June 2017

Estimation of Human Poses Categories and Physical Object Properties from Motion Trajectories

Mona Fathollahi Ghezelghieh

University of South Florida, mona2@mail.usf.edu

Follow this and additional works at: http://scholarcommons.usf.edu/etd

Part of the Computer Sciences Commons

Scholar Commons Citation

http://scholarcommons.usf.edu/etd/6835

This Dissertation is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
Estimation of Human Poses Categories and Physical Object Properties from Motion
Trajectories

by

Mona Fathollahi Ghezelghieh

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy
Department of Computer Science and Engineering College of Engineering University of South Florida

Co-Major Professor: Rangachar Kasturi, Ph.D.
Co-Major Professor: Sudeep Sarkar, Ph.D.
Dmitry Goldgof, Ph.D.
Richard Gitlin, Sc.D.
Toru Shimizu, Ph.D.

Date of Approval:
June 7, 2017

Keywords: 3D Human Pose, Camera Viewpoint, Deep Neural Network

Copyright © 2017, Mona Fathollahi Ghezelghieh
DEDICATION

I dedicate this dissertation to my wonderful husband, Hamid, who has been my greatest supporter throughout this journey. He cheered me up during my weakest moments and believed in me when I doubted my capabilities. I also dedicate this dissertation to my lovely parents, who quietly and patiently waited for me to achieve my goals. Their unconditional love and faith have been great motivation for me.
ACKNOWLEDGMENTS

First of all, I would like to express my sincere appreciation to my very kind PhD advisors, Prof. Kasturi and Prof. Sarkar for guiding and supporting me over the past years. You have been a tremendous mentor for me, thank you for allowing me to grow as a research scientist by giving me freedom to pursue various projects without objection. I would like to express my gratitude to the member of my committees, Prof. Goldgof, Prof. Gitlin and Prof. Shimizu. I am very grateful to all of my fellow graduate students for making my stay in University of South Florida pleasurable.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>iii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>iv</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>vi</td>
</tr>
<tr>
<td>CHAPTER 1 : INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Contributions</td>
<td>2</td>
</tr>
<tr>
<td>1.2.1 Coarse Body Orientation Estimation in Still Images</td>
<td>2</td>
</tr>
<tr>
<td>1.2.2 Body Orientation Estimation in Video</td>
<td>2</td>
</tr>
<tr>
<td>1.2.3 Human 3D Pose Estimation Approach</td>
<td>3</td>
</tr>
<tr>
<td>1.2.4 Object Physical Property Estimation</td>
<td>7</td>
</tr>
<tr>
<td>1.3 Dissertation Roadmap</td>
<td>8</td>
</tr>
<tr>
<td>CHAPTER 2 : LITERATURE SURVEY</td>
<td>9</td>
</tr>
<tr>
<td>2.1 Object Coarse Pose Estimation</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Object 3D Pose Estimation</td>
<td>10</td>
</tr>
<tr>
<td>CHAPTER 3 : ESTIMATION OF BODY ORIENTATION IN STILL IMAGES</td>
<td>19</td>
</tr>
<tr>
<td>3.1 Background on Deep Neural Networks</td>
<td>20</td>
</tr>
<tr>
<td>3.1.1 Regularization</td>
<td>24</td>
</tr>
<tr>
<td>3.2 Our ConvNet Methodology</td>
<td>26</td>
</tr>
<tr>
<td>3.3 Dataset</td>
<td>29</td>
</tr>
<tr>
<td>3.4 Experimental Results</td>
<td>31</td>
</tr>
<tr>
<td>CHAPTER 4 : ESTIMATION OF BODY ORIENTATION IN VIDEOS</td>
<td>34</td>
</tr>
<tr>
<td>4.1 Background on Recurrent Neural Networks</td>
<td>35</td>
</tr>
<tr>
<td>4.2 Our Recurrent Model</td>
<td>39</td>
</tr>
<tr>
<td>4.3 Experimental Results</td>
<td>40</td>
</tr>
<tr>
<td>CHAPTER 5 : HUMAN 3D POSE ESTIMATION IN STILL IMAGES</td>
<td>46</td>
</tr>
<tr>
<td>5.1 2D Human Body Joint Detection</td>
<td>47</td>
</tr>
<tr>
<td>5.2 Part-based 3D Regression Model</td>
<td>48</td>
</tr>
<tr>
<td>5.2.1 Gaussian Process</td>
<td>50</td>
</tr>
<tr>
<td>5.2.2 Twin Gaussian Process</td>
<td>52</td>
</tr>
<tr>
<td>5.3 Experimental Results</td>
<td>54</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Classification error of body orientation estimation in still images</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>Mean and standard deviation of continuous orientation</td>
<td>41</td>
</tr>
<tr>
<td>4.2</td>
<td>Error rate of CNN and RNN based orientation estimation approaches</td>
<td>42</td>
</tr>
<tr>
<td>5.1</td>
<td>Efficacy of joint-set regression on 3D pose estimation error in millimeters</td>
<td>57</td>
</tr>
<tr>
<td>6.1</td>
<td>Accuracy of 3D and 1D object trajectory classification</td>
<td>66</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1.1 Categorized human body orientation 3
Figure 1.2 Possible camera locations in our setup 7
Figure 3.1 VGG-F architecture 27
Figure 3.2 Our synthetic characters generation framework 29
Figure 3.3 Samples of our synthetic characters 30
Figure 3.4 Samples of our training dataset 32
Figure 4.1 Proposed framework to estimate body orientation in videos 35
Figure 4.2 Input to the RNN model 36
Figure 4.3 Our proposed MLP-LSTM framework 40
Figure 4.4 Point-wise predictions of body orientation by LSTM 44
Figure 4.5 Learning curve of LSTM and MLP-LSTM models 45
Figure 5.1 Our 3D pose framework 47
Figure 5.2 Joint sets used in our regression model 49
Figure 5.3 3D error of each body joint 56
Figure 5.4 Effect of non-perfect 2D pose on 3D pose reconstruction error 57
Figure 5.5 3D pose estimation error in still images in millimeters 58
Figure 5.6 3D pose estimation error in video sequence in millimeters 59
Figure 6.1  Selected frames of dynamic objects on the road  61
Figure 6.2  Object trajectory synthesis in Blender  63
Figure 6.3  Samples of synthesized trajectory data set  65
Figure 6.4  Classification accuracy versus trajectory length  66
Figure 6.5  Plots of real-world trajectories  67
Figure 6.6  Samples of real-world trajectories  68
ABSTRACT

Despite the impressive advancements in people detection and tracking, safety is still a key barrier to the deployment of autonomous vehicles in urban environments [1]. For example, in non-autonomous technology, there is an implicit communication between the people crossing the street and the driver to make sure they have communicated their intent to the driver. Therefore, it is crucial for the autonomous car to infer the future intent of the pedestrian quickly. We believe that human body orientation with respect to the camera can help the intelligent unit of the car to anticipate the future movement of the pedestrians. To further improve the safety of pedestrians, it is important to recognize whether they are distracted, carrying a baby, or pushing a shopping cart. Therefore, estimating the fine-grained 3D pose, i.e. (x,y,z)-coordinates of the body joints provides additional information for decision-making units of driverless cars.

In this dissertation, we have proposed a deep learning-based solution to classify the categorized body orientation in still images. We have also proposed an efficient framework based on our body orientation classification scheme to estimate human 3D pose in monocular RGB images.

Furthermore, we have utilized the dynamics of human motion to infer the body orientation in image sequences. To achieve this, we employ a recurrent neural network model
to estimate continuous body orientation from the trajectories of body joints in the image plane.

The proposed body orientation and 3D pose estimation framework are tested on the largest 3D pose estimation benchmark, Human3.6m (both in still images and video), and we have proved the efficacy of our approach by benchmarking it against the state-of-the-art approaches.

Another critical feature of self-driving car is to avoid an obstacle. In the current prototypes the car either stops or changes its lane even if it causes other traffic disruptions. However, there are situations when it is preferable to collide with the object, for example a foam box, rather than take an action that could result in a much more serious accident than collision with the object. In this dissertation, for the first time, we have presented a novel method to discriminate between physical properties of these types of objects such as bounciness, elasticity, etc. based on their motion characteristics. The proposed algorithm is tested on synthetic data, and, as a proof of concept, its effectiveness on a limited set of real-world data is demonstrated.
CHAPTER 1: INTRODUCTION

1.1 Motivation

Autonomous technology is growing quickly and many large companies are racing to deploy their first self-driving car by 2020. There has been significant progress on how to efficiently use the information from car sensors such as LIDAR, radar and camera array to accurately localize the self-driving car on the map, estimate its distance to other cars and avoid obstacles.

Despite the impressive advancements in people detection and tracking, safety is still a key barrier to the deployment of autonomous vehicles in urban environments [1]. For example, in non-autonomous technology, there is an implicit communication between the people crossing the street and the driver to make sure they have communicated their intent to the driver. This critical safety feature is missing in self-driving cars. Although tracking is an effective solution when the person is moving, it gets challenging when the area is crowded and pedestrians get occluded or they change their direction too fast. Therefore, it is crucial for the autonomous car to infer the future intent of the pedestrian quickly.
1.2 Contributions

1.2.1 Coarse Body Orientation Estimation in Still Images

Estimating human body orientation with respect to the camera into one of eight directions, as illustrated in Figure 1.1, can help the intelligent unit of the car to anticipate the future movement of the pedestrian. Body orientation estimation has been explored in other research works, [2], [3], [4]. However, they either limited their approach to the upper body or the limited number of characters. More specifically, since these methods use the traditional approaches in Computer Vision to extract image features they would not be able to estimate the body orientation of a pedestrian who has not been in the training set. In addition, they either collect data with white background or collect lots of background images separately. In this dissertation, we provide a deep learning solution to estimate the categorized human body orientation in monocular RGB images. The model learns the implicit coarse pose of the person and be invariant to body shape, clothing and background. Therefore, we have created synthetic characters with different fine-grained poses, body shapes and clothing. By augmenting the training dataset, we have been able to improve the accuracy on unseen subjects.

