs as our regressors, as they prove to be effective in various regression tasks.

More formally, let m be an image patch extracted from a bounding box returned by DPM. An ideal regressor $\psi(\cdot)$ for our purpose would return the horizontal and vertical shifts δu and δv of the person from the center of m: $\psi(m) = (\delta u, \delta v)$. In practice, to make the learning task easier, we introduce two separate regressors $\psi_{coarse}(\cdot)$ and $\psi_{fine}(\cdot)$. We train the first one to handle large shifts and the second to refine them. We use them iteratively as illustrated by Algorithm 1. After each iteration, we shift the images by the computed amount and estimate a new shift. This process typically takes only 4 iterations, 2 using $\psi_{coarse}(\cdot)$ and 2 using $\psi_{fine}(\cdot)$.

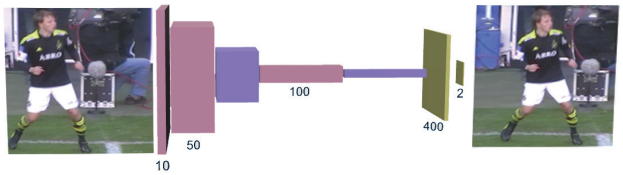


Figure 3.4: **Motion compensation CNN architecture.** The network consists of convolution (dark red), pooling (purple) and fully connected (yellow) layers. The output of the network is a two-dimensional vector that describes horizontal and vertical shifts of the person from the center of the patch.

Algorithm 1 Object-centric motion compensation.

Input: image I, initial location estimate (i, j)
$\psi_*(\cdot) = \begin{cases} \psi_{coarse}(\cdot) \text{ for the first 2 iterations,} \\ \psi_{fine}(\cdot) \text{ for the other 2,} \end{cases}$
$(i^0, j^0) = (i, j)$
for $o = 1 : MaxIter$ **do**
  $(\delta u^o, \delta v^o) = \psi_*(I(i^{o-1}, j^{o-1}))$, *with* $I(i^{o-1}, j^{o-1})$ *the image patch in* I *centered on* (i^{o-1}, j^{o-1})
  $(i^o, j^o) = (i^{o-1} + \delta u^o, j^{o-1} + \delta v^o)$
end for
$(i, j) = (i^{MaxIter}, j^{MaxIter})$

Both CNNs feature the same architecture, which comprises fully connected, convolutional, and pooling layers, as depicted by Fig. 3.2(b) and Fig. 3.4. Pooling layers are usually used to make the regressor robust to small image translations. However, while reducing the number of parameters to learn, they could negatively impact performance as our goal is precise localization. We therefore do not use pooling at the first convolutional layer, only in the subsequent ones. This yields accurate results while keeping the number of parameters small enough to prevent overfitting.

Training our CNNs requires a set of image windows centered on a subject, shifted versions, such as the one depicted by Fig. 3.4, and the corresponding shift amounts $(\delta u, \delta v)$. We generate them from training data by randomly shifting ground truth bounding boxes in horizontal and vertical directions. For ψ_{coarse}, these shifts are large, whereas for ψ_{fine}, they are small, thus representing the specific tasks of each regressor.

Using our CNNs requires an initial estimate of the bounding box for every person, which is given by DPM. However, applying the detector to every frame of the video is time consuming. Thus, we decided to apply DPM only to the first frame. The position of the detection is then refined and the resulting bounding box is used as an initial estimate in the second frame. Similarly, its position is then corrected and the procedure is iterated in subsequent frames. The initial person detector provides rough location estimates and our motion compensation algorithm naturally compensates even for relatively large positional inaccuracies using the regressor, ψ_{coarse}.

3.1.4 Pose Regression

We cast 3D pose estimation in terms of finding a mapping $\mathbf{Z} \rightarrow \mathbf{f}(\mathbf{Z}) \approx \mathbf{Y}$, where $\mathbf{Z}$ is the 3D HOG descriptor computed over a spatiotemporal volume and $\mathbf{Y}$ is the 3D pose in its central frame. To learn $\mathbf{f}$, we considered Kernel Ridge Regression (KRR) [74], Kernel Dependency Estimation (KDE) [37] as they were used in previous works on this task [79, 81], and Deep Networks.

Kernel Ridge Regression (KRR) trains a model for each dimension of the pose vector separately. To find the mapping from spatiotemporal features to 3D poses, it solves a regularized least-squares problem of the form,

$$\underset{\mathbf{W}}{\operatorname{argmin}} \sum_i ||\mathbf{Y}_i - \mathbf{W}\Phi_Z(\mathbf{Z}_i)||_2^2 + ||\mathbf{W}||_2^2 \ , \tag{3.1}$$

where $(\mathbf{Z}_j, \mathbf{Y}_j)$ are training pairs and Φ_Z is the Fourier approximation to the exponential-χ^2 kernel [81]. This problem can be solved in closed-form by $\mathbf{W} = (\Phi_Z(\mathbf{Z})^T \Phi_Z(\mathbf{Z}) + \mathbf{I})^{-1} \Phi_Z(\mathbf{Z})^T \mathbf{Y}$.

Kernel Dependency Estimation (KDE) is a structured regressor that accounts for correlations in 3D pose space. To learn the regressor, not only the input as in the case of KRR, but also the output vectors are lifted into high-dimensional Hilbert spaces using kernel mappings Φ_Z and Φ_Y, respectively [37, 81]. The dependency between high dimensional input and output spaces is modeled as a linear function. The corresponding matrix $\mathbf{W}$ is computed by standard kernel ridge regression,

$$\underset{\mathbf{W}}{\operatorname{argmin}} \sum_i ||\Phi_Y(\mathbf{Y}_i) - \mathbf{W}\Phi_Z(\mathbf{Z}_i)||_2^2 + ||\mathbf{W}||_2^2 \ , \tag{3.2}$$

To produce the final prediction $\mathbf{Y}$, the difference between the predictions and the mapping of the output in the high dimensional Hilbert space is minimized by finding

$$\hat{\mathbf{Y}} = \underset{\mathbf{Y}}{\operatorname{argmin}} \, ||\mathbf{W}^T \Phi_Z(\mathbf{Z}) - \Phi_Y(\mathbf{Y})||_2^2 \ . \tag{3.3}$$

Although the problem is non-linear and non-convex, it can nevertheless be accurately solved given the KRR predictors for individual outputs to initialize the process. In practice, we use an input kernel embedding based on 15,000-dimensional random feature maps corresponding to an exponential-χ^2 kernel, a 4000-dimensional output embedding corresponding to radial basis function kernel as in [107].

Deep Networks (DN) rely on a multilayered architecture to estimate the mapping to 3D poses. We use 3 fully-connected layers with the rectified linear unit (ReLU) activation function in the first 2 layers and a linear activation function in the last layer. The first two layers consist of 3000 neurons each and the final layer has 51 outputs, corresponding to 17 3D joint positions. We performed cross-validations across the network's hyperparameters and choose the ones with the best performance on a validation set. We minimize the squared difference between the prediction and the ground-truth 3D positions to find the mapping $\mathbf{f}$ parameterized by Θ:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}} \sum_i ||\mathbf{f}_\Theta(\mathbf{Z}_i) - \mathbf{Y}_i||_2^2 \ . \tag{3.4}$$

We used the ADAM [94] gradient update method to steer the optimization problem with a learning rate of 0.001 and dropout regularization to prevent overfitting. We will show in the results section that our DN-based regressor outperforms KRR and KDE [79, 81].

3.2 Results

We evaluate our approach on the Human3.6m [81], HumanEva-I/II [180], and KTH Multiview Football II [26] datasets. *Human3.6m* is a recently released large-scale motion capture dataset that comprises 3.6 million images and corresponding 3D poses within complex motion scenarios. 11 subjects perform 15 different actions under 4 different viewpoints. In Human3.6m, different people appear in the training and test data. Furthermore, the data exhibits large variations in terms of body shapes, clothing, poses and viewing angles within and across training/test splits [81]. The *HumanEva-I/II* datasets provide synchronized images and motion capture data and are standard benchmarks for 3D human pose estimation. We further provide results on the *KTH Multiview Football II* dataset to demonstrate the performance of our method in a non-studio environment. In this dataset, the cameraman follows the players as they move around the pitch. We compare our method against several state-of-the-art algorithms in these datasets. We chose them to be representative of different approaches to 3D human pose estimation, as discussed in Section 2.1. For those which we do not have access to the code, we used the published performance numbers and ran our own method on the corresponding data.

3.2.1 Evaluation on Human3.6m

To quantitatively evaluate the performance of our approach, we first used the Human3.6m [81] dataset. On this dataset, the regression-based method of [81] performed best at the time and we therefore use it as a baseline. That method relies on a Fourier approximation of 2D HOG features

Method	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting
e^{χ_2}-HOG+KRR [81]	140.00 (42.55)	189.36 (94.79)	157.20 (54.88)	167.65 (60.16)	173.72 (60.93)	159.25 (52.47)	214.83 (86.36)	193.81 (69.29)
e^{χ_2}-HOG+KDE [81]	132.71 (61.78)	183.55 (121.71)	132.37 (90.31)	164.39 (91.51)	162.12 (83.98)	150.61 (93.56)	171.31 (141.76)	151.57(93.84)
CNN-Regression [112]	-	148.79 (100.49)	104.01 (39.20)	127.17 (51.10)	-	-	-	-
RSTV+KRR (Ours)	119.73 (37.43)	159.82 (91.81)	113.42 (50.91)	144.24 (55.94)	145.62 (57.78)	136.43 (44.49)	166.01 (69.94)	178.93 (69.32)
RSTV+KDE (Ours)	103.32 (55.29)	158.76 (119.16)	89.22 (37.45)	127.12 (76.58)	119.35 (53.53)	115.14 (65.21)	**108.12** (84.10)	**136.82** (91.25)
RSTV+DN (Ours)	**102.41** (36.13)	**147.72** (90.32)	**88.83** (32.13)	**125.28** (51.78)	**118.02** (51.23)	**112.38** (42.71)	129.17 (65.93)	138.89 (66.18)

Method:	Sitting Down	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Average
e^{χ_2}-HOG+KRR [81]	279.07 (102.81)	169.59 (60.97)	211.31 (83.72)	174.27 (82.99)	108.37 (30.63)	192.26 (90.63)	139.76 (38.86)	178.03 (67.47)
e^{χ_2}-HOG+KDE [81]	243.03 (173.51)	162.14 (91.08)	205.94 (111.28)	170.69 (96.38)	96.60 (40.61)	177.13(130.09)	127.88 (69.35)	162.14 (99.38)
CNN-Regression [112]	-	-	189.08 (93.99)	-	77.60 (23.54)	146.59 (75.38)	-	-
RSTV+KRR (Ours)	247.21 (101.14)	140.54 (56.04)	192.75 (84.85)	156.84 (78.13)	70.98 (22.69)	152.01 (76.16)	91.47 (26.30)	147.73 (61.52)
RSTV+KDE (Ours)	**206.43** (163.55)	119.64 (69.67)	185.96 (116.29)	146.91 (98.81)	66.40 (20.92)	128.29 (95.34)	78.01 (28.70)	126.03 (78.39)
RSTV+DN (Ours)	224.9 (100.63)	**118.42** (54.28)	**182.73** (80.04)	**138.75** (77.24)	**55.07** (18.95)	**126.29** (73.89)	**65.76** (24.41)	**124.97** (57.72)

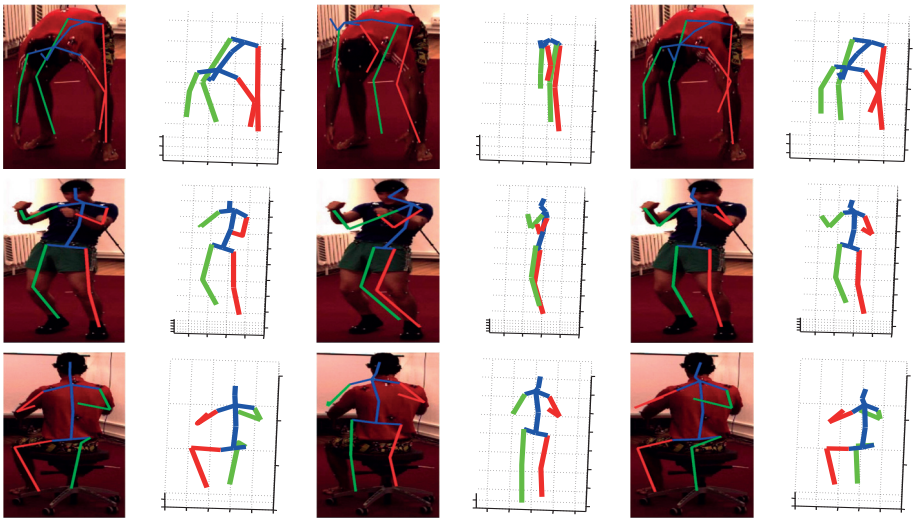
Table 3.1: **Comparison of our RSTV-Regression approach to the state-of-the-art.** 3D joint position errors in *Human3.6m* using the metric of average Euclidean distance between the ground truth and predicted joint positions (in mm) to compare our results, obtained with the different regressors described in Section 3.1.4, to those of [81] and [112]. Our method achieves significant improvement over state-of-the-art discriminative regression approaches by exploiting appearance and motion cues from motion compensated sequences. '-' indicates that the results are not reported for the corresponding action class. Standard deviations are given in parantheses.

using the χ^2 comparison metric, and we will refer to it as "e^{χ_2}-HOG+KRR" or "e^{χ_2}-HOG+KDE", depending on whether it uses KRR or KDE. Since then, even better results have been obtained for some of the actions by using CNNs [112]. We denote it as CNN-Regression. We refer to our method as "RSTV+KRR", "RSTV+KDE" or "RSTV+DN", depending on whether we use respectively KRR, KDE, or deep networks on the features extracted from the Rectified Spatiotemporal Volumes (RSTV). We report pose estimation accuracy in terms of average Euclidean distance between the ground-truth and predicted joint positions (in millimeters) as in [81, 112] and exclude the first and last $T/2$ frames (0.24 seconds for $T = 24$ at 50 fps).

The authors of [112] reported results on subjects S9 and S11 of Human3.6m and those of [81] made their code available. To compare our results to both of those baselines, we therefore trained our regressors and those of [81] for 15 different actions. We used 5 subjects (S1, S5, S6, S7, S8) for training purposes and 2 (S9 and S11) for testing. Training and testing is carried out in all camera views for each separate action, as described in [81]. Recall from Section 3.1.1 that 3D body poses are represented by skeletons with 17 joints. Their 3D locations are expressed relative to that of a root node in the coordinate system of the camera that captured the images.

Table 3.1 summarizes our results[1] on Human3.6m and Figs. 3.5-3.6 depict some of them on selected frames. Overall, our method significantly outperforms e^{χ_2}-HOG+KDE [81] for all actions, with the mean error reduced by about 23%. It also outperforms the method of [79],

[1]The sequence corresponding to Subject 11 performing *Directions* action on camera 1 in trial 2 is removed from evaluation due to video corruption.



(a) Ground Truth (b) Ionescu et. al [81] (c) Our method

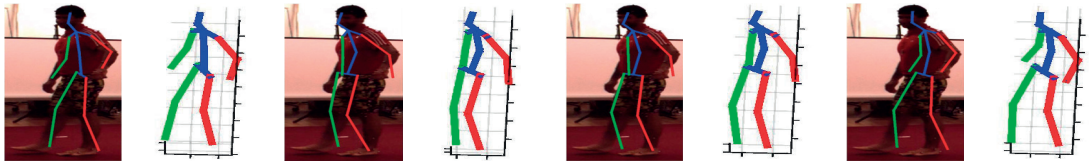
Figure 3.5: **Pose estimation results of our RSTV-Regression approach on Human3.6m.** The rows correspond to the *Buying*, *Discussion* and *Eating* actions. **(a)** Reprojection in the original images and projection on the orthogonal plane of the ground-truth skeleton for each action. **(b,c)** The skeletons recovered by the approach of [81] and our method. Note that our method can recover the 3D pose in these challenging scenarios, which involve significant amounts of self occlusion and orientation ambiguity.

which itself reports an overall performance improvement of 17% over e^{χ_2}-HOG+KDE and 33% over plain HOG+KDE on a subset of the dataset consisting of single images. Furthermore, it improves on CNN-Regression [112] by a margin of more than 5% for all the actions for which accuracy numbers are reported. The improvement is particularly marked for actions such as *Walking* and *Eating*, which involve substantial amounts of predictable motion. For *Buying*, *Sitting* and *Sitting Down*, using the structural information of the human body, RSTV+KDE yields better pose estimation accuracy. On 12 out of 15 actions and in average over all actions in the dataset, RSTV+DN yields the best pose estimation accuracy.

In the following, we analyze the importance of motion compensation and of the influence of the temporal window size on pose estimation accuracy. Additional analyses and qualitative results can be found in the appendix.

Importance of Motion Compensation. To highlight the importance of motion compensation, we recomputed our features without it. We will refer to this method as STV. We also tried using a recent optical flow (OF) algorithm for motion compensation [136].

We provide results in Table 3.2 for two actions, which are representative in the sense that the *Walking Dog* one involves a lot of movement while subjects performing the *Greeting* action tend



(a) Ground Truth (b) RSTV+KRR (c) RSTV+KDE (d) RSTV+DN

Figure 3.6: **3D human pose estimation with different regressors on RSTVs in Human3.6m. (a)** Reprojection in the original images and projection on the orthogonal plane of the ground truth skeletons for *Walking Pair* action class. **(b,c,d)** The 3D body pose recovered using the KRR, KDE or DN regressors applied to RSTV.

not to walk much. Even without the motion compensation, regression on the features extracted from spatiotemporal volumes yields better accuracy than the method of [81]. Motion compensation significantly improves pose estimation performance as compared to STVs. Furthermore, our CNN-based approach to motion compensation (RSTV) yields higher accuracy than optical-flow based motion compensation [136].

Action:	[81]	STV	STV+OF [136]	RSTV
Greeting	164.39	144.48	140.97	**127.12**
Walking Dog	177.13	138.66	134.98	**126.29**

Table 3.2: **Importance of motion compensation.** The results of [81] are compared against those of our method, without motion compensation and with motion compensation using either optical flow (OF) of [136] or our algorithm introduced in Section 3.1.3.

Influence of the Size of the Temporal Window. In Table 3.3, we report the effect of changing the size of our temporal windows from 12 to 48 frames, again for two representative actions. Using temporal information clearly helps and the best results are obtained in the range of 24 to 48 frames, which corresponds to 0.5 to 1 second at 50 fps. When the temporal window is small, the amount of information encoded in the features is not sufficient for accurate estimates. By contrast, with too large windows, overfitting can be a problem as it becomes harder to account for variation in the input data. Note that a temporal window size of 12 frames already yields better results than the method of [81]. For the experiments we carried out on Human3.6m, we use 24 frames as it yields both accurate reconstructions and efficient feature extraction.

3.2.2 Evaluation on HumanEva

We further evaluated our approach on HumanEva-I and HumanEva-II datasets. The baselines we considered are frame-based methods of [16, 46, 76, 99, 184, 185, 214], frame-to-frame-tracking

Action:	[81]	RSTV			
		12 frames	24 frames	36 frames	48 frames
Walking	96.60	58.78	55.07	**53.68**	54.36
Eating	132.37	93.97	88.83	87.23	**85.36**

Table 3.3: **Influence of the size of the temporal window on the pose estimation accuracy.** We compare the results of [81] against those obtained using our method, RSTV+DN, with increasing temporal window sizes.

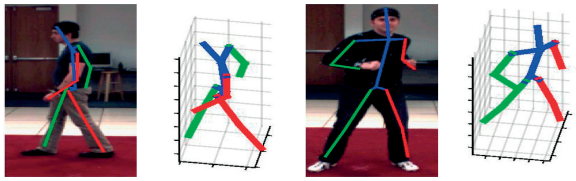


Figure 3.7: **Results of our RSTV-Regression approach on HumanEva-I.** The recovered 3D poses and their projection on the image are shown for *Walking* and *Boxing* actions. More results are provided in the appendix.

approaches which impose dynamical priors on the motion [183,194] and the tracking-by-detection framework of [8]. The mean Euclidean distance between the ground-truth and predicted joint positions is used to evaluate pose estimation performance. As the size of the training set in HumanEva is too small to train a deep network, we use RSTV+KDE instead of RSTV+DN.

We demonstrate in Tables 3.4 and 3.5 that using temporal information earlier in the inference process in a discriminative bottom-up fashion yields more accurate results than the above-mentioned approaches that enforce top-down temporal priors on the motion.

HumanEva-I: For the experiments we carried out on HumanEva-I, we train our regressor on training sequences of Subject 1, 2 and 3 and evaluate on the "validation" sequences in the same manner as the baselines we compare against [13, 16, 46, 99, 183, 184, 185, 194, 214]. Spatiotemporal features are computed only from the first camera view. We report the performance of our approach on cyclic and acyclic motions, more precisely *Walking* and *Boxing*, in Table 3.4 and depict example 3D pose estimation results in Fig. 3.7. The results show that our method outperforms the state-of-the-art approaches on this benchmark as well.

HumanEva-II: On HumanEva-II, we compare against [8,76] as they report the best monocular pose estimation results on this dataset. HumanEva-II provides only a test dataset and no training data, therefore, we trained our regressors on HumanEva-I using videos captured from different camera views. This demonstrates the generalization ability of our method to different camera

	Walking				*Boxing*			
Method:	S1	S2	S3	Avg.	S1	S2	S3	Avg.
Taylor et al. [194]	48.8	47.4	49.8	48.7	75.35	-	-	-
Sigal et al. [183]	66.0	69.0	-	-	-	-	-	-
Simo-Serra et al.'12 [185]	99.6	108.3	127.4	111.8	-	-	-	-
Simo-Serra et al.'13 [184]	65.1	48.6	73.5	62.2	-	-	-	-
Wang et al. [214]	71.9	75.7	85.3	77.6	-	-	-	-
Belagiannis et al. [13]	68.3	-	-	-	62.70	-	-	-
Elhayek et al. [46]	66.5	-	-	-	60.0	-	-	-
Kostrikov et al. [99]	44.0	30.9	**41.7**	38.9	-	-	-	-
Bo et al. [16]	45.4	28.3	62.3	45.33	**42.5**	64.0	69.3	58.6
Ours	**37.5**	**25.1**	49.2	**37.3**	50.5	**61.7**	**57.5**	**56.6**

Table 3.4: **Comparison of our RSTV-Regression approach to the state-of-the-art on HumanEva-I.** We report 3D joint position errors (in mm) on the *Walking* and *Boxing* sequences of HumanEva-I. We compare our approach against methods that rely on discriminative regression [16, 99], 2D pose detectors [184, 185, 214], 3D pictorial structures [13], CNN-based markerless motion capture method of [46] and methods that rely on top-down temporal priors [183, 194]. '-' indicates that the results are not reported for the corresponding sequences.

views. Following [8], we use subjects *S*1, *S*2 and *S*3 from HumanEva-I for training and report pose estimation results in the first 350 frames of the sequence featuring subject S2. Global 3D joint positions in HumanEva-I are projected to camera coordinates for each view. Spatiotemporal features extracted from each camera view are mapped to 3D joint positions in its respective camera coordinate system, as done in [151]. Whereas [8] uses additional training data from the "People" [154] and "Buffy" [51] datasets, we only use the training data from HumanEva-I. We evaluated our approach using the official online evaluation tool. We illustrate the comparison in Table 3.5, where our method achieves the state-of-the-art performance.

Method:	S2/C1	S2/C2	S2/C3	Average
Andriluka et al. [8]	107	101	-	-
Howe [76]	81	**73**	143	99
Ours	**79.6**	79.0	**79.2**	**79.3**

Table 3.5: **Comparison of our RSTV-Regression approach to the state-of-the-art on HumanEva-II.** 3D joint position errors (in mm) on the Combo sequence of the HumanEva-II dataset. We compare our approach against the tracking-by-detection framework of [8] and recognition-based method of [76]. '-' indicates that the result is not reported for the corresponding sequence.

Camera 1 Camera 2 Camera 3

Figure 3.8: **Results of our RSTV-Regression approach on KTH Multiview Football II.** The 3D skeletons are recovered from Camera 1 images and projected on those of Camera 2 and 3, which were not used to compute the poses.

Body parts:	[26] (Mono)	[26] (Stereo)	[13](Stereo)	Ours (Mono)
Pelvis	97	97	-	**99**
Torso	87	90	-	**100**
Upper arms	14	53	64	**74**
Lower arms	06	28	**50**	49
Upper legs	63	88	75	**98**
Lower legs	41	**82**	66	77
All parts	43	69	-	**79**

Table 3.6: **Comparison of our RSTV-Regression approach to the state-of-the-art on KTH Multiview Football II.** On the KTH Multiview Football II we have compared our method using a single camera to those of [26] using either single or two cameras and to the one of [13] using two cameras. '-' indicates that the result is not reported for the corresponding body part.

3.2.3 Evaluation on KTH Multiview Football Dataset

As in [13, 26], we evaluate our method on the sequence containing Player 2. The first half of the sequence is used for training and the second half for testing, as in the original work [26]. To compare our results to those of [13, 26], we report pose estimation accuracy in terms of percentage of correctly estimated parts (PCP) score. As in the HumanEva experiments, we provide results for RSTV+KDE. Fig. 3.8 depicts example pose estimation results. As shown in Table 3.6, we outperform the baselines even though our algorithm is monocular, whereas they use both cameras. This is due to the fact that the baselines instantiate 3D pictorial structures relying on 2D body part detectors, which may not be precise when the appearance-based information is weak. By contrast, collecting appearance and motion information simultaneously from rectified spatiotemporal volumes, we achieve better 3D pose estimation accuracy.

3.3 Conclusion

We have demonstrated that taking into account motion information very early in the modeling process yields significant performance improvements over doing it *a posteriori* by linking pose estimates in individual frames. We have shown that extracting appearance and motion cues

from rectified spatiotemporal volumes disambiguate challenging poses with mirroring and self-occlusion, which brings about substantial increase in accuracy over the state-of-the-art methods on several 3D human pose estimation benchmarks. Our proposed framework is generic and could be used for other kinds of articulated motions.

4 Learning Structured Latent Representations of 3D Human Pose with Deep Neural Networks

In spite of much recent progress, estimating 3D human pose from a single ordinary image remains challenging because of the many ambiguities inherent to monocular 3D reconstruction. They include occlusions, complex backgrounds, and, more generally, the loss of depth information resulting from the projection from 3D to 2D.

Recent regression-based methods can directly and efficiently predict the 3D pose given the input image [108] or images [197] but often ignore the underlying body structure and resulting joint dependencies, which makes them vulnerable to ambiguities. Several methods have recently been proposed to account for these dependencies [81, 109, 170]. In particular, by leveraging the power of Deep Learning, the method of [109] achieves high accuracy. However, it involves a computationally expensive search procedure to estimate the 3D pose.

Since pose estimation is much better-posed in 2D than in 3D, an alternative way to handle ambiguities is to use discriminative 2D pose regressors [28, 31, 43, 59, 83, 132, 140, 142, 201, 215, 221] to extract the 2D pose and then infer a 3D one from it [17, 46, 223, 233]. This however also involves fitting a 3D model in a separate optimization step, and is thus more expensive than direct regression.

In this chapter, we demonstrate that we can account for the human pose structure within a deep learning regression framework. To this end, we propose to first train an overcomplete autoencoder that projects body joint positions to a high dimensional space represented by its middle layer, as depicted by Fig. 4.1(a). We then learn a CNN-based mapping from the image to this high-dimensional pose representation as shown in Fig. 4.1(b). Finally, as illustrated in Fig. 4.1(c), we connect the decoding layers of the autoencoder to the CNN, and fine-tune the whole model for pose estimation. This procedure is inspired by Kernel Dependency Estimation

(KDE) in that it can be understood as replacing the high-dimensional feature maps in kernel space by autoencoder layers that represent the pose in a high-dimensional space encoding complex dependencies between the different body parts. However, our approach has the advantage over KDE of directly providing us with a mapping back to the pose space, thus avoiding the need for a computationally expensive optimization at test time. Altogether, and as will be demonstrated by our experiments, our framework enforces implicit constraints on the human pose, preserves the human body statistics, and improves prediction accuracy.

With the growing availability of large training datasets, 2D pose estimation algorithms have achieved tremendous success [132, 142, 215] by relying on Deep Learning. They exploit the fact that finding 2D joint locations in a color image is easier than direct 3D pose prediction, which is fraught with depth ambiguities. To leverage the well-posedness of the 2D localization problem, we therefore use the reliable 2D joint location heatmaps produced by [132] as input to our autoencoder-based regression architecture. We show that this improves 3D pose accuracy upon direct regression from an RGB image. We further show that our autoencoder-based regression approach scales to very deep architectures and achieves state-of-the-art performance when used with ResNet architecture [66].

Because we can perform 3D pose-estimation using a single CNN, our approach can easily be extended to handling sequences of images instead of single ones. To this end, we introduce two LSTM-based architectures: one that acts on the pose predictions in consecutive images, and one that models temporal information directly at the feature level. Our experiments evidence the additional benefits of modeling this temporal information over our single-frame approach.

In short, our contribution is to show that combining traditional CNNs for supervised learning with autoencoders for structured learning preserves the power of CNNs while also accounting for dependencies, resulting in increased performance. In the remainder of the chapter, we first briefly discuss earlier approaches. We then present our structured prediction framework in more detail, introduce our LSTM-based architectures and finally demonstrate that our approach achieves competitive performance with the state-of-the-art methods on standard 3D human pose estimation benchmarks.

Previous work. Following recent trends in Computer Vision, human pose estimation is now usually formulated within a Deep Learning framework. The switch away from earlier representations started with 2D pose estimation by learning a regressor from an input image either directly to pose vectors [201] or to heatmaps encoding 2D joint locations [83, 140, 200]. This has been exploited very effectively to infer 3D poses by fitting a 3D model to the 2D predictions [17, 46, 223, 233]. This approach currently yields some of the best results, but involves a separate, typically expensive model-fitting stage, outside of the Deep Learning framework.

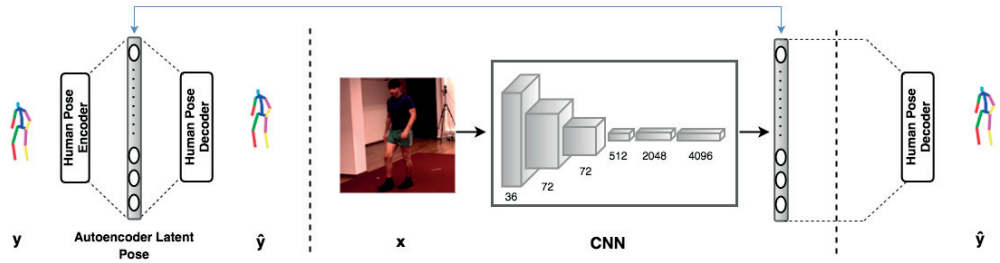


Figure 4.1: **Our architecture for the structured prediction of the 3D human pose. (a)** An autoencoder whose hidden layers have a larger dimension than both its input and output layers is pretrained. In practice we use either this one or more sophisticated versions that are described in more detail in Section 4.1.1 **(b)** A CNN maps either a monocular image or a 2D joint location heatmap to the latent representation learned by the autoencoder. **(c)** The latent representation is mapped back to the original pose space using the decoder.

