University of California

Los Angeles

# Generating Human Images and Ground Truth using Computer Graphics

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Statistics

by

**Weichao Qiu**

2016

## Abstract of the Thesis

# Generating Human Images and Ground Truth using Computer Graphics

by

**Weichao Qiu**

Master of Science in Statistics

University of California, Los Angeles, 2016

Professor Alan Loddon Yuille, Chair

How to provide high quality data for computer vision is a challenge. Researchers spent a lot of effort creating image datasets with more images and more detailed annotation. Computer graphics (CG) is a way of creating synthetic images, during the image synthesis many types of information of the CG scene can be exported as ground truth annotation. In this paper, we develop a pipeline to synthesize realistic human images and automatically generate detailed annotation at the same time. We use 2D annotation to control the pose of the CG human model, which enables our images to contain more poses than motion capture based method. The synthetic images are used to train and evaluate human pose estimation algorithm to show its usefulness.

The thesis of Weichao Qiu is approved.

Yingnian Wu

Hongjing Lu

Alan Loddon Yuille, Committee Chair

University of California, Los Angeles

2016

*To my parents*

*who provide courage and support for me to pursue my dream*

*To my grandpa*

*who sparked my interest in science*

# Table of Contents

# List of Figures

# List of Tables

# Acknowledgments

First, I would like to thank my advisor Prof. Alan Yuille. He is very supportive for this project and provided continuous support to make the idea come true. He also spent his precious time giving detailed feedback to the draft of this thesis.

My sincere thanks also goes to my lab-mates, Peng Wang and Xianjie Chen. The idea of this project came from the brainstorm between us. They also provided guidance and support when the project got stuck.

I also want to thank the mentors in my research career. They are Prof. Zhuowen Tu, Prof. Xiang Bai, Prof. Hongjing Lu, Prof. Xinggang Wang and Prof. Jiayi Ma. I learnt a lot by working with them.

Last but not least, I want to thank Prof. Yingnian Wu, Chunyu Wang, Junhua Mao, Lingxi Xie and Jiamei Shuai for helpful discussion and suggestion.

# CHAPTER 1

# Introduction

Computer vision benefits a lot from the development of datasets by using them to train and benchmark algorithms. Thus, researchers are energetic at creating better image datasets. Datasets are becoming larger and include more detailed annotation. But creating a large dataset with detailed annotation is expensive. Computer graphics provides an alternative approach to solve this problem. Using synthetic images to build datasets has many benefits and can be a good complement to traditional approach. In this paper we built a pipeline to synthesize human images with detailed annotation. In addition to using motion capture data, we propose a novel way to use 2D joint annotation to control the human pose in synthetic images. This technique enables us to create many variable poses.

Researchers have spent a lot of effort creating better datasets [6, 7, 8]. The number of images is growing bigger. Making dataset larger can cover more cases in the real world, which is helpful to avoid dataset bias [9] and overfitting. More images also enable us to train models with a higher capacity [10]. Datasets are also becoming more detailed [11, 12, 13, 14, 15, 16, 17]. At the beginning, objects in images were only annotated by bounding box. But now, people are more interested in finding the boundary of an object and its semantic parts. Richer annotation allows algorithms to extract more information from an image and do fine grain tasks. Building an algorithm to fully understand an image and pass the Visual Turing Test [18] is the Holy Grail for computer vision researchers.

Many techniques and tools were developed to make annotation easier and faster. Crowd-sourcing services such as Amazon Turk make annotation much cheaper. But annotating a lot of images in detail is still hard and expensive. By contrast, computer graphics is good at creating large and detailed datasets. After defining the 3D scene, rendering a new image charges no extra cost. Many types of annotation such as depth and boundary can be exported from the 3D scene directly. Information not contained in the 3D scene can be manually annotated. In short, after annotating the 3D scene, many images and their annotation can be synthesized at almost no cost.

Besides being larger and more detailed, computer graphics based methods have some advantages that traditional approach can not compete with. First, computer graphics can generate hard to annotate information of a scene, such as depth, edge and optical flow. Fig 1.1 shows two types of annotations that are hard for human labelers to label, but easy to obtain using computer graphics. Second, every variable of a scene, such as material, lighting and camera viewpoint, can be controlled. This enables us to generate images with different configurations to perform diagnosis of a model [19, 20]. Third, instead of providing a fixed set of images, computer graphics based datasets can provide data according to the needs of an algorithm. This enables us to start training the algorithm from simple and clean images and then move to harder cases by adding occlusion and motion blur. This also enables the learning algorithm to interact with the synthesize system, which is similar to how we human learn. We human can explore the world by freely moving our head to find interesting or hard examples for learning.

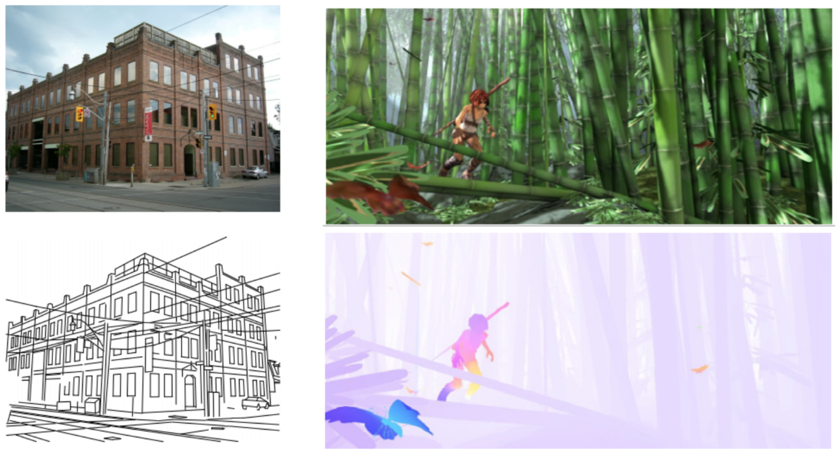

(a) (b)

Figure 1.1: Two type of information that are difficult to annotate (a) Line segments in an image, from an unpublished work (b) Optical flow in a temporal sequence, from [1]

Training computer vision systems in a virtual world is not a new idea and can date back to the beginning of this field, e.g., Blocks World [21]. But the computer graphics based approach was not popular for a long time for various reasons. It was hard to render realistic images, expensive to create a virtual environment and slow to render large number of images. The recent advances of the CG industry make synthetic images much better. In particular, fast GPUs make it possible to use physically realistic rendering at a large scale. More high quality models, open source movies and games make it easier to generate synthetic images.