1.2.2 Body Orientation Estimation in Video

The performance of our body orientation estimation from a single image depends highly on the training dataset. Similar to any machine learning approach, the dataset should have different variety of images to be able to generalize well on unseen data. Suppose the
Figure 1.1: Categorized human body orientation. The angles displayed in the figure are obtained by discretizing the yaw angle of human subject

subject in the test image wears skirt. In order to estimate her 3D pose reliably, the training set should also contain subjects who wear skirt. Processing the frames in a video instead of a single image is usually a promising way to improve the accuracy. To estimate human body orientation in image sequences, one solution is to independently estimate body orientation in each frame and merge the decisions in a sequence of frames. This approach suffers from the shortcoming explained above. The other approach is to utilize the temporal information in the video to estimate body orientation in each frame. In this dissertation, we propose a novel approach to leverage the trajectories of body joints in a temporal window to infer body orientation.

1.2.3 Human 3D Pose Estimation Approach

To further improve the safety of pedestrians, it is important to recognize whether they are distracted, carrying a baby, or pushing a shopping cart. Therefore, estimating
the fine-grained 3D pose, x,y,z coordinates of body joints, would be more informative for decision-making units of driverless cars. Consequently, in this dissertation, we have also studied the 3D pose estimation approaches in monocular RGB images. Before we explain our contribution in human 3D pose estimation, we will provide a brief introduction on this area.

Human 3D pose estimation has a wide spectrum of applications. In a surveillance camera, suspicious or hostile human activities such as a security boundary intrusion or shoplifting can be detected. In clinical studies, 3D pose estimation has been used very effectively in the analysis of the gait of patients with walking abnormalities to identify the underlying causes of abnormality. In computer animation, 3D modeling of different activities can be used to simulate them in a realistic way, which will be applied to the animated characters in videos. Some tech companies are trying to provide their customers with a technology that overlays virtual reality avatars of friends who are in remote locations on top of their vision to facilitate a real-time experience of "hanging out" together, wherein both parties can express emotions and play games. Another popular application of 3D pose estimation is in Sports, where the reconstructed 3D pose can be used for scoring and professional training.

In Computer Vision, perspective projection is used to generate a 2D planar image from 3D space. This modeling is not linear due to intrinsic distortion parameters of camera and rigid transformation between the world and the camera. On the other hand, finding the inverse of perspective projection is an ill-conditioned problem because a point on the
2D image is transformed to a line vector in 3D space. Therefore, lack of depth information makes the 3D pose estimation difficult. The other factor that makes the 3D pose estimation challenging is the large variability in shape and clothing of humans. It is very important for the model to be invariant to geometric and photometric variations in the image. Occlusion by objects or self-occlusion by other body parts is another problem in 2D and 3D pose estimation. However, one can utilize the strong correlation in body parts to infer the position of joints that are not directly observable.

Methods in estimating the 3D human pose can be broadly categorized into top-down and bottom-up models.

Top-down or generative models estimate 3D pose by explicitly rendering the synthetic 3D human model candidate and matching it to the observed images. They accurately represent the human body by modeling both 3D pose and 3D shape. The 3D pose is represented as a vector with global rotation and relative joint angles for each body part. The generative models employ tapered cylinders or truncated cones to represent shape models. In these approaches, 3D pose parameters and 3D shape latent parameters are estimated by searching in the parameter space and generating several candidate 3D poses and shapes. Since the search space is usually high dimensional, the kinematic priors are exploited to constrain the search space to a range of values. 3D pose hypotheses are then evaluated by direct 2D image rendering. Each hypothesis is evaluated by calculating the image evidence of synthetic 3D model. Construction of a likelihood function requires realistic 3D shape and pose models, camera projection model, and an efficient image matching function. Finally, the
3D pose/shape with highest probability is chosen as the estimated 3D pose. The accuracy of the generative 3D pose estimation is highly dependent on the accurate perspective projection and effective feature extraction. In practice, despite the ability of these approaches to generalize to unseen data, they fail due to bad initialization, high dimensionality of the parameter space, and specific observation likelihood [5].

Bottom-up or discriminative methods learn statistical models to predict 3D pose from the feature vectors extracted from the 2D image. Most of these models train separate classifiers to detect each body part in the image. Regression models are learned to map these features to the 3D pose space. To reduce the training data, the strong correlations between joints are taken into account. The 3D pose is mapped into a low dimensional space and the mapping between the image features and this compact representation of 3D pose is learnt. The limitations of these approaches are requirements to have large training set, lack of robust image descriptors, and their dependency on the learning algorithm to avoid over-fitting.

In this dissertation, we propose an effective discriminative 3D pose estimation framework that directly learns the mapping from image features to the corresponding 3D pose. In this framework, we attempt to improve the accuracy of the model by addressing the limitations of the top-down approach. We propose to map the image to features that encodes both depth and spatial relation of body parts.
We represent depth by human body orientation with respect to the camera, which provides a cue on coarse relative depth of body joints. For example, if the person orientation with respect to the camera is $45^\circ$, we could reason that the depth of his or her left shoulder is more than the right shoulder. This inference in combination with 2D pose could serve as an expressive feature in constructing human 3D pose from a monocular image. Our method does not require calibrating the camera, but the person upper body should be in upright position. Figure 1.2 illustrates possible locations of the camera in our setup.

1.2.4 Object Physical Property Estimation

As discussed earlier, human body orientation has a potential application to improve the safety in the self-driving technology by inferring the intent of the pedestrians. An-
other requirement for of a self-driving car is to avoid an obstacle. Currently, the prototype self-driving cars either stop or change their lane even if it causes other traffic disruptions. However, there are situations where it is preferable to collide with the object, for example a foam box, rather than taking an action that could result in a much more serious accident than a collision with that object. In this work, for the first time, we present a novel method to discriminate between different classes of object physical properties such as bounciness and elasticity, based on their motion trajectories. We test the algorithm on synthetic data, and as a proof of concept, demonstrate its effectiveness on a limited set of real-world data.

1.3 Dissertation Roadmap

The rest of this dissertation is structured as follows: chapter 2 includes a summery of prior research on human orientation estimation and 3D pose estimation. In chapters 3 and 4, we describe our body orientation estimation methodology in still images and video respectively. Our 3D human pose estimation framework is described in chapter 5. In chapter 6, we introduce our approach to infer the category of physical properties of objects from their motion trajectories. Finally, in Chapter 7 we present conclusions and possible future directions of our work.
CHAPTER 2 : LITERATURE SURVEY

In this chapter, we first review the previous research in human body orientation estimation and discuss their challenges. Next, we review the state-of-the-art approaches for estimation of human 3D pose from monocular RGB still images and video.

2.1 Object Coarse Pose Estimation

Prediction of human body orientation is one of the challenging tasks for a mobile robot. For example, in [4] a framework is proposed to assist a robot in deciding which direction to approach the human. The authors replaced the binary decision maker in Support Vector Machines (SVM) with a decision tree to enable the distinction of eight upper-body orientations in addition to person detection. Around 1.5 million background images have been collected and used as a new class in addition to eight orientation classes in the model. Fifteen subjects have participated in the experiments. The characters in the training and test sets are the same. In contrast, our proposed approach allows different characters in the test and training sets.

To improve the safety of pedestrians in the driver assistance system in automotive industry, a random forests classifier based on Histogram of Oriented Gradients (HOG) features is built into automotive camera processing unit to estimate human body orientation
in [3]. The orientation is classified into four classes: right, left, front and back. Also, the background of all images is white and the characters in the training and test set are similar. A partial least squares (PLS) model, based on gradient and texture features is proposed in [2] to estimate the human upper body orientation. The authors have employed PLS to obtain a low dimensional latent space of the observed variables, which maximizes the separation between samples with different characteristics. The training set is cropped of INRIA data, which leads to accuracy of 70%. The movement of the person is further tracked by Unscented Kalman Filter-based tracker and is fused to upper body orientation estimation to build more robust estimator. The idea of global pose estimation has been extended to other domains in Computer Vision. For example, in [6] the Euler angles (azimuth, elevation, cyclorotation) of a rigid object is estimated using deep convolutional neural network architecture. The authors further used this inference to improve the rigid object’s keypoints estimation.

2.2 Object 3D Pose Estimation

Human 3D pose estimation from a monocular RGB images is one of the well-established research areas in Computer Vision. There are generally two methodologies used in this area: bottom-down or generative model and top-down or discriminative model. The first category has better performance in generalizing to unseen data, but they suffer from manual initialization and priors on human activities requirements, while the second category is very much dependent on the training data and effective image feature extraction. Leveraging the emerging large datasets, the second category shows more promising results. Therefore, we
only review the discriminative approaches. We have roughly divided these approaches into three categories depending on how they attempt to address missing depth.

1. In the first category, the goal is to simultaneously learn or extract image features that incorporate both the human body spatial relation and depth cue. For example, in [7], first the background is subtracted; next, human body parts are segmented and are described by the second-order label-sensitive pooling method [8]. ConvNets have also been exploited to learn image features and the regression model simultaneously. For example, in [9] a model is built that takes an image and 3D pose and generates a match score. This score is high when the pose matches the image. To achieve this, first a ConvNets is trained to extract image features; next these image features and 3D pose are transformed into a joint embedding. The score function is defined as the dot product between the image and joint embedding.

Li et al [10] have considered two strategies to train deep convolutional neural networks to estimate 3D pose of human. In the first strategy, a multi-task learning framework is proposed to jointly estimate the 3D pose and a set of body part detectors. The other scheme is to first pre-train the network using the detection tasks, and then fine-tune it to the pose regression.

The approach proposed in [11], implicitly model the long-range dependencies between the variables in articulated pose estimation. The proposed solution is a sequential model that operates on the estimates from the previous stages. This model is called the “pose machine” that consists of a sequence of multi-class predictors that are trained
to predict the location of each body joint. In each stage, the classifier predicts the belief of assigning a location to each part based on the feature extracted at location \( z \) and contextual information from the preceding classifier. The network architecture composed of five convolutional layers followed by two \( 1 \times 1 \) convolutional layers, which leads to a fully convolutional layer [12]. The design of the second stage network should be such that it achieves a large enough receptive field to learn the long-range correlations between the body parts. These convolutional layers allow the classifier to combine contextual information. The convolutional layer of the first stage has small receptive field, which increases drastically in the second stage. Large receptive fields can be achieved by either pooling, increasing the kernel size of convolutional filters or increasing the number of convolutional layers at the expense of increasing the risk of vanishing gradients [13] [14] [15] during the training. The magnitude of back-propagated gradients decreases if the number of layers between output and input layer is increased. However, the sequential prediction in the approach avoids this problem. Because, each stage is trained to produce belief maps for the location of each part by defining a loss function at the output of each stage.