In parallel, there has been a trend towards performing direct 3D pose estimation [81, 108], formulated as a regression problem. In other words, the algorithms output continuous 3D joint locations, because discretizing the 3D space is more challenging than the 2D one.

Our work fits in that line research, which involves dealing with the ambiguities inherent to inferring a 3D pose from a 2D input. To resolve them, recent algorithms have sought to encode the dependencies between the different joints within Deep Learning approaches, thus effectively achieving structured prediction. In particular, [75] uses autoencoders to learn a shared representation for 2D silhouettes and 3D poses. This approach, however, relies on accurate foreground masks and exploits handcrafted features, which mitigates the benefits of Deep Learning. In the context of hand pose estimation, [134] introduces a bottleneck, low dimensional layer that aims at accounting for joint dependencies. This layer, however, is obtained directly via PCA, which limits the range of dependencies it can model.

The work of [109] constitutes an effective approach to encoding dependencies within a Deep Learning framework for 3D human pose estimation. This approach extends the structured SVM model to the Deep Learning setting by learning a similarity score between feature embeddings of the input image and the 3D pose. This process, however, comes at a high computational cost at test time, since, given an input image, the algorithm needs to search for the highest-scoring pose. Furthermore, the final results are obtained by averaging over multiple high-scoring ground-truth training poses, which might not generalize well to unseen data since the prediction can thus only be in the convex hull of the ground-truth training poses.

To achieve a similar result effectively, we drew our inspiration from earlier KDE-based approaches [80, 81], which map both image and 3D pose to high-dimensional Hilbert spaces

and learn a mapping between these spaces. In this study, we show how to do this in a Deep Learning context by combining CNNs and autoencoders. Not only does this allow us to leverage the power of learned features, which have proven more effective than hand-designed ones such as HOG [1] and 3D HOG [217], but it yields a direct and efficient regression between the two spaces. Furthermore, it also allows us to learn the mapping from high-dimensional space to pose space, thus avoiding the need of KDE-based methods to solve an optimization problem at test time.

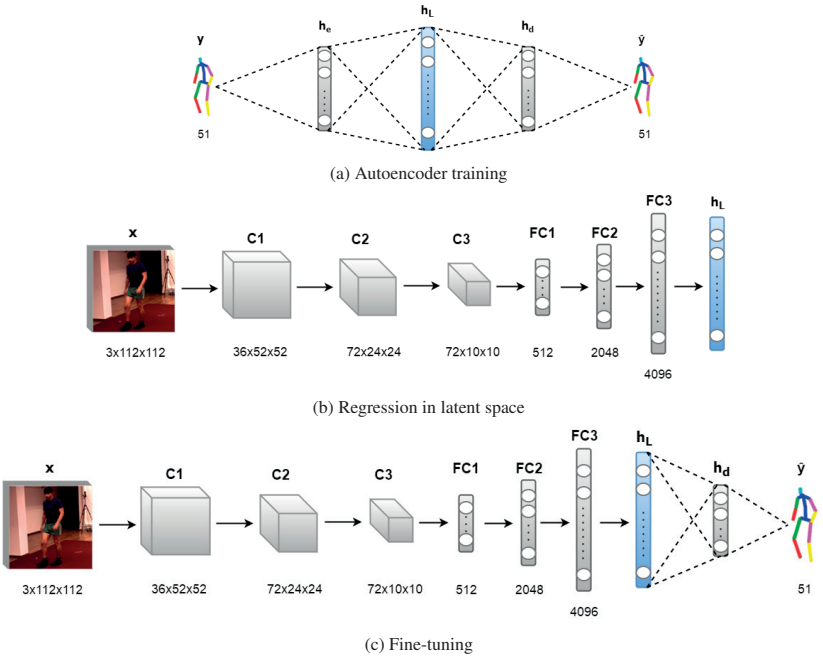
Using autoencoders for unsupervised feature learning has proven effective in several recognition tasks [98, 211]. In particular, denoising autoencoders [210] that aim at reconstructing the perfect data from a corrupted version of it have demonstrated good generalization ability. Similarly, contractive autoencoders have been shown to produce intermediate representations that are robust to small variations of the input data [160]. All these methods, however, rely on autoencoders to learn features for recognition tasks. By contrast, here, we exploit them to model the output structure for regression purposes.

In this study, we further investigate the use of Recurrent Neural Networks (RNNs), and in particular LSTMs, to model temporal information. RNNs have recently been used in many Natural Language Processing [97, 193] and Computer Vision [113, 141] tasks, and, at the intersection of these fields, for image captioning and video description [42, 85]. More closely related to our work, in [52, 84], RNNs have been employed to model human dynamics. Nevertheless, these methods do not tackle human pose estimation, but motion capture generation, video pose labeling and forecasting for [52], and human-object interaction prediction for [84]. To the best of our knowledge [110] is the only method that exploits RNNs for 3D human pose estimation from images. However, this approach operates on single images and makes use of RNNs to iteratively refine the pose predictions of [109]. By contrast we leverage the power of RNNs at modeling long term temporal dependencies across image sequences.

4.1 Approach

In this work, we aim at directly regressing from an input image or heatmap x to a 3D human pose. As in [16, 81, 108], we represent the human pose in terms of the 3D locations $y \in \mathbb{R}^{3J}$ of J body joints relative to a root joint. An alternative would have been to predict the joint angles and limb lengths. However, this is a less homogeneous representation and is therefore rarely used for regression purposes.

As discussed above, a straightforward approach to creating a regressor is to train a conventional CNN such as the one used in [108]. However, this fails to encode dependencies



(a) Autoencoder training

(b) Regression in latent space

(c) Fine-tuning

Figure 4.2: **Overview of our structured prediction approach. (a)** We train a stacked denoising autoencoder that learns the structural information and enforces implicit constraints about human body in its latent middle layer h_L. **(b)** Our CNN architecture maps the raw image or the 2D joint location heatmap predicted from the input image to the latent representation h_L learned by the autoencoder. **(c)** We stack the decoding layers of the autoencoder on top of the CNN for reprojection from the latent space to the original pose space and fine-tune the entire network by updating the parameters of all layers.

between joint locations. In [109], this limitation was overcome by introducing a substantially more complex, deep architecture for maximum-margin structured learning. Here, we encode dependencies in a simpler, more efficient, and, as evidenced by our experiments, more accurate way by learning a mapping between the output of a CNN and a latent representation obtained using an overcomplete autoencoder, as illustrated in Fig. 4.2. The autoencoder is pre-trained on human poses and comprises a hidden layer of *higher dimension* than its input and output. In effect, this hidden layer and the CNN-based representation of the image play the same role as the kernel embeddings in KDE-based approaches [37, 80, 81], thus allowing us to account for structure within a direct regression framework. Once the mapping between these two high-dimensional embeddings is learned, we further fine-tune the whole network for the final pose estimation task, as depicted at the bottom of Fig. 4.2.

In the remainder of this section, we describe the different stages of our single-frame approach. We then extend this framework to modeling temporal consistency in Section 4.2.

4.1.1 Structured Latent Representations via Autoencoders

We encode the dependencies between human joints by learning a mapping of 3D human pose to a high-dimensional latent space. To this end, we use a denoising autoencoder that can have one or more hidden layers.

Following standard practice [211], given a training set of pose vectors $\{y_i\}$, we add isotropic Gaussian noise to create noisy versions $\{\tilde{y}_i\}$ of these vectors. We then train our autoencoder to take as input a noisy $\tilde{y}_i$ and return a denoised y_i. The behavior of the autoencoder is controlled by the set $\theta_{ae} = (W_{enc,j}, b_{enc,j}, W_{dec,j}, b_{dec,j})_{j=1}^{L}$ of weights and biases for L encoding and decoding layers.

We take the middle layer to be our latent pose representation and denote it by $h_L = g(\tilde{y}, \theta_{ae})$, where $g(\cdot)$ represents the encoding function. For example, with a single layer, the latent representation can be expressed as

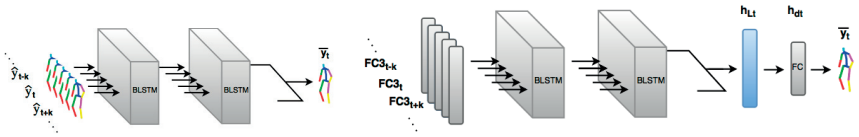
$$h_L = g(\tilde{y}, W_{enc}, b_{enc}) = r(W_{enc}\tilde{y} + b_{enc}) \,, \tag{4.1}$$

where $r(\cdot)$ is the activation function. In practice, we use ReLU as the activation function of the encoding layers. This favors a sparse hidden representation [60], which has been shown to be effective at modeling a wide range of human poses [5, 153]. For the decoding part of the autoencoder, we use a linear activation function to be able to predict both negative and positive joint coordinates. To keep the number of parameters small and reduce overfitting, we use tied weights for the encoder and the decoder, that is, $W_{dec,j} = W_{enc,j}^T$.

To learn the parameters θ_{ae}, we rely on the square loss between the reconstruction, $\hat{y}$, and the true, noise-free pose, y, over the N training examples. To increase robustness to small pose changes, we regularize the cost function by adding the squared Frobenius norm of the Jacobian of the hidden mapping $g(\cdot)$, that is, $J(\tilde{y}) = \frac{\partial g}{\partial \tilde{y}}(\tilde{y})$. Training can thus be expressed as finding

$$\theta_{ae}^* = \underset{\theta_{ae}}{\operatorname{argmin}} \sum_{i=1}^{N} ||y_i - f(\tilde{y}_i, \theta_{ae})||_2^2 + \lambda \|J(\tilde{y}_i)\|_F^2 \,, \tag{4.2}$$

where $f(\cdot)$ represents the complete autoencoder function, and λ is the regularization weight. Unlike when using KDE, we do not need to solve a complex problem to go from the latent pose representation to the pose itself. This mapping, which corresponds to the decoding part of our autoencoder, is learned directly from data.



(a) (b)

Figure 4.3: **Our (B)LSTM networks to enforce temporal consistency.** (a) The (B)LSTM-Pose approach involves refining 3D human pose predictions by feeding those obtained as described in Fig. 4.2(c) into a (B)LSTM network, which yields the final 3D poses. (b) The (B)LSTM-Feature approach maps the features obtained from the last fully-connected layer of a CNN trained to directly regress 3D pose from monocular images to the latent representation h_L of Fig. 4.2(a) via a (B)LSTM network. The final pose is recovered by the decoder part of the autoencoder.

4.1.2 Regression in Latent Space

Once the autoencoder is trained, we aim to learn a mapping from the input image or heatmap to the latent representation of the human pose. To this end, and as shown in Fig. 4.2(b), we use a CNN to regress the image to a high-dimensional representation, which is itself mapped to the latent pose representation.

More specifically, let θ_{cnn} be the parameters of the CNN, including the mapping to the latent pose representation. Given an input image or heatmap x, we consider the square loss between the representation predicted by the CNN, $f_{cnn}(x,\theta_{cnn})$, and the one that was previously learned by the autoencoder, h_L. Given our N training samples, learning amounts to finding

$$\theta^*_{cnn} = \underset{\theta_{cnn}}{\operatorname{argmin}} \sum_{i=1}^{N} ||f_{cnn}(x_i,\theta_{cnn}) - h_{L,i}||_2^2 \,. \tag{4.3}$$

In practice, we either rely on a standard CNN architecture shown in Fig. 4.2(b), similar to the one of [108,201] or a very deep network architecture, e.g. ResNet-50 [66]. In our implementation, the input volume is a three channel image of size 128×128 or a 16 channel heatmap of size 128×128. The last fully-connected layer of the base network is mapped linearly to the latent pose embedding. Except for this last linear layer, each layer uses a ReLU activation function. When we use images as input, we initialize the convolutional layers of our CNN from those of a network trained for the detection of body joints in 2D as in [108, 124].

In the case of 3D pose prediction from 2D joint location heatmaps, we rely on the stacked hourglass network design [132], which assigns high confidence values to most likely joint positions in the image. In practice, we have observed a huge performance improvement in overall 3D pose estimation accuracy when using reliable 2D joint location heatmaps produced by stacked hourglass networks compared to directly using RGB images as input to our standard CNN architecture in Fig. 4.2(b).

4.1.3 Fine-Tuning the Whole Network

Finally, as shown in Fig. 4.2(c), we append the decoding layers of the autoencoder to the CNN discussed above, which maps the latent pose estimates to the original pose space. We then fine-tune the resulting complete network for the task of human pose estimation. We take the cost function to be the squared difference between the predicted and ground-truth 3D poses, which yields the optimization problem

$$\theta_{ft}^{*} = \underset{\theta_{ft}}{\operatorname{argmin}} \sum_{i}^{N} ||f_{ft}(x_i, \theta_{ft}) - y_i||_2^2 , \tag{4.4}$$

where θ_{ft} are the model parameters, including θ_{cnn} and the decoding weights and biases $(W_{dec,j}, b_{dec,j})_{j=1}^{L}$, and f_{ft} is the mapping function.

At test time, a new input image or heatmap is then simply passed forward through this fine-tuned network, which predicts the 3D pose via the learned latent representation.

4.2 Modeling Temporal Consistency

We have so far focused on predicting 3D poses from single images or heatmaps. However, it is well known that accounting for temporal consistency increases robustness. In this section, we show that our approach naturally allows us to use Long Short-Term Memory Units (LSTMs) to this end. Below, we first briefly review LSTMs and then introduce two different ways to exploit them to encode temporal information in our framework.

4.2.1 LSTMs

Recurrent Neural Networks (RNNs) have become increasingly popular to model temporal dynamics. In their simplest form, they map a sequence of inputs to a sequence of hidden states, each connected to its temporal neighbors, which are in turn mapped to a sequence of outputs. In theory, simple memory units and backpropagation through time (BPTT) allow RNNs to capture the temporal correlations between distant data points. However, in practice, longer sequences often cause the gradients to either vanish or explode, thus making optimization impossible. LSTMs [72] were introduced as a solution to this problem. Although they have four times as many parameters as traditional RNNs, they can be trained efficiently thanks to their sharing of parameters across

time slices. An LSTM unit is defined by the recurrence equations

$$\begin{aligned} i_t &= \sigma_i(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ f_t &= \sigma_f(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\ o_t &= \sigma_o(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \sigma_c(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ h_t &= o_t \odot \sigma_h(c_t)\,, \end{aligned} \tag{4.5}$$

where x_t, c_t and h_t are the input, hidden/cell state and output at time t, respectively, and i_t, f_t and o_t represent gate vectors to forget/select information. $\sigma_.(\cdot)$ are sigmoids and $\odot$ denotes the Hadamard or element-wise product.

In practice, we use either LSTMs or Bidirectional LSTMs (BLSTMs). A BLSTM comprises two LSTMs with information traveling in opposite temporal directions [61]. They have been shown to boost performance when the quantity to be predicted depends on contextual information coming from both forward and backward in time [61]. This is typically the case for human pose estimation, where the estimate at time t is correlated to those at time $t-1$ and $t+1$.

4.2.2 Recurrent Pose Estimation

We tested two different ways to incorporate (B)LSTMs into our framework.

4.2.2.1 Constraining the Final Poses

The first is to refine the pose estimates by imposing temporal consistency on the output of the network introduced in the previous section, as shown in Fig. 4.3(a).

More specifically, let $S_t = [\hat{y}_{t-\frac{T}{2}+1}, \ldots, \hat{y}_t, \ldots, \hat{y}_{t+\frac{T}{2}}]$ be the input sequence of T predicted poses centered at time t. The network prediction can be expressed as

$$\bar{y}_t = f_p(S_t, \theta_p)\,, \tag{4.6}$$

where θ_p includes all the parameters of the network. During training, these parameters are taken to be

$$\theta_p^* = \underset{\theta_p}{\operatorname{argmin}} \sum_{t=T/2}^{N-T/2} ||f_p(S_t, \theta_p) - y_t||_2^2\,. \tag{4.7}$$

We refer to this method as *(B)LSTM-Pose*.

4.2.2.2 Constraining the Features

An alternative would be to enforce temporal consistency not on the poses, but earlier in the network on the features extracted from a direct CNN regressor. To this end, we made use of the features of the penultimate layer of our base network. This, for example, corresponds to FC3 features for the network shown in Fig. 4.2(b). These features act as input to the model depicted in Fig. 4.3(b), which stacks two BLSTM layers and maps the features to the latent representation learned by the autoencoder of Section 4.1.1. This is followed by the decoder to finally predict 3D poses.

Let $F_t = [\mathrm{FC}_{t-T/2+1}, \ldots, \mathrm{FC}_t, \ldots, \mathrm{FC}_{t+T/2}]$ be the sequence of such features. Then, training this network can be achieved by solving the problem

$$\theta_f^* = \underset{\theta_f}{\operatorname{argmin}} \sum_{t=T/2}^{N-T/2} ||f_f(F_t, \theta_f) - y_t||_2^2 , \tag{4.8}$$

where $f_f(F_t, \theta_f)$ represents the complete network mapping, with parameters θ_f. We refer to this method as *(B)LSTM-Feature*.

4.3 Results

In this section, we first describe the datasets we tested our approach on. We then give implementation details and describe the evaluation protocol. Finally, we compare our results against those of the state-of-the-art methods.

4.3.1 Datasets

We evaluate our method on the Human3.6m [81], HumanEva [182], KTH Multiview Football II [26] and Leeds Sports Pose (LSP) [86] datasets.

Human3.6m comprises 3.6 million image frames with their corresponding 2D and 3D poses. The subjects perform complex motion scenarios based on typical human activities such as discussion, eating, greeting and walking. The videos were captured from 4 different camera viewpoints. Following the standard procedure of [108], we collect the input images by extracting a square region around the subject using the bounding box present in the dataset and the output pose is a vector of 17 3D joint coordinates.

HumanEva-I comprises synchronized images and motion capture data and is a standard benchmark for 3D human pose estimation. The output pose is a vector of 15 3D joint coordinates.

KTH Multiview Football II is a recent benchmark to evaluate the performance of pose estimation algorithms in unconstrained outdoor settings. The camera follows a soccer player moving around the field. The videos are captured from 3 different camera viewpoints and the output pose is a vector of 14 3D joint coordinates.

LSP is a standard benchmark for 2D human pose estimation and does not contain any ground-truth 3D pose data. The images are captured in unconstrained outdoor settings. 2D pose is represented in terms of a vector of 14 joint coordinates. We report qualitative 3D pose estimation results on this dataset.

4.3.2 Implementation Details

We trained our autoencoder using a greedy layer-wise training scheme followed by fine-tuning as in [71, 211]. We set the regularization weight of Eq. 4.2 to $\lambda = 0.1$. We experimented with single-layer autoencoders, as well as with 2-layer ones. The size of the layers were set to 2000 and 300-300 for the 1-layer and 2-layer cases, respectively. We corrupted the input pose with zero-mean Gaussian noise with standard deviation of 40 for 1-layer and 40-20 for 2-layer autoencoders. In all cases, we used the ADAM optimization procedure [94] with a learning rate of 0.001 and a batch size of 128.

The number and individual sizes of the layers of our base architecture are given in Fig. 4.2. The filter sizes for the convolutional layers are consecutively 9×9, 5×5 and 5×5. Each convolutional layer is followed by a 2×2 max-pooling layer. The activation function is the ReLU in all the layers except for the last one that uses linear activation. As for the autoencoders, we used ADAM [94] with a learning rate of 0.001 and a batch size of 128. To prevent overfitting, we applied dropout with a probability of 0.5 after each fully-connected layer and augmented the data by randomly cropping 112×112 patches from the 128×128 image. When using 2D heatmaps as input, the 64×64 outputs of stacked hourglass network of [132] were upscaled to 128×128 before processing.

To demonstrate that our approach scales to very deep architectures, we also use ResNet-50 [66] as baseline CNN architecture. More specifically, we use it up to level 5, with the first three levels initialized on a 2D pose estimation task as in [124] and then kept constant throughout the 3D pose prediction process. We then use two additional convolutional layers of size 512 and

128 and a linear layer to regress the 3D pose from the convolutional features of level 4.

To train *Ours-LSTM-Feature* and *Ours-BLSTM-Feature*, we relied on the features extracted from the penultimate layer of a CNN trained to directly predict 3D pose, referred to later as *CNN-Direct*. We did not backpropagate the loss of our LSTM-based models through this network, but rather kept its weights fixed. By contrast, *Ours-LSTM-Pose* and *Ours-BLSTM-Pose* take as input the 3D pose predictions obtained using the network in Fig. 4.2(c). In all cases, we cascaded two (B)LSTM layers of size 512, whose output sequence was merged into a single fully-connected layer of size 51. The activation function was $tanh$ for the recurrent layers and linear for the fully-connected layer at the end. In all architectures, we used a temporal window of length $T = 5$ with a stride of 5 covering 0.5 seconds for 50 fps Human3.6m videos. The first $T/2-1$ and the last $T/2$ frames were excluded from the evaluation. We optimized the recurrent networks using the ADAM optimization procedure [94] with a learning rate of 0.001 and a batch size of 128.

4.3.3 Evaluation Protocol

On Human3.6m, for the comparison to be fair, we used the same data partition protocol as in earlier work [108, 109] to obtain the training and test splits. The data from 5 subjects (S1,S5,S6,S7,S8) was used for training and the data from 2 different subjects (S9,S11) was used for testing. We trained a single model for all actions. We evaluate the accuracy of 3D human pose estimation in terms of average Euclidean distance between the predicted and ground-truth 3D joint positions as in [108, 109]. To compare against [17, 171], we further evaluate the pose estimation accuracy after Procrustes transformation. The accuracy numbers are reported in milimeters for all actions. Training and testing were carried out monocularly in all camera views for each separate action.

On HumanEva-I, we trained our model on the Walking sequences of subjects S1, S2 and S3 as in [184, 233] and evaluate on the validation sequences of all subjects. We pretrained our network on the Walking sequences of Human3.6m and used only the first camera view for further training and validation.

On KTH Multiview Football II, we trained our model on the first half of the sequence containing Player 2 and test on the second half, as in [26]. We report accuracy using the percentage of correctly estimated parts (PCP) score with a threshold of 0.5 for a fair comparison. Since the training set is quite small, we pretrained our CNN model on the synthetic dataset introduced in [30], which contains images of sports players with their corresponding 3D poses.

On LSP, in order to generalize to the unconstrained outdoor settings, we trained our regressor

on the recently released synthetic dataset of [30] and tested on the actual data from the LSP dataset.

4.3.4 Evaluation

We first discuss our results on predicting 3D pose from a single image, and then turn to the case where we use multiple consecutive frames as input.

4.3.4.1 Human Pose from a Single Image

Fig. 4.4 depicts selected pose estimation results on Human3.6m. In Table 4.1, we report our single-image autoencoder-based results on this dataset along with those of the following state-of-the-art single image-based methods: KDE regression from HOG features to 3D poses [81], jointly training a 2D body part detector and a 3D pose regressor [108, 137], the maximum-margin structured learning framework of [109, 110], the deep structured prediction approach of [195], pose regression with kinematic constraints [232], pose estimation with mocap guided data augmentation [163], volumetric pose prediction approach of [138] and lifting 2D heatmap predictions to 3D human pose [199]. *ShallowNet-Autoencoder* refers to our autoencoder-based regression approach using the base architecture depicted in Fig. 4.2, and *ResNet-Autoencoder* to the one using the ResNet-50 architecture. For the shallow network architecture, we also evaluate the pose estimation accuracy using the 2D joint location heatmaps of [132] as input. This is referred to as *ShallowNet-Hm-Autoencoder*.

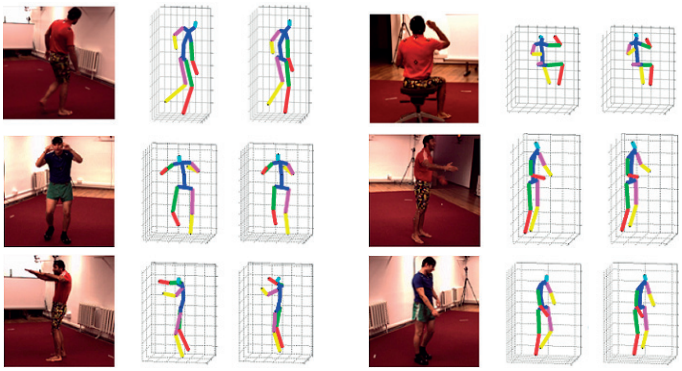


Figure 4.4: Example 3D pose estimation results of our structured prediction approach. Examples are from the *Walking, Eating, Taking Photo, Greeting, Discussion* and *Walking Dog* actions of the Human3.6m database. In each case, the first skeleton depicts the ground-truth pose and the second one the pose we recover. Best viewed in color.

The shallow network architecture provides satisfactory pose estimation accuracy with a fast computational runtime of 6 ms/frame, which corresponds to 166 fps real-time performance, whereas *ResNet-Autoencoder* comes at the cost of a three times slower runtime. Our autoen-

Method	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting	Sitting Down
Ionescu et al. [81]	132.71	183.55	132.37	164.39	162.12	150.61	171.31	151.57	243.03
Li & Chan [108]	-	148.79	104.01	127.17	-	-	-	-	-
Li et al. [109]	-	134.13	97.37	122.33	-	-	-	-	-
Li et al. [110]	-	133.51	97.60	120.41	-	-	-	-	-
Zhou et al. [233]	-	-	-	-	-	-	-	-	-
Rogez & Schmid [163]	-	-	-	-	-	-	-	-	-
Tekin et al. [195]	-	129.06	91.43	121.68	-	-	-	-	-
Park et al. [137]	100.34	116.19	89.96	116.49	115.34	117.57	106.94	137.21	190.82
Zhou et al. [232]	91.83	102.41	96.95	98.75	113.35	90.04	93.84	132.16	158.97
Tome et al. [199]	64.98	73.47	76.82	86.43	86.28	68.93	74.79	110.19	173.91
Pavlakos et al. [138]	67.38	71.95	66.70	69.07	71.95	65.03	68.30	83.66	**96.51**
OURS (ShallowNet-Autoencoder)	94.98	129.06	91.43	121.68	133.54	115.13	133.76	140.78	214.52
OURS (ShallowNet-Hm-Autoencoder)	69.64	93.79	69.02	96.47	103.42	83.36	85.22	116.62	147.57
OURS (ResNet-Autoencoder)	**57.84**	**64.62**	**59.41**	**62.83**	**71.52**	**57.50**	**60.38**	**80.22**	104.14

Method:	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Avg. (6 Actions)	Avg. (All)
Ionescu et al. [81]	162.14	205.94	170.69	96.60	177.13	127.88	159.99	162.14
Li & Chan [108]	-	189.08	-	77.60	146.59	-	132.20	-
Li et al. [109]	-	166.15	-	68.51	132.51	-	120.17	-
Li et al. [110]	-	163.33	-	73.66	135.15	-	121.55	-
Zhou et al. [233]	-	-	-	-	-	-	-	120.99
Rogez & Schmid [163]	-	-	-	-	-	-	-	121.20
Tekin et al. [195]	-	162.17	-	65.75	130.53	-	116.77	-
Park et al. [137]	105.78	149.55	125.12	62.64	131.90	96.18	111.12	117.34
Zhou et al. [232]	106.91	125.22	94.41	79.02	126.04	98.96	104.73	107.26
Tome et al. [199]	84.95	110.67	85.78	71.36	86.26	73.14	84.17	88.39
Pavlakos et al. [138]	71.74	**76.97**	65.83	59.11	74.89	63.24	69.78	71.90
OURS (ShallowNet-Autoencoder)	121.26	162.17	138.2	65.75	130.53	113.34	116.77	127.07
OURS (ShallowNet-Hm-Autoencoder)	87.17	120.50	95.31	55.87	85.69	64.66	86.89	91.62
OURS (ResNet-Autoencoder)	**66.31**	80.50	**61.20**	**52.55**	**69.97**	**60.08**	**61.20**	**67.27**

Table 4.1: **Comparison of our structured prediction approach with state-of-the-art algorithms on Human3.6m.** We report 3D joint position errors in mm, computed as the average Euclidean distance between the ground-truth and predicted joint positions. '-' indicates that the results were not reported for the respective action class in the original paper. Note that our method achieves the best overall accuracy.

Model	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting
Bogo et al. [17]	62.0	60.2	67.8	76.5	92.1	73.0	75.3	100.3
Sanzari et al. [171]	48.82	56.31	95.98	84.78	96.47	66.30	107.41	116.89
OURS (ResNet-Autoencoder)	**43.89**	**48.54**	**46.57**	**49.95**	**53.94**	**43.77**	**43.94**	**60.20**

Model	Sitting Down	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Average
Bogo et al. [17]	137.3	83.4	77.0	77.3	86.8	79.7	81.7	82.3
Sanzari et al. [171]	129.63	97.84	105.58	65.94	92.58	130.46	102.21	93.15
OURS (ResNet-Autoencoder)	**73.64**	**51.15**	**59.29**	**46.30**	**39.81**	**52.25**	**47.18**	**50.69**

Table 4.2: **Comparison of our structured prediction approach with state-of-the-art algorithms after Procrustes transformation on Human3.6m.** The error is given as the average Euclidean distance in mm between the ground-truth 3D joint locations and predictions.

coder-based regression approach using ResNet-50 as base network outperforms all the baselines.

In [17], the reconstruction error was evaluated by first aligning the estimated skeleton to the ground-truth one by Procrustes transformation, and we confirmed through personal communication that the same protocol was used in [171]. To compare our results to those of these state-of-the-art methods, we therefore also report in Table 4.2 the joint error after Procrustes transformation. Altogether, by leveraging the power of deep neural networks and accounting for the dependencies between body parts, *ResNet-Autoencoder* significantly outperforms the state-of-the-art.

We further evaluated our approach on the official test set of Human3.6m for two different actions. We obtained a pose reconstruction error of 64.38 and 63.86 mm for the *Directions* and *Discussion* actions, respectively. Our method currently ranks second in the leaderboard for these two actions. Note that the first ranking method [150] relies on the knowledge of body part segmentations whereas we do not use this additional piece of ground-truth information.

To validate our design choices, we report in Table 4.3, the pose estimation accuracies obtained with various autoencoder configurations using the shallow network depicted in Fig. 4.2. The results reported in Tables 4.1 and 4.2 were obtained using a two layer autoencoder. However, as discussed in Section 4.1.1 our formalism applies to autoencoders of any depth. Therefore, in Table 4.3(a), we also report results obtained using a single layer one obtained by turning off the final fine-tuning of Section 4.1.3. For completeness, we also report results obtained by using a CNN similar to the one of Fig. 4.2(b) to regress directly to a 51-dimensional 3D pose vector *without* using an autoencoder at all. We will refer to it as *CNN-Direct*. We found that both kinds of autoencoders perform similarly and better than *CNN-Direct*, especially for actions such as *Taking Photo* and *Walking Dog* that involve interactions with the environment and are thus physically more constrained. This confirms that the power of our method comes from autoencoding. Furthermore, as expected, fine-tuning consistently improves the results.