Recently, the authors of [22, 23] used computer graphics models from 3D warehouse to synthesize images for training state-of-the-art object detector. They mainly focused on rendering rigid objects. In this work, we study how to create synthetic human images with detailed annotations. Annotating images with different types of information can enable us to train a complex model to work on multiple tasks. Different types of information in a system can be shared. For example, after estimating the joint location of a human, it is easier to estimate the semantic part segmentation [24]. We also develop a novel method to control CG human model with 2D joint annotation. In Fig. 1.2, we give an example of a synthetic image and its corresponding annotation, including depth, joint location and semantic parts. The pose of the human model is obtained from the annotation of a real image.

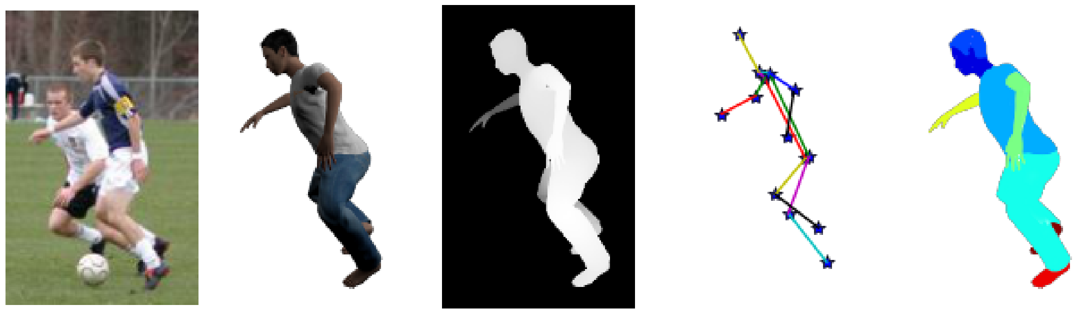


Figure 1.2: Generated ground truth during image synthesis. From left to right are real image, synthetic image, depth, joint location, semantic part annotation. The pose of the human model is obtained from the annotation of the real image.

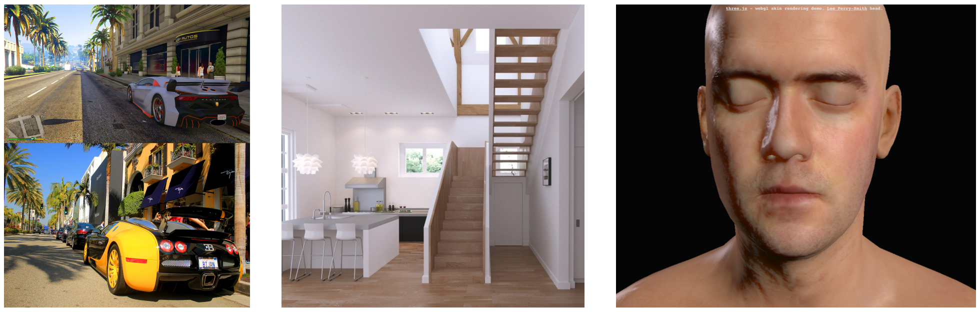
This paper is organized as follow. In Chapter 2, we describe the design of our image synthesis pipeline. We mainly focus on how to use 2D joint location to control the CG human model, which can provide more poses than motion capture based method. In Chapter 3, we report the performance of two pose estimation algorithms trained on our synthetic dataset. And discuss the transfer issue between real and synthetic datasets.

# CHAPTER 2

# Synthesize Human Images

The goal of computer graphics is synthesizing realistic images. The Movie, Game and VR industries spend lots of money to improve CG techniques and have made huge progress. Today CG can produce very realistic synthetic images sometimes hard to distinguish from photographs. Fig. 2.1 shows a few examples of CG generated images. If synthetic images can be realistic enough, why not use them as an alternative or complimentary approach for creating image datasets. In this chapter, we focus on creating synthetic human images with detailed annotation.

We first briefly survey previous work using synthetic images in computer vision. Then, the CG human model is described in Sec.2.2. We introduce a novel way to control CG human models using 2D body joint annotations in Sec.2.3 and how to automatically get annotation in Sec.2.4. Finally, the synthesis pipeline is concluded in Sec.2.5.



(a) (b) (c)

Figure 2.1: Realistic synthetic images. (a) The virtual scene (top) and its corresponding real photograph (bottom) in the game GTA5, (b) An indoor scene created by a virtual artist using Blender, (c) Digital human avatar

## 2.1 Related Work

Using synthetic images to train an intelligent system dates back to the early days of artificial intelligence. Researchers tried to teach a machine system in a simple virtual world, such as Blocks World [21]. But at that time, the simulated virtual world was too simple. Even if the algorithm can perform well in the toy world, it is not practical in the real world.

Synthetic images are getting much better today. Many researchers tried to utilize the rich information that computer graphics can provide. In [1], an open source CG movie was used to create a large optical flow dataset. In [22, 23], synthetic images are used to train object detectors and determine object pose. In [25], depth information of an object is exported to train a depth estimation system. In [26], the autonomous driving system is purely trained with images from a video game.