The review paper presented in [16] is a recent survey of approaches for 3D pose estimation.

2. The second group has focused on 3D pose inference from 2D body joint locations in RGB images. The reconstructed 3D poses should be disambiguated to account for missing depth information. For example, in [17], 3D human pose is represented as a
sparse embedding in an overcomplete dictionary. The authors proposed a matching pursuit algorithm to sequentially select pose bases which minimizes the reprojection error and refines the projective camera parameters. Fan et al [18] have utilized an unsupervised pose subspace clustering method to hierarchically construct a pose tree, where each node represents a pose subspace and the nodes with larger depths in the tree represent more specific pose subspaces. Concatenation of the basis poses from the entire pose subspaces forms block-structural pose dictionary. At the next step, the projected matching pursuit algorithm has been applied to estimate the most likely 3D human pose. Yasin et al in [19] combined two different datasets to generate many 3D-2D pairs as training examples. In the inference step, estimated 2D pose is used to retrieve the normalized nearest 3D pose. The final 3D pose is then estimated by minimizing the projection error under the constraint that the estimated 3D pose should be close to the retrieved pose. Akhter et al [20] proposed a new framework to estimate 3D pose from the ground truth 2D pose. To resolve the ambiguity, they first learn the pose-dependent joint angle limits by collecting a new mocap dataset which includes an extensive variety of stretching poses. Radwan et al in [21], imposed a set of kinematic constraints by projecting a 3D model onto the input image and pruning the parts that are incompatible with the anthropomorphism. To reduce the depth ambiguity, several 3D poses were generated by regressing the initial view to multiple oriented views. Estimated orientation from 2D body part detector is used to choose the final 3D pose. Simo-Serra et al in [22] proposed a Bayesian framework to jointly estimate
3D and 2D poses. The set of 3D pose hypotheses are generated using 3D generative kinematic model, which are weighted by a discriminative part model.

In [23], a deep multitask architecture has been proposed that is able to do several level of segmentation( background segmentation, body-part labeling), 2D and 3D pose estimation. To achieve this, the authors have designed a multi-task loss function at different stages: body joint detection and 2D/3D pose estimation and semantic body part segmentation. In this model, each stage of processing is split into semantic processing and 3D reconstruction. the semantic module is further divided into two sub-modules: 1. 2D pose estimation. 2. body part labeling and background segmentation.

Each task has six recurrent stages with the following inputs:

- Input RGB image
- The results of the previous stages of the same type
- Inputs from other stages: 2D pose estimation is fed into semantic body part segmentation and both are input to 3D pose reconstruction.

The inputs to each stage are individually processed and fused using a ConvNet to produce the corresponding outputs. They have been able to take advantage of multiple datasets and have shown the state-of-the-art results on 3D pose estimation and body part segmentation tasks.

The authors in [24] have taken similar approach to estimate the 3D pose of the object. The designed network is called 3D Interpreter Network where the 3D pose estimation
is decoupled into two sequential steps: 2D keypoint heatmaps detection, and inference step of 3D joint positions. The motivation behind this work is the difficulty of obtaining training images with ground truth 3D data. Therefore, they proposed a projection layer that calculates the 2D projection from a 3D skeleton, and uses the 2D projection as the supervision. This way, the system does not require 3D object annotations. However, due to the ambiguity in 2D-to-3D mapping, this might lead to recovery of 3D geometries that are unnatural even though the projected 2D key-points match the 2D image. Therefore, to teach the network the plausible shapes in 3D world, the network is presented with synthetic data as well. The first part of the network, key-point estimation, is trained with 2D-annotated real images and the second part is trained with the synthetic 3D data. Finally, the entire network is trained with the projection layer. The 3D pose of the object is defined as its key-points and their connections, which are manually designed for each object category. Several base shapes [25] are defined in each category and the network is required to learn the weights for each base shape.

3. The last group is similar to the second group except that a cue is learned from images to reduce the ambiguity in estimation of 3D poses from 2D images. For example, the authors in [26] estimated height-maps that encode the height of each pixel in the image with respect to a reference plane. The height-maps are used to improve 3D pose recovery in the image sequences. In [27], an end-to-end learning scheme is proposed
to estimate the 3D pose. Instead of direct joint coordinate regression, they map the 3D pose estimation to a key-point localization problem in the discretized 3D pose space. To achieve this goal, the 3D space is finely discretized around the subject and a ConvNet is trained to predict per-pixel likelihoods for each joint. Motivated by the success of iterative estimation and intermediate supervision [11] [28], they also proposed a model to improve the initial estimation by employing a coarse-to-fine estimation algorithm. The design choice that has been very effective in the case of 2D human pose estimation is to force the network to produce predictions in multiple processing stages instead of using a single component with a single output. The use of intermediate supervision of the earlier outputs allows for stronger learning capabilities. The authors also have considered a gradual refinement scheme. The network consists of multiple fully convolutional components. The first step is supervised by lower resolution 3D pose of the z-dimension. The ground truth for each joint is $64 \times 64 \times d$ where $d$ is the resolution in z dimension, which takes values from $1, 2, 4, 8, 16, 64$. By presenting the network an easier version of the task during the early processing stages, the authors have been able to reduce overfitting. In addition, the network first estimates the 2D heatmaps that is served as an intermediate supervision. The estimated heatmaps are combined with image features and used as the input to the second part of the network. In [29], a two-stream deep architecture has been presented: one to model uncertainty via probability maps of 2D joint locations and the other to exploit 3D cues by directly working on the image.
Our frame-based approach belongs to the third category. We use estimated 2D body joint locations to account for spatial relation of body parts and learn camera viewpoint to incorporate depth information.

There exists a plethora of frameworks that utilize the temporal information in image sequences to reliably estimate human 3D pose. For example, in [30] a set of motion compensated frames are first described by 3D HOG [31] that simultaneously encodes appearance and motion information. Next, a regressor based on Kernel Ridge Regression [32] or Kernel Dependency Estimation [33] is learned to estimate 3D pose.

In another line of work, 2D body part is estimated in each frame and used as the feature representation of that frame. Later, features from several consecutive frames are used as the input to the regression model [34]. They further investigated systematically how the number of consecutive frames influences 3D human pose estimation results. Recently, a 3D CNN is proposed in [35] to directly estimate 3D pose in a video sequence.

In [36], a novel representation of motion data has been proposed that encodes both temporal structures and the correlations between joints. The proposed approach is based on an auto-encoder framework that aims to project high-dimensional data onto a low-dimensional manifold. Approximating projection functions with multiplayer perceptrons does this. The encoder and decoder functions do not need to be symmetric. The model requires the encoder to take the local features into account and the decoder to learn a global valid structure. In that paper, the input and output are matrices that have skeleton of $\Delta t$. They have proposed different variation of temporal encoders: Symmetric coding, Time-scale
encoding and Hierarchy encoding. The filter is a temporal encoder that covers the whole joints and can be convolved in the time direction. The input data of time-scale encoder is convolved using filters with different sizes and the results are concatenated. Finally they are processed by fully connected layers.

In [37], the temporal dynamics have been leveraged to perform action classification. The input is the Cartesian skeleton data that are divided into five parts according to human physical structure: two arms, two legs, and one trunk. Each one is separately fed into a subnet. Each subnet is bidirectionally recurrently connected. To model the neighboring skeleton parts, the representation of the trunk subnet is combined with each of the other four subnets. The final representation is fed into a fully connected layer and Softmax layer for classification.

Our video-based framework is different from the mentioned approaches in a sense that we have utilized the temporal information to estimate human body orientation, which is later used in combination with the estimated 2D body pose to detect 3D joints’ locations.
CHAPTER 3 : ESTIMATION OF BODY ORIENTATION IN STILL IMAGES\(^1\)

Human body orientation with respect to the camera provides much information on the relative depth of body parts. For example, if we know that the orientation is 90 degrees, we can conclude that the depth of the left hand is more than the depth of the right hand. In this section, the body orientation angle is discretized into eight categories (0°, 45°, ..., 315°), Figure 1.1, and the problem is mapped to a classification problem.

Human body orientation estimation is a challenging task due to the wide variety of clothing textures and styles, body shapes and fine-grained pose variations. For this reason, we have considered Convolutional Neural Network (ConvNet) framework which has shown impressive performance in learning hierarchical and contextual features in other computer vision tasks such as classification [38] and scene labeling [39]. In the next section, we provide a brief introduction on these models and what makes them so successful in different domains.

\(^1\)Portions of this chapter were reprinted from the paper "Ghezelghieh, M.F., Kasturi, R. and Sarkar, S., 2016, October. Learning camera viewpoint using CNN to improve 3D body pose estimation. In 3D Vision (3DV), 2016 Fourth International Conference on (pp. 685-693). IEEE" Copyright (2016), with permission from IEEE. Permission is included in Appendix A.
3.1 Background on Deep Neural Networks

In classical machine learning models, the input representation, i.e. features, are manually extracted and the role of the data scientist is to choose and tune a model to map the features to the output labels. Designing good features plays a critical role in the performance of the model. The goal of Neural Network is to automatically learn this representation from raw data.

Neural Network models are multiple layers of non-linear functions that both extract features and classify them. Neuron are the basic computational block of Neural Network models. Each of these units applies nonlinearity function, also called activation function, on the weighted input. A common choice of the nonlinearity function is the logistic function which is defined as:

\[
\frac{1}{1 + e^{-w^T x}} \tag{3.1}
\]

where \( x \) is the input and \( w \) is the learnable weights. One of the drawbacks of the logistic function is that its derivative becomes increasingly small as the absolute value of its input increases. This subsequently causes vanishing gradient effect in training neural networks. In the last few years, Rectified Linear Unit or ReLU has been proposed which can be defined as the following function:

\[
f(x) = \max(0, x) \tag{3.2}
\]
The derivative of ReLU is constant which greatly accelerate the convergence of stochastic descent compared to logistic or tanh functions. The other benefit of ReLU is sparsity, because when the input value is negative the output of the ReLU is zero. Usually composition models are able to learn more complex features. Therefore, neurons are connected in layers, called hidden layers. Applying the weights in this layered architecture could be implemented with matrix multiplication which is very computationally efficient.