During fine-tuning, our complete network has more fully-connected layers than *CNN-Direct*. One could therefore argue that the additional layers are the reason why our approach outperforms it. To disprove this, we evaluated the baseline, *CNN-ExtraFC*, in which we simply add one more fully-connected layer. We also evaluated another baseline, *CNN-PCA*, in which we replace our autoencoder latent representation by a PCA-based one. In Table 4.3(b), we show that our approach significantly outperforms these two baselines on the *Taking Photo* action. This suggests that our overcomplete autoencoder yields a representation that is more discriminative than other latent ones. Among the different PCA configurations, the one with 40 dimensions performs the best. However, training an autoencoder with 40 dimensions outperforms it.

Model	Discussion	Eating	Greeting	Taking Photo	Walking	Walking Dog
CNN-Direct	135.36	105.98	133.35	177.62	77.73	153.02
OURS-Autoencoder, 1 layer no FT	134.02	96.01	127.58	158.73	68.55	146.28
OURS-Autoencoder, 2 layer no FT	129.67	98.57	124.80	162.69	73.47	146.46
OURS-Autoencoder, 1 layer with FT	130.07	94.08	121.96	**158.51**	65.83	135.35
OURS-Autoencoder, 2 layer with FT	**129.06**	**91.43**	**121.68**	162.17	**65.75**	**130.53**

(a)

Model	Joint error
CNN-Direct	177.62
CNN-ExtraFC[2000]	179.29
CNN-PCA[30]	170.74
CNN-PCA[40]	167.62
CNN-PCA[51]	182.64
OURS-Autoencoder[40]	165.11
OURS-Autoencoder[2000]	**158.51**

(b)

Table 4.3: **Ablation studies for our structured prediction approach.** Average Euclidean distance in mm between the ground-truth 3D joint locations and those computed **(a)** using either no autoencoder at all (CNN) or 1-layer and 2-layer encoders (OURS-Autoencoder), with or without fine-tuning (FT), **(b)** by replacing the autoencoder by either an additional fully-connected layer (*CNN-ExtraFC*) or a PCA layer (*CNN-PCA*) on the sequences of *Taking Photo* action class. The bracketed numbers denote the various dimensions of the additional layer we tested. Our approach again yields the most accurate predictions.

To learn a more powerful latent pose space, we exploit additional motion capture data from the MPI-INF-3DHP dataset [124] for training the autoencoder. In Table 4.4, we report results with and without this additional data. We achieve better pose estimation accuracy when we train on a wider range of poses. As Human3.6m already includes a large variety of poses and the marker placements between the two datasets do not exactly match each other, we only observe a slight improvement. However, our results suggest that training an autoencoder on a larger pose space without any dataset bias would result in an even more representative latent pose space and, eventually, a higher pose estimation accuracy. We further compare our autoencoder-based regression approach to a direct regression baseline. The relative contribution of the autoencoder on very deep neural networks is smaller than that on a shallower network. However, we still increase the accuracy by applying our autoencoder training on top of the ResNet architecture.

Model	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting
ResNet	56.77	64.73	60.94	63.49	74.98	57.65	61.08	81.29
ResNet-Autoencoder w/o ExtraMocap	57.84	64.62	59.41	62.83	**71.52**	57.50	60.38	**80.22**
ResNet-Autoencoder w/ ExtraMoCap	**55.87**	**63.65**	**59.08**	**62.64**	72.08	**56.15**	**58.88**	80.53

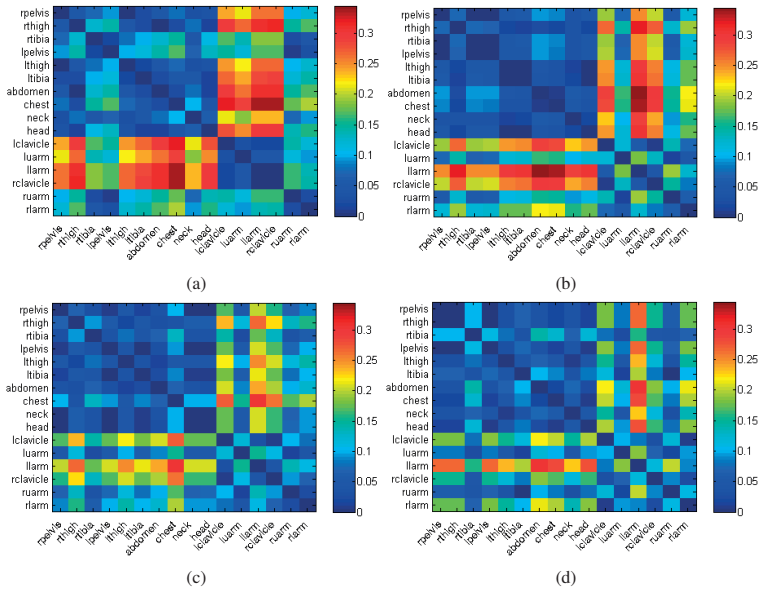
Model	Sitting Down	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Average
ResNet	102.45	66.65	80.96	60.87	53.26	70.27	60.95	68.29
ResNet-Autoencoder w/o ExtraMoCap	104.14	66.31	80.50	61.20	52.55	69.97	60.08	67.27
ResNet-Autoencoder w/ ExtraMoCap	**102.30**	**65.68**	**78.25**	**59.05**	**51.81**	**68.44**	**58.19**	**66.17**

Table 4.4: **Evaluation of our structured prediction approach with very deep network architectures.** Average Euclidean distance in mm between the ground-truth and predicted 3D joint locations of a direct ResNet regressor, ResNet-Autoencoder trained with only motion capture data from Human3.6m and ResNet-Autoencoder trained with motion capture data from Human3.6m and MPI-INF-3DHP.

Following [80], we show in Fig. 4.5 the differences between the ground-truth limb ratios and the limb ratios obtained from predictions based on KDE, *CNN-Direct* and our autoencoder-based approach. These results demonstrate that our predictions better preserve these limb ratios, and

thus better model the dependencies between joints.

In Fig. 4.6, we visualize the latent space learned by the autoencoder after embedding it in 2D using the t-SNE algorithm [120]. It can be seen that the upper left corner spans the downward-facing body poses, the diagonal includes mostly the upright body poses and the lower right corner clusters the forward-facing body poses sitting on the ground. Note that our latent representation covers the entire low-dimensional space, thus making it well-suited to discriminate between poses with small variations.



(a) (b)

(c) (d)

Model	Lower Body	Upper Body	Full Body
KDE [81]	1.02	7.18	16.43
CNN	0.57	6.86	14.97
OURS-Autoencoder no FT	0.62	5.30	11.99
OURS-Autoencoder with FT	0.77	5.43	11.90

(e)

Figure 4.5: **Analysis on structure preservation ability of our approach.** Matrix of differences between estimated log of limb length ratios and those computed from ground-truth poses. The rows and columns correspond to individual limbs. For each cell, the ratios are computed by dividing the limb length in the horizontal axis by the one in the vertical axis as in [80] for **(a)** KDE [81], **(b)** CNN-Direct as in Table 4.3, and **(c,d)** our method without and with fine-tuning. An ideal result would be one in which all cells are blue, meaning the limb length ratios are perfectly preserved. Best viewed in color. **(e)** Sum of the log of limb length ratio errors for different parts of the human body. All methods perform well on the lower body. However, ours outperforms the others on the upper body and when considering all ratios in the full body.

We further report single-image 3D pose estimation accuracy on the HumanEva-I dataset in Table 4.5 and show qualitative pose estimation results in Fig. 4.7. We follow the protocol adopted in the state-of-the-art approaches to 3D inference from 2D body part detections [184] and to

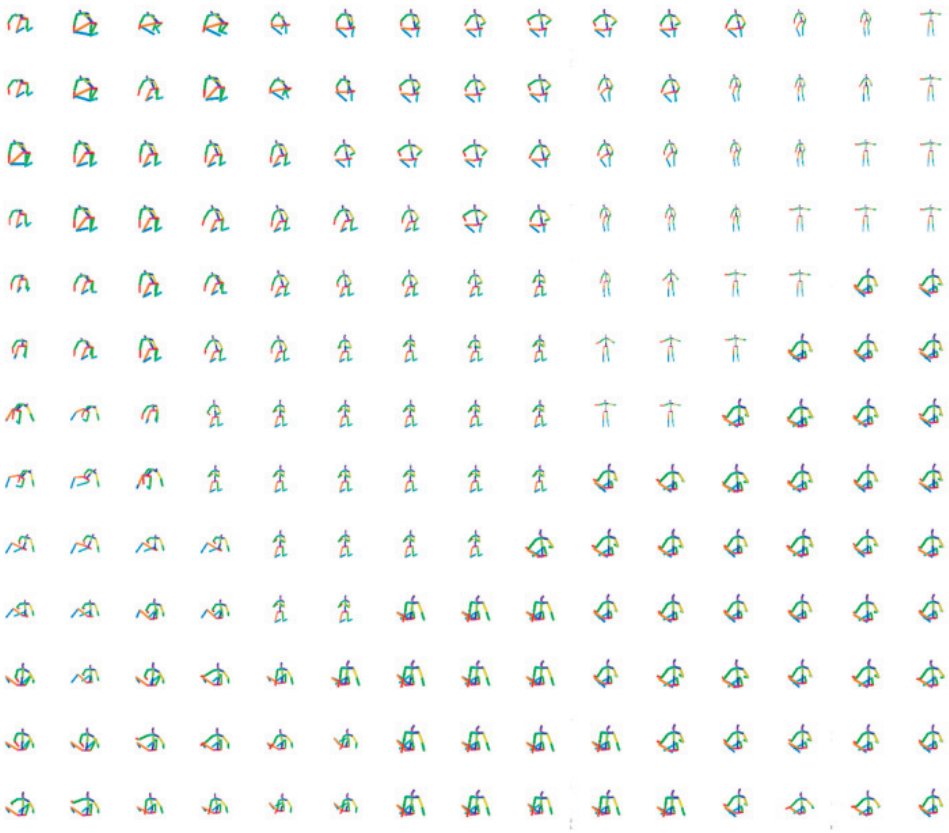


Figure 4.6: **Visualization of the learned latent pose space.** t-SNE embedding [120] for the latent representation of the poses from the *Sitting Down* category in Human3.6m.

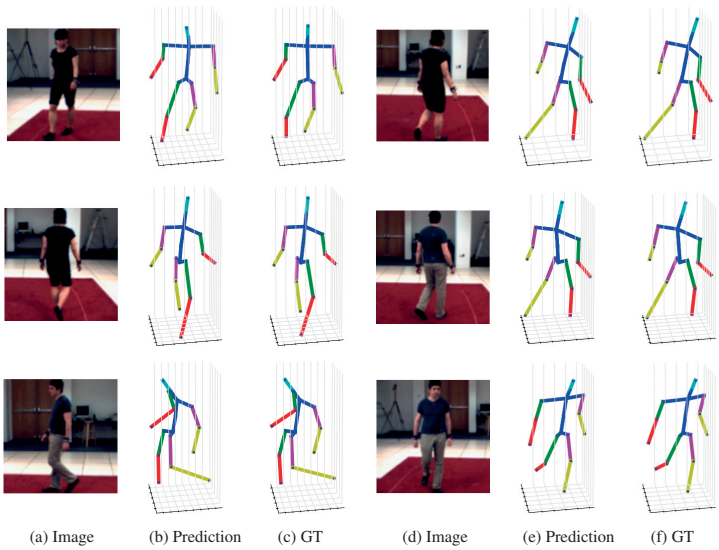


Figure 4.7: **Pose estimation results of our structured prediction approach on HumanEva-I. (a,d)** Input images. **(b,e)** Recovered pose. **(c,f)** Ground truth. Best viewed in color.

3D model-fitting [17, 233]. Following these methods, we measure 3D pose error after aligning the prediction to the ground-truth by a rigid transformation. Note that [233] uses video instead of a single frame for prediction. Our method outperforms the state-of-the-art on this standard benchmark.

Method	S1	S2	S3	Average
Simo-Serra et al. [184]	65.1	48.6	73.5	62.4
Bogo et al. [17]	73.3	59.0	99.4	77.2
Zhou et al. [233]	34.2	30.9	**49.1**	38.07
OURS-Autoencoder	**29.32**	**17.94**	59.51	**35.59**

Table 4.5: **Quantitative results of our structured prediction approach on Walking sequences of the HumanEva-I dataset** [182]. S1, S2 and S3 correspond to Subject 1, 2, and 3, respectively. The accuracy is reported in terms of average Euclidean distance (in mm) between the predicted and ground-truth 3D joint positions.

On the KTH Multiview Football II dataset, we compare our autoencoder-based approach against [26], which is the only monocular single-image 3D pose estimation method publishing results on this dataset so far. As can be seen in Table 4.6, we outperform the PCP accuracy of this baseline significantly on all body parts except for the pelvis. Fig. 4.8 depicts example pose estimation results on this dataset.

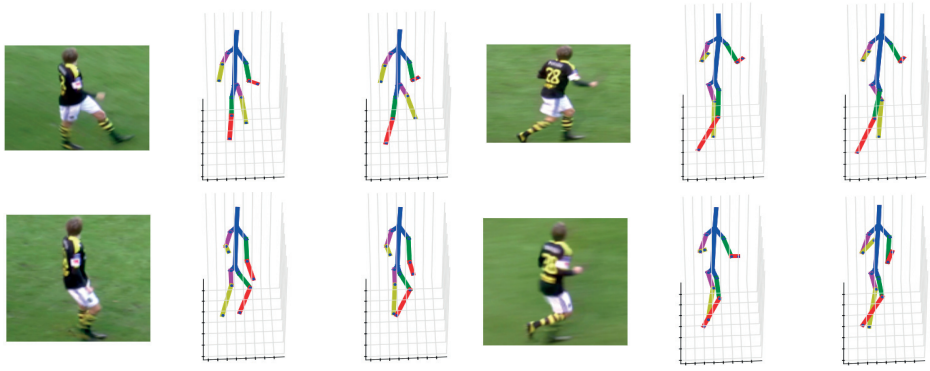
Method:	Pelvis	Torso	Upper Arms	Lower arms	Upper Legs	Lower Legs	All parts
[26]	**97**	87	14	6	63	41	43
OURS-Autoencoder	66	**100**	**66.5**	**16.5**	**83**	**66.5**	**63.1**

Table 4.6: **Evaluation of our structured prediction approach on KTH Multiview Football II.** On this dataset we compare our method that uses a single image to that of [26]. We rely on the percentage of correctly estimated parts (PCP) score to evaluate performance as in [26]. Higher PCP score corresponds to better 3D pose estimation accuracy.

In Fig. 5.11, we provide additional qualitative results on the LSP dataset, which features challenging poses. Our autoencoder-based regression approach nevertheless delivers accurate 3D predictions.

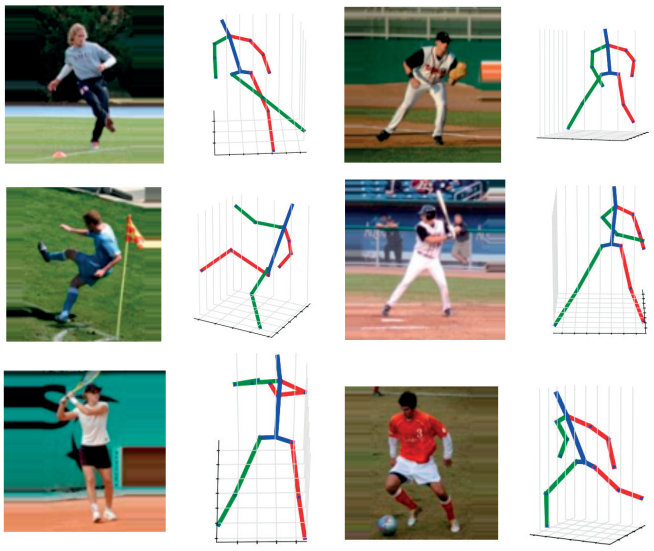
4.3.4.2 Human Pose from Video

In Table 4.7, we demonstrate the effectiveness of imposing temporal consistency using LSTMs on Human3.6m, as described in Section 4.2. We compare our results with and without LSTMs against those of [44, 197, 233], which also rely on video sequences. On average, our LSTM-based approaches applied to the 3D pose predictions of *ResNet-Autoencoder* bring an improvement over single-image results, with the one of Section 4.2.2.2 that enforces temporal consistency at pose level being significantly better than the other. Using standard LSTMs instead of BLSTMs



(a) Image (b) Prediction (c) GT (d) Image (e) Prediction (f) GT

Figure 4.8: **Pose estimation results of our structured prediction approach on KTH Multi-view Football II**. **(a, d)** Input images. **(b, e)** Recovered pose. **(c, f)** Ground truth. Best viewed in color.



(a) Image (b) Prediction (c) Image (d) Prediction

Figure 4.9: **Pose estimation results of our structured prediction approach on LSP**. **(a,c)** Input images. **(b,d)** Recovered pose. We trained our network on the recently released synthetic dataset of [30] and tested it on the LSP dataset. The quality of the 3D pose predictions demonstrates the generalization of our method. In the last row, we show failure cases in the 3D pose prediction of lower legs due to foreshortening (left) and orientation ambiguities (right).

degrades the accuracy but eliminates the latency involved in working on image batches, which can be a worthwhile trade-off if real-time performance is required.

Model	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting
OURS (ResNet-Autoencoder)	57.84	64.62	59.41	62.83	71.52	57.50	60.38	80.22
OURS-LSTM-Pose	55.63	64.55	57.56	62.20	70.71	56.52	57.37	78.93
OURS-BLSTM-Pose	54.93	63.26	57.26	62.30	70.28	56.66	57.08	78.98
OURS-LSTM-Feature	71.34	68.88	67.12	75.87	79.36	66.19	61.49	83.28
OURS-BLSTM-Feature	70.01	68.74	64.64	75.90	78.99	64.21	60.50	83.10

Model	Sitting Down	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Average
OURS (ResNet-Autoencoder)	104.14	66.31	80.50	61.20	52.55	69.97	60.08	67.27
OURS-LSTM-Pose	98.47	64.43	77.18	62.32	50.12	67.50	66.77	66.02
OURS-BLSTM-Pose	97.13	64.29	77.40	61.94	49.76	67.11	62.26	65.37
OURS-LSTM-Feature	97.66	71.51	83.93	78.67	63.69	73.23	69.03	74.08
OURS-BLSTM-Feature	96.44	70.29	83.51	77.83	62.02	71.11	69.55	73.52

Table 4.7: **Analysis of our different (B)LSTM architectures.** Error is given in average Euclidean distance in mm between the ground-truth 3D joint locations and the predictions obtained by our ResNet-Autoencoder approach evaluated using different LSTM architectures on video data.

As shown in Table 4.8, our LSTM units improves the pose estimation accuracy on average by approximately 3% and our ResNet-based results are significantly more accurate than the other methods, with an average pose estimation accuracy of 65.37 mm vs 124.97 mm for [197], 113.01 mm for [233] and 126.47 mm for [44]. Fig. 4.10 depicts example pose estimation results of our BLSTM approach compared to our autoencoder-based approach based on a single image.

We further compare our *OURS-BLSTM-Pose* model with a network where the BLSTM was replaced by two fully-connected layers, thus giving it a similar capacity as the BLSTM one, but not explicitly modeling temporal consistency. This model gives an average pose estimation accuracy on all Human3.6m actions of 77.96 mm, whereas our BLSTM-based model achieves 65.37 mm. Our method significantly outperforms this baseline, thus showing that the better performance of our LSTM-based networks does not just come from their larger number of parameters, but truly from their ability to model temporal information.

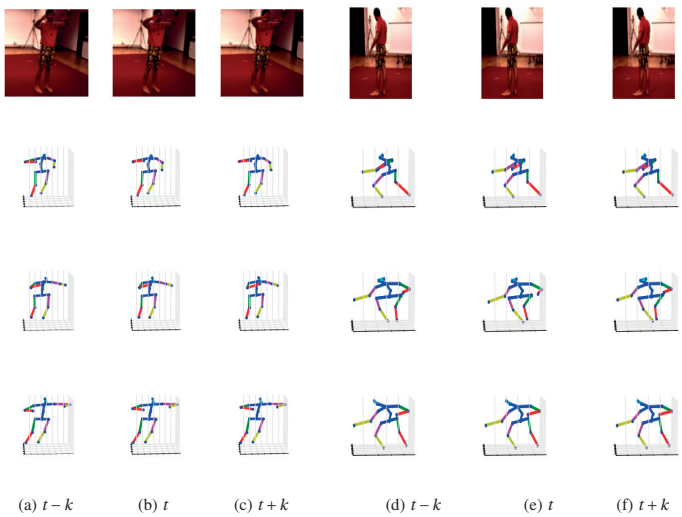
4.3.5 Comparison Between KDE and Autoencoders

In Table 4.9, we compare two structured 3D human pose estimation methods: Our autoencoder-based deep network approach and kernel dependency estimation (KDE) [80, 81]. In the earlier works of [80] and [81], KDE is applied to handcrafted HOG features, whereas in our approach we rely on deep features. In order to compare the structured regression performance of KDE to our autoencoder-based approach, we also applied KDE to the deep features extracted from a CNN. We extract either the features from the last convolutional layer (Conv3) or the last fully-connected layer (FC3) of the network depicted in Fig. 4.2(b). As can be seen in Table 4.9, we consistently

Model	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting
Du et al. [44]	85.07	112.68	104.90	122.05	139.08	105.93	166.16	117.49
Tekin et al. [197]	102.41	147.72	88.83	125.28	118.02	112.3	129.17	138.89
Zhou et al. [233]	87.36	109.31	87.05	103.16	116.18	106.88	99.78	124.52
OURS (ResNet-Autoencoder)	57.84	64.62	59.41	62.83	71.52	57.50	60.38	80.22
OURS-BLSTM-Pose	**54.93**	**63.26**	**57.26**	**62.30**	**70.28**	**56.66**	**57.08**	**78.98**

Model	Sitting Down	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Average
Du et al. [44]	226.04	120.02	135.91	117.65	99.26	137.36	106.54	126.47
Tekin et al. [197]	224.90	118.42	182.73	138.75	55.07	126.29	65.76	124.97
Zhou et al. [233]	199.23	107.42	143.32	118.09	79.39	114.23	97.70	113.01
OURS (ResNet-Autoencoder)	104.14	66.31	80.50	**61.20**	52.55	69.97	**60.08**	67.27
OURS-BLSTM-Pose	**97.13**	**64.29**	**77.40**	61.94	**49.76**	**67.11**	62.26	**65.37**

Table 4.8: **Comparison of our (B)LSTM-based architectures to the state of the art**. Error is given average Euclidean distance in mm between the ground-truth 3D joint locations and the predictions obtained by our ResNet-Autoencoder approach with and without BLSTM regularization on output poses, compared to [44, 197, 233].



(a) $t-k$ (b) t (c) $t+k$ (d) $t-k$ (e) t (f) $t+k$

Figure 4.10: **Pose estimation results with LSTMs on Human3.6m. (a,d)** $t-k^{th}$ frame. **(b,e)** t^{th} frame. **(c,g)** $t+k^{th}$ frame. k denotes the stride between consecutive frames. Top row: Input image, Second row: Our pose estimate from the single image, Third row: Our BLSTM pose estimate, Last row: Ground truth. Our BLSTM network can correct for the errors made by the autoencoder by accounting for the temporal consistency. Best viewed in color.

outperform all the baselines, which demonstrates the power of autoencoding.

Model	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting
HOG + KDE [81]	132.71	183.55	132.37	164.39	162.12	150.61	171.31	151.57
Conv3 Feat. + KDE	99.13	160.84	112.10	137.32	137.97	118.16	137.13	153.79
FC3 Feat. + KDE	99.06	160.39	104.53	132.01	132.35	118.13	144.36	149.80
CNN-Direct	106.23	161.54	108.42	136.15	136.21	123.37	148.68	157.15
OURS-Autoencoder	**94.98**	**129.06**	**91.43**	**121.68**	**133.54**	**115.13**	**133.76**	**140.78**

Model	Sitting Down	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Average
HOG + KDE [81]	243.03	162.14	205.94	170.69	96.60	177.13	127.88	162.14
Conv3 Feat. + KDE	190.48	137.06	181.77	151.15	93.97	149.81	120.46	138.74
FC3 Feat. + KDE	206.35	133.91	169.31	150.76	86.44	144.83	113.20	136.36
CNN-Direct	217.88	136.59	169.42	157.71	88.75	149.58	115.02	140.85
OURS-Autoencoder	**214.52**	**121.26**	**162.17**	**138.2**	**65.75**	**130.53**	**113.34**	**127.07**

Table 4.9: **Comparison of our structured prediction approach to KDE.** Error is given in average Euclidean distance in mm between the ground-truth 3D joint locations and those predicted by competing methods [81] and ours.

Layer Configuration	Greeting
[40]	129.49
[500]	123.95
[1000]	121.96
[2000]	**121.96**
[3000]	123.49
[250-250]	125.61
[300-300]	**121.68**
[250-500]	128.98
[500-1000]	126.52
[200-200-200]	126.78
[500-500-500]	127.73

Table 4.10: **Analysis on the parameter choices for our structured prediction approach.** Error is given in average Euclidean distance in mm between the ground-truth 3D joint locations and the ones predicted by our approach trained using autoencoders in various configurations, with different number of layers and number of channels per layer as indicated by the bracketed numbers. This validation was performed on the *Greeting* action and the optimal values used for all other actions.

4.3.6 Parameter Choices

In Table 4.10, we compare the results of different autoencoder configurations in terms of number of layers and channels per layer on the *Greeting* action. Similarly to what we did in Table 4.3(b), the bracketed numbers denote the dimension of the autoencoder's hidden layers. We obtained the best result for 1 layer with 2000 channels or 2 layers with 300-300 channels. These values are those we used for all the experiments described above. They were chosen for a single action and used unchanged for all others, thus demonstrating the versatility of our approach.

4.4 Conclusion

We have introduced a novel Deep Learning regression architecture for structured prediction of 3D human pose from a monocular image or a 2D joint location heatmap. We have shown that our approach to combining autoencoders with CNNs accounts for the dependencies between the human body parts efficiently and significantly improves accuracy. We have also shown that accounting for the temporal information with LSTMs further increases the accuracy of our pose estimates. Since our framework is generic, in future work, we intend to apply it to other structured prediction problems, such as deformable surface reconstruction.

5 Learning to Fuse 2D and 3D Image Cues for Monocular Body Pose Estimation

Monocular 3D human pose estimation is a longstanding problem of Computer Vision. Over the years, two main classes of approaches have been proposed: Discriminative ones that directly regress 3D pose from image data [1, 16, 89, 138, 166, 203] and generative ones that search the pose space for a plausible body configuration that aligns with the image data [54, 177, 204]. With the advent of ever larger datasets [81], models have evolved towards deep architectures, but the story remains largely unchanged. The state-of-the-art approaches can be roughly grouped into those that directly regress 3D pose from images [81, 108, 195, 197] and those that first predict a 2D pose in the form of joint location confidence maps and fit a 3D model to this 2D prediction [17, 233].

Since detecting the 2D image location of joints is easier than directly inferring the 3D pose, it can be done more reliably. However, inferring a 3D pose from these 2D locations is fraught with ambiguities and the above-mentioned methods usually rely on a database of 3D models to resolve them, at the cost of a potentially expensive run-time fitting procedure. By contrast, the methods that regress directly to 3D avoid this extra step but also do not benefit of the well-posedness of the 2D joint detection location problem.

In this chapter, we propose the novel architecture depicted by Fig. 5.1 designed to deliver the best of both worlds. The first stream, which we will refer to as the *Confidence Map Stream*, first computes a heatmap of 2D joint locations and then infer the 3D poses from it. The second stream, which we will dub the *Image Stream*, is designed to produce features that complement those computed by the first stream and can be used in conjunction with them to compute the 3D pose, that is, guide the regression process given the 2D locations.

However, for this approach to be beneficial, effective fusion of the two streams is crucial. In theory, it could happen at any stage of the two streams, ranging from early to late fusion, with no

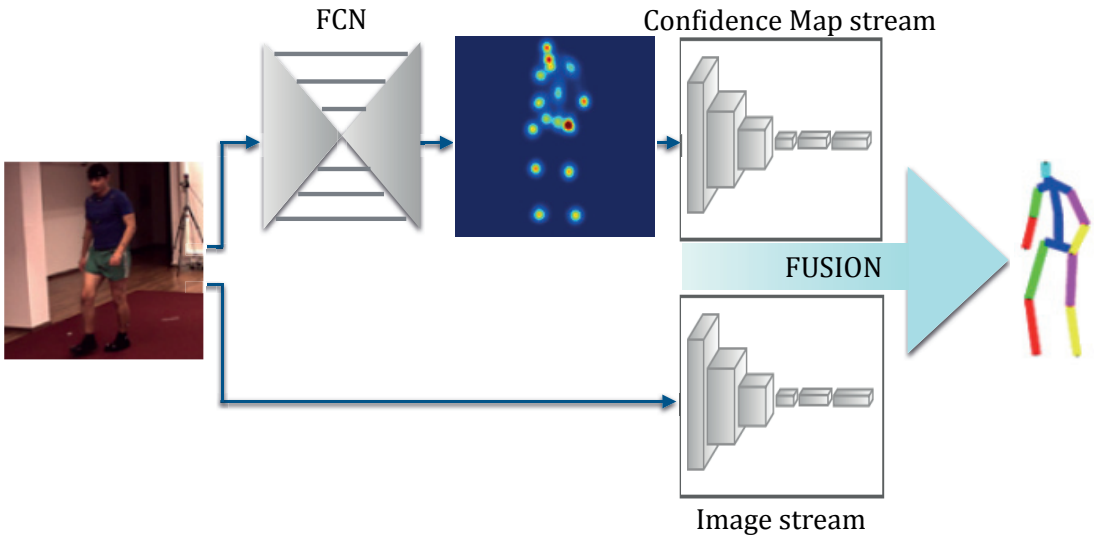


Figure 5.1: **Overview of our fusion approach for 3D human pose estimation.** One stream of our network accounts for the 2D joint locations and the corresponding uncertainties. The second one leverages all 3D image cues by directly acting on the image. The outputs of these two streams are then fused to obtain the final 3D human pose estimate.