Non-rigid objects such as human are more challenging to synthesize. It is harder to make sure the deformed object still looks valid. In [27], the author synthesized a large number of images and for an input image, the algorithm retrieves the most similar image and its pose from the dataset. In [28], the authors use synthetic depth images to train pose estimation system.

Researchers have already spent a lot of effort creating human images datasets. Our synthesis pipeline can provide more images and detailed annotation. Synthetic images can be a compliment for existing human image dataset. We show a comparison between our pipeline and some existing datasets in Table. 2.1. It is worth noticing that in the Human3.6M dataset[29], data capture is performed in a studio, subjects need to wear sensors and special equipment is needed to capture data.

| Name | Type of images | No. of images | Ground truth |
|---|---|---|---|
| LSP[3] | sports | 2000 | joint location (2D) |
| MPII[30] | diverse | 40522 | joint location (2D) |
| Fashionista[31] | fashion | 685 | joint location (2D), semantic part |
| Human3.6M[29] | indoor | 3.6M | joint location (2D/3D), semantic part, depth |
| Ours | synthetic | unlimited | joint location (2D/3D), semantic part, depth |

Table 2.1: Comparison of human image dataset

## 2.2 Computer Graphics Human Model

Designing a realistic human model from scratch is not easy. A static body mesh can be designed by a virtual artist from scratch or reconstructed from depth scanning. But how to make a static body mesh deformable is more challenging. One of the challenges is how to make the model still looks realistic under different poses. Another is how to make the motion of clothes and hair realistic and follow physical rules. Creating realistic digital humans is still an active research area for computer graphics researchers. The Movie and Game industries have spent a lot of money building better human models and the most advanced models are proprietary.

There are a few options for creating CG human. The first is to use models created by 3D modeling software. Many models have been created and uploaded by amateurs and artists. These models can be downloaded from user forums or marketplaces, such as 3D Warehouse, or TurboSquid. The quality of these models varies a lot. The simple models only contains mesh data without any texture. More advanced models contain skin texture photographed from real humans. The price ranges from free to a few thousand dollars. Those high quality models can be used for movie production.

The second is to use models generated by specific software designed to produce human models. This type of software can produce human models with various body parameters, such as weight, height, torso shape. The exported human models are fully rigged. Rigging means defining the skeleton and how the mesh is controlled by each bone. These models can be easily animated by generic computer graphics softwares such as Blender, Maya, or by game engines. Popular softwares include MakeHuman, Daz3D, SMPL[32]. A visual comparison of synthetic images using different models is shown in Figure 2.2. Among these options, Daz3D is the most realistic and also provide different types of wearables, such as clothes, pants. SMPL is a research project and can simulate realistic soft tissue motion, but it does not contain clothes and hair.

In this project we chose MakeHuman to provide human models. Since the MakeHuman project can create human bodies with various shape parameters and high quality fully rigged model. Another advantage of MakeHuman project is that it is open source and written with Python, so that we can modify its source code and integrate it with our rendering pipeline which is based on Blender.

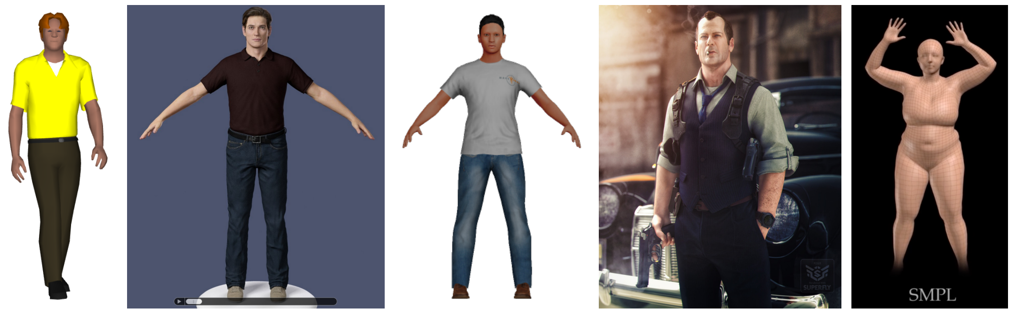


Figure 2.2: Typical images from different sources, from left to right, 3D Warehouse, TurboSquid, MakeHuman, Daz 3D, SCAPE

## 2.3 Control Pose of the Human Model

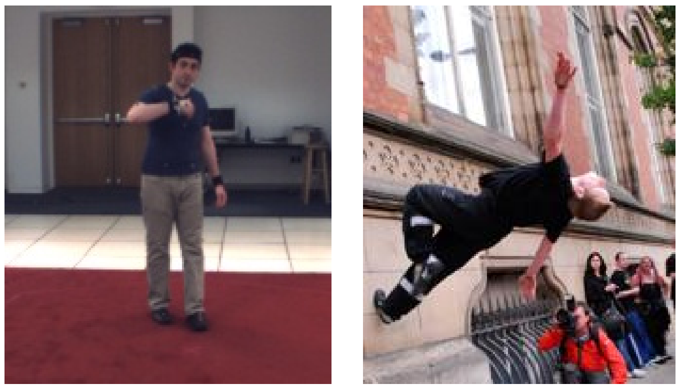
Synthetic data is less popular than real data for computer vision. One of the reasons is that synthetic data contains less variability than real. In particular, in terms of synthetic human images, the real data is richer in terms of pose, environment and appearance.

The human body is a highly flexible non-rigid object, shown in Figure 2.3 (b). This flexibility of the human body in a challenge for pose estimation. Seeing human images with different poses is helpful for the learning of a pose estimation algorithm, so it is important to making sure that the dataset contains various poses.

In our rendering pipeline, we have two approaches to control the pose of the human model.

**Pose from Motion Capture Data** The common practice to control the pose of CG human model is using motion capture data. Motion capture defines the rotation parameters for each bone of the body. It is widely used in movie and game to make the virtual character behave the same as an actor.