Suppose the input and the corresponding output data is represented with $x_i$ and $y_i$. The operation done by all the layers of Neural Network is represented by function $f$. In a classification problem, the goal is to find the weights of different layers of the network such that the estimated value of outputs, $\hat{y}_i$, is close to $y_i$. To measure the performance of the classification, we choose a loss function $l$ and minimize the total loss function, $L$, over all training data:

$$ L = \sum_{i=0}^{N} l(y_i - f(x_i)) $$

(3.3)

Let’s represent the weights of the network by vector $\theta$; this vector fully determines $f$. In a neural network, Gradient descent algorithm is used to learn these weights by minimizing the loss function. In each iteration, the error in the final output layer is propagated backward throughout the network layers to compute the weights’ updates. This algorithm is called backpropagation. Since gradient descent computes the gradient using the whole dataset, it is computationally very expensive. To overcome this issue, Stochastic Gradient Descent
SGD is used to compute the gradient using mini-batches consisting of several samples. This technique is more applicable when the loss function has lots of local minima. This way, the noisy calculation of the gradient shifts the model out of local minima to a more global minimum. The other advantage of SGD is to increase the convergence speed. In cases where the training dataset is large and cannot be loaded on the RAM, SGD only requires to load a batch of data on the RAM. Batch size, $k$, is one of the hyper-parameters that requires to be tuned when training a neural network with SGD. Therefore, at each time step, $t$, the parameter $\theta_{t-1}$ is updated according to:

$$\theta_t = \theta_{t-1} + g_t$$  \hspace{1cm} (3.4)

where $g_t$ is the moving average of previous weights and is calculated as:

$$g_t = -\eta \sum_{i=0}^{k-1} \nabla_{\theta} l(y_i, f(x_i, \theta_{t-1})) + \gamma g_{t-1}$$  \hspace{1cm} (3.5)

In this equation, $\eta$ is another hyper-parameter called learning rate, and $\gamma$ is a real value between 0 and 1, called momentum. Generally $\gamma$ starts with 0.5 and is increased to 0.9 or higher values. The momentum hyper-parameter causes less variations of the gradient. For example, when the network is not well conditioned the error surface has a lot of different curvatures in different directions. On the other hand, when the gradient is almost pointing to a same direction, momentum will increase the size of the step taken toward the minimum.

Convolutional Neural Network, or convnets, are a class of neural networks that process
images. In these models, the linear transformation is implemented as a convolution operation, which shares the parameters across the space. This leads to statistical invariance of the image location which renders the model to be translational invariant. There are several architecture designs that help the network to learn more effective features with limited resources:

- Pooling layer can be inserted between successive convolution layers to reduce the spatial size of the feature map without adding extra parameters. Pooling is applied on each depth independently. The most common form of a pooling layer is MAX pooling.

- 1x1 convolution is another less expensive trick to reduce the dimensionality of the network. For example, if we apply 20 filters of $1 \times 1$ on an image of $200 \times 200$ with 50 features map, it would result in size of $200 \times 200 \times 20$. In fact, this is feature pooling across various channels or feature maps.

- Inception module is a concatenation of several convolutional layers and an average pooling layer at the top. The idea of inception module’s design aligns with the intuition that visual information should be processed at various scales and aggregated so that the next stage can abstract features from different scales simultaneously. This module helps to have deeper network with less parameters. For example Google Lenet [40] has 22 layers with 12 times less parameters than the AlexNet [41], while being more accurate.

Usually the classification accuracy of ConvNets improves by adding more layers [42], because the early layers represent low-level features such as edges and color contrasts while
deeper layers try to capture more complex shapes and global context [43]. However, training a very deep network is challenging. Because, by increasing the number of convolutional layers, the number of parameters will be increased, and it is harder to optimize the loss function. This might lead to higher error rates [44]. In addition, there is a higher chance of experiencing vanishing or exploding gradients in deeper networks [15].

In [45] a mechanism called ResNet has been proposed that effectively reduces these problems. In this architecture, there is a shortcut connection parallel to the normal convolutional layer. These shortcut connections or AKA skip connections are always alive and the gradients can easily back propagate through them that lead to faster training time and less chance of vanishing gradient.

3.1.1 Regularization

One of the common issues in training a deep neural network is the lack of enough training data, which causes overfitting. In the following, we list several techniques that could prevent overfitting:

- L2 regularization is the most popular form of controlling overfitting. L2 regularization is implementing by adding the sum of squared magnitude of all parameters in the objective function. Every weight of the network is scaled by $\frac{1}{2}\lambda$, where $\lambda$ is the regularization strength. The L2 regularization penalizes peaky weight vectors and prefers diffuse vectors. This is an appealing property to encourage the network to use all its inputs a little, rather some of them a lot.
In addition, in the backpropagation step every weight is updated linearly:

\[ dW = -\lambda \times W \]  \hspace{1cm} (3.6)

- L1 regularization is similar to the L2 regularization with the difference that for each weight, \( \lambda |W| \) is added to the objective function. L1 regularization's main advantage is to make the weight vectors to be very close to zero during optimization. In other words, neurons only use a subset of their input and become more invariant to the "noisy" inputs, while the final weight vectors in L2 regularization are diffused and small. It is possible to combine the L1 regularization with L2 regularization as:

\[ \lambda_1 \times |W| + \lambda_2 \times |W|^2 \]  \hspace{1cm} (3.7)
• Max norm constraints is another form of regularization that enforces an absolute upper bound on the magnitude of weight vector of each neuron and uses projected gradient descent to apply this upper bound. In details, parameters are updated similar to the standard gradient descent, but after the update we enforce the upper bound by clamping the weight vector, i.e. \(||W||_2 < c\). Typical values of \(c\) are in the orders of 3 or 4. One of the appealing properties of “Max norm constraints” is that the network cannot “explode” even with high learning rates, because the updates are always bounded.

• Dropout is an effective regularization technique that has been recently proposed. It is implemented by keeping a neuron active only with some probability \(p\) or setting it to zero otherwise.

3.2 Our ConvNet Methodology

In this section, we will use well-known network architecture called VGG-F [46] model shown in Figure 3.1. This architecture is proposed by Visual Geometry Group and has achieved one of the best performances in image classification task on ImageNet object recognition dataset. This architecture contains 5 convolutional layers (C1-C5) and two fully connected layers (F1-F2). The activation function for all weight layers is the REctification Linear Unit (RELU).

Two other architectures that have been proposed in [46] are VGG-m and VGG-s. Even though these two architectures are more complex in terms of number of layers and deliver better accuracy on object recognition task, their performance is worse on our dataset. We
believe this is mainly due to the larger kernel size in VGG-F architecture compared to the other two (i.e. $11 \times 11$ kernel size versus $7 \times 7$). Because to infer body orientation, the model should consider the correlation between body joints, so having larger kernel size in the first layer provides more chance to the network to learn the relative position of left and right joints in different categories. Neural Network models require large and diverse datasets to demonstrate good generalization. There exist several large datasets with 2D and 3D pose ground truth, but the number of human characters performing those activities is limited. This would causes poor performance on test images with different characters. Because, if the training set does not contain different clothing, body shapes and background variations, the network would not be able to generalize well to unseen characters. There are two strategies to tackle this problem in a ConvNet model: fine-tuning and collecting more training data.

In Fine-tuning approach, first the ConvNet is pretrained on a very large dataset with thousands of categories. The learned weights are subsequently used for initializing the parameters when the network is re-trained on the target dataset. In this work, the target dataset is the body orientation dataset.
Collecting more training data is one of the solutions to reduce overfitting in ConvNet models. This solution is always effective, but it is expensive in real world. Therefore, we have rendered synthetic 3D human characters with different clothing, skeleton shapes and various 3D poses.

We have performed several experiments on neural networks with different number of layers, filter sizes and network architectures. In all cases, the performance of VGG-f network was similar or better compared to other architectures. We could justify this as follows:

• By increasing the number of layers or using a more complex architecture, the number of parameters that should be learned increases and this requires a larger training dataset.

• Since we have limited number of training images, we use a pre-trained network for body orientation estimation task. On the other hand, the architecture of VGG, GoogleNet or LeNet network have been optimized to deliver the best performance on object recognition tasks. Therefore, it is best to use these architectures instead of customized ones.

Synthetic data generation is another solution to create training data. To generate a diverse set of characters, we have utilized MakeHuman [47], open source 3D computer graphic software. The 3D morphing feature of MakeHuman facilitates human character creation with variant attributes and clothing which would be otherwise a very time-consuming task. The CMU mocap dataset \(^1\) in BHV (Biovision Hierarchical Data), format is used as 3D skeleton.

\(^1\)The CMU data was obtained from http://mocap.cs.cmu.edu
Figure 3.2: Our synthetic characters generation framework

of the rendered characters. Next, these 3D poses are applied to all characters using “retarget” feature of MakeWalk add-on in Blender software \(^2\). Finally, different viewpoint images are obtained by rotating the human characters in 45 angular steps. Figure 3.2 illustrates our framework to generate synthesized training data. Figure 3.3 illustrates eight out of ten examples of the generated characters. In the experiments section, we evaluate the efficacy of the proposed synthetic data augmentation in estimation of body orientation category.

3.3 Dataset

In this dissertation we have used Human3.6m dataset [48], which is the largest benchmark of 3D pose estimation. It includes video recordings of 11 subjects performing daily activities such as Direction, Discussion, Eating, Walking Together, etc. Aside from the dataset size, the clothing variation of subjects performing the activities are more diverse

\(^2\)https://www.blender.org/
compared to other datasets such as Human Eva [49]. This is a very important factor to guarantee good performance of our body orientation framework. Only the activities that mainly consist of upright upper-body are considered i.e. Direction, Discussion, Greeting, Walking and Walking Together.

To analyze our single image 3D pose estimation approach, we follow the procedure in [7] and use a downsampled subset of Human3.6m dataset that is called H80K. In addition, to be consistent with other published work on these benchmarks, i.e. Human3.6m and H80k, the 17 joints skeleton representation in the camera coordinate system are used. Both 3D pose and estimated 2D pose are transferred to the pelvis point.

![Figure 3.3: Samples of our synthetic characters](image)

Figure 3.3: Samples of our synthetic characters
There are several datasets with human body orientation in depth domain, however to the best of our knowledge there is no publicly available dataset in RGB domain. Therefore, we should label the Human3.6M dataset with categorized body orientation for our training and test purposes. Manual labeling of the images is cumbersome and would not be feasible in such a large dataset. Therefore, we have calculated the yaw angle using 3D coordinates of markers on the right and left shoulders of the subject in each frame. In an ideal scenario where the mocap sensors are located on the shoulders, this angle would reliably represent the body orientation. But, these sensors are actually located on the upper arm that makes the automatic labeling noisy. We believe that by cleaning the dataset, the accuracy would improve. In Figure 3.4 we have shown some examples of the first four classes of our human body orientation dataset.