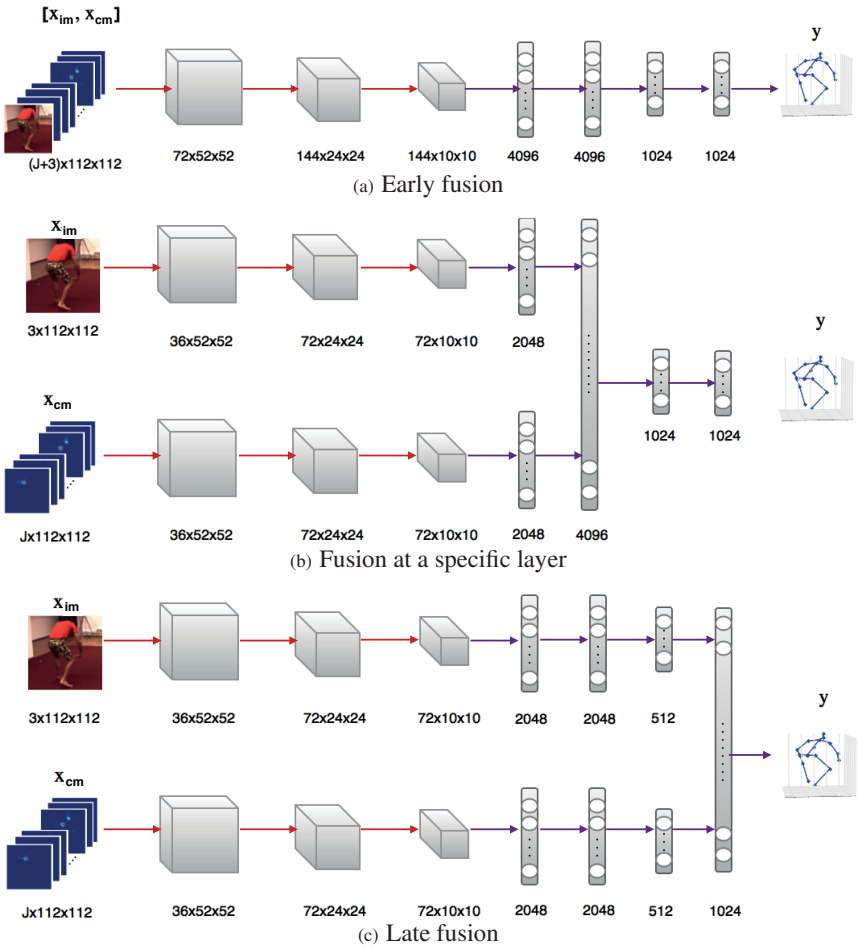
principled way to choose one against the other. We therefore also developed a *trainable fusion* scheme that learns how to fuse the two streams.

Ultimately, our approach allows the network to still exploit image cues while inferring 3D poses from 2D joint locations. As we demonstrate in our experiments, the features computed by both streams are decorrelated and therefore truly encode complementary information. Our contributions can be summarized as follows:

- We introduce a discriminative fusion framework to simultaneously exploit 2D joint location confidence maps and 3D image cues for 3D human pose estimation.

- We introduce a novel trainable fusion scheme, which automatically learns where and how to fuse these two sources of information.

We show that our approach significantly outperforms the state-of-the-art results on standard benchmarks and yields accurate pose estimates from images acquired in unconstrained outdoors environments.



(a) Early fusion

(b) Fusion at a specific layer

(c) Late fusion

Figure 5.2: **Three different instances of hard-coded fusion for 3D human pose estimation.** The fusion strategies combine 2D joint location confidence maps with 3D cues directly extracted from the input image.

5.1 Approach

Our goal is to increase the robustness and accuracy of monocular 3D pose estimation by exploiting image cues to the full while also taking advantage of the fact that 2D joint locations can be reliably detected by modern CNN architectures. To this end, we designed the two stream architecture depicted by Fig. 5.1. The Confidence Map Stream shown at the top first computes a heatmap of 2D joint locations from which feature maps can be computed. The Image Stream shown at the bottom extracts additional features directly from the image and all these features are fused to produce a final 3D pose vector.

As shown in Fig. 5.2, there is a whole range of ways to perform the fusion of these two data streams, ranging from early to late fusion with no obvious way to choose the best, which might

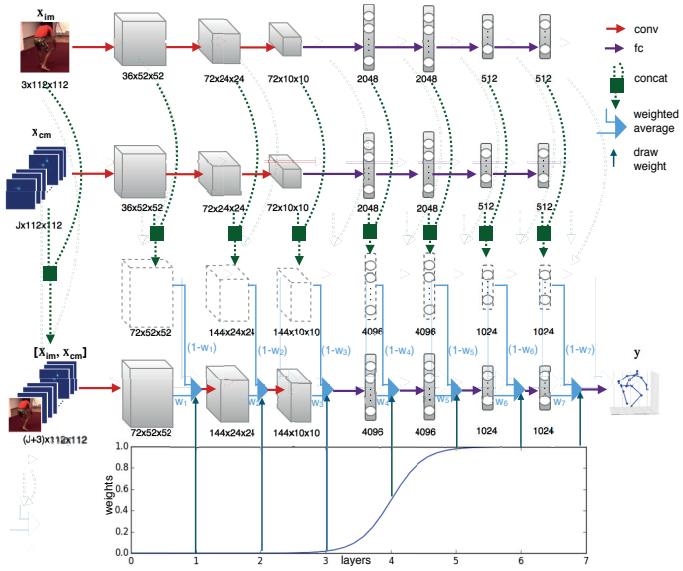


Figure 5.3: **Trainable fusion architecture.** The first two streams take as input the image and 2D joint location confidence maps, respectively. The combined feature maps of the image and confidence map stream are fed into the fusion stream and linearly combined with the outputs of the previous fusion layer. The linear combination of the streams is controlled by a weight vector shown at the bottom part of the figure. The numbers below each layer represent the corresponding size of the feature maps for convolutional layers and the number of neurons for fully connected ones.

well be problem-dependent anyway. To solve this conundrum, we rely on the fusion architecture depicted by Fig. 5.3, which involves introducing a third *fusion stream* that combines the feature maps produced by the two data streams in a trainable way. Each layer of the fusion stream acts on a linear combination of the previous fusion layer with the concatenation of the two data stream outputs. In effect, different weight values for these linear combinations correspond to different fusion strategies.

In the remainder of this section, we formalize this generic architecture and study different ways to set these weights, including learning them along with the weights of the data streams, which is the approach we advocate.

5.1.1 Fusion Network

Let $\{\mathbf{I}_l\}_{l=0}^{L}$ be the feature maps of the *image stream* and $\{\mathbf{X}_l\}_{l=0}^{L}$ be the feature maps of the *confidence map stream*. As special cases, $\mathbf{I}_0 : [1,3] \times [1,H] \times [1,W] \rightarrow [0,1]$ is the input RGB image, and $\mathbf{X}_0 : [1,J] \times [1,H] \times [1,W] \rightarrow_+$ are the confidence maps encoding the probability of observing each one of J body joints at any given image location. The feature maps $\mathbf{I}_l$ and $\mathbf{X}_l$ at each layer l must coincide in width and height but can have different number of channels. In the

following, we denote each feature map at level l as both the output of layer l and the input to layer $l+1$.

Let $\{\mathbf{Z}_l\}_{l=0}^{L+1}$ be the feature maps of the *fusion stream*. The feature map $\mathbf{Z}_l$ is the output of layer l, but, unlike in the data streams, the input to layer $l+1$ is a linear combination of $\mathbf{Z}_l$ with $\mathbf{I}_l$ and $\mathbf{X}_l$ given by

$$(1 - w_l) \cdot \text{concat}(\mathbf{I}_l, \mathbf{X}_l) + w_l \cdot \mathbf{Z}_l, \quad 1 \leq l \leq L, \tag{5.1}$$

where $\text{concat}(\cdot,\cdot)$ is the concatenation of the given feature maps along the channel axis, and w_l is the l-th element of the fusion weights $\mathbf{w} \in [0,1]^L$ controlling the mixture. For this mixture to be possible, $\mathbf{Z}_l$ must have the same size as $\mathbf{I}_l$ and $\mathbf{X}_l$ and a number of channels equal to the sum of the number of channels of $\mathbf{I}_l$ and $\mathbf{X}_l$. As special cases, $\mathbf{Z}_0 = \text{concat}(\mathbf{I}_0, \mathbf{X}_0)$, and $\mathbf{Z}_{L+1} \in \mathbb{R}^{3J}$ is the output of the network, that is, the J predicted 3D joint locations.

In essence, the fusion weights $\mathbf{w}$ control where and how the fusion of the data streams occurs. Different settings of these weights lead to different fusion strategies. We illustrate this with two special cases below, and then introduce an to automatically learn these weights together with the other network parameters.

Early fusion. If the fusion weights are all set to one, $\mathbf{w} = \mathbf{1}$, the two data streams are ignored, and only the fusion one is considered to compute the output. Since the fusion stream takes the concatenation of the image $\mathbf{I}_0$ and the confidence maps $\mathbf{X}_0$ as input, this is equivalent to the early fusion architecture of Fig. 5.2(a).

Fusion at a specific layer. Instead of fusing the streams in the very first layer, one might want to postpone the fusion point to a later layer $\beta \in \{0, \cdots, L\}$. In our formalism, this can be achieved by setting the fusion weights to $w_l = \mathbb{I}[l > \beta]$, where $\mathbb{I}$ is the indicator function. For example, when $\beta = 4$, our network becomes equivalent to the one depicted by Fig. 5.2(b). The early and late fusion architectures of Fig. 5.2(a, c) can also be represented in this manner by setting $\beta = 0$ and $\beta = L$, respectively.

Ultimately, the complete fusion network encodes a function $f(\mathbf{i}, \mathbf{x}; \theta, \mathbf{w}) = \mathbf{Z}_{L+1}|_{\mathbf{I}_0=\mathbf{i}, \mathbf{X}_0=\mathbf{x}}$ mapping from an image $\mathbf{i}$ and confidence maps $\mathbf{x}$ to the 3D joint locations, parametrized by layer weights θ and fusion weights $\mathbf{w}$. With manually-defined fusion weights, given a set of N training pairs $(\mathbf{i}_n, \mathbf{x}_n)$ with corresponding ground-truth joint positions $\mathbf{y}_n$, the parameters θ can be learnt by minimizing the square loss expressed as

$$L(\theta) = \sum_{n=1}^{N} \left\| f(\mathbf{i}_n, \mathbf{x}_n; \theta, \mathbf{w}) - \mathbf{y}_n \right\|_2^2 . \tag{5.2}$$

Trainable fusion. Setting the weights manually, which in our formalism boils down to choosing β, is not obvious; the best value for β will typically depend on the network architecture, the problem and the nature of the input data. A straightforward approach would consist of training networks for all possible values of β to validate the best one, but this quickly becomes impractical. To address this issue, we introduce a trainable fusion approach, which aims to learn β from data jointly with the network parameters. To this end, however, we cannot directly use the indicator function, which has zero derivatives almost everywhere, thus making it inapplicable to gradient-based optimization. Instead, we propose to approximate the indicator function by a sigmoid function

$$w_l = \frac{1}{1 + e^{-\alpha \cdot (l - \beta)}}, \tag{5.3}$$

parameterized by α and β. As above, β determines the stage at which fusion occurs and α controls how sharp the transition between weights with value 0 and with value 1 is. When $\alpha \to \infty$, the function in Eq. 5.3 becomes equivalent to the indicator function[1], while, when $\alpha = 0$, the network mixes the data and fusion streams in equal proportions at every layer.

In practice, mixing the data and fusion streams at every layer is not desirable. First, by contrast to having binary weights $\mathbf{w}$, which deactivate some of the layers of each stream, it corresponds to a model with a very large number of active parameters, and thus prone to overfitting. Furthermore, after training, a model with binary weights can be pruned, by removing the inactive layers in each stream, that is all layers l from the fusion stream where $w_l \approx 0$, and all layers l from the data streams where $w_l \approx 1$. This yields a more compact, and thus more efficient network for test-time prediction.

To account for this while learning where to fuse the information sources, we modify the loss function of Eq. 5.2 by incorporating a term that penalizes small values of α and favors sharp fusions. This yields a loss of the form

$$L(\theta, \alpha, \beta) = \sum_{n=1}^{N} \left\| f(\mathbf{i}_n, \mathbf{x}_n; \theta, \alpha, \beta) - \mathbf{y}_n \right\|_2^2 + \frac{\lambda}{\alpha^2}, \tag{5.4}$$

with α and β as trainable parameters, in addition to θ, and a hyperparameter λ weighing the penalty term. Altogether, this loss lets us simultaneously find the most suitable fusion layer β for the given data and the corresponding network parameters θ, while encouraging a sharp fusion function to mimic the behavior of the indicator function.

[1] Except at $l = \beta$.

In practice, we initialize α with a small value of 0.1 and β to the middle layer of the complete network. We use the ADAM [94] gradient update method with a learning rate of 10^{-3} to guide the optimization. We set the regularization parameter to $5 \cdot 10^3$, which renders the magnitude of both the regularization term and the main cost comparable. We use dropout and data augmentation to prevent overfitting.

5.1.2 2D Joint Location Confidence Map Prediction

Our approach depends on generating heatmaps of thfe 2D joint locations that we can feed as input to the confidence map stream. To do so, we rely on a fully-convolutional network with skip connections [132]. Given an RGB image as input, it performs a series of convolutions and pooling operations to reduce its spatial resolution, followed by upconvolutions to produce pixel-wise confidence values for each pixel. We employed the stacked hourglass network design of [132], which carries out repeated bottom-up, top-down processing to capture spatial relationships in the image. We perform heatmap regression to assign high confidence values to the most likely joint positions. In our experiments, we fine-tuned the hourglass network initially trained on the MPII dataset [7] using the training data specific to each experiment as a preliminary step to training our fusion network. In practice, we have observed that using the more accurate 2D joint locations predicted by the stacked network architecture improves the overall 3D prediction accuracy over using those predicted by a single-stage fully-convolutional network, such as [164]. Ultimately, these predictions provide reliable intermediate features for the 3D pose estimation task.

5.2 Results

In this section, we first describe the datasets we tested our approach on and the corresponding evaluation protocols. We then compare our approach against the state-of-the-art methods and provide a detailed analysis of our general framework.

5.2.1 Datasets

We evaluate our approach on the Human3.6m [81], HumanEva-I [182], KTH Multiview Football II [26] and Leeds Sports Pose (LSP) [86] datasets described below.

Human3.6m is a large and diverse motion capture dataset including 3.6 million images with their corresponding 2D and 3D poses. The poses are viewed from 4 different camera angles. The subjects carry out complex motions corresponding to daily human activities. We use the standard 17 joint skeleton from Human3.6m as our pose representation.

HumanEva-I comprises synchronized images and motion capture data and is a standard benchmark for 3D human pose estimation. The output pose is a vector of 15 3D joint coordinates.

KTH Multiview Football II provides a benchmark to evaluate the performance of pose estimation algorithms in unconstrained outdoor settings. The camera follows a soccer player moving around the pitch. The videos are captured from 3 different camera viewpoints. The output pose is a vector of 14 3D joint coordinates.

LSP is a standard benchmark for 2D human pose estimation and does not contain any ground-truth 3D pose data. The images are captured in unconstrained outdoor settings. 2D pose is represented in terms of a vector of 14 joint coordinates. We report qualitative 3D pose estimation results on this dataset.

5.2.2 Evaluation Protocol

On Human3.6m, we used the same data partition and evaluation protocol as in earlier work [44, 108, 109, 110, 137, 138, 163, 195, 197, 199, 232, 233] for a fair comparison. The data from 5 subjects (S1, S5, S6, S7, S8) was used for training and the data from 2 different subjects (S9, S11) was used for testing. We evaluate the accuracy of 3D human pose estimation in terms of average Euclidean distance between the predicted and ground-truth 3D joint positions. Training and testing were carried out monocularly in all camera views.

In [17][2] and [171][3] the estimated skeleton was first aligned to the ground-truth one by Procrustes transformation before measuring the joint distances. This is therefore what we also do when comparing against [17, 171].

On HumanEva-I, following the standard evaluation protocol [17, 184, 197, 223, 233], we trained our model on the training sequences of subjects S1, S2 and S3 and evaluated on the validation sequences of all subjects. We pretrained our network on Human3.6m and used only the first camera view for further training and validation.

On the KTH Multiview Football II dataset, we evaluate our method on the sequence containing Player 2, as in [13, 26, 138, 197]. Following [13, 26, 138, 197], the first half of the sequence from camera 1 is used for training and the second half for testing. To compare our results to those of [13, 26, 138, 197], we report accuracy using the percentage of correctly estimated parts (PCP) score. Since the training set is quite small, we propose to pretrain our network on the recent

[2]The pose estimation network in [17] is not trained on the Human3.6m data, however we also include their quantitative results for completeness.

[3]This it is not explicitly stated in [171], but the authors confirmed this to us by email.

synthetic dataset [30], which contains images of sports players with their corresponding 3D poses. We then fine-tuned it using the training data from KTH Multiview Football II. We report results with and without this pretraining.

5.2.3 Comparison to the State-of-the-Art

We first compare our approach with state-of-the-art baselines on the *Human3.6m [81], HumanEva [182] and KTH Multiview Football [26]* datasets.

Human3.6m. In Table 5.1, we compare the results of our trainable fusion approach with those of the following state-of-the-art single image-based methods: KDE regression from HOG features to 3D poses [81], jointly training a 2D body part detector and a 3D pose regressor [108, 137], the maximum-margin structured learning framework of [109, 110], the deep structured prediction approach of [195], pose regression with kinematic constraints [232], pose estimation with mocap guided data augmentation [163], volumetric pose prediction approach of [138] and lifting 2D heatmap predictions to 3D human pose [199]. For completeness, we also compare our approach to the following methods that rely on either multiple consecutive images or impose temporal consistency: regression from short image sequences to 3D poses [197], fitting a sparse 3D pose model to 2D confidence map predictions across frames [233], and fitting a 3D pose sequence to the 2D joints predicted by images and height-maps that encode the height of each pixel in the image with respect to a reference plane [44].

As can be seen from the results in Table 5.1, our approach improves upon the state-of-the-art in overall pose estimation accuracy. In particular, we outperform the image-based regression methods of [81, 108, 109, 110, 137, 195, 199, 232], as well as the model-fitting strategy of [109, 110, 233]. This, we believe, clearly evidences the benefits of fusing 2D joint location confidence maps with 3D image cues, as done by our approach. By leveraging reliable 2D joint location estimates, [138] also yields accurate 3D pose estimates, however our approach outperforms it on average across the entire dataset. Furthermore, we also achieve lower error than the method of [163], despite the fact that it relies on additional training data. Even though our algorithm uses only individual images, it also outperforms the methods that rely on sequences [44, 197, 233].

Since results are reported in [17, 171] for the average accuracy over all actions using the Procrustes transformation, as explained in Section 5.2.2, we do the same when comparing against these methods. Table 5.2 shows that we also outperform these baselines by a large margin.

Input	Method	Directions	Discussion	Eating	Greeting	Phone Talk	Posing	Buying	Sitting	Sitting Down
Single-Image	Ionescu et al. [81]	132.71	183.55	132.37	164.39	162.12	150.61	171.31	151.57	243.03
	Li & Chan [108]	-	148.79	104.01	127.17	-	-	-	-	-
	Li et al. [109]	-	134.13	97.37	122.33	-	-	-	-	-
	Li et al. [110]	-	133.51	97.60	120.41	-	-	-	-	-
	Zhou et al. [233]	-	-	-	-	-	-	-	-	-
	Rogez & Schmid [163]	-	-	-	-	-	-	-	-	-
	Tekin et al. [195]	-	129.06	91.43	121.68	-	-	-	-	-
	Park et al. [137]	100.34	116.19	89.96	116.49	115.34	117.57	106.94	137.21	190.82
	Zhou et al. [232]	91.83	102.41	96.95	98.75	113.35	90.04	93.84	132.16	158.97
	Tome et al. [199]	64.98	73.47	76.82	86.43	86.28	68.93	74.79	110.19	173.91
	Pavlakos et al. [138]	67.38	71.95	66.70	69.07	**71.95**	65.03	**68.30**	83.66	**96.51**
Video	Tekin et al. [197]	102.41	147.72	88.83	125.28	118.02	112.3	129.17	138.89	224.90
	Zhou et al. [233]	87.36	109.31	87.05	103.16	116.18	106.88	99.78	124.52	199.23
	Du et al. [44]	85.07	112.68	104.90	122.05	139.08	105.93	166.16	117.49	226.94
Single-Image	Ours (GM)	**53.91**	62.19	61.51	66.18	80.12	64.61	83.17	70.93	107.92
Single-Image	Ours (ASM)	54.23	**61.41**	**60.17**	**61.23**	79.41	**63.14**	81.63	**70.14**	107.31

Input	Method:	Smoking	Taking Photo	Waiting	Walking	Walking Dog	Walking Pair	Avg. (6 Actions)	Avg. (All)
Single-Image	Ionescu et al. [81]	162.14	205.94	170.69	96.60	177.13	127.88	159.99	162.14
	Li & Chan [108]	-	189.08	-	77.60	146.59	-	132.20	-
	Li et al. [109]	-	166.15	-	68.51	132.51	-	120.17	-
	Li et al. [110]	-	163.33	-	73.66	135.15	-	121.55	-
	Zhou et al. [233]	-	-	-	-	-	-	-	120.99
	Rogez & Schmid [163]	-	-	-	-	-	-	-	121.20
	Tekin et al. [195]	-	162.17	-	65.75	130.53	-	116.77	-
	Park et al. [137]	105.78	149.55	125.12	62.64	131.90	96.18	111.12	117.34
	Zhou et al. [232]	106.91	125.22	94.41	79.02	126.04	98.96	104.73	107.26
	Tome et al. [199]	84.95	110.67	85.78	71.36	86.26	73.14	84.17	88.39
	Pavlakos et al. [138]	71.74	**76.97**	**65.83**	59.11	74.89	63.24	69.78	71.90
Video	Tekin et al. [197]	118.42	182.73	138.75	55.07	126.29	65.76	120.99	124.97
	Zhou et al. [233]	107.42	143.32	118.09	79.39	114.23	97.70	106.07	113.01
	Du et al. [44]	120.02	135.91	117.65	99.26	137.36	106.54	118.69	126.47
Single-Image	Ours (GM)	70.44	79.45	68.01	52.81	77.81	**63.11**	66.66	70.81
Single-Image	Ours (ASM)	**69.29**	78.31	70.27	**51.79**	**74.28**	63.24	**64.53**	**69.73**

Table 5.1: **Comparison of our fusion approach with the state-of-the-art algorithms on *Human3.6m*.** We report 3D joint position errors in mm, computed as the average Euclidean distance between the ground-truth and predicted joint positions. (ASM) refers to an action-specific model in which a separate regressor is trained for each action and (GM) refers to a single general model trained on the whole training set. While [138, 199, 232] train single models, the rest carry out action-specific training.

Method:	3D Pose Error
Sanzari et al. [171]	93.15
Bogo et al. [17]	82.3
Ours	**50.12**

Table 5.2: **Comparison of our fusion approach to the state-of-the-art methods that use Procrustes transformation on Human3.6m.** We report 3D joint position errors (in mm).

HumanEva. In Table 5.3, we present the performance of our fusion approach on the HumanEva-I dataset [182]. We adopted the evaluation protocol described in [17, 184, 223, 233] for a fair comparison. As in [17, 184, 223, 233], we measure 3D pose error as the average joint-to-joint distance after alignment by a rigid transformation. Our approach also significantly outperforms the state-of-the-art on this dataset.

Method	S1	S2	S3	Average
Simo-Serra et al. [184]	65.1	48.6	73.5	62.4
Bogo et al. [17]	73.3	59.0	99.4	77.2
Zhou et al. [233]	34.2	30.9	49.1	38.07
Yasin et al. [223]	35.8	32.4	41.6	36.6
Tekin et al. [197]	37.5	25.1	49.2	37.3
Ours	**27.24**	**14.26**	**31.74**	**24.41**

Table 5.3: **Quantitative results of our fusion approach on the Walking sequences of the HumanEva-I dataset [182].** S1, S2 and S3 correspond to Subject 1, 2, and 3, respectively. The accuracy is reported in terms of average Euclidean distance (in mm) between the predicted and ground-truth 3D joint positions.

KTH Multiview Football. In Table 5.4, we compare our approach to [13, 26, 138, 197] on the KTH Multiview Football II dataset. Note that [13] and [26] rely on multiple views, and [197] makes use of video data. As discussed in Section 5.2.2, we report the results of two instances of our model: one trained on the standard KTH training data, and one pretrained on the synthetic 3D human pose dataset of [30] and fine-tuned on the KTH dataset. Note that, while working with a single input image, both instances outperform all the baselines. Note also that pretraining on synthetic data yields the highest accuracy. We believe that this further demonstrates the generalization ability of our method.

In Fig. 5.4, we provide representative poses predicted by our approach on the Human3.6m, HumanEva and KTH Multiview Football datasets.

5.2.4 Detailed Analysis

We now analyze two different aspects of our approach. First, we compare our trainable fusion approach to early fusion, depicted in Fig. 5.2(a), and late fusion, depicted in Fig. 5.2(c). Then, we analyze the benefits of leveraging both 2D joint locations with their corresponding uncertainty and additional image cues. To this end, we make use of two additional baselines. The first one consists of a single stream CNN regressor operating on the image only. We refer to this baseline as *Image-Only*. The second is a CNN trained to predict 3D pose from only the 2D confidence map (CM) stream. We refer to this baseline as *CM-Only*.

Method:	[26]	[26]	[13]	[197]	[138]	Ours-NoPretraining	Ours-Pretraining
Input:	Image	Image	Image	Video	Image	Image	Image
Num. of cameras:	1	2	2	1	1	1	1
Pelvis	97	97	-	99	-	66	**100**
Torso	87	90	-	**100**	-	**100**	**100**
Upper arms	14	53	64	74	94	74	**100**
Lower arms	06	28	50	49	80	**100**	88
Upper legs	63	88	75	98	96	**100**	**100**
Lower legs	41	82	66	77	84	77	**88**
All parts	43	69	-	79	-	83.2	**95.2**

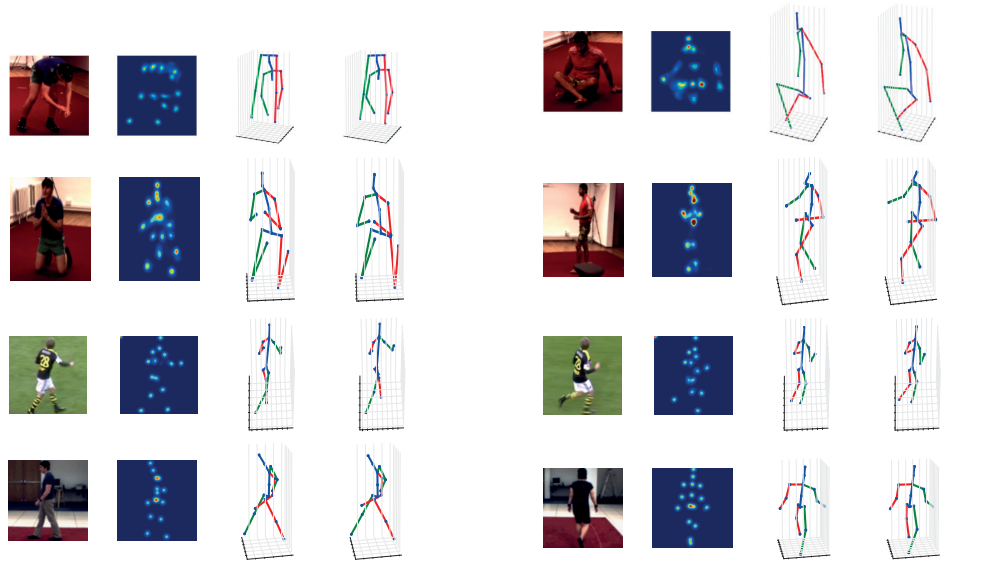
Table 5.4: **Comparison of our fusion approach to the-state-of-the-art on KTH Multiview Football II.** On this dataset, we compare our method that uses a single image to those of [26, 138, 197] that use either one or two images, the one of [13] that uses two, and the one of [197] that operates on a sequence. As in [13, 26, 138, 197], we measure performance as the percentage of correctly estimated parts (PCP) score. A higher PCP score corresponds to better 3D pose estimation accuracy.

Method:	3D Pose Error
Image-Only	124.13
CM-Only	79.28
Early Fusion	76.41
Late Fusion	74.12
Trainable Fusion	**69.73**

Table 5.5: **Comparison of different fusion strategies and single-stream baselines on Human3.6m.** We report the 3D joint position errors (in mm). The fusion networks perform better than those that use only the image or only the confidence map as input. Our trainable fusion achieves the best accuracy overall.

In Table 5.5, we report the average pose estimation errors on Human3.6m for all these methods. Our trainable fusion strategy yields the best results. Note also that, in general, all fusion strategies yield accurate pose estimates. Importantly, the *Image-Only* and *CM-Only* baselines perform worse than our approach, and all fusion-based methods. This demonstrates the importance of fusing 2D joint location confidence maps along with 3D cues in the image for monocular pose estimation.

In Fig. 5.5, we depict the evolution throughout the training iterations of **(a)** the parameters α and β that define the weight vector in our trainable fusion framework as given by Eq. 5.3, and **(b)** the weight vector itself. An increasing value of α, expected due to our regularizer, indicates that fusion becomes sharper throughout the training. An increasing β, which is the typical behavior, corresponds to fusion occurring in the later stages of the network. We conjecture that this is due to the fact that features learned by the image and confidence map streams at later layers become less correlated, and thus yield more discriminative power.



(a) Image (b) Heatmap (c) Prediction (d) Ground-truth (e) Image (f) Heatmap (g) Prediction (h) Ground-truth

Figure 5.4: **Pose estimation results of our fusion approach on Human3.6m, HumanEva and KTH Multiview Football II. (a, e)** Input images. **(b, f)** 2D joint location confidence maps. **(c, g)** Recovered pose. **(d, h)** Ground truth. Note that our method can recover the 3D pose in these challenging scenarios, which involve significant amounts of self occlusion and orientation ambiguity. Best viewed in color.

To analyze this further, we show in Fig. 5.7 the squared Pearson correlation coefficients between all pairs of features of the confidence map stream and of the image stream at the last convolutional layer of our trainable fusion network. As can be seen in the figure, the image and confidence map streams produce decorrelated features that are complementary to each other allowing to effectively account for different input modalities.

We analyze further the effect of the regularization term that encourages sharp fusion in Eq. 5.4. In the absence of the regularization term, the network mixes the data and fusion streams without necessarily fusing them at a specific layer. As discussed in Section 5.1.1, this corresponds to a model with many active parameters. Therefore it is prone to overfitting and computationally less efficient at test-time. In Table 5.6, we compare the results of our approach with and without this regularization term. For the latter, we do not parametrize the weights of the network with a sigmoid function and do not constrain the network to have a sharp fusion. The results confirm that encouraging sharp fusion yields both better accuracy and faster prediction.

We carried out our experiments on a machine equipped with an Intel Xeon CPU E5-2680 and an NVIDIA TITAN X Pascal GPU. It takes 90 ms to compute 2D joint location confidence maps and 6 ms to predict 3D pose with our fusion network. Therefore, the total runtime of our method is 0.096 sec/frame (over 10 fps), which compares favorably with the recent model-based methods ranging from 0.04 fps to 1 fps [171, 223, 233].