Motion capture data is generated by asking human subjects to wear motion capture sensors and perform certain tasks indoor, such as walking and running. The number of available action categories is limited [2, 29]. It is hard to cover as many poses as a traditional image dataset [3]. Many human poses are hard for actors to do, such as Parkour and Gymnastics. In [33], gymnasts are asked to perform some extreme poses to explore the limit of each body joint, but it is still not rich enough to capture poses in the daily life.



(a) (b)

Figure 2.3: (a) Motion capture data is recorded indoor and subjects are asked to do a limited number of tasks, image from [2] (b) In the real world, humans perform a large range of poses by interacting with the environment, image from [3]

Motion capture data is widely used. Most computer graphics software provides built-in support for animating rigged models with motion capture data. We use the MakeHuman plugin in Blender to read motion capture data.

**Estimate 3D Pose from 2D Annotation** We already have many good human image datasets with joint location annotated, such as [3, 30]. If we can make our CG human model perform the same pose as those images, then we can have numerous human poses. Moreover, we can generate new 2D poses by changing the camera viewpoint. To achieve this goal, we designed a method to use 2D annotations to control the pose of our 3D human model.

The pose of a human is defined by his joint location. The 2D joint location $\mathbf{x} \in \mathcal{R}^2$ is the projection from the 3D joint location $\mathbf{X} \in \mathcal{R}^3$. Using an orthographic camera model $\mathbf{x} = s\mathbf{R}\mathbf{X} + t$, where $\mathbf{R}$ is a projection matrix, $s$ and $t$ are the scale and translation parameters of the camera.

But the 2D annotations are not sufficient to recover the underlying 3D joint location, because this is an underdetermined problem. Extra prior knowledge about human poses can help us solve this problem. A few methods were developed to solve this problem [33, 34].

Here we use the method in [33] to recover the 3D pose. The 3D pose is modeled with a sparse representation $\mathbf{X} = \mathbf{B}\mathbf{w} + \mu$. Here $\mathbf{B}$ is an over-complete basis, $\mathbf{w}$ is a vector of coefficient, $\mu$ is the mean pose. The 3D pose is recovered by minimizing an energy function. The energy function is defined as follow:

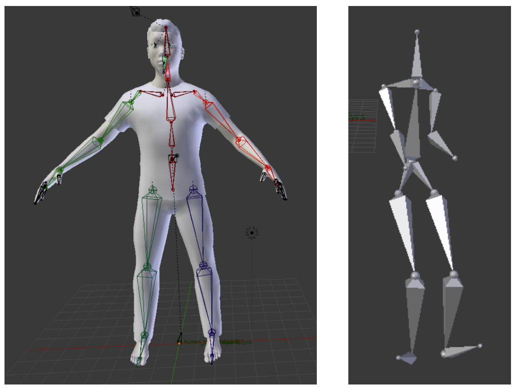
$$\mathbf{E}(\mathbf{w}, s, \mathbf{R}) = C_r + C_p + \beta C_l \tag{2.1}$$

In Eq. 2.1, $C_r(\mathbf{w}, s, \mathbf{R}) = ||x - s\mathbf{R}(\mathbf{B}\mathbf{w} + \mu)||_2^2$, this term ensures the projection of the estimated 3D joint location $\mathbf{X}$ is consistent with the 2D observation $\mathbf{x}$.

$C_l = \sum_{i=1}^{N} \left| ||\delta(\hat{\mathbf{X}}_i)||_2^2 - l_i^2 \right|$, $l_i$ is the normalized length for bone $i$ in a training set. This term ensures the bone length of the recovered pose $\delta(\hat{\mathbf{X}})$ is consistent with the training data.

$C_p$ use a function learnt from the training set to determine whether the estimated pose is valid. The bones of human body can not rotate arbitrary. $C_p$ is 0 if all the bones of $\mathbf{X}$ are valid, and inf if one of them is in an invalid configuration. After solving $\mathbf{R}, \mathbf{B}, \mathbf{w}$, the 3D joint location can be recovered using $\mathbf{X} = \mathbf{B}\mathbf{w} + \mu$. More details about this method can be found in [33].

**Inverse Kinematics (IK) to Control Human Body** The number of bones in our human model is more than the amount of data we have. From the 2D joint locations we can recover the 3D joint location, but the number of parameters is still not enough. There are more bones in our human model than the recovered bone configurations, see Fig. 2.4. The human model contains more bones in the spine and shoulder.



(a) (b)

Figure 2.4: (a) The bone structure of a MakeHuman model (b) the 3D joint information recovered from 2D annotation. The human model contains more bones in the shoulder and spine

This is a typical problem in robotics and graphics. Given the location of a target, we want to solve for the configuration of the whole robotic arm to reach the target. The solution is not unique, but with a sensible prior, we can get a good solution. We use the inverse kinematics (IK) solver in Blender to solve the IK constraints.

The motion capture dataset only contains indoor motion performed by a few human subjects. The ability of using existing 2D joint annotation hugely enriches the poses in our synthetic dataset. By changing the camera viewpoint and making small perturbations, we can have more poses than the 2D annotations can provide. Fig. 2.5 shows the original real image, a synthetic image with a similar pose and new poses by changing the camera viewpoint.