3.4 Experimental Results

In the following, we analyze the performance of two proposed approaches to estimate body orientation. VGG convolutional neural network architecture [46] is used to predict discretized body orientation. As illustrated in Figure 1.1, eight discretization levels are defined for the still image framework. For example, if the yaw angle is between 40° and 50° degree it gets labeled as second class or $\theta = 45^\circ$ orientation.

We have only considered 3D pose estimation for upright activity. This constraint along with categorization, has drastically reduced the number of training images in H80k dataset. Our training set contains 1516 images per category and 12128 in total. Our valida-
tion set has 683 images per category and 5464 images in total. The number of subjects in
the training and validation sets is five and two respectively. Therefore, to prevent overfitting
due to small number of training examples, we fine-tune the network that is pre-trained on
ImageNet ILSVRC-2012 dataset. The first row in Table 3.1 summarizes the results of this
model on two different scenarios:

- Within Subject scenario: the subjects present in the test set are the same in the
training set. The differences are only in poses, background and illumination.

![Figure 3.4: Samples of our training dataset](image-url)
Table 3.1: Classification error of body orientation estimation in still images

<table>
<thead>
<tr>
<th>Training dataset</th>
<th>Scenario</th>
<th>Within subjects</th>
<th>Across subjects</th>
</tr>
</thead>
<tbody>
<tr>
<td>H80K dataset</td>
<td></td>
<td>8.5%</td>
<td>34%</td>
</tr>
<tr>
<td>Centered H80K dataset</td>
<td></td>
<td>5%</td>
<td>30%</td>
</tr>
<tr>
<td>H80K dataset + Synthetic dataset</td>
<td></td>
<td>3.7%</td>
<td>20%</td>
</tr>
</tbody>
</table>

- Across Subjects scenario: the subjects in the test set are different from the ones in the training set. This is harder task, because the model should also learn to be independent to body shape and clothing.

The second column in Table 3.1 shows that the performance of the first model degrades in the across subjects scenario. Therefore, as discussed before, we have use synthetic characters to increase the variation of clothing in the training set. In each experiments, we increase the number of synthetic subjects by one (305 images in each category per subject), train the network again and test it on the real-world dataset. This leads to the improvement of across subject scenario’s error rate. However after some point accuracy improvement was not noticeable. In the final experiments, we create ten different synthetic subjects (five males and five females) which in total increases the number of training set images by 24400. This data augmentation leads to 14% reduction in camera viewpoint estimation error (second row of Table 3.1).

In this chapter, we presented our methodology to annotate real-world images automatically with body orientation; create a synthetic dataset; and train appropriate ConvNet to estimate categorized body orientation in still images. In the next chapter, we will discuss our approach to extract continuous body orientation in videos.
In the previous chapter, we employed a ConvNet model to estimate categorized human body orientation. To make the model more invariant to human body shape and clothing variations, we generated synthetic characters with different appearance. However, the model is still limited to the real and synthetic datasets. For example, if the character in the test image wears skirt, the training set should have a similar example to enable the model to generalize well on this type of clothing. In this section, we leverage the temporal information in the video and hypothesize that the variation of body joints over time, provides a strong clue regarding a person’s intended direction of movement. In this case, the model will be less affected by the appearance of the subjects.

There are different models to analyze the sequential data such as Kalman filters, Markov models and conditional random fields, but they are not designed to learn long-range dependencies between data. Also, there are other models that work on the engineered features that require domain knowledge. Neural networks are powerful models that learn both data representations and classifiers from data. In this chapter, we employ recurrent neural network (RNN) model [50], [51] to analyze human dynamics. RNNs are powerful and increasingly used in different applications. For example, in [52], a model has been presented to estimate the distance to the leading vehicle using the temporal measurements of LIDAR,
gyroscopes and GPS in a self-driving car. Similarly, Zachary et. al in [53] has presented the first study to evaluate the ability of RNN models to recognize patterns in clinical medical data of intensive care units. The input to the model is sensor data and lab test results and the goal is to train a model to classify 128 diagnoses. Motivated by these approaches, we employ RNN to estimate continuous body orientation from trajectories of body joints.

Figure 4.1 illustrates our proposed 3D pose estimation framework in video, where the input to the RNN module is 2D body joint locations in the last $l$ frames and the output is the continuous human body orientation angle. Figure 4.2 shows the input to the RNN module.

### 4.1 Background on Recurrent Neural Networks

Recurrent Neural Network can model the contextual information of a temporal sequence using a feedback loop that is embedded in its structure. The loop allows information
to be passed from one time step to the next time step, acting similar to memory. In this model, the input data is mapped to the hidden states, followed by their projection onto output space using a nonlinear function. The problem is formulated as a mapping from a sequence, $\bar{x}_i = [x_1^i, x_2^i, \ldots, x_l^i]$ of length $l$ image frames to the real-valued body orientation angle, $\theta_i$. $x_t^i$ is a $d$-dimensional feature representation of the $t$-th frame in the $i$-th sequence that is basically 2D locations of body joints (16 joints). Let $\mathbf{X} = \{\bar{x}_i\}_{i=1}^n$ represents the training inputs and $\theta = \{\theta_i\}_{i=1}^n$ be a collection of corresponding body orientation angles. $\bar{h}_t$ and $\bar{x}^t$ that are the image evidences at frame $(t-1)$-th. Following equations shows the relation between these variables and the orientation angles:

$$\bar{\theta}^t = \psi \left( \bar{W}^T_{hy} \bar{h}_t^{t-1} \right)$$  \hspace{1cm} (4.1)

$$\bar{h}_t = \phi \left( \bar{W}^T_{hh} \bar{h}_t^{t-1}, \bar{W}^T_{xh} \bar{x}^{t-1} \right)$$  \hspace{1cm} (4.2)

Figure 4.2: Input to the RNN model
In these equations, $\vec{W}_{xh}, \vec{W}_{hh}, \vec{W}_{hy}$ are learnable weight matrices, and $\phi(.)$ and $\psi(.)$ are non-linearity functions. Even though this model is powerful, it is hard to train it when the input sequence length grows, especially with tanh and sigmoid activation functions.

To train the RNN model, the full input sequence is treated as a single training example and the error is the sum of errors at each time step. Therefore, the gradient for one training example is the sum of the gradients at each time step. This is similar to the standard backpropagation algorithm in feedforward neural networks, except that at each time step, the gradient is the sum of previous time steps. This method is called BackPropagation Through Time (BPTT).

The nonlinearities in the RNN model have derivatives of zero when the absolute value of their input increases. These zero derivatives will make the gradients in early layers becomes zero as well. Therefore in RNN, the contributions of “far away” steps becomes zero which renders the model unable to learn long-range dependencies. This is called vanishing gradient problem. One technique to overcome this problem is to use ReLU activation function, because its derivative is constant. A preferred technique is to use Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) [54] architecture, both are designed to deal with vanishing gradient and help with learning long-range dependencies.

Long Short Term Memory or LSTM [55] is one the effective models to tackle vanishing gradient problem in RNN by simulating internal memory and learning some gates to enable reading, writing or erasing from memory. The input, forget, and output gates are represented as $\vec{i}$, $\vec{f}$, and $\vec{o}$ respectively. These vectors have exact same equations with different
parameter matrices and are called gate because sigmoid function maps them between 0 and 1. Therefore, multiplying them with another vector will specify how much of the other vector should be let through. The dimension of these gates is the same and equal to the hidden state dimension, \( d \).

- The input gate limits the newly computed state for the current input.
- The forget gate limits the previous state.
- The output gate limits the internal state that is exposed to the higher layers in the next time step.

Following equations show the LSTM model and how to compute gates, hidden state, and finally the output.

\[
\vec{i}_t = \sigma \left( \vec{U}_i \vec{x}_t + \vec{W}_i \vec{s}_{t-1} \right), \\
\vec{f}_t = \sigma \left( \vec{U}_f \vec{x}_t + \vec{W}_f \vec{s}_{t-1} \right), \\
\vec{o}_t = \sigma \left( \vec{U}_o \vec{x}_t + \vec{W}_o \vec{s}_{t-1} \right), \\
\vec{g}_t = \tanh \left( \vec{U}_g \vec{x}_t + \vec{W}_g \vec{s}_{t-1} \right), \\
\vec{c}_t = \vec{c}_{t-1}.\vec{f} + \vec{g}.\vec{i}, \\
\vec{s}_t = \tanh(\vec{c}_t).\vec{o}
\]
\[ g_t \] is called ‘candidate’ hidden state and is similar to the equation in RNN for updating hidden state. \[ c_t \] is called "internal memory" and is a summation of the previous memory \[ c_{t-1} \] multiplied by the forget gate, and the gated version of the newly computed hidden state \[ g_t \]. \[ s_t \] is the output hidden state that is equal to multiplication of the "internal memory" with the output gate.

4.2 Our Recurrent Model

In our baseline method, x and y locations of all 16 body joints in a frame sequence of length \( l \) used as the input to the LSTM. Hence, the input is a \( 32 \times l \) matrix that requires large amount of training data to robustly optimize the objective function. The number of parameters of standard LSTM is equal to \( 4 \times d \times (d+n) \) where \( d \) is the number of hidden states and \( n \) is the input dimension. In this section, we try to reduce the dimensionality of the input, since intuitively not all the body joints would contribute to the body orientation inference. For example, the neck joint could have almost random location in each of the body orientation categories. Therefore, we feed the input data which is the 2D pose in each frame, to a Multi Layer Perceptron (MLP) and train the network to learn the weights of MLP and LSTM simultaneously. Figure 4.3 illustrates our framework. We have performed a grid search to find the number of hidden layers of MLP. Two hidden layers with 16 output dimensions yields the best performance.
4.3 Experimental Results

In this section, we illustrate the performance of our proposed model to estimate the continuous human body orientation in video. We have used Human3.6M dataset [48] (more details are provided in Chapter 3), for training and validation purposes. Videos are downsampled from their original frames by selecting every third frame. The input to the RNN model is a sequence of 2D joints (32 dimension) and the output is body orientation of the subject in the last frame in degrees. Each sequence contains N=10 consecutive frames. The model is implemented using the Keras \footnote{https://github.com/fchollet/keras} library. All data points in all upright activities are used for training and the model is tested on each activity separately. The number of
training sequences is 21039, and each test activity consists of approximately 2,242 sequences. The model is a two-layers LSTM with 100 hidden units in each layer. RMSProp is used during optimization to modulate the per-parameter learning rate. The batch size is 50 and mean-absolute-error is used as the loss function. The continuous body orientation estimation and ground truth are displayed in Figure 4.4. Each subplot illustrates qualitative results of each upright activity. Figure 4.4 shows that the model is able to follow the variation of body orientation, except when these variations are very fast. For example, in frame 750 of the "Discussion" activity we observe a high error rates occurs when the person suddenly turns.