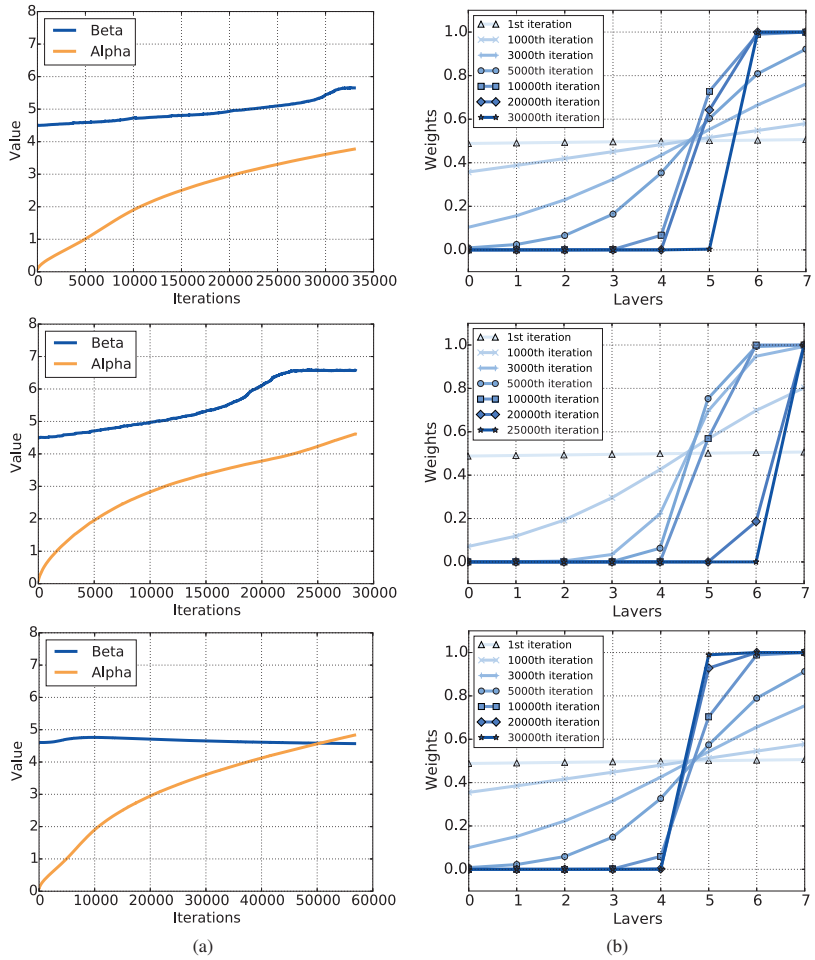


Figure 5.5: **Evolution of (a) α and β, and (b) the fusion weights in Human3.6m during training.** Top row: Directions; Middle row: Discussion; Bottom row: Sitting Down.

Method	3D Pose Error	Runtime
Without regularization	68.30	0.013
With regularization	60.17	0.006

Table 5.6: **Quantitative results of our fusion approach with and without the regularization term encouraging sharp fusion.** These experiments were carried out on the *Eating* action class of Human3.6m. 3D pose error is computed as the average Euclidean distance (in milimeters) between the predicted and ground-truth 3D joint positions. Runtime denotes the computational time spent, in sec/frame, during testing for the fusion network with and without the regularization term. With the regularization term, inactive layers are pruned after training, which yields a more efficient network for test-time prediction.

In Fig. 5.6, we present qualitative pose estimation results on the Leeds Sports Pose dataset. We trained our network on the synthetic dataset of [30] and tested on images acquired outdoors in unconstrained settings. The accurate 3D predictions of the challenging poses demonstrate the generalization ability and robustness of our method.

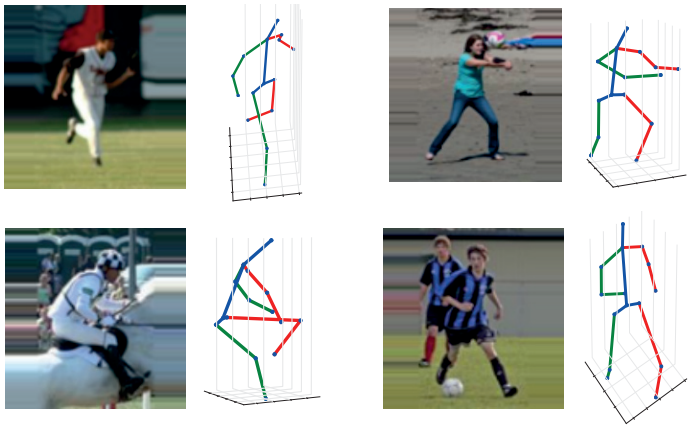


Figure 5.6: **Pose estimation results of our fusion approach on the Leeds Sports Pose dataset.** We show the input image and the predicted 3D pose for four images. Best viewed in color.

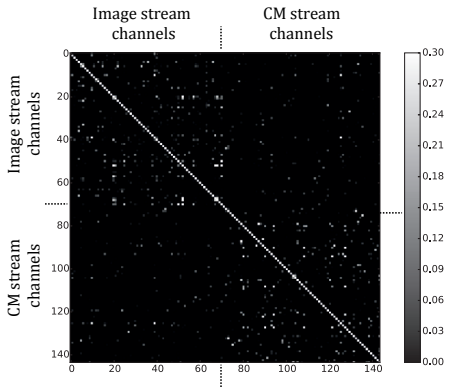
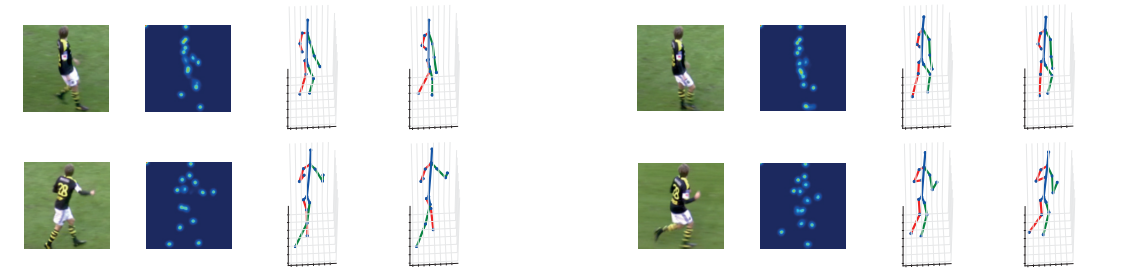


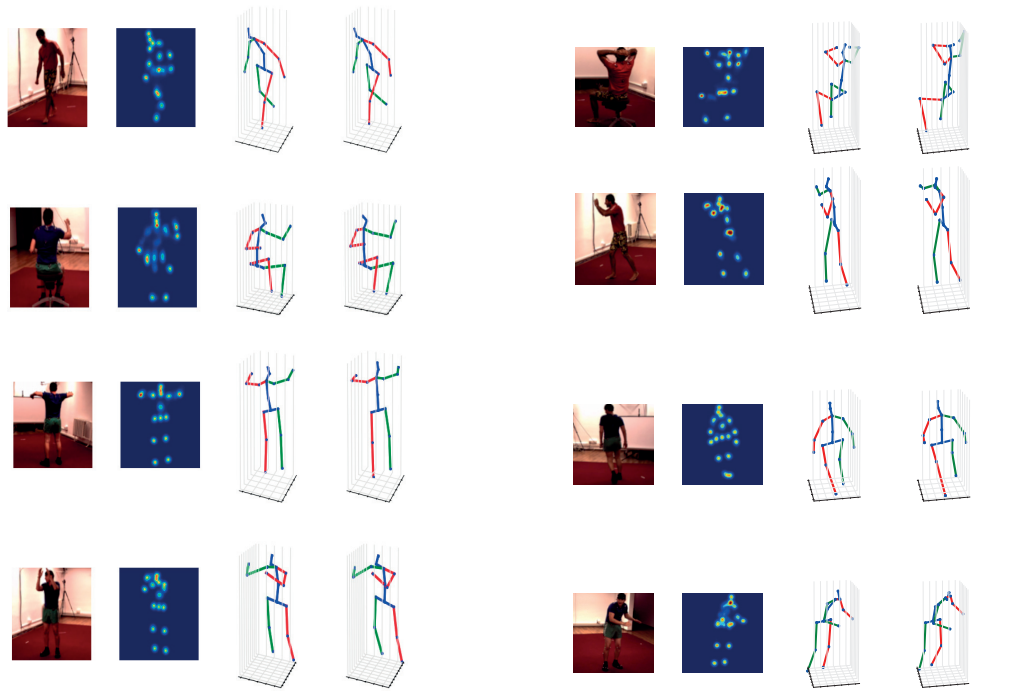
Figure 5.7: **Feature correlation between the two streams of the network for our fusion approach.** We report squared Pearson correlation coefficients (R^2) between each pair of the features learned at the last convolutional layer of our trainable fusion network computed from 128 randomly selected images in Human3.6m. As can be seen in the lower left and upper right submatrices, the feature maps of the image and the confidence map streams are decorrelated.

We provide additional qualitative results for the KTH Multiview Football II [26], Human3.6m [81] and HumanEva [182] datasets in Figs. 5.8, 5.9 and 5.10, respectively. Finally, we demonstrate that our regressor trained on the recently released synthetic dataset of [30] generalizes well to real images obtained from the Leeds Sports Pose dataset [86] in Fig. 5.11.



(a) Image (b) Heatmap (c) Prediction (d) Ground-truth (e) Image (f) Heatmap (g) Prediction (h) Ground-truth

Figure 5.8: **Example pose estimation results of our fusion approach on KTH Multiview Football II. (a, e)** Input images. **(b, f)** 2D joint location confidence maps. **(c, g)** Recovered pose. **(d, h)** Ground truth. Best viewed in color.



(a) Image (b) Heatmap (c) Prediction (d) Ground-truth (e) Image (f) Heatmap (g) Prediction (h) Ground-truth

Figure 5.9: **Example pose estimation results of our fusion approach on Human3.6m. (a, e)** Input images. **(b, f)** 2D joint location confidence maps. **(c, g)** Recovered pose. **(d, h)** Ground truth. Note that our method can recover the 3D pose in these challenging scenarios, which involve significant amounts of self occlusion and orientation ambiguity. Best viewed in color.

(a) Image (b) Heatmap (c) Prediction (d) Ground-truth (e) Image (f) Heatmap (g) Prediction (h) Ground-truth

Figure 5.10: **Example pose estimation results of our fusion approach on HumanEva-I. (a, e)** Input images. **(b, f)** 2D joint location confidence maps. **(c, g)** Recovered pose. **(d, h)** Ground truth. Best viewed in color.

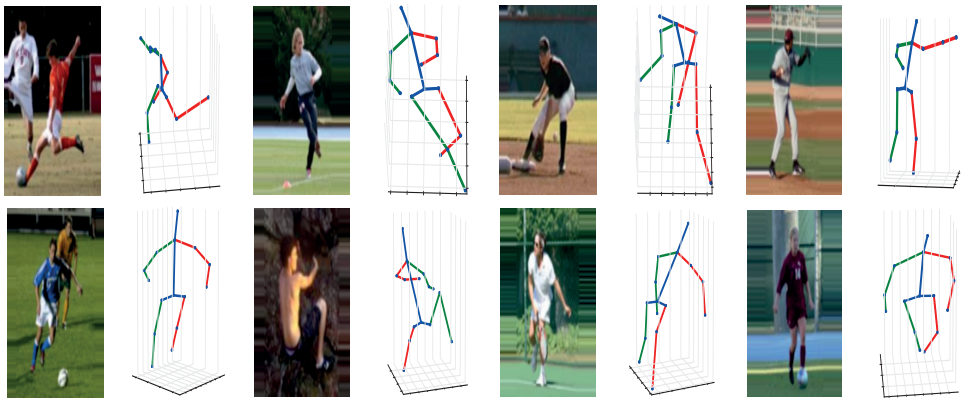


Figure 5.11: **Example pose estimation results of our fusion approach on LSP.** We trained our network on the recently released synthetic dataset of [30] and tested it on the LSP dataset. The quality of the 3D pose predictions demonstrates the generalization of our method. Best viewed in color.

6 Real-Time Seamless Single Shot 6D Object Pose Prediction

Real-time object detection and 6D pose estimation is crucial for augmented reality, virtual reality, and robotics. Currently, methods relying on depth data acquired by RGB-D cameras are quite robust [20, 33, 34, 92, 100]. However, active depth sensors are power hungry, which makes 6D object detection methods for passive RGB images more attractive for mobile and wearable cameras. There are many fast keypoint and edge-based methods [118, 167, 213] that are effective for textured objects. However, they have difficulty handling weakly textured or untextured objects and processing low-resolution video streams, which are quite common when dealing with cameras on wearable devices.

Deep learning techniques have recently been used to address these limitations [91, 152]. BB8 [152] is a 6D object detection pipeline made of one CNN to coarsely segment the object and another to predict the 2D locations of the projections of the object's 3D bounding box given the segmentation, which are then used to compute the 6D pose using a PnP algorithm [106]. The method is effective but slow due to its multi-stage nature. SSD-6D [91] is a different pipeline that relies on the SSD architecture [115] to predict 2D bounding boxes and a very rough estimate of the object's orientation in a single step. This is followed by an approximation to predict the object's depth from the size of its 2D bounding box in the image, to lift the 2D detections to 6D. Both BB8 and SSD-6D require a further pose refinement step for improved accuracy, which increases their running times linearly with the number of objects being detected.

In this chapter, we propose a single-shot deep CNN architecture that takes the image as input and directly detects the 2D projections of the 3D bounding box vertices. It is end-to-end trainable and accurate even without any *a posteriori* refinement. And since, we do not need this refinement step, we also do not need a precise and detailed textured 3D object model that is needed by other methods [91, 152]. We only need the 3D bounding box of the object shape for training. This can

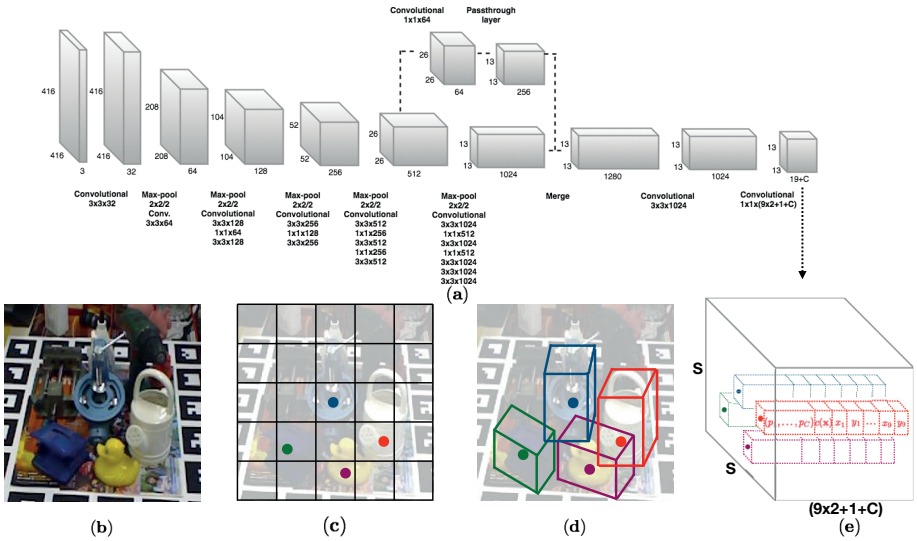


Figure 6.1: **Overview of our single shot 6D object pose estimation approach.** (a) The proposed CNN architecture. (b) An example input image with four objects. (c) The $S \times S$ grid showing cells responsible for detecting the four objects. (d) Each cell predicts 2D locations of the corners of the projected 3D bounding boxes in the image. (e) The 3D output tensor from our network, which represents for each cell a vector consisting of the 2D corner locations, the class probabilities and a confidence value associated with the prediction.

be derived from other easier to acquire and approximate 3D shape representations.

We demonstrate state-of-the-art accuracy on the LINEMOD dataset [70], which has become a *de facto* standard benchmark for 6D pose estimation. However, we are much faster than the competing techniques by a factor of more than five, when dealing with a single object. Furthermore, we pay virtually no time-penalty when handling several objects and our running time remains constant whereas that of other methods grow proportional to the number of objects, which we demonstrate on the OCCLUSION dataset [20].

Therefore, our contribution is an architecture that yields a fast and accurate one-shot 6D pose prediction without requiring any post-processing. It extends single shot CNN architectures for 2D detection in a seamless and natural way to the 6D detection task. Our implementation is based on YOLO [158] but the approach is amenable to other single-shot detectors such as SSD [115] and its variants.

6.1 Approach

With our goal of designing an end-to-end trainable network that predicts the 6D pose in real-time, we were inspired by the impressive performance of single shot 2D object detectors such as

YOLO [157, 158]. This led us to design the CNN architecture [157, 158] shown in Fig. 6.1. We designed our network to predict the 2D projections of the corners of the 3D bounding box around our objects. The main insight was that YOLO was originally designed to regress 2D bounding boxes and to predict the projections of the 3D bounding box corners in the image, a few more 2D points had to be predicted for each object instance in the image. Then given these 2D coordinates and the 3D ground control points for the bounding box corners, the 6D pose can be calculated algebraically with an efficient PnP algorithm [106]. BB8 [152] takes a similar approach. However, they first find a 2D segmentation mask around the object and present a cropped image to a second network that predicts the eight 2D corners in the image. We now describe our network architecture and explain various aspects of our approach in details.

6.1.1 Model

We formulate the 6D pose estimation problem in terms of predicting the 2D image coordinates of virtual 3D control points associated with the 3D models of our objects of interest. Given the 2D coordinate predictions, we calculate the object's 6D pose using a PnP algorithm. We parameterize the 3D model of each object with 9 control points. For these control points, we select the 8 corners of the tight 3D bounding box fitted to the 3D model, similar to [152]. In addition, we use the centroid of the object's 3D model as the 9th point. This parameterization is general and can be used for any rigid 3D object with arbitrary shape and topology. In addition, these 9 control points are guaranteed to be well spread out in the 2D image and could be semantically meaningful for many man-made objects.

Our model takes as input a single full color image, processes it with a fully-convolutional architecture shown in Figure 6.1(a) and divides the image into a 2D regular grid containing $S \times S$ cells as shown in Figure 6.1(c). In our model, each grid location in the 3D output tensor will be associated with a multidimensional vector, consisting of predicted 2D image locations of the 9 control points, the class probabilities of the object and an overall confidence value. At test time, predictions at cells with low confidence values, ie. where the objects of interest are not present, will be pruned.

The output target values for our network are stored in a 3D tensor of size $S \times S \times D$ visualized in Fig. 6.1(e). The target values for an object at a specific spatial cell location $i \in S \times S$ is placed in the i-th cell in the 3D tensor in the form of a D dimensional vector $\mathbf{v}_i$. When N objects are present in different cells, we have N such vectors, $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ in the 3D tensor. We train our network to predict these target values. The 9 control points in our case are the 3D object model's center and bounding box corners but could be defined in other ways as well. To train our network, we only need to know the 3D bounding box of the object, not a detailed mesh or an associated

texture map.

As in YOLO, it is crucial that a trained network is able to predict not only the precise 2D locations but also high confidence values in regions where the object is present and low confidence where it isn't present. In case of 2D object detection, YOLO uses for its confidence values, an intersection over union (IoU) score associated with the predicted (and true 2D rectangles) in the image. In our case, the objects are in 3D and to compute an equivalent IoU score with two arbitrary cuboids, we would need to calculate a 3D convex hull corresponding to their intersections. This would be tedious and would slow down training, as also analyzed in Section 6.3. Therefore, we take a different approach. We model the predicted confidence value using a confidence function shown in Figure 6.2. The confidence function, $c(\mathbf{x})$, returns a confidence value for a predicted 2D point denoted by $\mathbf{x}$ bad on its distance $D_T(\mathbf{x})$ from the ground truth i.e. target 2D point. Formally, we define the confidence function $c(\mathbf{x})$ as follows:

$$c(\mathbf{x}) = \begin{cases} e^{\alpha(1-\frac{D_T(\mathbf{x})}{d_{th}})}, & \text{if } D_T(\mathbf{x}) < d_{th} \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

The distance $D_T(\mathbf{x})$ is defined as the 2D Euclidean distance in the image space. To achieve precise localization with this function, we choose a sharp exponential function with a cut-off value d_{th} instead of a monotonically decreasing linear function. The sharpness of the exponential function is defined by the parameter α. In practice, we apply the confidence function to all the control points and calculate the mean value and assign it as the confidence.

As mentioned earlier, we also predict C conditional class probabilities at each cell. The class probability is conditioned on the cell containing an object. Overall, our output 3D tensor depicted in Figure 6.1(**e**) has dimension $S \times S \times D$, where the 2D spatial grid corresponding to the image dimensions has $S \times S$ cells and each such cell has a D dimensional vector. Here, $D = 9 \times 2 + C + 1$, because we have 9 (x_i, y_i) control points, C class probabilities and one confidence value.

Our network architecture follows the fully convolutional YOLO v2 architecture [158]. Thus, our network has 23 convolutional layers and 5 max-pooling layers. Similar to YOLO v2, we choose $S = 13$ and have a 13×13 2D spatial grid on which we make our predictions. We also allow higher layers of our network to use fine-grained features by adding a passthrough layer. Specifically, we bring features from an earlier layer at resolution 26×26, apply batch normalization and resize the input image during training on-the-fly. As the network downsamples the image by a factor of 32, we change the input resolution to a multiple of 32 randomly chosen from the set $\{320, 352, \ldots, 608\}$ to be robust to objects of different size.

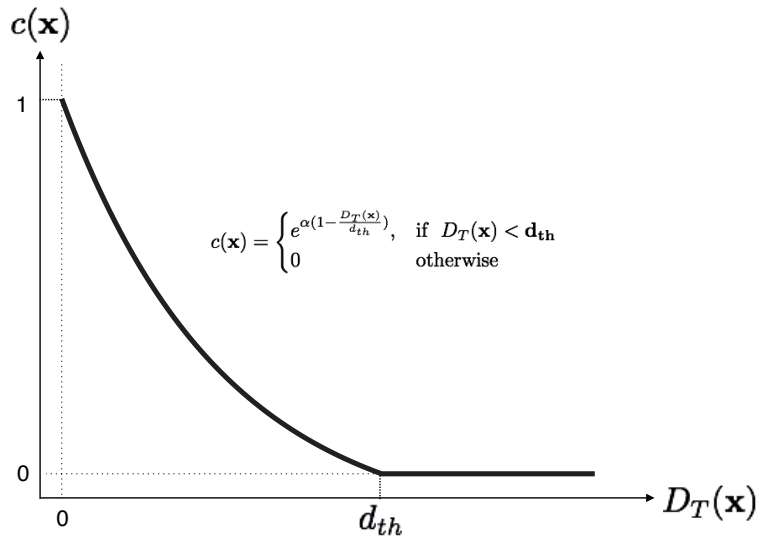


Figure 6.2: **Confidence $c(\mathbf{x})$ of our control point predictions as a function of the distance $D_T(\mathbf{x})$ between a predicted point and the true point.**

6.1.2 Training Procedure

Our final layer outputs class probabilities, (x, y) coordinate locations for the control points, and the overall confidence score. During training, this confidence value is computed on the fly using the function defined in Eq. 6.1 to measure the distance between the current coordinate predictions and the ground-truth, $D_T(\mathbf{x})$. We predict offsets for the 2D coordinates with respect to (c_x, c_y), the top-left corner of the associated grid cell. For the centroid, we constrain this offset to lie between 0 and 1. However, for the corner points, we do not constrain the network's output as those points should be allowed to fall outside the cell. The predicted control point (g_x, g_y) is defined as

$$g_x = f(x) + c_x \tag{6.2}$$

$$g_y = f(y) + c_y \tag{6.3}$$

where $f(\cdot)$ is chosen to be a 1D sigmoid function in case of the centroid and the identity function in case of the eight corner points. This has the effect of forcing the network to first find the approximate cell location for the object and later refine its eight corner locations. We minimize

the following loss function to train our complete network.

$$\mathscr{L} = \lambda_{pt}\mathscr{L}_{pt} + \lambda_{conf}\mathscr{L}_{conf} + \lambda_{id}\mathscr{L}_{id} \tag{6.4}$$

Here, the terms $\mathscr{L}_{pt}$, $\mathscr{L}_{conf}$ and $\mathscr{L}_{id}$ denote the coordinate, confidence and the classification loss, respectively. We use mean-squared error for the coordinate and confidence losses, and cross entropy for the classification loss. As suggested in [157], we downweight the confidence loss for cells that don't contain objects by setting λ_{conf} to 0.1. This improves model stability. For cells that contain objects, we set λ_{conf} to 5.0. We set λ_{pt} and λ_{id} simply to 1.

When multiple objects are located close to each other in the 3D scene, they are more likely to appear close together in the images or be occluded by each other. In these cases, certain cells might contain multiple objects. To be able to predict the pose of such multiple objects that lie in the same cell, we allow up to 5 candidates per cell and therefore predict five sets of control points per cell. This essentially means that we assumed that at most 5 objects could occlude each other in a single grid cell. This is a reasonable assumption to make in practical pose estimation scenarios. As in [158], we precompute with k-means, five anchor boxes that define the size, ie. the width and height of a 2D rectangle tightly fitted to a masked region around the object in the image. During training, we assign whichever anchor box has the most similar size to the current object as the responsible one to predict the 2D coordinates for that object.

6.1.3 Pose Prediction

We detect and estimate the pose of objects in 6D by invoking our network only once. At test time, we estimate the class-specific confidence scores for each object by multiplying the class probabilities and the score returned by the confidence function. Each grid cell produces predictions in one network evaluation and cells with predictions with low confidence are pruned using a confidence threshold. For large objects and objects whose projections lie at the intersection of two cells, multiple cells are likely to predict highly confident detections. To obtain a more robust and well localized pose estimate, we inspect the cells in the 3×3 neighborhood of the cell which has the maximum confidence score. We combine the individual corner predictions of these adjacent cells by computing a weighted average of the individual detections, where the weights used are the confidence scores of the associated cells.

At run-time, the network gives the 2D projections of the object's centroid and corners of its 3D bounding box along with the object identity. We estimate the 6D pose from the correspondences between the 2D and 3D points using a Perspective-n-Point (PnP) pose estimation method [106].

In our case, PnP uses only 9 such control point correspondences and provides an estimate of the 3D rotation **R** and 3D translation **t** of the object in camera coordinates.

6.2 Implementation Details

We initialize the parameters of our network by training the original network on the ImageNet classification task. As the pose estimates in the early stages of training are inaccurate, the confidence values computed using Eq. 6.1 are initially unreliable. To remedy this, we pretrain our network parameters by setting the regularization parameter for confidence to 0. Subsequently, we train our network by setting λ_{conf} to 5 for the cells that contain an object, and to 0.1 otherwise, to have more reliable confidence estimates in the early stages of the network. In practice, we set the sharpness of the confidence function α to 2 and the distance threshold to 30 pixels. We use stochastic gradient descent for optimization. We start with a learning rate of 0.001 and divide the learning rate by 10 at every 100 epochs. To avoid overfitting, we use extensive data augmentation by randomly changing the hue, saturation and exposure of the image by up to a factor of 1.5. We also randomly scale and translate the image by up to a factor of 20% of the image size. Our implementation is based on PyTorch. We will make our code publicly available for the sake of reproducibility.

6.3 Experiments

We first evaluate our method for estimating the 6D pose of single objects and then we evaluate it in the case where multiple objects are present in the image. We use the same datasets and evaluation protocols as in [21, 91, 152], which we review below. We then present and compare our results to the state of the art methods.

6.3.1 Datasets

We test our approach on two datasets that were designed explicitly to benchmark 6D object pose estimation algorithms. We describe them briefly below.

LineMod [70] has become a *de facto* standard benchmark for 6D object pose estimation of textureless objects in cluttered scenes. The central object in each RGB image is assigned a ground-truth rotation, translation, and ID. A full 3D mesh representing the object is also provided. There are 15783 images in LINEMOD for 13 objects. Each object features in about 1200 instances.

OCCLUSION [20] is a multi-object detection and pose estimation dataset that contains additional annotations for *all* objects in a subset of the LINEMOD images. As its name suggests, several objects in the images are severely occluded due to scene clutter, which makes pose estimation extremely challenging. With the exception of [91, 152], it has primarily been used to test algorithms that require depth images.

6.3.2 Evaluation Metrics

We use three standard metrics to evaluate 6D pose accuracy, namely – 2D reprojection error, IoU score and average 3D distance of model vertices (referred to as ADD metric) as in [21, 91, 152]. In all cases, we calculate the accuracy as the percentage of *correct* pose estimates for certain error thresholds.

When using the reprojection error, we consider a pose estimate to be correct when the mean distance between the 2D projections of the object's 3D mesh vertices using the estimate and the ground truth pose is less than 5 pixels [21]. This measures the closeness of the true image projection of the object to that obtained by using the estimated pose. This metric is suitable for augmented reality applications.

To compute the IoU score, we measure the overlap between the projections of the 3D model given the ground-truth and predicted pose and accept a pose as correct if the overlap is larger than 0.5, as in [91].

When comparing 6D poses using the ADD metric, we take a pose estimate to be correct if the mean distance between the true coordinates of 3D mesh vertices and those estimated given the pose is less than 10% of the object's diameter [70]. For most objects, this is approximately a 2cm threshold but for smaller objects, such as *ape*, the threshold drops to about 1cm. For rotationally symmetric objects whose pose can only be computed up to one degree of rotational freedom, we modify slightly the metric as in [21, 70] and compute

$$s = \frac{1}{|\mathcal{M}|} \sum_{x_1 \in \mathcal{M}} \min_{\mathcal{M}} \|(\mathbf{R}\mathbf{x} + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{t}})\|, \tag{6.5}$$

where $(\mathbf{R}, \mathbf{t})$ are the ground-truth rotation and translation, $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$ the predicted ones, and $\mathcal{M}$ the vertex set of the 3D model. We use this metric when evaluating the pose accuracy for the rotationally invariant objects, *eggbox* and *glue* as in [21, 70].

Method	w/o Refinement			w/ Refinement	
Object	Brachmann [21]	BB8 [152]	OURS	Brachmann [21]	BB8 [152]
Ape	-	***95.3***	92.10	85.2	**96.6**
Benchvise	-	80.0	**95.06**	67.9	90.1
Cam	-	80.9	**93.24**	58.7	86.0
Can	-	84.1	**97.44**	70.8	91.2
Cat	-	97.0	**97.41**	84.2	98.8
Driller	-	74.1	***79.41***	73.9	**80.9**
Duck	-	81.2	**94.65**	73.1	92.2
Eggbox	-	87.9	**90.33**	83.1	91.0
Glue	-	89.0	**96.53**	74.2	92.3
Holepuncher	-	90.5	**92.86**	78.9	95.3
Iron	-	78.9	***82.94***	83.6	**84.8**
Lamp	-	74.4	**76.87**	64.0	75.8
Phone	-	77.6	**86.07**	60.6	85.3
Average	69.5	83.9	**90.37**	73.7	89.3

Table 6.1: **Comparison of our single shot 6D pose estimation approach with state-of-the-art algorithms on LINEMOD in terms of 2D reprojection error.** We report percentages of correctly estimated poses. In Tables 6.1, 6.2 and 6.4 **bold face** numbers denote the best overall methods, ***bold italic*** numbers denote the best methods among those that do not use refinement as opposed to the ones that use, if different. Note that even though we do not rely on the knowledge of a detailed 3D object model our method consistently outperforms the baselines.

6.3.3 Single Object Pose Estimation

We first estimate the 6D pose of the central object in the RGB only LINEMOD images, without reference to the depth ones. We compare our approach to those of [21, 91, 152], which operate under similar conditions.