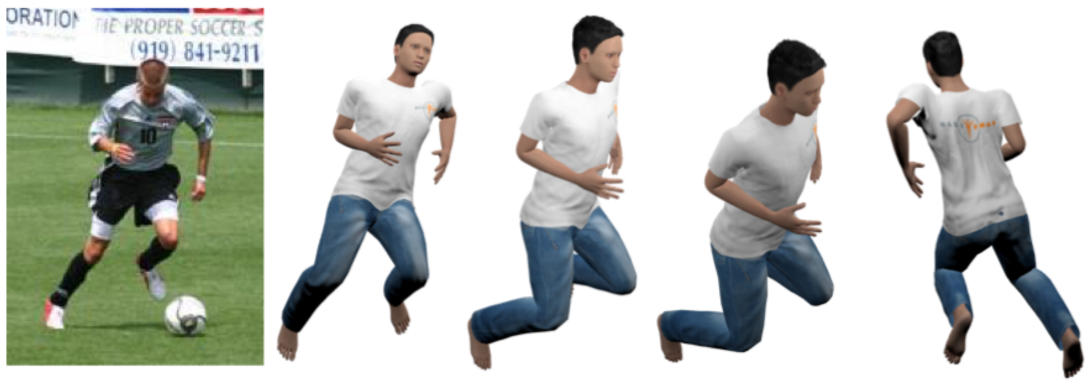


Figure 2.5: The corresponding real image and synthetic images based on the estimated 3D pose, under different camera viewpoints

## 2.4 Automatic Ground Truth Generation

In this project, we want to synthesize realistic human images, while also generating the corresponding annotations automatically at the same time. More complex visual tasks require more detailed annotation for training and evaluation. But it is expensive to annotate all the detail in the scene. With the help of computer graphics, we can get very rich annotation of the scene for almost free.

The body joint annotation is exported as the endpoint location of a bone in the human model, shown in Fig. 2.4 (a). The bone location is in 3D, which is useful for 3D human pose estimation. It can be projected to the camera space to get the 2D joint location corresponding to the synthetic image.

Semantic part annotation is useful for fine-grain tasks [24, 35]. Labelers are required to annotate the regions of semantic body part, such as head, arm and torso. The time required for each image is the same. This will be a large number if we trying to create a large dataset. Also the annotation quality for each image is inconsistent. Instead we label the computer graphics human model, then this labeled model can be used to generate a nearly infinite number of images without the need of labeling it again. This is a huge gain for creating large scale dataset. The semantic part of the body is not defined in the CG human model. To annotate the semantic parts, we assign color to each mesh. This can done by using the paint brush provided by Blender. In our pipeline, we use the part definition and color scheme defined in [35].

Depth ground truth can be generated by exporting the z-buffer of the scene. The boundaries of human can be computed from the depth. Line segment and occlusion information can also be generated automatically by modifying the rendering engine, but are beyond the discussion of this paper. The generated ground truth is shown in Fig. 1.2.

## 2.5 Image Synthesis Pipeline

Synthetic images which only contain humans are not very interesting. The boundaries are too sharp and the human foreground can be easily computed. So we randomly pick a background image and use alpha blending to combine the foreground and background. Synthesizing human in more realistic virtual scenes is a future work. The scene is lit with 16 point lights surrounding the human. The intensity of each light is randomly sampled from a Gaussian distribution. The colors of skin, hair and clothes are randomly sampled from a set of colors. We use Blender's internal rendering engine to render images. The source code will be publicly available[1].

The pipeline of the synthesis is as follow. First, the human mesh model is generated using MakeHuman. Then, the semantic body part is annotated in Blender. Next, 2D joint annotation or motion capture data is used to control the pose of the human model. Images are rendered with different rendering modes to

[1]www.bitbucket.org/qiuwch/tenon

produce different types of annotation. The background is added by alpha blending to the synthetic image. The synthesize pipeline can be seen in Fig. 2.6.

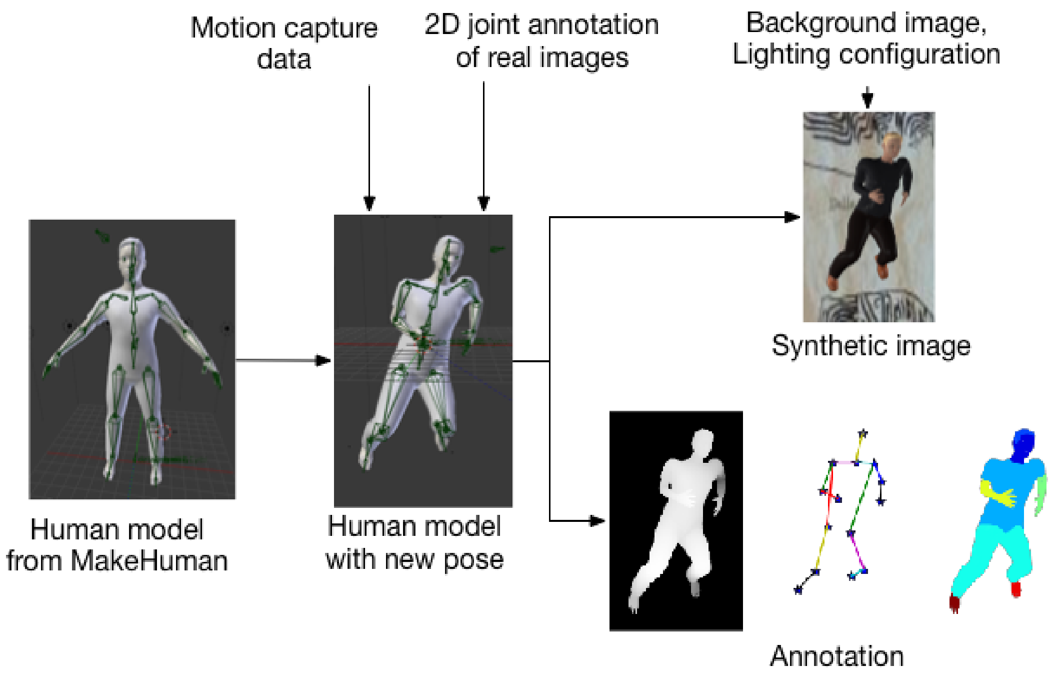

Motion capture data
2D joint annotation of real images
Background image, Lighting configuration
Synthetic image
Human model from MakeHuman
Human model with new pose
Annotation

Figure 2.6: Illustration of the image synthesis pipeline

# CHAPTER 3

# 2D Human Pose Estimation

In this section, we use our synthetic images to train pose estimation algorithms. We take popular pose estimation algorithms and train them with our synthetic data to see whether they can work on real images. This chapter is organized as follows: First, the pose estimation algorithms are briefly overviewed. Then a comparison between training using synthetic images and real images is shown. We also discuss issues, such as transfer between real and synthetic datasets. We then give some diagnostic result.