We have presented the quantitative results of body orientation estimation in video in Table 4.1. The first column is mean absolute error and the second one is standard deviation. The error in “Walking” and “Walking Together” activities are higher compared to the rest of activities. The subjects in these activities walk vary fast in the circles, as previously mentioned our algorithm is not able to follow fast variations. One possible future direction could be to train a separate model for these activities with smaller input sequence length.

In this section, we compare our two frameworks of body orientation estimation: the first model is based on convNet which takes a still image as an input (as explained in Chapter
Table 4.2: Error rate of CNN and RNN based orientation estimation approaches

<table>
<thead>
<tr>
<th>Method</th>
<th>Direction</th>
<th>Discussion</th>
<th>Greeting</th>
<th>Walking</th>
<th>Walking together</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>22.91%</td>
<td>20.19%</td>
<td>32.29%</td>
<td>✅ 25.37%</td>
<td>✅ 29.26%</td>
<td>✅ 26.00%</td>
</tr>
<tr>
<td>RNN</td>
<td>15.95%</td>
<td>16.47%</td>
<td>28.46%</td>
<td>30.73%</td>
<td>32.20%</td>
<td>24.76%</td>
</tr>
</tbody>
</table>

3), and the second one is the proposed model in this chapter. The output of RNN model should be discretized into eight bins to be consistent with the ConvNet categorical body orientation. The results in Table 4.2 illustrate the superior performance of our RNN model in most upright activities. The ConvNet model has better performance in "Walking" and "Walking Together" activities. This could be due to the fast variations of the viewpoint angle as the person walks in a circular path in these activities. Overall, utilizing the temporal data leads to 1.24% improvement.

To prove the effectiveness of MLP-LSTM model compared to the LSTM, we have shown the learning curves of both models in Figure 4.5. In both subplots, the blue line is the performance on the training set and red line is on validation set. The thick lines represent our new approach, MLP-LSTM. Left subplot shows the error on continuous human body orientation in degree which is less for both training and validation datasets in MLP-LSTM. The right subplot shows the loss versus epoch. In both models, the training loss and validation loss decrease with the same speed which shows that the model has not overfitted. In addition, the loss in MLP-LSTM is less than LSTM which shows the superior performance of our approach.

In this chapter, we employed an RNN-LSTM based approach to utilize the human movement dynamics to infer body orientation from the 2D joints locations in an image
sequence. We also improved the model by pre-processing the image inference by MLP. Our experiments’ results prove the effectiveness of our approach, particularly when the variations of body orientation is not fast. In the next chapter, we will discuss human 3d pose estimation and we will show how the estimated body orientation by the methods explained in the current and previous chapters can be used to improve human 3D pose estimation.
Figure 4.4: Point-wise prediction of body orientation by LSTM. Dashed lines correspond to the ground truth; solid lines are predictions. WalkTogether(left), Greeting(right)
Figure 4.5: Learning curve of LSTM and MLP-LSTM models
In this chapter, we illustrate our method to estimate 3D human pose with respect to the camera in a single RGB image. Our approach belongs to the learning-based or bottom-up category of approaches. Our goal is to both learn expressive image features and a regression model to account for missing depth information and correlation among body joints. We hypothesize that human body orientation with respect to the camera provides coarse information on the relative depth of body joints. In addition, we have directly used the body joint locations in the image plane to express the relative location of them is X, Y coordinates.

Therefore, the image evidence, in our work, is the concatenation of 2D joints (32 dimensions) and human body orientation. Figure 5.1 outlines our approach that can be split into three major parts: viewpoint estimation\(^1\), 2D joint localization, and regression model. In Chapters 3 and 4, we have provided our proposed approaches to estimate body orientation in still images and video. Employing the body orientation has constrained our framework to only activities with upright upper-body.

\(^1\) Portions of this chapter were reprinted from international conference on 3D vision, Ghezelghieh, M.F., Kasturi, R. and Sarkar, S., 2016, October. Learning camera viewpoint using CNN to improve 3D body pose estimation. In 3D Vision (3DV), 2016 Fourth International Conference on (pp. 685-693). IEEE, Copyright (2016), with permission from IEEE.

\(^1\) In this paper we use body orientation and camera viewpoint angle interchangeably.
The naive approach is to directly append body orientation to 2D features, but this could cause distance ambiguities. For example, let us assume that the orientation of the subject in the first image is frontal, $\theta = 0^\circ$. The orientation of the subject in the second image is $\theta = 315^\circ$ and third image is backward, $\theta = 180^\circ$. In this case, the first image is more similar to the second image than the third image in terms of viewpoint angle (see Figure 1.1). To resolve this problem, we map the orientation angle to the $(\sin \theta, \cos \theta)^T$ vector. This vector is further scaled by a fixed coefficient $M$ to account for the influence of viewpoint in 2D feature representation. In our experiments, $M$ is found by grid search to maximize average accuracy over all activities.

5.1 2D Human Body Joint Detection

The hypothesis behind our approach is that the combination of human body orientation with 2D joint locations would serve as an expressive feature representation to regress
3D joint locations. The state-of-the-art 2D pose estimation model [56], "hourglass network" is used in this dissertation. The idea behind this approach is to design a convolutional neural network such that it processes the image locally and globally. Therefore it is able to estimate body joints and utilize the correlation between them to predict the location of occluded joints. Therefore, the hourglass module processes and combines features across different scales [57] [58]. In this architecture, to get to a lower resolution image, convolutional and max pooling layers are used. Before each pooling layer, the network branches off and applies more convolutional layers. After reaching to the lowest resolution, the network begins top-down sequence (going from low to high resolution) by upsampling and combining features across scales. The architecture is symmetric; therefore, for every layer in bottom-up path there is one layer in top-down path. After reaching to the output resolution, two consecutive $1 \times 1$ convolutions are applied to produce final predictions. The output is a heatmap for each body joint. Every pixel represents the probability of presence of that joint in that pixel. More details of the 2D pose estimation model are provided in [56]. To test our 3D pose estimation framework in more realistic scenario, we have not fine-tuned this model on our target dataset.

5.2 Part-based 3D Regression Model

The goal in this section is to build a regression model to estimate 3D pose in a single RGB frame. To account for high articulation in human body, we have proposed “joint-set” regression model [59]. In this model, the body joints are divided into three sets: left-hand,
right-limb, and extended torso (spine, head, neck and legs), as shown in Figure 5.2. For each group a regression model is learned. This will increase the pose variations in the training set. For example, consider a training set that includes an image of a walking person with hands in the pocket and a person who is sitting on a chair and talking on the phone. Now, if the test image is a walking person who is talking on the phone, the "joint-set" regression model can exploit both training samples to regress the target pose of unseen input.

The proposed structured prediction method in [60] is adopted to estimate the 3D pose. In this method, the distributions of input and output features are modeled by Gaussian Processes. The target pose at the given test input is predicted by minimizing the Kullback-Leibler (KL) divergence between these distributions. In the following subsection we provide a brief introduction on Gaussian processes [61].
5.2.1 Gaussian Process

A Gaussian processes (GP) is a collection of random variables, any finite number of which have joint Gaussian distribution [62]. In other words, a function $f$ is a GP random variable with mean function $\mu$ and a covariance $k$:

$$f \sim GP(\mu, k)$$

(5.1)

if for any input vector $[r_1, r_2, ..., r_n]$, the corresponding vector of function values is Gaussian with mean $\mu$ and covariance matrix $K_{R,R}$:

$$[f(r_1), f(r_2), ..., f(r_n)] \sim \mathcal{N}(\mu, K_{R,R})$$

(5.2)

where $K_{R,R}(i, j) = k(R_i, R_j)$ is covariance function or kernel.

The basic assumption in supervised learning is that similar input points are likely to have similar target values, and thus training points that are close to a test point should be very informative on the target value of that test point. This is why the notion of similarity between data points plays an important role in the performance of the model. In GP model, covariance function defines the closeness and similarity.
In the Gaussian process regression model the goal is to predict \( y_* \) at test point \( x_* \).

Assuming noisy observations, the regression function \( f \) is given as:

\[
y = f(x) + \mathcal{N}(0, \sigma_n^2)
\]  

(5.3)

\( f(x) \) can be any function such as linear, quadratic or even polynomial function. In GPR, \( f(x) \) is a Gaussian Process. Based on the key assumption in GP, the data has the following multivariate Gaussian distribution:

\[
\begin{bmatrix}
  y \\
  y_*
\end{bmatrix} \sim \mathcal{N}(0, \begin{bmatrix} K & K^T \\ K_* & K_{**} \end{bmatrix}).
\]  

(5.4)

The three matrices are calculated as follows:

\[
K = \begin{bmatrix}
  k(x_1, x_1) & k(x_1, x_2) & k(x_1, x_3) & \ldots & k(x_1, x_n) \\
  k(x_2, x_1) & k(x_2, x_2) & k(x_2, x_3) & \ldots & k(x_2, x_n) \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  k(x_n, x_1) & k(x_n, x_2) & k(x_n, x_3) & \ldots & k(x_n, x_n)
\end{bmatrix}
\]  

(5.5)

\[
K_* = \begin{bmatrix}
  k(x_*, x_1) & k(x_*, x_2) & \ldots & k(x_*, x_n)
\end{bmatrix}
\]  

(5.6)
\[ K_{**} = k(x_*, k_*), \quad (5.7) \]

In the regression problem, we are interested in likelihood probability \( p(y_*|y) \) that has also Gaussian distribution:

\[ y_*|y \sim \mathcal{N}(K_* K_*^{-1} y, K_{**} - K_* K_*^{-1} K_*^T) \quad (5.8) \]

Therefore the best estimate of \( y_* \) is the mean of this distribution:

\[ y_* = K_* K_*^{-1} y, \quad (5.9) \]

The uncertainty in the estimate is given by:

\[ \text{var}(y_*) = K_{**} - K_* K_*^{-1} K_*^T \quad (5.10) \]

### 5.2.2 Twin Gaussian Process

Let’s represent the input feature by \( r \) and the corresponding 3D pose by \( z \). Similarly, \( R = (r_1, r_2, ..., r_N) \) and \( Z = (z_1, z_2, ..., z_N) \) stands for training inputs and outputs respectively.