In this dataset, the training images are selected such that the relative orientation between corresponding pose annotations are larger than a threshold. As in [21, 91, 152], to avoid being influenced by the scene context and overfitting to the background, we segment the training images using the segmentation masks provided with the dataset and replace the background by a random image from the PASCAL VOC dataset [48].

We use exactly the same training/test splits as in [152]. We report our results in terms of 2D reprojection error in Table 6.1, 6D pose error in Table 6.2 and IoU metric in Table 6.4. We provide example pose predictions of our approach in Figure 6.3.

6.3.3.1 Comparative Accuracy

6D Accuracy in terms of projection error. In Table 6.1, we compare our results to those of Brachmann et al. [21] and to BB8 [152]. Both of these competing methods involve a multi-stage

pipeline that comprises a 2D detection step followed by pose prediction and refinement. Since we do not have a refinement stage, we show in the table their results without and with it. In both cases, we achieve better 6D pose estimation accuracies.

In Table 6.4, we perform a similar comparison with SSD-6D [91], whose authors report their projection accuracy in terms of the IoU metric. That method also requires *a posteriori* refinement and our results are again better in both cases, even though SSD-6D relies on a large training set of rendered images that are sampled over a wide range of viewpoints and locations.

6D Accuracy in terms of the ADD metric. In Tables 6.2 and 6.3, we compare our methods against the other in terms of the average of the 3D distances, as described in Section 6.3.2. In Table 6.2, we give numbers before and after refinement for the competing methods. Before refinement, we outperform all the methods by a significant margin of at least 12%. After refinement, our pose estimates are still better than Brachmann et al. [21]. By assuming the additional knowledge of a full 3D CAD model and using it to further refine the pose, BB8 [1] and SSD-6D [2] boost their pose estimation accuracy.

Without any bells and whistles, our approach achieves state-of-the-art pose estimation accuracy in all the metrics without refinement. When compared against methods that rely on the additional knowledge of full 3D CAD models and pose refinement, it still achieves state-of-the-art performance in 2D projection error and IoU metrics and yields comparable accuracy in the ADD metric. Our approach could be used in conjunction with such refinement strategies to further increase the accuracy however this comes at a heavy computational cost as we describe below.

6.3.3.2 Accuracy / Speed Trade-off

In Table 6.5, we report the computational efficiency of our approach for single object pose estimation in comparison to the state-of-the-art approaches [21, 91, 152]. Our approach runs at real-time performance in contrast to the existing approaches which fall short of it. In particular, our algorithm runs at least 5 times faster than the state-of-the-art techniques for single object pose estimation.

As can be seen in Table 6.2, pose refinement in Brachmann et al. increase the accuracy significantly by 17.9% at an additional run-time of 100 miliseconds per object. BB8 also gets a substantial improvement of 19.1% in accuracy at an additional run-time of 21 miliseconds per

[1] The authors do not report results without refinement, however they provided us with the accuracy numbers reported in Table 6.2.

[2] The authors were not able to provide their accuracy numbers without refinement for this metric, but made their code publicly available. We ran their code with provided pretrained models to obtain the 6D pose errors.

Method	w/o Refinement				w/ Refinement		
Object	Brachmann [21]	BB8 [152]	SSD-6D [91]	OURS	Brachmann [21]	BB8 [152]	SSD-6D [91]
Ape	-	***27.9***	0	21.62	33.2	40.4	**65**
Benchvise	-	62.0	0.18	***81.80***	64.8	**91.8**	80
Cam	-	***40.1***	0.41	36.57	38.4	55.7	**78**
Can	-	48.1	1.35	***68.80***	62.9	64.1	**86**
Cat	-	***45.2***	0.51	41.82	42.7	62.6	**70**
Driller	-	58.6	2.58	***63.51***	61.9	**74.4**	73
Duck	-	***32.8***	0	27.23	30.2	44.3	**66**
Eggbox	-	40.0	8.9	***69.58***	49.9	57.8	**100**
Glue	-	27.0	0	***80.02***	31.2	41.2	**100**
Holepuncher	-	42.4	0.30	***42.63***	52.8	**67.2**	49
Iron	-	67.0	8.86	***74.97***	80.0	**84.7**	78
Lamp	-	39.9	8.20	***71.11***	67.0	**76.5**	73
Phone	-	35.2	0.18	***47.74***	38.1	54.0	**79**
Average	32.3	43.6	2.42	***55.95***	50.2	62.7	**79**

Table 6.2: **Comparison of our single shot pose estimation approach with state-of-the-art algorithms on LINEMOD in terms of ADD metric.** We report percentages of correctly estimated poses.

object. Even without correcting for the pose error, our approach outperforms Brachmann et al. and yields close accuracy to BB8 while being 16 times faster for single object pose estimation. As discussed also in [91], the unrefined poses computed from the bounding boxes of the SSD 2D object detector, are rather approximate. We confirmed this by running their publicly available code with the provided pretrained models. We report the accuracy numbers without the refinement using the ADD metric in Table 6.3 for different thresholds. While providing a good initialization for the subsequent pose processing, the pose estimates of SSD-6D without refinement are much less accurate than our approach. The further refinement increases the pose estimation accuracy significantly, however at the cost of a computational time of 24 miliseconds per object. Moreover, in contrast to our approach, the refinement requires the knowledge of the full 3D object CAD model.

In Figure 6.3, we show example results of our method on the LINEMOD.

6.3.4 Multiple Object Pose Estimation

We use the OCCLUSION dataset to compare our approach to Brachmann et al. [21] for multi-object detection and report pose estimation accuracy as in [152]. The identity of the objects cannot be assumed to be known *a priori* and has to be guessed. To this end, the method of [152] assumes that it has access to image crops based on the ground-truth 2D bounding boxes [3]. We make no such assumptions. Instead, we jointly detect the object in 2D, estimate its identity and predict its 6D pose. We generate our training images with the approach explained in Section 6.3.2.

[3] This it is not explicitly stated in [152], but the authors confirmed this to us in private email communication.

Threshold	10%		30%		50%	
Object	[91]	OURS	[91]	OURS	[91]	OURS
Ape	0	**21.62**	5.62	**70.67**	19.95	**88.10**
Benchvise	0.18	**81.80**	2.07	**91.07**	10.62	**98.85**
Cam	0.41	**36.57**	34.52	**81.57**	63.54	**94.80**
Can	1.35	**68.80**	61.43	**99.02**	85.49	**99.90**
Cat	0.51	**41.82**	36.87	**90.62**	64.04	**98.80**
Driller	2.58	**63.51**	56.01	**99.01**	84.86	**99.80**
Duck	0	**27.23**	5.56	**70.70**	32.65	**89.39**
Eggbox	8.9	**69.58**	24.61	**81.31**	48.41	**98.31**
Glue	0	**80.02**	14.18	**89.00**	26.94	**97.20**
Holepuncher	0.30	**42.63**	18.23	**85.54**	38.75	**96.29**
Iron	8.86	**74.97**	59.26	**98.88**	88.31	**99.39**
Lamp	8.20	**71.11**	57.64	**98.85**	81.03	**99.62**
Phone	0.18	**47.74**	35.55	**91.07**	61.22	**98.85**
Average	2.42	**55.95**	31.65	**88.25**	54.29	**96.78**

Table 6.3: **Comparison of our single shot pose estimation approach with SSD-6D [91] without refinement using different thresholds for the 6D pose metric.**

Method	w/o Refinement		w/ Refinement
Object	SSD-6D [91]	OURS	SSD-6D [91]
Ape	98.46	**99.81**	99
Benchvise	**100**	99.90	**100**
Cam	99.53	**100**	99
Can	**100**	99.81	**100**
Cat	99.34	**99.90**	99
Duck	99.04	**100**	98
Glue	97.24	**99.81**	98
Holepuncher	98.95	**99.90**	99
Iron	99.65	**100**	99
Lamp	99.38	**100**	99
Phone	99.91	**100**	100
Average	99.22	**99.92**	99.4
Driller	-	**100**	99
Eggbox	-	**99.91**	99

Table 6.4: **Comparison of our single shot pose estimation approach against the state-of-the-art [91] on LINEMOD using IoU metric.** The authors of [91] were able to provide us the results of our approach w/o the refinement.

We further augment the LINEMOD training data by adding into the images objects extracted from other training sequences. We report our pose estimation accuracy in Figure 6.4 and demonstrate that even without assuming ground-truth information as in the case of [152], our method yields satisfactory pose accuracy in the case of severe occlusions. For object detection purposes, we consider an estimate to be correct if its detection IoU is larger than 0.5. Note that here the detection IoU corresponds to the overlap of the 2D bounding boxes of the object, rather than the overlap of the projected masks as is the case for the IoU metric defined in Sec 6.3.2. In Table 6.6, we report a mean average precision (MAP) of 0.48 which is similar to the accuracy reported by [21] and outperforms the ones reported by [68, 91].

Our approach provides accurate 6D poses with real-time performance. Upon one network

Method	Overall Speed	Refinement runtime
Brachmann et al. [21]	2 fps	100 ms/object
Rad & Lepetit [152]	3 fps	21 ms/object
Kehl et al. [91]	10 fps	24 ms/object
OURS	50 fps	-

Table 6.5: **Comparison of the overall computational runtime of our single shot pose estimation approach in comparison to the state-of-the-art [21, 91, 152].** We further provide the computational runtime induced by the pose refinement stage of [21, 91, 152]

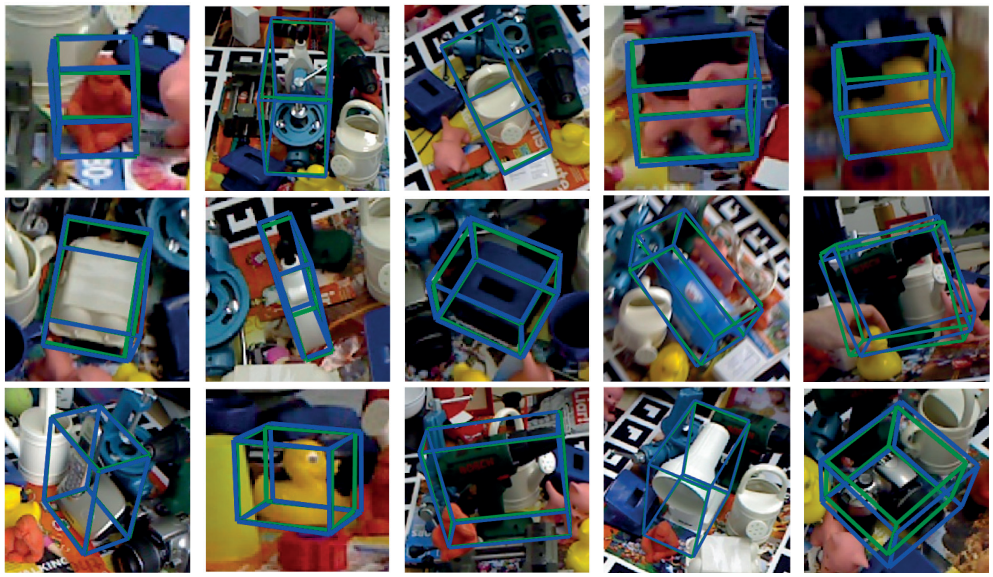


Figure 6.3: **Example results of our single shot pose estimation approach.** Note that our method can recover the 6D pose in these challenging scenarios, which involve significant amounts of clutter, occlusion and orientation ambiguity. In the last column, we show failure cases due to motion blur, severe occlusion and specularity (this figure is best viewed on a computer screen).

Method	MAP
Hinterstoisser et al. [68]	0.21
Brachmann et al. [21]	0.51
Kehl et al. [91]	0.38
OURS	0.48

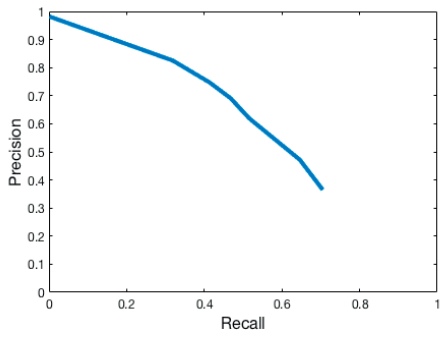


Table 6.6: **The object detection experiment on the Occlusion dataset [21].** (Left) Precision-recall plot. (Right)

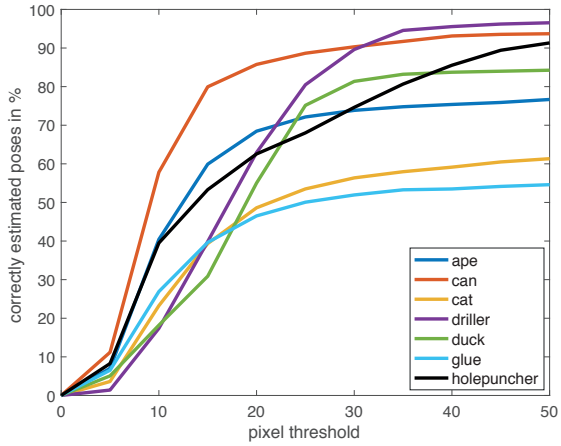


Figure 6.4: **Percentage of correctly estimated poses as a function of the projection error for different objects of the Occlusion dataset [21].**

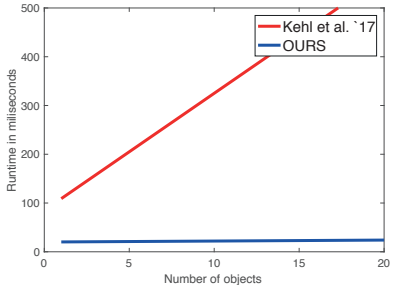


Figure 6.5: **The runtime of our approach with increasing number of objects as compared to the state-of-the-art [91].**

invocation, our only computational overhead is an efficient PnP algorithm which operates on just 9 points per object. Furthermore we do not require full 3D colored object models to further refine our initial pose estimates. Our approach is therefore scalable to handle multiple objects as shown in Figure 6.5 and has only a negligible computational overhead of PnP (0.2 miliseconds/object) while the competing approaches [91] have a linear runtime growth.

We also evaluated the accuracy and speed of our approach for different input resolutions. As explained in Section 6.1.1, we adopt a multi-scale training procedure and change the input resolution during training randomly as in [158]. This allows us to be able to change the input resolution at test-time and predict from images with higher resolution. This is especially useful for predicting the pose of small objects more robustly. As we do not have an initial step for 2D object detection and produce image crops which are then resized to higher resolutions for pose prediction as in [152], our approach requires better handling of the small objects. In Table 6.7, we compare the accuracy and computational efficiency of our approach for different input resolutions. With only 1% decrease in accuracy the average runtime per image is 94 ms and the runtime

virtually remains the same for estimating the pose of multiple objects.

Resolution	2D projection metric	Speed
416 × 416	89.71	94 fps
480 × 480	90.00	67 fps
544 × 544	90.37	50 fps
688 × 688	90.65	43 fps

Table 6.7: **Accuracy/speed trade-off of our single shot pose estimation method on the LINEMOD dataset.** Accuracy reported is the percentage of correctly estimated poses w.r.t the 2D projection error. The same network model is used for all four input resolutions. Timings are on a Titan X (Pascal) GPU.

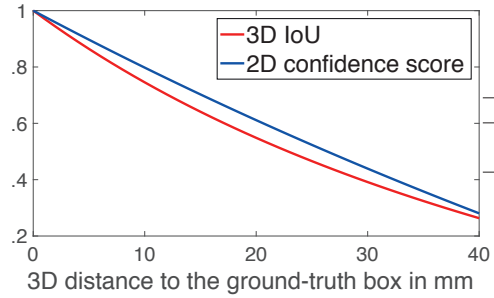
6.3.5 Further Analysis and Visualizations

Training Images. As discussed in Section 6.3.3, we segment the foreground object in the images of the training set, using the segmentation masks provided and paste the segmented image over a random image as in [21, 91, 152]. Examples of such images, which are given as input to the network at training time are shown in Figure 6.6. This operation of removing the actual background prevents the network from overfitting to the background, which is similar for training and test images of LINEMOD. When we train a model without eliminating the background, in practice, we observe about 1% improvement in the 2D projection score.

Figure 6.6: **Training images of our single shot pose estimation method.** Using segmentation masks given in LINEMOD, we extract the foreground objects in our training images and composite them over random images from PASCAL VOC [48]. We also augment the training set by combining images of multiple objects taken from different training images.

Confidence function. We analyze in Figure 6.7 our confidence function in comparison to 3D cube IoU in terms of its value and runtime. We show that our confidence function closely approximates the actual 3D cube IoU while being much faster to compute.

Confidence-weighted prediction. In the final step of our method, we compute a weighted sum of multiple sets of predictions for the corners and the centroid, using associated confidence values as weights. On LINEMOD, this gave a 1–2% improvement in accuracy with the 2D projection metric. The first step involves scanning the full 17×17 grid to find the cell with the highest



Method	Runtime per object (ms)
3D IoU	5.37
2D Conf. Score	0.18

Figure 6.7: **Comparison of the 3D IoU and our 2D confidence score in value (Left) and runtime (Right).** The model for the *Cam* object is shifted in x-dimension synthetically to produce a distorted prediction and projected on the image plane with randomly chosen 20 transformation matrices from LINEMOD. Scores are computed between the ground-truth references and distorted predictions. Results are averaged over all the trials. The runtime for 3D IoU is computed using the optimized PyGMO library that relies on [23].

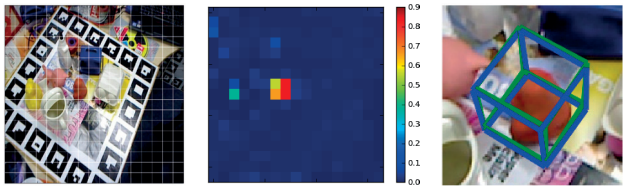


Figure 6.8: **Confidence weighted prediction of our single shot pose estimation method.** (Left) The 17×17 grid on a 544×544 image. (Middle) Confidence values for predictions of the *ape* object on the grid. (Right) Cropped view of our pose estimate (shown in blue) and the ground truth (shown in green). Here, three cells next to the best cell have good predictions and their combination gives a more accurate pose than the best prediction alone (best viewed in color).

confidence for each potential object. We then consider a 3×3 neighborhood around it on the grid and prune the cells with confidence values lower than the detection threshold of 0.5. On the remaining cells, we compute a confidence-weighted average of the associated predicted 18-dimensional vectors, where the eight corner points and the centroid have been stacked to form the vector. The averaged coordinates are then used in the PnP method. This *s*ub-pixel refinement on the grid usually improves the pose of somewhat large objects that occupy several adjoining cells in the grid. Figure 6.8 shows an example where the *ape* object lies between two adjoining cells and the confidence weighting improves the pose accuracy.

Qualitative Results. We show additional qualitative results from the OCCLUSION [20] and LINEMOD [70] datasets in Figures 6.9 to 6.14. These examples show that our method is robust to severe occlusions, rotational ambiguities in appearance, reflections, viewpoint change and scene clutter.

Figure 6.9: **Results of our single shot pose estimation method on the OCCLUSION dataset (I).** Our method is quite robust against severe occlusions in the presence of scene clutter and rotational pose ambiguity for symmetric objects. (left) Input images, (middle) 6D pose predictions of multiple objects, (right) A magnified view of the individual 6D pose estimates of six different objects is shown for clarity. In each case, the 3D bounding box is rendered on the input image. The following color coding is used – APE (gold), BENCHVISE (green), CAN (red), CAT (purple), DRILLER (cyan), DUCK (black), GLUE (orange), HOLEPUNCHER (blue). In addition to the objects from the OCCLUSION dataset, we also visualize the pose predictions of the *Benchvise* object from the LINEMOD dataset. As in [152], we do not evaluate on the *Eggbox* object, as more than 70% of close poses are not seen in the training sequence. This image is best viewed on a computer screen.

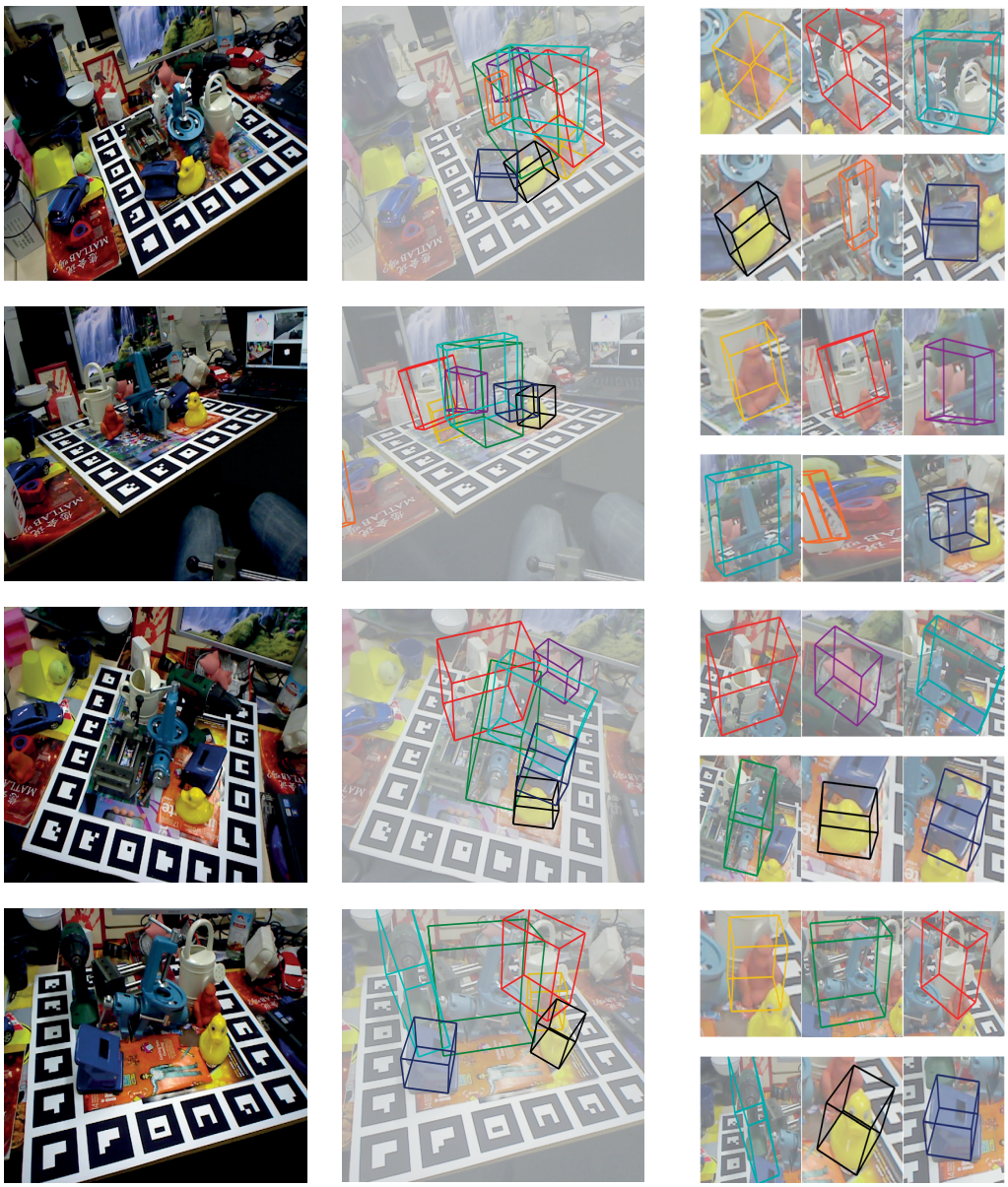


Figure 6.10: **Results of our single shot pose estimation method on the OCCLUSION dataset (II).** Our method is quite robust against severe occlusions in the presence of scene clutter and rotational pose ambiguity for symmetric objects. (left) Input images, (middle) 6D pose predictions of multiple objects, (right) A magnified view of the individual 6D pose estimates of six different objects is shown for clarity. In each case, the 3D bounding box is rendered on the input image. The following color coding is used – APE (gold), BENCHVISE (green), CAN (red), CAT (purple), DRILLER (cyan), DUCK (black), GLUE (orange), HOLEPUNCHER (blue). In addition to the objects from the OCCLUSION dataset, we also visualize the pose predictions of the *Benchvise* object from the LINEMOD dataset. As in [152], we do not evaluate on the *Eggbox* object, as more than 70% of close poses are not seen in the training sequence. This image is best viewed on a computer screen.

Figure 6.11: **Example results of single shot pose estimation method on the LINEMOD dataset (I).** (left) APE, (middle) BENCHVISE, (right) CAM. The projected 3D bounding boxes are rendered over the image and they have been cropped and resized for ease of visualization. The blue cuboid is rendered using our pose estimate whereas the green cuboid is rendered using the ground truth object pose. Note that the input image dimension is 640 × 480 pixels and the objects are often quite small. Noticeable scene clutter and occlusion makes these examples challenging.

Figure 6.12: **Example results of single shot pose estimation method on the LINEMOD dataset (II).** (left) CAN, (middle) CAT, (right) DRILLER. The projected 3D bounding boxes are rendered over the image and they have been cropped and resized for ease of visualization. The blue cuboid is rendered using our pose estimate whereas the green cuboid is rendered using the ground truth object pose. Note that the input image dimension is 640 × 480 pixels and the objects are often quite small. Noticeable scene clutter and occlusion makes these examples challenging.

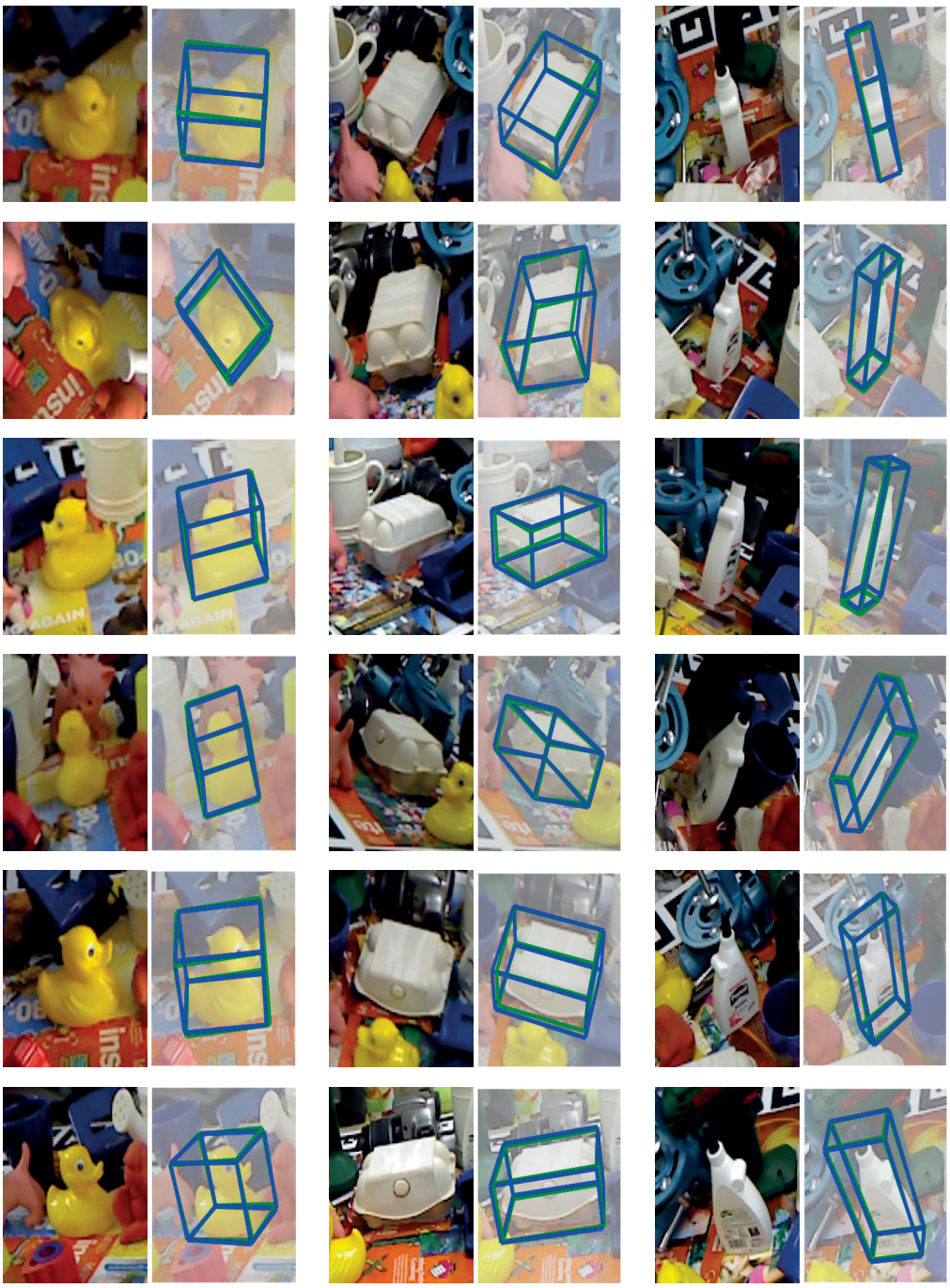


Figure 6.13: **Example results of single shot pose estimation method on the LINEMOD dataset (III).** (left) DUCK, (middle) EGGBOX, (right) GLUE. The projected 3D bounding boxes are rendered over the image and they have been cropped and resized for ease of visualization. The blue cuboid is rendered using our pose estimate whereas the green cuboid is rendered using the ground truth object pose. Note that the input image dimension is 640 × 480 pixels and the objects are often quite small. Noticeable scene clutter and occlusion makes these examples challenging.