## 3.1 Model Review

The pose of a human is defined by the joint location $l_i = (x, y)$. 2D pose estimation tries to estimate the joint location $l_i$ from an image $I$. The first model we use is [5]. It uses a deformable part model, where each part is represented by a mixture of HOG feature templates. Parts are connected in a tree structure and the model is trained by a structure SVM. During inference, it first computes the unary response for each part, then use dynamic programming to combine the score of each part together to find the optimal solution. The energy function for the model is below:

$$S(I, l, t) = S(t) + \sum_{i \in V} w_i^{t_i} \phi(I, l_i) + \sum_{i,j \in E} w_{ij}^{t_i t_j} \psi(l_i - l_j) \tag{3.1}$$

$$S(t) = \sum_{i \in V} b_i^{t_i} + \sum_{ij \in E} b_{ij}^{t_i t_j} \tag{3.2}$$

In Eq. 3.1, $t$ is the type for each joint. The first term $S(t)$ computes the compatibility when assigning type $t_i$ for joint $i$. $\phi(I, l_i)$ is the HOG feature vector extracted at location $l_i$. The second term $w_i^{t_i}\phi(I, l_i)$ computes the unary score. $\psi(l_i - l_j) = [dx, dx^2, dy, dy^2]$. $\psi(l_i - l_j)$ measures the distance between joint $i$ and $j$. The third term $w_{ij}^{t_i t_j}\psi(l_i - l_j)$ computes the penalty for the displacement of joints.

During training, $w_i^{t_i}$, $w_{ij}^{t_i t_j}$ and $b_i^{t_i}$, $b_{ij}^{t_i t_j}$ are learnt using a structure SVM solver described in [36]. During inference, the $S(I, l, t)$ in Eq. 3.1 is maximized over $l$ and $t$ with dynamic programming.

The second method we tried is [4]. In this method, the HOG feature is replaced by convolutional neural network [10], which can produce more accurate part localization. The author also proposed a novel IDPR (image dependent pairwise relations) term to measure the consistency between each part of the model.

The details for these two models can be found in [5, 4].

## 3.2 Experiment Setup

In the experiment, the performance is measured by PCKh. PCK (Probability of Correct Keypoint) is initially defined in [5]. In PCK, a prediction is considered correct if it falls within a threshold $\alpha \cdot \max(h, w)$. Here $h$ and $w$ are the height and width of the human bounding box, $\alpha$ is a parameter to control the percentage of the size. In PCKh, the threshold is changed to 50% of the head segment length. This change makes PCKh more robust to the articulation of the human body than PCK.

The real image dataset we used is the Leeds Sport Dataset (LSP) [3]. The synthetic images are generated using the pose estimated from the ground truth annotation of LSP dataset. For each image in the LSP dataset, we generate one image with a similar pose. That is 2000 synthetic images, 1000 images for training and 1000 for testing. We call this synthetic dataset the L23 dataset.

We show a few examples of the L23 dataset and their corresponding real images in Fig. 3.1.

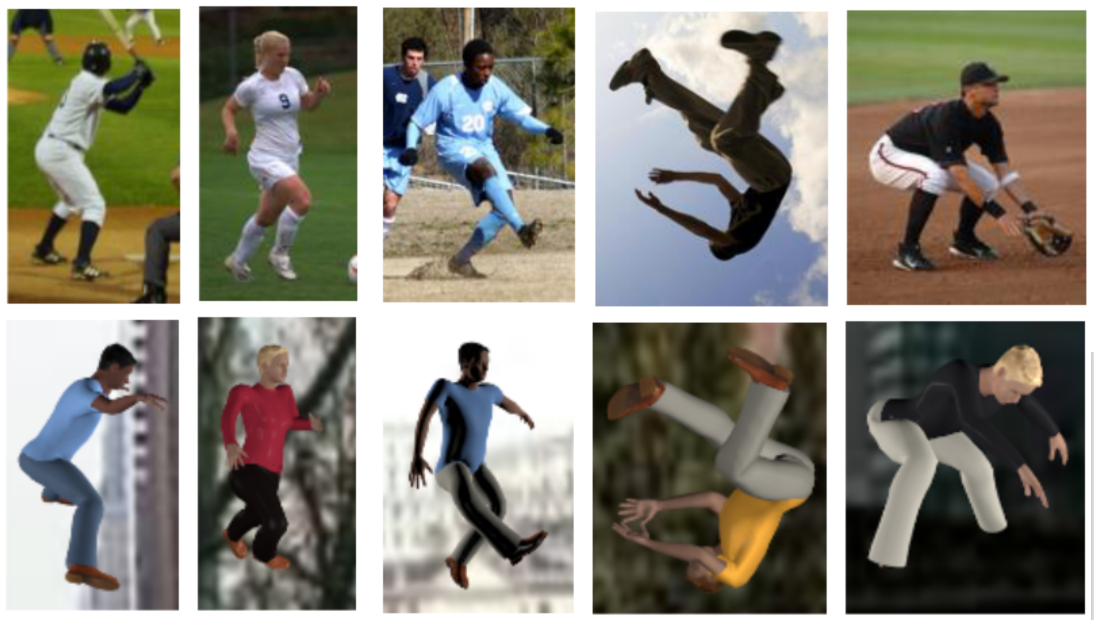


Figure 3.1: Visual comparison between the real LSP and the synthetic L23 dataset

## 3.3 Transfer Between Dataset

We split the L23 dataset into a training set and a test set following the split of the LSP dataset. The methods in [4, 5] are used to test whether the model trained on the synthetic data can also work on the real test set.