We follow the approach in [60], to model each dimension of the observation vector by a Gaussian process [63].
The joint distribution of test input, $r$, and training inputs, $R$, is represented as Gaussian distribution too:

$$
\mathcal{N}_R \left( 0, \begin{bmatrix}
K_R & K^r_R \\
(K^r_R)^T & K_R(r, r)
\end{bmatrix} \right)
$$

(5.11)

where $K_R$ is the $N \times N$ covariance matrix of training features and $K^r_R$ is a covariance function of test input and training inputs ($N \times 1$ vector). The elements of covariance matrix are given by covariance function which is usually chosen to be Radial Basis Function (RBF):

$$
K_R(r_i, r_j) = \exp \left( -\gamma_r \| r_i - r_j \|^2 \right) + \lambda_r \delta_{ij}.
$$

(5.12)

where $\gamma_r$ is the kernel width parameter, $\lambda_r$ is the variance of the noise and $\delta_{ij}$ is the Kronecker delta function. Similarly, the joint distribution of target training data and unknown test data, $z$ is shown by $\mathcal{N}_Z(0, K_{Z\cup z})$, where the covariance matrix is estimated as:

$$
K_{Z\cup z} = \begin{bmatrix}
(Z^{(d)})^T \\
Z^{(d)}
\end{bmatrix}
\begin{bmatrix}
(Z^{(d)}) & Z^{(d)}
\end{bmatrix}
$$

(5.13)
Following the derivations in [60], the Kullback-Leibler divergence measure between these two distributions is given by:

\[
L(z) = D_{KL}(\mathcal{N}_Z\|\mathcal{N}_R) = K_Z(z, z) - 2(K_Z^z)^\top K_R^{-1}K_R^z - [K_R(r, r) - (K_R^r)^\top K_R^{-1}K_R^r] \times \log \left[ K_Z(z, x) - (K_Z^z)^\top K_Z^{-1}K_Z^x \right]
\] (5.14)

where \(K_Z^z\) is a \(N \times 1\) column vector defined as:

\[
(K_Z^z)_i = K_Z(Z_i, z)
\] (5.15)

Therefore, estimated 3D pose, \(z^*\), is obtained by minimizing this divergence measure [60]:

\[
z^* = \arg \min_z [L(z) = D_{KL}(\mathcal{N}_Z\|\mathcal{N}_R)]
\] (5.16)

### 5.3 Experimental Results

In this section, first, we prove the efficacy of each step of our proposed framework. Next, we provide the comparison with state-of-the-art approaches on both single monocular
RGB image and video. In our experiments, we have used the Human3.6m dataset [48]. This is the largest dataset with 3.6 millions images and corresponding 3D pose annotations. Please refer to chapter 3, for more detail on the dataset. To fully utilize our body orientation framework, only activities that consist mainly upright poses are considered i.e. Direction, Discussion, Greeting, Walking and Walking together. In future, we are planning to extend the body orientation model to sitting down or lying down activities.

In video domain, we have only compared our results with those approaches that have utilized temporal information in video, even though the still image schemes can be easily extended to the video. For this reason, similar to other papers in 3D pose estimation, the same baseline is chosen in still image and video. In the baseline method [48], Fourier approximation of the HOG features of each image is used as input feature. The corresponding 3D pose is mapped to a reproducing kernel Hilbert space and used as output in the ridge regression model. In addition, the error measure explained at the following is used as the comparison metric:

Similar to other papers that reported on 3D pose estimation benchmark, we calculate MPJPE (Mean Per Joint Position Error) metric. For each image, this metric is given by

$$E_{\text{MPJPE}} = \frac{1}{N_S} \sum_{i=1}^{N_s} \left\| \mathbf{m}_{\text{est}}(i) - \mathbf{m}_{\text{gt}}(i) \right\|_2$$

(5.17)

where $N_S$ is the number of joints in the skeleton, $\mathbf{m}_{\text{gt}}(i)$ is the 3D coordinate of $i$th joint and $\mathbf{m}_{\text{est}}(i)$ is the estimated coordinate.
Figure 5.3: 3D error of each body joint

Figure 5.3 depicts the results of mean 3D pose estimation per body joint for the “Direction activity”, with and without utilizing body orientation. We observe a significant improvement in the right and left hand joints but only a slight improvement in the legs and torso. This can be attributed to the higher degree of freedom in hands compared to legs and torso, which leads to a higher chance of ambiguity in inferring 3D from 2D coordinates.

We have investigated the effect of non-perfect 2D pose estimation. To make the experiments isolated from our body orientation estimation, ground truth body orientation is used in this experiment. Figure 5.4 illustrates the mean 3D pose estimation error for each joint set. The performance drops for the right and left hand joints. Part of this performance drop could be due to the frequent occlusion of the hands.
Table 5.1: Efficacy of joint-set regression on 3D pose estimation error in millimeters.

<table>
<thead>
<tr>
<th>Regression Model</th>
<th>All-joint</th>
<th>Joint-set</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject 5</td>
<td>74.1</td>
<td>70.45</td>
<td>5.2%</td>
</tr>
<tr>
<td>Subject 6</td>
<td>105.7</td>
<td>99.89</td>
<td>5.8%</td>
</tr>
</tbody>
</table>

Our joint set regression is more effective compared to the approaches that estimate all joints with one regression model. Similar to the previous experiment, in this part, ground truth body orientation is used in the 3D pose estimation to isolate the effect of regression model from body orientation estimation error. Table 5.1 shows our results on validation set subjects. Our 'joint-set' regression model improves the accuracy in 3D pose estimation for both subjects.

Our comparison with the state-of-the-art approaches on 3D pose estimation in still images is summarized in Figure 5.5. Some approaches have not reported the 3D pose error on “Direction” activity; the corresponding bar is therefore is missing in this Figure. We have been able to improve baseline by 31.15% on these six activities. Our approach has superior performance compared to the rest except “DMHS” approach which has recently

![Figure 5.4: Effect of non-perfect 2D pose on 3D pose reconstruction error](image)

57
been proposed in [23]. This framework has been trained for multiple tasks on several dataset: body parts segmentation, background segmentation, 2D and 3D pose estimation.

We have evaluated our video based 3D pose estimation approach on H3.6M benchmark and compared it with state-of-the-art methods. All the approaches listed in Figure 5.6, except the baseline [48], have utilized temporal information in video to estimate 3D pose. In average, our approach improves the baseline and state-of-the-art by 31.84% and 12.8% respectively.

In this chapter, we proposed our framework to estimate 3D coordinates of body joints in upright activities. We performed extensive experiments to prove the efficacy of our approach compared to the state-of-art methods. In the next chapter, we will propose our method to estimate physical properties of an object using its motion trajectory.

![Figure 5.5: 3D pose estimation error in still images in millimeters. Some approaches have not reported the 3D pose error on "Direction" and "Walking Together" activities.](image-url)
Figure 5.6: 3D pose estimation error in video sequence in millimeters.
CHAPTER 6: ESTIMATION OF OBJECTS PHYSICAL PROPERTY

CLASSES

6.1 Introduction

In recent years, the technology of self-driving cars has made dramatic progress. One of the critical challenges of this emerging technology is the safety of both car occupants and other road users. The current prototype of autonomous cars is equipped with advanced sensors such as ultrasonic, vision, radar and LIDAR. These sensors along with sophisticated data fusion algorithms are able to detect and track obstacles in real-time with very good resolution.

When an obstacle is detected in the planned path, either its planned route should be modified or the vehicle should come to stop. Depending on the traffic situation and vehicle speed, this policy could cause collision with other vehicles. Therefore, obstacle avoidance may not always be the safest action. Similar challenge has been discussed in [64].

The intuitive solution would be to recognize the object before taking an action. The intelligent unit should predict whether it is safe to pass over the object or it should inevitably follow avoiding policy.

---

1 Portions of this chapter were reprinted from the paper "Fathollahi, Mona and Kasturi, Rangachar, Autonomous driving challenge: To Infer the property of a dynamic object based on its motion pattern using recurrent neural network", Copyright (2016), Permission is included in Appendix A.
Figure 6.1: Selected frames of dynamic objects on the road. (a) a plastic container which is safe to collide, (b) a heavy object that should be avoided.

A sample video for each scenario is downloaded from Youtube and a few frames are shown in the Figure 6.1. In the first video, an empty plastic container is bouncing in the road that is safe to pass. In the second video, a heavy object is falling out of the front car that should definitely be avoided.

The immediate solution that one might consider is to formulate the problem as a regular image classification task and collect a dataset of collision safe and unsafe objects. While there is much progress in object detection/recognition methods [65], this approach has several challenges which makes it ineffective for this particular application.

First, collecting a dataset that contains different objects in different lighting conditions and viewpoints is a difficult task by itself. Second, it is almost impossible to infer the
weight of an object from its visual cues. For example, two very similar boxes with one of
them filled with metal pieces and the other one that is empty have similar images.

Finally, there is a high possibility of recognition failure because the image resolution
is usually poor for far away objects. Also, the classifier should decide in a short period of
time, where motion blur might make the problem even more challenging. For example, the
white plastic container in the first column of Figure 6.1 could be classified as a gas cylinder.

A human easily resolves these challenges by observing the trajectory of empty box
versus heavy box (e.g. plastic container versus a gas cylinder). Therefore, assuming that the
real-time trajectory of the dynamic object is available [66], we claim that motion pattern
provides strong insight to infer the object dynamics accurately and to classify it as a "safe
to pass over" or "must avoid" object.

6.2 Method

In this section, our goal is to design a classifier to infer object’s bounciness char-
acteristic based on its trajectory when it hits the ground. Our approach is based on the
observation that the bouncing pattern of objects is directly affected by their mass.