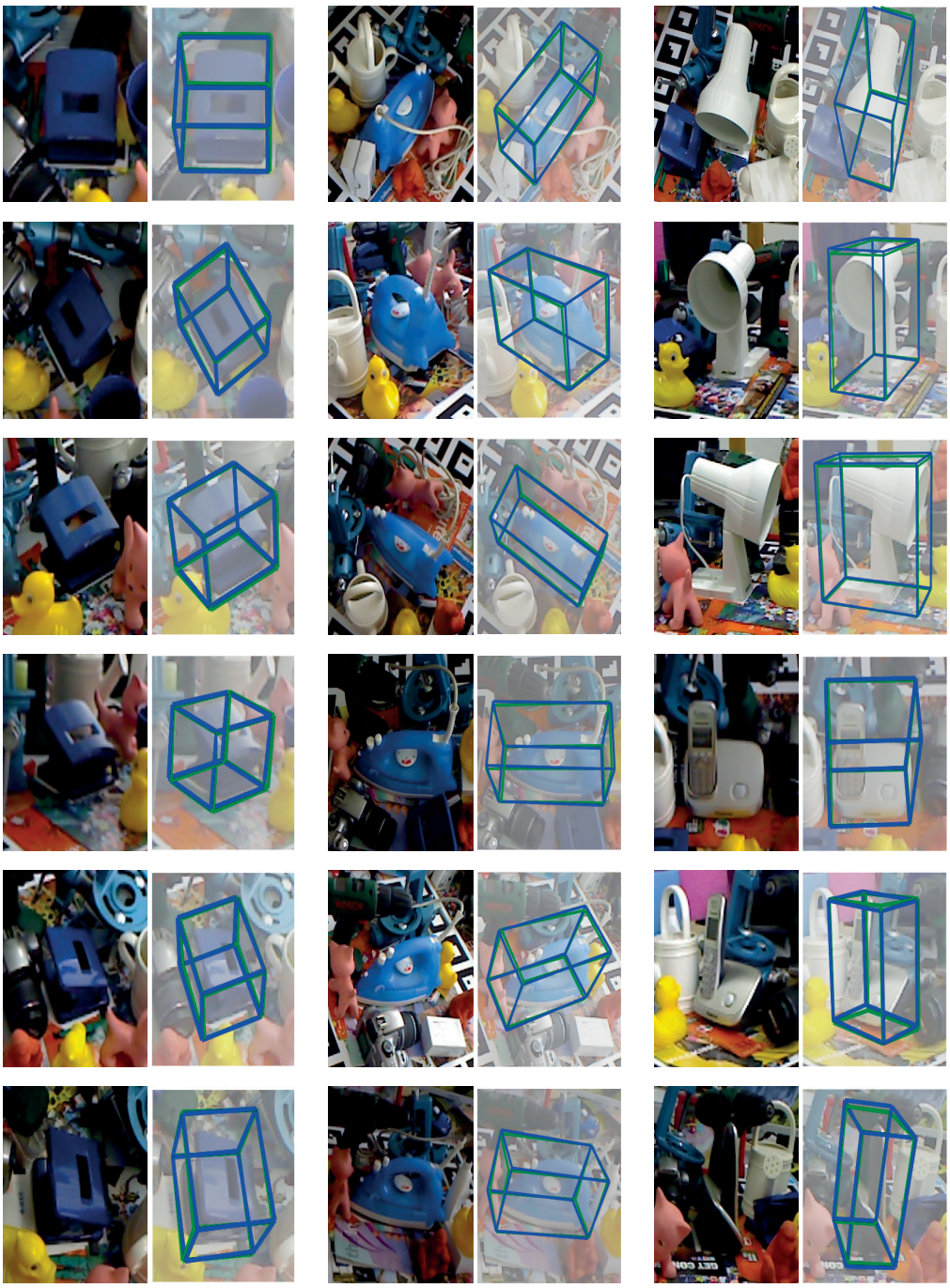


Figure 6.14: **Example results of single shot pose estimation method on the LINEMOD dataset (IV).** (left) HOLEPUNCHER, (middle) IRON, (right) LAMP and PHONE. The projected 3D bounding boxes are rendered over the image and they have been cropped and resized for ease of visualization. The blue cuboid is rendered using our pose estimate whereas the green cuboid is rendered using the ground truth object pose. Note that the input image dimension is 640 × 480 pixels and the objects are often quite small. Noticeable scene clutter and occlusion makes these examples challenging.

7 Concluding Remarks

In this thesis, we have presented several solutions to the 3D pose estimation, which is a challenging problem that finds several applications in augmented reality and human-computer interaction. In the following, we first give a brief summary of the achievements and contributions presented in this thesis. We then discuss some limitations of our approaches and identify potential directions for future research.

7.1 Summary

In Chapter 3, we introduce a novel video-based 3D human pose estimation approach from spatiotemporal features. Majority of existing video-based approaches rely on pose estimates in individual frames and regularize them to enforce temporal consistency. When the individual pose estimates are not accurate enough, regularizing them a posteriori, corrects for the errors only to a limited extent and results in inaccurate pose estimation performance. By contrast, we propose to exploit the temporal information very early in the pose estimation process by extracting spatiotemporal features. This allows us to encode richer and more descriptive features than the ones that could be obtained from single images. We further develop a CNN-based motion compensation approach that improves the reliability and stability of spatiotemporal features by factoring out the camera and global body motion, while preserving nonrigid motions that serve as useful cues for pose estimation.

In Chapter 4, recent 3D human pose estimation approaches either train a Convolutional Neural Network to directly regress from image to 3D pose, which ignores the dependencies between human joints, or model these dependencies via a max-margin structured learning framework, which involves a high computational cost at inference time. We introduce a Deep Learning regression architecture for structured prediction of 3D human pose from monocular images that

relies on an overcomplete autoencoder to learn a high-dimensional latent pose representation and account for joint dependencies. We demonstrate that our approach outperforms state-of-the-art ones both in terms of structure preservation and prediction accuracy.

In Chapter 5, we present a two-stream CNN that simultaneously exploits 2D pose estimates and image features. We demonstrate that the intermediate 2D pose supervision improves the 3D pose estimation accuracy significantly. We further demonstrate that the choice of where and how to combine features of a two-stream CNN is critical. We propose a *trainable fusion* scheme that learns where to fuse information coming from two different input modalities. This then allows to select the optimal network architecture on-the-fly and improves the pose estimation accuracy in comparison to standard hard-coded network architectures.

In Chapter 6, we present a real-time single shot approach for 3D object pose estimation. Previous approaches typically rely on heavy pipelines that involve consecutive 2D object detection and 3D pose estimation stages. Instead, we propose to simultaneously detect an object in 2D and predict its 3D pose in one shot. At the heart of our approach is a new CNN architecture that directly predicts the 2D image locations of the projected vertices of the object's 3D bounding box. The 3D pose of the object is then estimated using a Perspective-n-Point algorithm. While achieving state-of-the-art accuracy, our approach does not require any post-processing on initial poses. As a result, it is much faster and performs at real-time speed which existing methods fall short of.

7.2 Limitations and Future Directions

In this section we discuss the main limitations of the proposed methods and suggest potential directions for the future work.

Harvesting Ground-Truth 3D Pose Data. State-of-the-art methods for estimating the pose of objects [91, 198] and humans [192, 196] rely on deep learning techniques that require a large amount of training data. Currently, motion capture datasets, such as [81], provide large-scale data, however they are only limited to controlled studio environments. As a result, deep learning models trained on them do not generalize well enough to in-the-wild images. Because manual annotation of 3D poses is both tedious and time-consuming, the scale of the datasets captured in unconstrained outdoor environments is still far from enough to cover a wide range of articulations, appearances and backgrounds. However, recently there has been some progress in this direction by the means of automated annotation techniques that exploit multi-view data [139] or large-scale manual annotation efforts [64].

Weakly Supervised Learning for Generalizibility. At the lack of 3D ground-truth pose data in unconstrained outdoor environments, readily available 2D pose estimation datasets provide valuable mid and high level features for 3D pose estimation. To this end, a weakly-supervised transfer learning approach that uses mixed 2D and 3D data, as in [124, 231], would allow us to have additional 2D pose supervision and transfer 3D pose labels in controlled lab environments to in-the-wild images. One could also use additional *depth* information from RGB-D data or crude *body part segmentation* annotations as a form of weak supervision for the 3D pose estimation task.

Multiple 3D Human Pose Estimation and Modelling Human-Human Interactions. In Chapter 3, 4 and 5, we have assumed that we have a rough bounding box around the subject and estimated the 3D pose of a single person from the image crop. Our algorithms could directly be used for multi-person 3D pose estimation by operating on individual bounding boxes. However, this would discard the scene context and interactions of people in the scene which are valuable cues for 3D human pose estimation. To both benefit from scene context and interactions, one could jointly predict the pose of multiple people from the full image with a single-shot architecture [115]. Furthermore, our autoencoder-based algorithm to model the dependencies among different body parts could be extended to also account for the dependencies between the poses of multiple interacting people.

3D Human Body Shape Estimation. We have formulated the 3D human pose estimation problem in terms of predicting the 3D joint locations of a person. Although this captures the configuration of body parts, it does not encode the shape information. 3D shape information is important for applications such as augmented reality, virtual try-on and health monitoring. One exciting direction for future research is to predict the full 3D shape of a person by a voxelized output representation or fitting a detailed 3D body model to the initial 3D pose predictions.

Modelling Human-Object Interactions. We have so far addressed the human and object pose estimation problems individually. However, these two problems are highly intertwined. For example, if we are sitting on a chair, our pose significantly constrains the 3D pose of the chair. Similarly, the 3D pose of the object provides helpful cues to disambiguate the 3D human pose. One very promising direction thus is to jointly solve these two problems. It would ultimately allow us to have a more semantic understanding of the scene, introduce additional constraints on the pose of interacting entities and improve pose estimation accuracy.

A Analysis of Spatiotemporal Feature Regression and Motion Compensation for 3D Human Pose Estimation

In this appendix, we first describe implementation details of our motion compensation algorithm. Then, we provide further visualizations and analysis of our 3D human pose estimation approach.

A.1 Analysis on Motion Compensation with CNNs

Centering the body. As explained in Section 3.1.3, we train Convolutional Neural Networks to predict the shifts of the person from the center of the bounding box. In order to obtain training images centered on the subject, we use the foreground masks that are part of the datasets to compute and center the bounding box at the root position of the person.

Scale of the person. We rely on masks' height to compute the scale of the person. While it gives only a rough estimate, it is sufficient to handle scale changes when they occur. In the future, we plan to train a single regressor to compensate for both shift and scale changes.

Initialization. In our approach, DPM (trained on VOC 2010) is used in the first image of a sequence to provide an initial estimate of the bounding box. The initial person detector provides rough location estimates of the person and our motion compensation algorithm naturally compensates even for relatively large positional inaccuracies using the regressor, ψ_{coarse}, as can be seen in Fig. A.1.

Motion Compensation vs. Centering the Detections at Each Frame. For the alignment of the body across time, it is also possible to compute the center of the root part of a DPM-based pose estimator. However, it is time-consuming and computationally heavy to detect body parts at each frame. Therefore, instead of computing DPM in all the frames of the sequence, we use it *only in the first frame* and use motion compensation to iteratively center the body in subsequent frames. This is a more efficient and elegant solution than detecting body parts at each time

Figure A.1: **Examples of our motion compensation algorithm.** For each pair of images, the left one depicts the initial bounding box, and the right one depicts the aligned bounding box using our motion compensation algorithm.

instant. In order to justify the efficiency of our CNN-based motion compensation approach (CNN-MC), we compare its timing to that of DPM detections in Table A.1. Additionally, we provide comparisons to the timings of conventional optical-flow techniques [24, 119, 136]. We show that our approach to image alignment is substantially faster than these approaches.

Method:	Time (sec)
DPM [50], run sequentially	9.645
Large Displacement Optical Flow [24]	0.967
Lucas-Kanade Optical Flow [119, 136]	0.140
CNN-MC	0.006

Table A.1: **Timings (in seconds per image) of our motion compensation algorithm (CNN-MC) in comparison to DPM body part detector [50] and optical-flow [24, 119, 136].** Our motion compensation algorithm aligns the body in subsequent frames by shifting the body to the center of the bounding box. DPM takes the center of the root part of the part detector at each frame. Our approach to aligning the body is orders of magnitude faster.

Temporal Heuristic. Although we treat each frame independently, we exploit additional temporal heuristic in our approach by initializing the motion compensation algorithm using the bounding box from the previous frame. Simultaneous motion compensation in multiple frames not only increases model complexity but could yield incorrect estimates when the motion direction changes fast.

A.2 Further Analysis and Visualizations

In this section, we provide additional analysis of our experimental results and further visualizations for our 3D body pose recovery method.

Evaluation. The parameters of Deep Network regressor are cross-validated on a validation set and used for all the actions in the dataset. We consider the average error excluding the first and last $T/2$ frames (0.24 seconds for $T = 24$ at 50 fps) to evaluate the performance.

Additional Comparisons on HumanEva-I. On HumanEva [180], we trained our regressors on training sequences of subjects S1, S2 and S3 and evaluated on the "validation" sequences as in [13, 16, 46, 183] as explained in Section 4.2. [6, 222] followed a different experimental procedure where they use the same subject for training and testing purposes. In order to compare our results to these baselines as well, we employ the same subject-specific experimental setup and provide analysis in Table A.2. The results demonstrate that our method yields state-of-the-art 3D human pose estimation accuracy, also with this experimental setting.

Method:	S1	S2	S3	Average
Yao et al. [222]	41.6	64.0	46.5	50.7
Amin et al. [6]	56.7	52.1	62.4	57.1
Ours	38.4	27.9	52.1	**39.5**

Table A.2: **Additional comparisons of our RSTV-Regression approach against existing approaches.** We report 3D joint position errors (in mm) on the *Walking* sequences of HumanEva-I. We compare our approach against [6, 222].

Additional Comparisons on Human3.6m. [109] is a recently published structured deep learning method and uses the correlations among joint points for 3D human pose estimation. As shown in Section 3.2.2, we outperform all pose estimation methods on Human3.6m, HumanEva or KTH Multiview Football II that do not use structural dependencies. In Table A.3, we further show that we also outperform [109] on average over the action classes for which the authors reported accuracy numbers even though our algorithm do not rely on using the dependencies among the human body parts.

Method:	Discussion	Eating	Greeting	Taking Photo	Walking	Walking Dog	Average
Li et al. [109]	**136.88**	96.94	**124.74**	**168.68**	69.97	132.17	121.56
RSTV+DN (Ours)	147.72	**88.83**	125.28	182.73	**55.07**	**126.29**	**120.98**

Table A.3: **Additional comparisons of our RSTV-Regression approach against existing approaches on HumanEva-I.** We report 3D joint position errors (in mm) on Human3.6m. We compare our approach against [109].

Stability. 3D pose predictions of our approach are stable as can be seen in accompanying videos, as they are obtained for each overlapping temporal window with 1 frame shift. The comparison in Fig. A.2 further demonstrates that RSTV+DN obtains the most stable and accurate predictions.

Generalization. We demonstrate the generalization ability of our approach in the following ways.

- HumanEva-II provides only a test dataset and no training data, therefore, we trained our

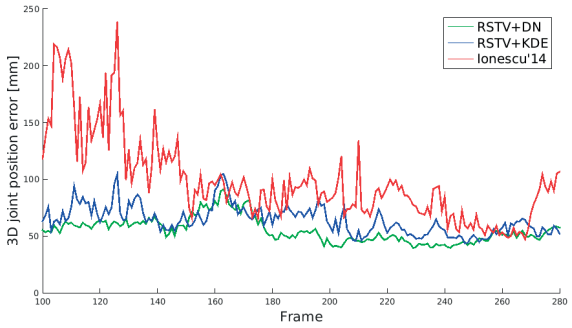
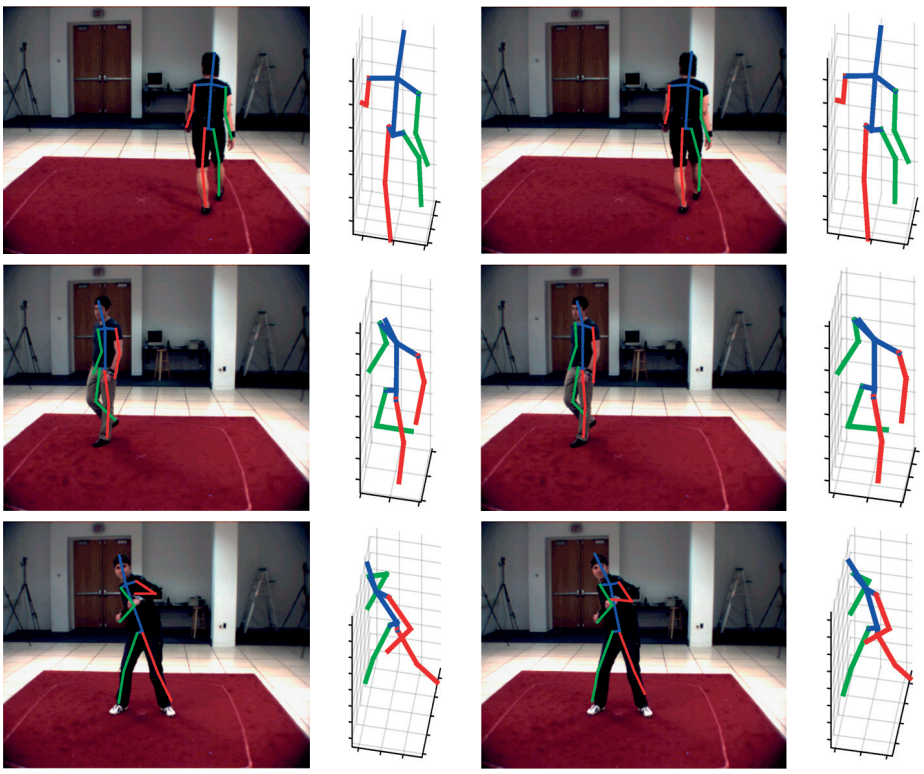


Figure A.2: **3D joint position errors across frames for our RSTV-Regression approach.** We report our results on the *Walking* sequence corresponding to Subject 9, Trial 1, Camera 1 in the Human3.6m dataset and compare our RSTV+KDE and RSTV+DN methods to [81]. RSTV+DN yields the best accuracy on average with the added advantage of temporal consistency. Best viewed in color.

regressors on HumanEva-I using videos captured from different camera views;

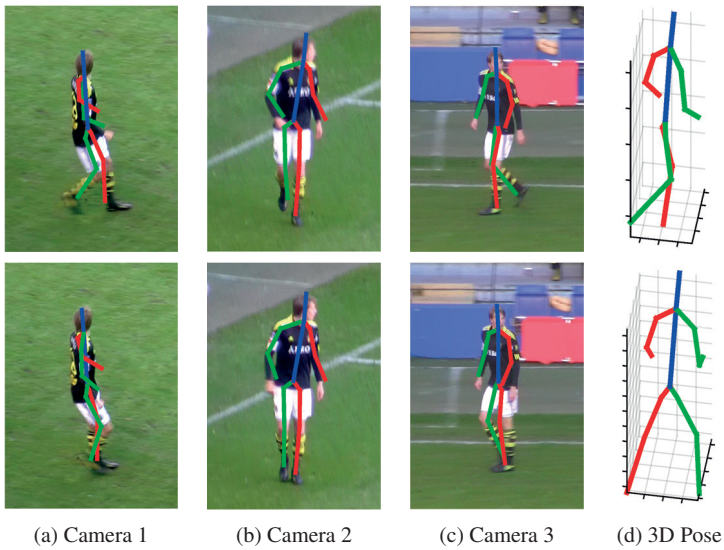
- Data from Human3.6m exhibit large variations in terms of body shapes, clothing, poses and viewing angles within and across training/test splits [81]. Also, different people appear in the training and test data.
- The size of the training set in HumanEva is too small to train a deep network. However, we tested on HumanEva-I using Deep Network regressors trained on Human3.6m and report a pose estimation accuracy of 75.4 mm. on Subject 1. As skeleton configurations for Human3.6m and HumanEva do not exactly match each other, the error contains a constant offset. However, we still obtain accurate pose estimates and outperform [185], which reports a 99.6 mm. accuracy, even though it is trained on HumanEva.

Visualization. We provide additional qualitative results for the Human3.6m, HumanEva and KTH Multiview Football II datasets in Fig. A.5, Fig. A.3 and Fig. A.4.



(a) Ground Truth (b) Our method

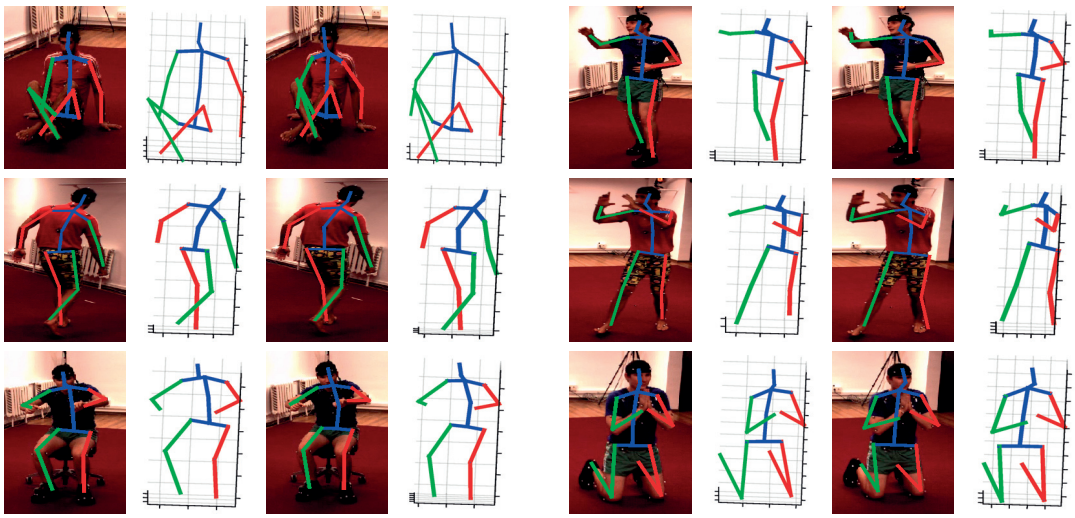
Figure A.3: **3D human pose estimation results of RSTV-Regression on HumanEva.** The rows correspond to the *Walking* and *Box* actions. **(a)** Reprojection in the images and ground-truth 3D pose. **(b)** The skeletons recovered by our method and their projection on the image plane. Best viewed in color.



(a) Camera 1 (b) Camera 2 (c) Camera 3 (d) 3D Pose

Figure A.4: **Example 3D human pose estimation results of RSTV-Regression on KTH Multiview Football II.** The 3D skeletons are recovered from Camera 1 images and projected on those of Camera 2 and 3, which were not used to compute the poses. Best viewed in color.

Appendix A. Analysis of Spatiotemporal Feature Regression and Motion Compensation for 3D Human Pose Estimation



(a) Ground Truth (b) Our method (c) Ground Truth (d) Our method

Figure A.5: **3D human pose estimation with RSTV-Regression on Human3.6m for several different action categories. (a,c)** Ground-truth 3D poses and their projection in the images. **(b,d)** The skeletons recovered by our method and their projection on the image plane. We can reliably recover the 3D pose of the body in case of ambiguities, such as self-occlusions and mirroring. Best viewed in color.

Bibliography

[1] A. Agarwal and B. Triggs. 3D Human Pose from Silhouettes by Relevance Vector Regression. In *CVPR*, 2004.

[2] A. Agarwal and B. Triggs. A Local Basis Representation for Estimating Human Pose from Cluttered Images, 2006.

[3] A. Agarwal and B. Triggs. Recovering 3D Human Pose from Monocular Images. *PAMI*, 28(1):44–58, 2006.

[4] J. Aggarwal and Q. Cai. Human Motion Analysis: A Review. In *Nonrigid and Articulated Motion Workshop*, 1997.

[5] I. Akhter and M. J. Black. Pose-Conditioned Joint Angle Limits for 3D Human Pose Reconstruction. In *CVPR*, 2015.

[6] S. Amin, M. Andriluka, M. Rohrbach, and B. Schiele. Multi-View Pictorial Structures for 3D Human Pose Estimation. In *BMVC*, 2013.

[7] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In *CVPR*, 2014.

[8] M. Andriluka, S. Roth., and B. Schiele. Monocular 3D Pose Estimation and Tracking by Detection. In *CVPR*, 2010.

[9] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: Shape Completion and Animation of PEople. *ACM Transactions on Graphics (TOG)*, 24(3):408–416, 2005.

[10] Ascension. trakStar Tracking Systems. https://www.ascension-tech.com/products/trakstar-drivebay/. Accessed: 2018-05-13.

[11] A. Baak, M. Müller, G. Bharaj, H. P. Seidel, and C. Theobalt. A Data-Driven Approach for Real-Time Full Body Pose Reconstruction from a Depth Camera. In *ICCV*, 2011.

[12] A. O. Balan, L. Sigal, M. J. Black, J. E. Davis, and H. W. Haussecker. Detailed Human Shape and Pose from Images. In *CVPR*, 2007.

[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] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic. 3D Pictorial Structures Revisited: Multiple Human Pose Estimation. *PAMI*, 38(10):1929–1942, 2015.

[15] V. Belagiannis, X. Wang, B. Schiele, P. Fua, S. Ilic, and N. Navab. Multiple Human Pose Estimation with Temporally Consistent 3D Pictorial Structures. In *ECCV*, September 2014.

[16] L. Bo and C. Sminchisescu. Twin Gaussian Processes for Structured Prediction. *IJCV*, 2010.

[17] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep It SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image. In *ECCV*, 2016.

[18] F. Bogo, J. Romero, M. Loper, and M. J. Black. FAUST: Dataset and Evaluation for 3D Mesh Registration. In *CVPR*, 2014.

[19] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black. Dynamic FAUST: Registering Human Bodies in Motion. In *CVPR*, 2017.

[20] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D Object Pose Estimation Using 3D Object Coordinates. In *ECCV*, 2014.

[21] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother. Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image. In *CVPR*, 2016.

[22] J. Brauer, W. Gong, J. Gonzalez, and M. Arens. On the Effect of Temporal Information on Monocular 3D Human Pose Estimation. In *ICCV*, 2011.

[23] K. Bringmann and T. Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Computational Geometry: Theory and Applications*, 43:601–610, 2010.

[24] T. Brox and J. Malik. Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation. *PAMI*, 33(3):500–513, 2011.

[25] A. Bulat and G. Tzimiropoulos. Human Pose Estimtation via Convolutional Part Heatmap Regression. In *ECCV*, 2016.

[26] M. Burenius, J. Sullivan, and S. Carlsson. 3D Pictorial Structures for Multiple View Articulated Pose Estimation. In *CVPR*, 2013.

[27] X. Burgos-Artizzu, D. Hall, P. Perona, and P. Dollár. Merging Pose Estimates Across Space and Time. In *BMVC*, 2013.

[28] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human Pose Estimation with Iterative Error Feedback. In *CVPR*, 2016.

[29] S. Chang, R. Ji, and Y. Wang. Label Propagation from Imagenet to 3D Point Clouds. In *CVPR*, pages 3135–3142, 2013.

[30] W. Chen, H. Wang, Y. Li, H. Su, Z. Wang, C. Tu, D. Lischinski, D. Cohen-or, and B. Chen. Synthesizing Training Images for Boosting Human 3D Pose Estimation. In *3DV*, 2016.

[31] X. Chen and A. L. Yuille. Articulated Pose Estimation by a Graphical Model with Image Dependent Pairwise Relations. In *NIPS*, 2014.

[32] Y. Chen, T. Kim, and R. Cipolla. Inferring 3D Shapes and Deformations from Single Views. In *ECCV*, 2010.

[33] C. Choi and H. I. Christensen. 3D Textureless Object Detection and Tracking: An Edge-Based Approach. In *IROS*, 2012.

[34] C. Choi and H. I. Christensen. RGB-D Object Pose Estimation in Unstructured Environments. *Robotics and Autonomous Systems*, 75:595–613, 2016.

[35] X. Chu, W. Ouyang, H. Li, and X. Wang. Structured Feature Learning for Pose Estimation. In *CVPR*, 2016.

[36] A. Collet, M. Martinez, and S. S. Srinivasa. The MOPED Framework: Object Recognition and Pose Estimation for Manipulation. *The International Journal of Robotics Research*, 30(10):1284–1306, 2011.

[37] C. Cortes, M. Mohri, and J. Weston. A General Regression Technique for Learning Transductions. In *ICML*, 2005.

[38] A. Criminisi and J. Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013.

[39] A. Crivellaro and V. Lepetit. Robust 3D Tracking with Descriptor Fields. In *CVPR*, 2014.

[40] A. Crivellaro, Y. Verdie, K. Yi, P. Fua, and V. Lepetit. [DEMO] Tracking Texture-Less, Shiny Objects with Descriptor Fields. In *ISMAR*, 2014.

[41] B. Daubney and X. Xie. Tracking 3D Human Pose with Large Root Node Uncertainty. In *CVPR*, 2011.

[42] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. In *CVPR*, 2015.

[43] M. Du and R. Chellappa. Face Association Across Unconstrained Video Frames Using Conditional Random Fields. In *ECCV*, 2012.

[44] Y. Du, Y. Wong, Y. Liu, F. Han, Y. Gui, Z. Wang, M. Kankanhalli, and W. Geng. Marker-Less 3D Human Motion Capture with Monocular Image Sequence and Height-Maps. In *ECCV*, 2016.

[45] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing Action at a Distance. In *ICCV*, pages 726–733, October 2003.

[46] A. Elhayek, E. Aguiar, A. Jain, J. Tompson, L. Pishchulin, M. Andriluka, C. Bregler, B. Schiele, and C. Theobalt. Efficient Convnet-Based Marker-Less Motion Capture in General Scenes with a Low Number of Cameras. In *CVPR*, 2015.

[47] A. Elhayek, E. Aguiar, A. Jain, J. Tompson, L. Pishchulin, M. Andriluka, C. Bregler, B. Schiele, and C. Theobalt. MARCOnI: ConvNet-Based MARker-Less Motion Capture in Outdoor and Indoor Scenes. *PAMI*, 39(3):501–514, 2017.

[48] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (Voc) Challenge. *IJCV*, 88(2):303–338, 2010.

[49] X. Fan, K. Zheng, Y. Zhou, and S. Wang. Pose Locality Constrained Representation for 3D Human Pose Reconstruction. In *ECCV*, 2014.

[50] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part Based Models. *PAMI*, 32(9):1627–1645, 2010.

[51] V. Ferrari, M. Martin, and A. Zisserman. Progressive Search Space Reduction for Human Pose Estimation. In *CVPR*, 2008.

[52] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent Network Models for Human Dynamics. In *ICCV*, 2015.

[53] O. Gabai and H. Primo. Acoustic motion capture, Jan. 13 2011. US Patent App. 12/746,532.

[54] J. Gall, B. Rosenhahn, T. Brox, and H.-P. Seidel. Optimization and Filtering for Human Motion Capture. *IJCV*, 2010.

[55] J. Gall, A. Yao, and L. Van Gool. 2D Action Recognition Serves 3D Human Pose Estimation. In *ECCV*, 2010.

[56] S. Gammeter, A. Ess, T. Jaeggli, K. Schindler, B. Leibe, and L. Van Gool. Articulated Multi-Body Tracking Under Egomotion. In *ECCV*, 2008.

[57] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real-Time Human Pose Tracking from Range Data. In *ECCV*, 2012.

[58] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient Regression of General-Activity Human Poses from Depth Images. In *ICCV*, 2011.

[59] G. Gkioxari, A. Toshev, and N. Jaitly. Chained Predictions Using Convolutional Neural Networks. In *ECCV*, 2016.

[60] X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *AISTATS*, 2011.

[61] A. Graves, S. Fernandez, and J. Schmidhuber. Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. In *ICANN*, 2005.

[62] D. Grest, V. Krüger, and R. Koch. Single View Motion Tracking by Depth and Silhouette Information. In *SCIA*, 2007.

[63] P. Guan, A. Weiss, A. Balan, and M. Black. Estimating Human Shape and Pose from a Single Image. In *ICCV*, 2009.

[64] R. Güler, N. Neverova, and I. Kokkinos. Densepose: Dense Human Pose Estimation in the Wild. *arXiv Preprint*, 2018.

[65] A. Haque, B. Peng, Z. Luo, A. Alahi, S. Yeung, and L. Fei-Fei. Towards Viewpoint Invariant 3D Human Pose Estimation. In *ECCV*, 2016.

[66] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016.