We first describe the result by using the method in [5]. We use this method to train two models with different training sets, the real and the synthetic training set. These two models are tested on real test set. Table. 3.1 shows the performance of the two models on the LSP test set. From the results we observe that the model trained on synthetic images performs worse than the one trained on real images.

But it is worth noticing that the synthetic data is generated almost for free.

More images can be synthesized by changing lighting, body shape, textures and camera viewpoint. The synthetic images can be considered as a compliment for real images. Also our synthetic L23 dataset contains richer annotation than the LSP dataset making it suitable for many visual tasks.

| Training data | Head | Shou | Elbo | Wris | Hip | Knee | Ankle | Mean |
|---|---|---|---|---|---|---|---|---|
| L23 | 40.3 | 38.6 | 30.6 | 42.9 | 37.6 | 43.7 | 53.5 | 41.0 |
| LSP | 57.2 | 60.4 | 46.2 | 65.5 | 62.2 | 65.8 | 80.0 | 62.3 |

Table 3.1: Test performance on the real LSP dataset, The two models are based on [5] and trained on synthetic images and real images respectively

We also tested the two models on the synthetic test set. The result is shown in Table. 3.2. It is not surprising to see that the model trained on the synthetic training set can work better on the synthetic test set. But the margin between these two models is smaller than shown in Table. 3.1, which might indicate that the appearance in the real images is more varied, so that the trained model is easier to generalize.

| Training data | Head | Shou | Elbo | Wris | Hip | Knee | Ankle | Mean |
|---|---|---|---|---|---|---|---|---|
| L23 | 50.7 | 50.9 | 41.0 | 56.3 | 51.5 | 61.2 | 74.6 | 55.2 |
| LSP | 44.0 | 42.0 | 33.4 | 42.8 | 44.2 | 54.2 | 66.6 | 47.4 |

Table 3.2: Test performance on the synthetic L23 dataset, The two models are based on [5] and trained on synthetic images and real images respectively

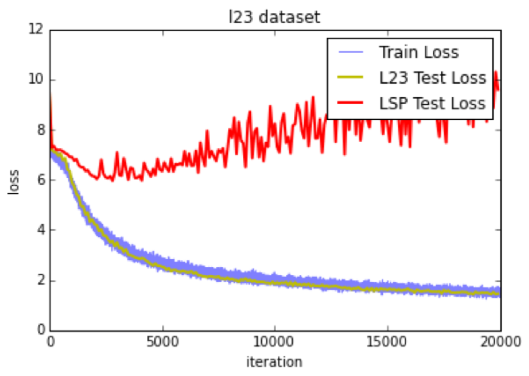
We also trained the method [4] with our synthetic data. We observed the model trained on synthetic images can barely work at all on the real data, but the model trained on real data can work well on synthetic data. We conjecture this is because the CNN model has a much higher learning capacity compared with the method based on HOG feature. HOG feature relies on edge cues, which are fairly similar between real and synthetic images. The higher capacity of CNN makes the algorithm easier to overfit to some specific features that are unique to synthetic data but not present in the real images. We did some diagnostic experiments to show the overfit issue.

## 3.4 Diagnostic Result for Appearance Transfer

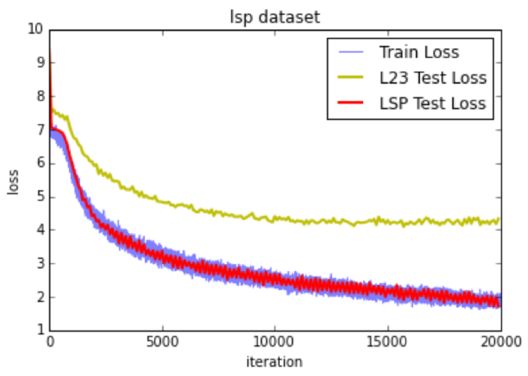
In [4], the authors first use a CNN to produce a unary score and IDPR score for each body part. The unary score is then combined in a tree structure using dynamic programming. Since the poses between the real and synthetic data are similar, we conjecture the problem is the unary scores produced by the CNN model. So we separate the CNN module from the algorithm in [4] and plot the train and test loss when training on synthetic and real images.

In Fig 3.2, we can clearly observe that the CNN trained on synthetic data overfit. The train and test loss on synthetic images keeps going down, but the test loss on the real images go up. But training on real data can still generalize to synthetic data.

This poses a challenge to design a better learning algorithm or better training strategy to be able to transfer between dataset, such as the method proposed in [37, 38]. Or make the synthetic dataset richer in appearance to overcome this issue. This remains as a challenge and we will address it in a future work.



(a)



(b)

Figure 3.2: The loss for training unary CNN in [4]. (a) Train on synthetic data (b) Train on real data

# CHAPTER 4

# Conclusion

In this paper, we built a pipeline to synthesize human images using computer graphics. Besides using motion capture data, we use the 2D annotation of images to control the pose of our CG model. This enables us to create a dataset with richer poses. Our pipeline also enables us to automatically export ground truth during image synthesis. Pose estimation algorithms were trained with the synthetic images to show the usefulness of the data.

**Limitations** Although the image synthesis technique has many benefits for creating dataset, it still has many limitations. Creating a realistic scene is time consuming. How to do physically real simulation is challenging.

For human synthesis, it is hard to simulate the hair and clothes. Realistic soft tissue simulation is still an active research topic in computer graphics [32]. The types of clothes are not as varied as those we observe in real images.

In our synthesis pipeline, the 3D pose estimated from 2D is not perfect. Sometimes it will produce unreasonable poses. The human model we use is not the most realistic and can be improved.