6.2.1 Data

To collect data, we should throw different objects with different masses and shapes
and record their trajectories. On the other hand, since the bounciness of the object is also
related to initial velocity, we should collect a large amount of data to be able to learn the
Figure 6.2: Object trajectory synthesis in Blender. The first few frames are to generate initial velocity (Linear motion). The second part, physics simulation, is recorded as trajectory of the object. Therefore, dataset collection in this case is cumbersome and expensive.

Therefore, we generate synthetic videos with binary labels denoting heavy or light object trajectory. We utilize open source 3D creation suite, Blender [67], to generate motion data of bouncing objects. Blender uses “Bullet Physics Library” for collision detection, rigid body dynamic simulation and other Physics simulations tasks.

Each trajectory starts from random coordinates and Euler angles and the object has random initial linear and angular velocities. To generate random initial velocities, two key-frames are inserted at first and seventh frames. Also, the height of object at first frame and both linear and angular positions at seventh frame are randomized. (Figure 6.2).
physics engine takes over the object animation after seventh frame. The world coordinates of the object after the seventh frame are recorded as object trajectory time series.

We only consider two object categories; the first class is the trajectory of light objects that have a high tendency to bounce when they hit the ground, and the second class are the objects that are heavy and have more tendency to slide than bounce. We generated 1000 training videos, and 1000 test videos for both categories.

Some randomly chosen examples of the generated trajectory data are shown in Figure 6.3. Even though there is a clear distinction between Z dimension of the trajectories, we still see subtle difference in X and Y components. For example, for a light object (higher bounciness) it takes more time to come to a full stop and this is reflected in X and Y coordinates and this justify the superior performance of classification when 3D data is used (Table 6.1). Finally, although some statistical differences are detectable between the two categories, the plots in this figure suggests that no simple rule can be proposed based on, for example, the number of bounces or time series duration; therefore, a more involved classification algorithm is required.

6.2.2 Classifier

In this section, we assume that a tracking algorithm estimates the trajectory of the object. Therefore the problem is reduced to time series classification.

We adopted, Recurrent Neural Network models (RNN) for trajectory classification. RNN is a type of artificial neural network that is able to process data with arbitrary input
sequence lengths. Their internal memory units and feedback loops have made them a very successful tool in sequential data prediction in various domains [68] [69]. Recently, they have been used in the context of time series classification [70]. We also use RNN architecture with Long Short-Term Memory (LSTM) units [71] for motion trajectory classification.

The input to the network is the first $T$ seconds of objects’ trajectory. Training and test time series are normalized by their standard deviations in each dimension. Our network architecture is a two-layered LSTM with 64 hidden units in each layer. The hidden state at the last time step of LSTM is fed into a softmax layer. We have also added a dropout layer between second LSTM layer and softmax layer with rate 0.8. To compute parameter gradients, the truncated back-propagation-through-time (BPTT) approach [72] is used to reduce the probability of vanishing gradient problem for long input sequences. The entire implementation is done using Tensorflow [73] package.
In the first experiment, the impact of input dimension (XYZ vs Z) on the classification accuracy is studied. We perform grid search on the parameters to get the maximum accuracy when only z-coordinate of the trajectories is used for training. The same parameters are used to train the network with 3D input, the results are compared in Table 6.1. Superior performance was achieve with 3D inputs, because it takes more time for a light object to stop along X and Y direction.

In the second experiment, we study the influence of input sequence length on the accuracy. If it is too short, the classifier has limited data and might not be able to learn distinguishable pattern. On the other hand, increasing the input sequence beyond some limit could cause the gradient of LSTM network to start vanishing or exploding, which

Table 6.1: Accuracy of 3D and 1D object trajectory classification

<table>
<thead>
<tr>
<th>Trajectory dimension(s)</th>
<th>Best Accuracy(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X, Y, Z</td>
<td>81</td>
</tr>
<tr>
<td>only Z</td>
<td>78</td>
</tr>
</tbody>
</table>

Figure 6.4: Classification accuracy versus trajectory length
consequently leads to the accuracy drop. In the Figure 6.4, we have shown the classification accuracy for different lengths of input sequence.

6.2.3 Experiment on Real-world Data

In this section, we leverage the trained network on synthetic data, to analyze the trajectories of real-world objects. One extreme example is chosen from each category: golf ball as an object with high bounciness, and wooden cube as an object with low bounciness. We throw them from different heights with different initial velocities and record the video, Figure 6.6. The objects are marked with a distinct color to be able to use a simple color tracker. Lastly, when the frames are blurred due to the fast motion of the object, missing
Figure 6.6: Samples of real-world trajectories. (Left) golf ball (Right) wooden cube. Parts of the trajectory are reconstructed by a simple interpolation. The trajectories that are shorter than input sequence length are zero-padded. For each category, we collected 20 videos and plotted the trajectories in Figure 6.5. In this experiment, the trajectories are recorded with a single RGB camera and only trajectories along the z direction are used for decision making. Therefore, we used the trained network on z channel as well. We obtained an accuracy of 93% on the ball and 100% on the wooden cube.
CHAPTER 7 : CONCLUSIONS AND FUTURE WORK

In this dissertation, we have studied two challenging problems in Computer Vision: human pose estimation and object physical property inference. Any improvement in these two domains may have potential applications for improving the safety of pedestrians and passengers in autonomous car technology. In this section, we summarize methods that we have developed, possible limitations and future works.

In human pose estimation domain, we have accomplished three goals:

- We have proposed a framework to estimate categorical body orientation in still RGB images. To account for clothing and body shape variations in human appearance, we have used synthetic 3D characters to systematically augment training dataset. One of the limiting factors of this approach is that improvement in error rate is not linear with respect to the number of synthetic characters and at some point, adding more synthetic images to the training set does not show a noticeable effect. This could be justified by the gap between feature distribution of real images and synthetic images. To improve accuracy, one possible future direction is to create more realistic synthetic data or to design a model to map the extracted features from synthetic and real data onto a common space.
• To be more invariant to the appearance of a person and increase angular resolution, we have proposed a MLP-RNN model to map trajectories of body joints to continuous body orientation. One of the shortcomings of our proposed framework is that it sometimes fails to follow the variation of body orientation when the person changes his or her direction quickly. A direction of future work includes designing a model to simultaneously utilize RGB images and 2D trajectories of body joints to estimate body orientation in video.

• We have illustrated that 2D body joints’ locations in combination with body orientation is an effective lightweight feature representation to estimate 3D human pose from a monocular RGB image. To prove the efficiency of our body orientation in improving 3D human pose, we have performed extensive experiments on the largest 3D human pose benchmark dataset, Human3.6M. This dataset is collected in an experimental setting and each image contains only one person. In real-world scenarios, usually there are multiple people in a scene that might cause occlusions, particularly if they are interacting with each other. Therefore, collecting a dataset that includes multi-person scenarios would help in developing new models that are more practical.

In the second phase of this dissertation, we have shown that motion pattern of an object provides a strong insight into some of object’s physical properties. For example, light objects tend to bounce more compared to heavy objects. This has a potential application in the autonomous driving technology and can reduce the number of dangerous stops or maneuvers when an object suddenly appears in front of the vehicle.
Our preliminary experiments show promising results on a synthetic small set of real-world data. One of the limitations of this framework is lack of clear definition of objects that are safe to collide with. In this dissertation, trajectories of objects with low and high restitution are simulated in a realistic 3D physics engine to model two categories of safe and dangerous objects. However, in a real-world scenario, shape, weight and texture of an object should be carefully considered on deciding whether it is a safe object to collide with or not. Extensive future experiments are required to investigate the effect of each parameter towards building a more reliable real-world dynamic object classification algorithm.
REFERENCES


APPENDIX A: COPYRIGHT CLEARANCE FORMS

Below is the permission for Chapters 3, 5, 6.
Learning Camera Viewpoint Using CNN to Improve 3D Body Pose Estimation

Conference: 3D Vision (3DV), 2016 Fourth International Conference on

Author: Mona Fathollahi Ghezelghieh

Publisher: IEEE

Date: Oct. 2016

Copyright © 2016, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.

2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.

3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author’s approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]

2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.

3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity’s name goes here]’s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.
Order Completed

Thank you for your order.

This Agreement between Mona Fathollahi ("You") and Springer ("Springer") consists of your license details and the terms and conditions provided by Springer and Copyright Clearance Center.

Your confirmation email will contain your order number for future reference.

Printable details.

<table>
<thead>
<tr>
<th>License Number</th>
<th>4107111202584</th>
</tr>
</thead>
<tbody>
<tr>
<td>License date</td>
<td>May 13, 2017</td>
</tr>
<tr>
<td>Licensed Content</td>
<td>Springer</td>
</tr>
<tr>
<td>Publication</td>
<td>Springer ebook</td>
</tr>
<tr>
<td>License Content Title</td>
<td>Autonomous Driving Challenge: To Infer the Property of a Dynamic Object Based on Its Motion Pattern</td>
</tr>
<tr>
<td>License Content Author</td>
<td>Mona Fathollahi</td>
</tr>
<tr>
<td>Licensed Content Date</td>
<td>Jan 1, 2016</td>
</tr>
<tr>
<td>Type of Use</td>
<td>Thesis/Dissertation</td>
</tr>
<tr>
<td>Portion</td>
<td>Full text</td>
</tr>
<tr>
<td>Number of copies</td>
<td>1</td>
</tr>
<tr>
<td>Yes and you are a contributor of the new work</td>
<td></td>
</tr>
<tr>
<td>Author of this Springer article</td>
<td></td>
</tr>
<tr>
<td>Order reference number</td>
<td></td>
</tr>
<tr>
<td>Title of your thesis / dissertation</td>
<td>Estimation of Human Pose Categories and Physical Object Properties from Motion Trajectories</td>
</tr>
<tr>
<td>Expected completion date</td>
<td>Jun 2017</td>
</tr>
<tr>
<td>Estimated size(pages)</td>
<td>88</td>
</tr>
<tr>
<td>Requestor Location</td>
<td>Mona Fathollahi</td>
</tr>
</tbody>
</table>
ABOUT THE AUTHOR

Mona Fathollahi received her Bachelor of Science in Electrical Engineering from Sharif University of Technology, Iran in 2006. She completed her Master of Science in Computer Science from University of South Florida in 2012. She is currently pursuing her PhD at University of South Florida in Computer Science and Engineering Department. Her research interests include machine learning/computer vision with a focus on Deep Learning methods.