[67] T. Helten, A. Baak, G. Bharaj, M. Müller, H. P. Seidel, and C. Theobalt. Personalization and Evaluation of a Real-Time Depth Based Full Body Tracker. In *3DV*, 2014.

[68] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal Templates for Real-Time Detection of Texture-less Objects in Heavily Cluttered Scenes. In *ICCV*, 2011.

[69] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *ACCV*, 2012.

[70] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model Based Training, Detection and Pose Estimation of Texture-less 3D Objects in Heavily Cluttered Scenes. In *ACCV*, 2012.

[71] G. Hinton and R. Salakutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 2006.

[72] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[73] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis. T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects. In *WACV*, 2017.

[74] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel Methods in Machine Learning. *The Annals of Statistics*, 2008.

[75] C. Hong, J. Yu, J. Wan, D. Tao, and M. Wang. Multimodal Deep Autoencoder for Human Pose Recovery. *TIP*, 2014.

[76] N. R. Howe. A Recognition-Based Motion Capture Baseline on the Humaneva II Test Data. *MVA*, 2011.

[77] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *PAMI*, 15(9):850–863, 1993.

[78] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele. Deepercut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model. In *ECCV*, 2016.

[79] C. Ionescu, J. Carreira, and C. Sminchisescu. Iterated Second-Order Label Sensitive Pooling for 3D Human Pose Estimation. In *CVPR*, 2014.

[80] C. Ionescu, F. Li, and C. Sminchisescu. Latent Structured Models for Human Pose Estimation. In *ICCV*, 2011.

[81] C. Ionescu, I. Papava, V. Olaru, and C. Sminchisescu. Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. *PAMI*, 2014.

[82] A. Jain, T. Thormahlen, H. Seidel, and C. Theobalt. Moviereshape: Tracking and Reshaping of Humans in Videos. In *SIGGRAPH*, 2010.

[83] A. Jain, J. Tompson, M. Andriluka, G. W. Taylor, and C. Bregler. Learning Human Pose Estimation Features with Convolutional Networks. In *ICLR*, 2014.

[84] A. Jain, A. Zamir, S. Savarese, and A. Saxena. Structural-Rnn: Deep Learning on Spatio-Temporal Graphs. In *CVPR*, 2016.

[85] J. Johnson, A. Karpathy, and L. Fei-fei. Densecap: Fully Convolutional Localization Networks for Dense Captioning. In *CVPR*, 2016.

[86] S. Johnson and M. Everingham. Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation. In *BMVC*, 2010.

[87] H. Y. Jung, S. Lee, Y. H. Seok, and I. Y. Dong. Random Tree Walk Toward Instantaneous 3D Human Pose Estimation. In *CVPR*, 2015.

[88] H. Y. Jung, Y. Suh, G. Moon, and K. M. Lee. A Sequential Approach to 3D Human Pose Estimation: Separation of Localization and Identification of Body Joints. In *ECCV*, 2016.

[89] A. Kanaujia, C. Sminchisescu, and D. N. Metaxas. Semi-Supervised Hierarchical Models for 3D Human Pose Reconstruction. In *CVPR*, 2007.

[90] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-fei. Large-Scale Video Classification with Convolutional Neural Networks. In *CVPR*, 2014.

[91] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. In *ICCV*, 2017.

[92] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. Deep Learning of Local RGB-D Patches for 3D Object Detection and 6D Pose Estimation. In *ECCV*, 2016.

[93] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF camera relocalization. In *ICCV*, 2015.

[94] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimisation. In *ICLR*, 2015.

[95] A. G. Kirk and J. F. O. D. A. Forsyth. Skeletal Parameter Estimation from Optical Motion Capture Data. In *CVPR*, 2005.

[96] A. Kläser, M. Marszałek, and C. Schmid. A Spatio-Temporal Descriptor Based on 3D-Gradients. In *BMVC*, 2008.

[97] S. Kombrink, T. Mikolov, M. Karafiat, and L. Burget. Recurrent Neural Network Based Language Modeling in Meeting Recognition. In *INTERSPEECH*, 2011.

[98] K. Konda, R. Memisevic, and D. Krueger. Zero-Bias Autoencoders and the Benefits of Co-Adapting Features. In *ICLR*, 2015.

[99] I. Kostrikov and J. Gall. Depth Sweep Regression Forests for Estimating 3D Human Pose from Images. In *BMVC*, 2014.

[100] K. Lai, L. Bo, X. Ren, and D. Fox. A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. In *ICRA*, 2011.

[101] K. Lai, L. Bo, X. Ren, and D. Fox. A Scalable Tree-Based Approach for Joint Object and Pose Recognition. In *AAAI*, 2011.

[102] I. Laptev. On Space-Time Interest Points. *IJCV*, 64(2-3):107–123, 2005.

[103] C. Lassner, J. Romero, M. Kiefel, F. Bogo, M. Black, and P. Gehler. Unite the People: Closing the Loop Between 3D and 2D Human Representations. In *CVPR*, 2017.

[104] V. Lepetit and P. Fua. *Monocular Model-Based 3D Tracking of Rigid Objects: A Survey*. Now Publishers, September 2005.

[105] V. Lepetit and P. Fua. Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, 2005.

[106] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An Accurate O(n) Solution to the PnP Problem. *IJCV*, 81(2):155–166, 2009.

[107] F. Li, G. Lebanon, and C. Sminchisescu. Asdf. In *CVPR*, 2012.

[108] S. Li and A. Chan. 3D Human Pose Estimation from Monocular Images with Deep Convolutional Neural Network. In *ACCV*, 2014.

[109] S. Li, W. Zhang, and A. B. Chan. Maximum-Margin Structured Learning with Deep Networks for 3D Human Pose Estimation. In *ICCV*, 2015.

[110] S. Li, W. Zhang, and A. B. Chan. Maximum-Margin Structured Learning with Deep Networks for 3D Human Pose Estimation. In *IJCV*, 2016.

[111] Y. Li, L. Gu, and T. Kanade. Robustly Aligning a Shape Model and Its Application to Car Alignment of Unknown Pose. *PAMI*, 33(9):1860–1876, 2011.

[112] Y. Li and R. S. Zemel. Mean Field Networks. In *ICML*, 2014.

[113] M. Liang and X. Hu. Recurrent Convolutional Neural Network for Object Recognition. In *CVPR*, 2015.

[114] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, and R. Chellappa. Fast Directional Chamfer Matching. In *CVPR*, 2010.

[115] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single Shot Multibox Detector. In *ECCV*, 2016.

[116] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. Black. SMPL: A Skinned Multi-Person Linear Model. *ACM SIGGRAPH Asia*, 34(6), 2015.

[117] D. G. Lowe. Fitting parameterized three-dimensional models to images. *PAMI*, 13(5):441–450, 1991.

[118] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[119] B. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *IJCAI*, pages 674–679, 1981.

[120] L. Maaten and G. Hinton. Visualizing High Dimensional Data Using t-SNE. *JMLR*, 2008.

[121] S. Mahendran, H. Ali, and R. Vidal. 3D Pose Regression using Convolutional Neural Networks. In *CVPRW*, 2017.

[122] E. Mansimov, N. Srivastava, and R. Salakhutdinov. Initialization Strategies of Spatio-Temporal Convolutional Neural Networks. *CoRR*, abs/1503.07274, 2015.

[123] J. Martinez, R. Hossain, J. Romero, and J. Little. A Simple Yet Effective Baseline for 3D Human Pose Estimation. In *ICCV*, 2017.

[124] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt. Monocular 3D Human Pose Estimation in the Wild Using Improved CNN Supervision. In *International Conference on 3D Vision*, 2017.

[125] MetaMotion. Gypsy 7 Motion Capture System. http://metamotion.com/gypsy/gypsy-motion-capture-system.htm. Accessed: 2018-05-13.

[126] F. Michel, A. Kirillov, E. Brachmann, A. Krull, S. Gumhold, B. Savchynskyy, and C. Rother. Global Hypothesis Generation for 6D Object Pose Estimation. In *CVPR*, 2017.

[127] Microsoft. HoloLens. https://www.microsoft.com/en-us/hololens. Accessed: 2018-05-17.

[128] G. Moon, J. Y. Chang, and K. M. Lee. V2V-PoseNet: Voxel-to-Voxel Prediction Network for Accurate 3D Hand and Human Pose Estimation from a Single Depth Map. In *CVPR*, 2018.

[129] F. Moreno-noguer. 3D Human Pose Estimation from a Single Image via Distance Matrix Regression. In *CVPR*, 2017.

[130] G. Mori and J. Malik. Estimating Human Body Configurations Using Shape Context Matching. In *ECCV*, 2002.

[131] G. Mori and J. Malik. Recovering 3D Human Body Configurations Using Shape Contexts. *PAMI*, 2006.

[132] A. Newell, K. Yang, and J. Deng. Stacked Hourglass Networks for Human Pose Estimation. In *ECCV*, 2016.

[133] H. Ning, W. Xu, Y. Gong, and T. Huang. Discriminative Learning of Visual Words for 3D Human Pose Estimation. In *CVPR*, 2008.

[134] M. Oberweger, P. Wohlhart, and V. Lepetit. Hands Deep in Deep Learning for Hand Pose Estimation. *arXiv Preprint*, abs/1502.06807, 2015.

[135] D. Ormoneit, H. Sidenbladh, M. Black, T. Hastie, and D. Fleet. Learning and Tracking Human Motion Using Functional Analysis. In *IEEE Workshop on Human Modeling, Analysis and Synthesis*, 2000.

[136] D. Park, C. L. Zitnick, D. Ramanan, and P. Dollár. Exploring Weak Stabilization for Motion Feature Extraction. In *CVPR*, pages 2882–2889, 2013.

[137] S. Park, J. Hwang, and N. Kwak. 3D Human Pose Estimation Using Convolutional Neural Networks with 2D Pose Information. In *ECCV*, 2016.

[138] G. Pavlakos, X. Zhou, K. Derpanis, G. Konstantinos, and K. Daniilidis. Coarse-To-Fine Volumetric Prediction for Single-Image 3D Human Pose. In *CVPR*, 2017.

[139] G. Pavlakos, X. Zhou, K. D. G. Konstantinos, and D. Kostas. Harvesting Multiple Views for Marker-Less 3D Human Pose Annotations. In *CVPR*, 2017.

[140] T. Pfister, J. Charles, and A. Zisserman. Flowing Convnets for Human Pose Estimation in Videos. In *ICCV*, 2015.

[141] P. Pinheiro and R. Collobert. Recurrent Neural Networks for Scenel Labelling. In *ICML*, 2014.

[142] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele. Deepcut: Joint Subset Partition and Labeling for Multi Person Pose Estimation. In *CVPR*, 2016.

[143] C. Plagemann, V. Ganapathi, D. Koller, and S. Thrun. Real-Time Identification and Localization of Body Parts from Depth Images. In *ICRA*, 2010.

[144] P. Poirson, P. Ammirato, C.-Y. Fu, W. Liu, J. Kosecka, and A. C. Berg. Fast Single Shot Detection and Pose Estimation. In *3DV*, 2016.

[145] Polhemus. Liberty Motion Capture System. https://polhemus.com/motion-tracking/all-trackers/liberty. Accessed: 2018-05-13.

[146] G. Pons-Moll, A. Baak, J. Gall, L. Leal-Taixe, M. Muller, H. Seidel, and B. Rosenhahn. Outdoor Human Motion Capture Using Inverse Kinematics and Von Mises-Fisher Sampling. In *ICCV*, 2011.

[147] G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black. Dyna: A Model of Dynamic Human Shape in Motion. In *SIGGRAPH*, 2015.

[148] G. Pons-Moll, J. Taylor, J. Shotton, A. Hertzmann, and A. Fitzgibbon. Metric Regression Forests for Human Pose Estimation. In *BMVC*, 2013.

[149] G. Pons-moll, J. Taylor, J. Shotton, A. Hertzmann, and A. Fitzgibbon. Metric Regression Forests for Correspondence Estimation. *IJCV*, 2015.

[150] A.-I. Popa, M. Zanfir, and C. Sminchisescu. Deep Multitask Architecture for Integrated 2D and 3D Human Sensing. In *CVPR*, 2017.

[151] R. Poppe. Evaluating Example-Based Pose Estimation: Experiments on the Humaneva Sets. In *CVPR*, 2007.

[152] M. Rad and V. Lepetit. BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth. In *ICCV*, 2017.

[153] V. Ramakrishna, T. Kanade, and Y. Sheikh. Reconstructing 3D Human Pose from 2D Image Landmarks. In *ECCV*, 2012.

[154] D. Ramanan. Learning to Parse Images of Articulated Bodies. In *NIPS*, 2006.

[155] D. Ramanan, A. Forsyth, and A. Zisserman. Strike a Pose: Tracking People by Finding Stylized Poses. In *CVPR*, 2005.

[156] K. Ramnath, S. N. Sinha, R. Szeliski, and E. Hsiao. Car make and model recognition using 3d curve alignment. In *WACV*, 2014.

[157] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *CVPR*, 2016.

[158] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. *CVPR*, 2017.

[159] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*, 2015.

[160] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *ICML*, 2011.

[161] R. Rios-Cabrera and T. Tuytelaars. Discriminatively trained templates for 3d object detection: A real time scalable approach. In *ICCV*, 2013.

[162] G. Rogez, J. Rihan, C. Orrite, and P. Torr. Fast Human Pose Detection Using Randomized Hierarchical Cascades of Rejectors. *IJCV*, 2012.

[163] G. Rogez and C. Schmid. Mocap Guided Data Augmentation for 3D Pose Estimation in the Wild. In *NIPS*, 2016.

[164] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015.

[165] R. Rosales and S. Sclaroff. Infering Body Pose Without Tracking Body Parts. In *CVPR*, June 2000.

[166] R. Rosales and S. Sclaroff. Learning Body Pose via Specialized Maps. In *NIPS*, 2002.

[167] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints. *IJCV*, 66(3):231–259, 2006.

[168] A. Rozantsev, V. Lepetit, and P. Fua. Flying Objects Detection from a Single Moving Camera. In *CVPR*, pages 4128–4136, 2015.

[169] M. Salzmann and R. Urtasun. Combining Discriminative and Generative Methods for 3D Deformable Surface and Articulated Pose Reconstruction. In *CVPR*, June 2010.

[170] M. Salzmann and R. Urtasun. Implicitly Constrained Gaussian Process Regression for Monocular Non-Rigid Pose Estimation. In *NIPS*, December 2010.

[171] M. Sanzari, V. Ntouskos, and F. Pirri. Bayesian Image Based 3D Pose Estimation. In *ECCV*, 2016.

[172] B. Sapp, A. Toshev, and B. Taskar. Cascaded Models for Articulated Pose Estimation. In *ECCV*, 2010.

[173] J. Shotton, A. Fitzgibbon, M. Cook, and A. Blake. Real-Time Human Pose Recognition in Parts from a Single Depth Image. In *CVPR*, 2011.

[174] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient Human Pose Estimation from Single Depth Images. *PAMI*, 35(12):2821–2840, 2013.

[175] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient Human Pose Estimation from Single Depth Images. *PAMI*, 35(12):2821–2840, 2013.

[176] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-Time Human Pose Recognition in Parts from Single Depth Images. *Communications of the ACM*, 56(1):116–124, 2013.

[177] H. Sidenbladh, M. J. Black, and D. J. Fleet. Stochastic Tracking of 3D Human Figures Using 2D Image Motion. In *ECCV*, 2000.

[178] L. Sigal. Human Pose Estimation. In *Encyclopedia of Computer Vision*. Springer, 2011.

[179] L. Sigal, A. Balan, and M. J. Black. Combined Discriminative and Generative Articulated Pose and Non-Rigid Shape Estimation. In *NIPS*, 2007.

[180] L. Sigal, A. Balan, and M. J. Black. Humaneva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion. *IJCV*, 2010.

[181] L. Sigal, S. Bhatia, S. Roth, M. Black, and M. Isard. Tracking Loose-Limbed People. In *CVPR*, 2004.

[182] L. Sigal and M. Black. Humaneva: Synchronized Video and Motion Capture Dataset for Evaluation of Articulated Human Motion. Technical report, Department of Computer Science, Brown University, 2006.

[183] L. Sigal, M. Isard, H. W. Haussecker, and M. J. Black. Loose-Limbed People: Estimating 3D Human Pose and Motion Using Non-Parametric Belief Propagation. *IJCV*, 2012.

[184] E. Simo-serra, A. Quattoni, C. Torras, and F. Moreno-noguer. A Joint Model for 2D and 3D Pose Estimation from a Single Image. In *CVPR*, 2013.

[185] E. Simo-serra, A. Ramisa, G. Alenya, C. Torras, and F. Moreno-noguer. Single Image 3D Human Pose Estimation from Noisy Observations. In *CVPR*, 2012.

[186] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Discriminative Density Propagation for 3D Human Motion Estimation. In *CVPR*, 2005.

[187] C. Sminchisescu, A. Kanaujia, and D. Metaxas. Learning Joint Top-Down and Bottom-Up Processes for 3D Visual Inference. In *CVPR*, 2006.

[188] J. Sock, S. H. Kasaei, L. S. Lopes, and T.-K. Kim. Multi-view 6D Object Pose Estimation and Camera Motion Planning using RGBD Images. In *CVPR*, 2017.

[189] C. Stoll, N. Hasler, J. Gall, H.-P. Seidel, and C. Theobalt. Fast Articulated Motion Tracking using a Sums of Gaussians Body Model. In *ICCV*, 2011.

[190] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views. In *ICCV*, 2015.

[191] M. Sun, P. Kohli, and J. Shotton. Conditional Regression Forests for Human Pose Estimation. In *CVPR*, 2012.

[192] X. Sun, J. Shang, S. Liang, and Y. Wei. Compositional human pose regression. In *ICCV*, volume 2, 2017.

[193] I. Sutskever, G. E. Hinton, and G. W. Taylor. Generating Text with Recurrent Neural Networks. In *ICML*, 2011.

[194] G. W. Taylor, L. Sigal, D. J. Fleet, and G. E. Hinton. Dynamical Binary Latent Variable Models for 3D Human Pose Tracking. In *CVPR*, 2010.

[195] B. Tekin, I. Katircioglu, M. Salzmann, V. Lepetit, and P. Fua. Structured Prediction of 3D Human Pose with Deep Neural Networks. In *BMVC*, 2016.

[196] B. Tekin, P. Márquez-neila, M. Salzmann, and P. Fua. Learning to Fuse 2D and 3D Image Cues for Monocular Body Pose Estimation. In *ICCV*, 2017.

[197] B. Tekin, A. Rozantsev, V. Lepetit, and P. Fua. Direct Prediction of 3D Body Poses from Motion Compensated Sequences. In *CVPR*, pages 991–1000, 2016.

[198] B. Tekin, S. Sinha, and P. Fua. Real-Time Seamless Single Shot 6D Object Pose Prediction. In *CVPR*, 2018.

[199] D. Tome, C. Russell, and L. Agapito. Lifting from the Deep: Convolutional 3D Pose Estimation from a Single Image. In *arXiv preprint, arXiv:1701.00295*, 2017.

[200] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation. In *NIPS*, 2014.

[201] A. Toshev and C. Szegedy. Deeppose: Human Pose Estimation via Deep Neural Networks. In *CVPR*, 2014.

[202] S. Tulsiani and J. Malik. Viewpoints and Keypoints. In *CVPR*, 2015.

[203] R. Urtasun and T. Darrell. Sparse Probabilistic Regression for Activity-Independent Human Pose Inference. In *CVPR*, 2008.

[204] R. Urtasun, D. Fleet, and P. Fua. 3D People Tracking with Gaussian Process Dynamical Models. In *CVPR*, 2006.

[205] R. Urtasun, D. Fleet, A. Hertzman, and P. Fua. Priors for People Tracking from Small Training Sets. In *ICCV*, 2005.

[206] L. Vacchetti, V. Lepetit, and P. Fua. Stable Real-Time 3D Tracking Using Online and Offline Information. *PAMI*, 26(10), 2004.

[207] J. Valmadre and S. Lucey. Deterministic 3D Human Pose Estimation Using Rigid Structure. In *ECCV*, 2010.

[208] G. Varol, D. Ceylan, B. Russell, J. Yang, E. Yumer, I. Laptev, and C. Schmid. Bodynet: Volumetric inference of 3d human body shapes. In *arXiv*, 2018.

[209] Vicon. Motion Capture System. https://www.vicon.com. Accessed: 2018-05-13.

[210] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In *ICML*, 2008.

[211] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *JMLR*, 2010.

[212] V. Vonikakis, D. Chrysostomou, R. Kouskouridas, and A. Gasteratos. A Biologically Inspired Scale-Space for Illumination Invariant Feature Detector. *Measurement Science and Technology*, 24(7):074024, 2013.

[213] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose Tracking from Natural Features on Mobile Phones. In *ISMAR*, pages 125–134, 2008.

[214] C. Wang, Y. Wang, Z. Lin, A. L. Yuille, and W. Gao. Robust Estimation of 3D Human Poses from a Single Image. In *CVPR*, 2014.

[215] S. E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional Pose Machines. In *CVPR*, 2016.

[216] X. Wei, P. Zhang, and J. Chai. Accurate Real-Time Full Body Motion Capture Using a Single Depth Camera. In *SIGGRAPH Asia*, 2012.

[217] D. Weinland, M. Ozuysal, and P. Fua. Making Action Recognition Robust to Occlusions and Viewpoint Changes. In *ECCV*, pages 635–648, 2010.

[218] S. Xia, Z. Zhang, and L. Su. Cascaded 3D Full Body Pose Regression from Single Depth Image at 100 FPS. In *arXiv*, 2017.

[219] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[220] Xsens. MVM Motion Capture System. https://www.xsens.com/products/xsens-mvn-animate/. Accessed: 2018-05-13.

[221] Y. Yang and D. Ramanan. Articulated Pose Estimation with Flexible Mixtures-Of-Parts. In *CVPR*, 2011.

[222] A. Yao, J. Gall, L. J. Van Gool, and R. Urtasun. Learning Probabilistic Non-Linear Latent Models for Tracking Complex Activities. In *NIPS*, 2011.

[223] H. Yasin, U. Iqbal, B. Kruger, A. Weber, and J. Gall. A Dual-Source Approach for 3D Pose Estimation from a Single Image. In *CVPR*, 2016.

[224] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys. Accurate 3D Pose Estimation from a Single Depth Image. In *ICCV*, 2011.

[225] M. Ye and R. Yang. Real-Time Simultaneous Pose and Shape Estimation for Articulated Objects using a Single Depth Camera. In *CVPR*, 2014.

[226] T.-H. Yu, T.-K. Kim, and R. Cipolla. Unconstrained Monocular 3D Human Pose Estimation by Action Detection and Cross Modality Regression Forest. In *CVPR*, 2013.

[227] C. Zach, A. Penate-Sanchez, and M.-T. Pham. A dynamic programming approach for fast and robust object pose recognition from range images. In *CVPR*, 2015.

[228] H. Zhang and Q. Cao. Combined Holistic and Local Patches for Recovering 6D Object Pose. In *ICCV*, 2017.

[229] F. Zhou and F. de la Torre. Spatio-Temporal Matching for Human Detection in Video. In *ECCV*, 2014.

[230] F. Zhou and F. De la Torre. Spatio-temporal matching for human pose estimation in video. *PAMI*, 38(8):1492–1504, 2016.

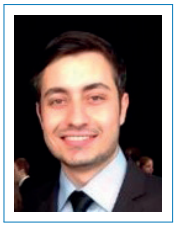
[231] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. We. Weakly-Supervised Transfer for 3D Human Pose Estimation in the Wild. *ICCV*, 2017.

[232] X. Zhou, X. Sun, W. Zhang, S. Liang, and Y. Wei. Deep Kinematic Pose Regression. In *ECCV*, 2016.

[233] X. Zhou, M. Zhu, S. Leonardos, K. Derpanis, and K. Daniilidis. Sparseness Meets Deepness: 3D Human Pose Estimation from Monocular Video. In *CVPR*, 2016.

[234] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis. Single Image 3D Object Detection and Pose Estimation for Grasping. In *ICRA*, 2014.

[235] S. Zuffi, J. Romero, C. Schmid, and M. J. Black. Estimating Human Pose with Flowing Puppets. In *ICCV*, 2013.



Bugra Tekin

Personal Data

Birth date 2 November 1988
E-Mail bugra.tekin@epfl.ch
Web http://bugratekin.info

Research Interests

Computer Vision, Augmented Reality, Machine Learning, Deep Learning, Human Pose Estimation, Action Recognition, 2D/3D Object Detection, 6 DOF Pose Estimation

Education & Diplomas

2013–present **EPFL (École Polytechnique Fédérale de Lausanne), Switzerland**, *Lausanne, Switzerland.*
Ph.D. Computer Science, Advisors: Prof. Pascal Fua, Prof. Vincent Lepetit
Dissertation Topic: Monocular 3D Human and Object Pose Estimation with Deep Learning

2011–2013 **EPFL (École Polytechnique Fédérale de Lausanne), Switzerland**, *Lausanne, Switzerland.*
M.Sc. Electrical & Electronics Engineering, Major: Information Technologies
Thesis Topic: Learning Separable Filters for Efficient Image Recognition, GPA: 5.63/6.00

2006–2011 **Bogazici University**, *Istanbul, Turkey.*
B.Sc. Electrical & Electronics Engineering, Major: Signal Processing and Communications
Thesis Topic: Acoustic Source Localization and Speech Recognition Using Microphone Arrays, GPA: 3.71/4.00

Professional Experience

2013–present **EPFL, Computer Vision Laboratory**, *Lausanne, Switzerland*, (full-time).
Research Assistant
Development of various algorithms for computer vision, machine learning and deep learning. Participation in teaching activities and international projects, supervision of graduate students. Research on 3D human pose estimation, motion capture, augmented reality, computer graphics and body tracking.

2017 **Microsoft Research**, *Seattle, Washington, USA*, (internship - 3 months).
Research Intern
Designed a new deep learning architecture that naturally extends the single-shot 2D object detection paradigm to 6D object pose estimation for efficient use in augmented reality glasses. It demonstrates state-of-the-art accuracy with real-time performance and is at least 5 times faster than the existing methods (50 to 94 fps depending on the input resolution).

2012–2013 **EPFL, LSM & Immersive Vision Technologies**, *Lausanne, Switzerland*, (internship - 6 months).
Research & Development Intern
Designed a parallelized panaromic image registration methodology on GPU with CUDA for a spherical multi-camera system: http://lsm.epfl.ch/page-52820-en.html. Obtained real-time performance for processing images simultaneously coming from 13 cameras.

2010 **Aselsan**, *Ankara, Turkey*, (internship - 3 months).
Engineering Intern in Signal and Image Processing Department
Implemented image stitching algorithms with a moving pan-tilt-zoom camera for augmented reality and object tracking applications.

2009 **National Research Council**, *Ankara, Turkey*, (internship - 3 months).
Engineering Intern in Optoelectronic Imaging Department
Developed signal and image processing software for high accuracy laser beam localization on optoelectronic devices.

Publications

B. Tekin, S. N. Sinha, P. Fua, "Real-Time Seamless Single Shot 6D Object Pose Prediction", *Computer Vision and Pattern Recognition (**CVPR**)*, Salt Lake City, UT, June 18-22, 2018.

I. Katircioglu*, **B. Tekin***, M. Salzmann, V. Lepetit, P. Fua, "Learning Latent Representations of 3D Human Pose with Deep Neural Networks", *International Journal of Computer Vision (**IJCV**)*, 2018.

B. Tekin, Pablo Márquez-Neila, M. Salzmann, P. Fua, "Learning to Fuse 2D and 3D Image Cues for Monocular Body Pose Estimation", *International Conference on Computer Vision (**ICCV**)*, Venice, Italy, October 22-29, 2017.

B. Tekin, Pablo Márquez-Neila, M. Salzmann, P. Fua, "Fusing 2D Uncertainty and 3D Cues for Monocular Body Pose Estimation", *arXiv preprint, arXiv:1611.05708*, November 2016.

B. Tekin*, I. Katircioglu*, M. Salzmann, V. Lepetit, P. Fua, "Structured Prediction of 3D Human Pose with Deep Neural Networks", *British Machine Vision Conference (**BMVC**)*, York, UK, September 19-22, 2016. **(oral presentation)**

B. Tekin, A. Rozantsev, V. Lepetit, P. Fua, "Direct Prediction of 3D Body Poses from Motion Compensated Sequences", *Computer Vision and Pattern Recognition (**CVPR**)*, Las Vegas, NV, June 27-30, 2016.

B. Tekin, X. Sun, X. Wang, V. Lepetit, P. Fua, "Predicting People's 3D Poses from Short Sequences", *arXiv preprint, arXiv:1504.08200*, April 2015.

A. Sironi*, **B. Tekin***, R. Rigamonti, V. Lepetit, P. Fua, "Learning Separable Filters", *Pattern Analysis and Machine Intelligence (**PAMI**)*, vol. 37, no. 1, pp. 94-106, January 2015.

B. Tekin, Ulugbek Kamilov, Emrah Bostan, M. Unser, "Benefits of Consistency in Image Denoising with Steerable Wavelets", *International Conference on Acoustics, Speech and Signal Processing (**ICASSP**)*, Vancouver, BC, May 26-31, 2013, pp. 1355-1358. **(oral presentation)**

* indicates equal contribution

Patent

P. Fua, V. Lepetit, A. Rozantsev, **B. Tekin**, "Method, System and Device for Direct Prediction of 3D Body Poses from Motion Compensated Sequence", ***US Patent***, Pub. No: US 2017-0316578 A1, Pub. Date: November 02, 2017.

Honors & Awards

2017 **Winner of Qualcomm Innovation Fellowship Europe** and recipient of an award of $40,000 for the most innovative project proposal in Europe in the field of mobile visual computing. The project is entitled "Real-Time 3D Human Pose Estimation in the Wild".

2011-2013 **International Master's Scholarship** from National Research Council for master studies

2011 **High Honors Distinction** upon graduation from bachelor studies

2011 **Ranking** in the 15^{th} place among 250,000 participants in National Graduate Education Entrance Examinations of Turkey

2006-2011 **Undergraduate Scholarship** from Tekfen Foundation

2006 **Ranking** in the 152^{th} place among more than 1.5 million participants in the National University Entrance Examinations of Turkey

Computer Skills

Languages Python, C/C++, Java, LaTeX, Matlab

OS Unix/Linux, Mac OS, Windows

Others Parallel programming using CUDA, Svn, Git, Deep Learning with PyTorch, Tensorflow, Theano

Reviewing

Conferences Computer Vision and Pattern Recognition (CVPR), Int. Conference on Computer Vision (ICCV), European Conference on Computer Vision (ECCV), International Conference on 3D Vision (3DV)

Journals Pattern Analysis and Machine Intelligence (PAMI), Int. Journal of Computer Vision (IJCV), Computer Vision and Image Understanding (CVIU)

Teaching

Deep Learning, Computer Vision, Numerical Methods for Visual Computing, Programming in C/C++/Java, Principles of Digital Communications, Circuits and Systems

Language Skills

English: fluent. Certified by TOEFL IBT (2010), **German:** communicative. Certified by Zertifikat-Deutsch (2010), **French:** elementary. Took several courses at EPFL, **Turkish:** native.