**Future Work** The synthetic quality of the images can be improved by using better computer graphics model. More realistic scene can be created by pasting the human model into a well designed scene [1], or pasting the human model into an image with estimated lighting and 3D structure [39].

It is also worthwhile to develop better model that can generalize well from synthetic images to real images.

## References

[1] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A Naturalistic Open Source Movie for Optical Flow Evaluation.," *ECCV*, vol. 7577, no. Chapter 44, pp. 611–625, 2012.

[2] L. Sigal, A. O. Balan, and M. J. Black, "HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion," *International journal of computer vision*, vol. 87, no. 1-2, pp. 4–27, 2009.

[3] S. Johnson and M. Everingham, "Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation.," *BMVC*, pp. 1–11, 2010.

[4] X. Chen and A. L. Yuille, "Articulated Pose Estimation by a Graphical Model with Image Dependent Pairwise Relations.," *NIPS*, pp. 1736–1744, 2014.

[5] Y. Yang and D. Ramanan, "Articulated Human Detection with Flexible Mixtures of Parts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 12, pp. 2878–2890, 2013.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2009.

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

[8] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context.," *ECCV*, vol. 8693, no. Chapter 48, pp. 740–755, 2014.

[9] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1521–1528, 2011.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks.," *NIPS*, pp. 1106–1114, 2012.

[11] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille, "Detect What You Can: Detecting and Representing Objects Using Holistic Models and Body Parts," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1979–1986, IEEE, 2014.

[12] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. Yuille, "Joint Object and Part Segmentation Using Deep Learned Potentials," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1573–1581, IEEE, 2015.

[13] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The Role of Context for Object Detection and Semantic Segmentation in the Wild," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 891–898, IEEE, 2014.

[14] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille, "The Secrets of Salient Object Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 280–287, IEEE, 2014.

[15] P. Wang and A. L. Yuille, "DOC: Deep OCclusion Recovering From A Single Image.," *CoRR abs/1511.06457*, 2015.

[16] J. Zhu, X. Chen, and A. L. Yuille, "DeePM: A Deep Part-Based Model for Object Detection and Semantic Part Localization," *arXiv.org*, Nov. 2015.

[17] J. Wang and A. Yuille, "Semantic part segmentation using compositional model combining shape and appearance," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1788–1797, IEEE, 2015.

[18] D. Geman, S. Geman, N. Hallonquist, and L. Younes, "Visual Turing test for computer vision systems," *Proceedings of the National Academy of Sciences*, vol. 112, pp. 3618–3623, Mar. 2015.

[19] A. Borji, S. Izadi, and L. Itti, "What can we learn about CNNs from a large scale controlled object dataset?," *arXiv.org*, Dec. 2015.

[20] J. Z. Leibo, Q. Liao, and T. A. Poggio, "Subtasks of Unconstrained Face Recognition.," *VISAPP*, pp. 113–121, 2014.

[21] L. G. Roberts, "Machine Perception of Three-Dimensional Solids," *Garland Publishing, New York 1963*, 1963.

[22] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views.," *ICCV*, pp. 2686–2694, 2015.

[23] X. Peng, B. Sun, K. Ali, and K. Saenko, "Learning Deep Object Detectors from 3D Models," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1278–1286, IEEE, 2015.

[24] F. Xia, J. Zhu, P. Wang, and A. L. Yuille, "Pose-Guided Human Parsing with Deep Learned Features.," *CoRR abs/1508.03881*, 2015.

[25] S. Song and J. Xiao, "Sliding Shapes for 3D Object Detection in Depth Images.," *ECCV*, vol. 8694, no. Chapter 41, pp. 634–651, 2014.

[26] C. Chen, A. Seff, A. L. Kornhauser, and J. Xiao, "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving.," *ICCV*, pp. 2722–2730, 2015.

[27] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter-sensitive hashing," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 750–757 vol.2, IEEE, 2003.

[28] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1297–1304, IEEE, 2011.

[29] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 7, pp. 1325–1339, 2014.

[30] M. Andriluka, L. Pishchulin, and P. Gehler, "2D Human Pose Estimation: New Benchmark and State of the Art Analysis," *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3686–3693, 2014.

[31] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg, "Parsing clothing in fashion photographs," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 3570–3577, IEEE, 2012.

[32] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: a skinned multi-person linear model," *ACM Transactions on Graphics (TOG)*, vol. 34, pp. 248–16, Nov. 2015.

[33] I. Akhter and M. J. Black, "Pose-conditioned joint angle limits for 3D human pose reconstruction," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1446–1455, IEEE, 2015.

[34] C. Wang, J. Flynn, Y. Wang, and A. L. Yuille, "Representing Data by a Mixture of Activated Simplices," *arXiv.org*, Dec. 2014.

[35] Y. Bo and C. C. Fowlkes, "Shape-based pedestrian parsing," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 2265–2272, IEEE, 2011.

[36] D. Ramanan, "Dual coordinate solvers for large-scale structural SVMs," *arXiv.org*, Dec. 2013.

[37] B. Sun, J. Feng, and K. Saenko, “Return of Frustratingly Easy Domain Adaptation,” *arXiv.org*, Nov. 2015.

[38] E. Tzeng, C. Devin, J. Hoffman, C. Finn, X. Peng, S. Levine, K. Saenko, and T. Darrell, “Towards Adapting Deep Visuomotor Representations from Simulated to Real Environments,” *arXiv.org*, Nov. 2015.

[39] K. Karsch, K. Sunkavalli, S. Hadap, N. Carr, H. Jin, R. Fonte, M. Sittig, and D. Forsyth, “Automatic Scene Inference for 3D Object Compositing,” *ACM Transactions on Graphics (TOG)*, vol. 33, pp. 32–15, May 2014.