Real-Time Human Motion Capture with Multiple Depth Cameras

Alireza Shafaei, James J. Little
Computer Science Department
The University of British Columbia
Vancouver, Canada
shafaei@cs.ubc.ca, little@cs.ubc.ca

***Abstract*—Commonly used human motion capture systems require intrusive attachment of markers that are visually tracked with multiple cameras. In this work we present an efficient and inexpensive solution to markerless motion capture using only a few Kinect sensors. Unlike the previous work on 3d pose estimation using a single depth camera, we relax constraints on the camera location and do not assume a co-operative user. We apply recent image segmentation techniques to depth images and use curriculum learning to train our system on purely synthetic data. Our method accurately localizes body parts without requiring an explicit shape model. The body joint locations are then recovered by combining evidence from multiple views in real-time. We also introduce a dataset of ~6 million synthetic depth frames for pose estimation from multiple cameras and exceed state-of-the-art results on the Berkeley MHAD dataset.**

***Keywords*-depth sensors; human motion capture;**

I. Introduction

Human motion capture is a process to localize and track the 3d location of body joints. It is used to acquire a precise description of human motion which could be used for a variety of tasks such as character animation, sports analysis, smart homes, human computer interaction, and health care.

In this work we are interested in extraction of 3d joint locations from multiple depth cameras. Unlike previous work we do not impose contextual assumptions for limited scenarios such as home entertainment. Instead, we focus on the problem of pose estimation to get comparable results to commercial motion capture systems. Our target is to have a real-time, inexpensive, and non-intrusive solution to the general human motion capture problem.

While single view pose estimation has been extensively studied [1]–[4], surprisingly, pose estimation with multiple depth sensors is relatively unexplored. One possible explanation is the challenge of overcoming the interference of multiple structured light sources. However, with the recent adoption of time-of-flight sensors in the Kinect devices the interference has become unnoticeable.

One of the main challenges in this field is the absence of datasets for training or even a standard benchmark for evaluation. At the time of writing the only dataset that provides depth from more than one viewpoint is the Berkeley Multimodal Human Action Database (MHAD) [5] with only two synchronized Kinect readings.

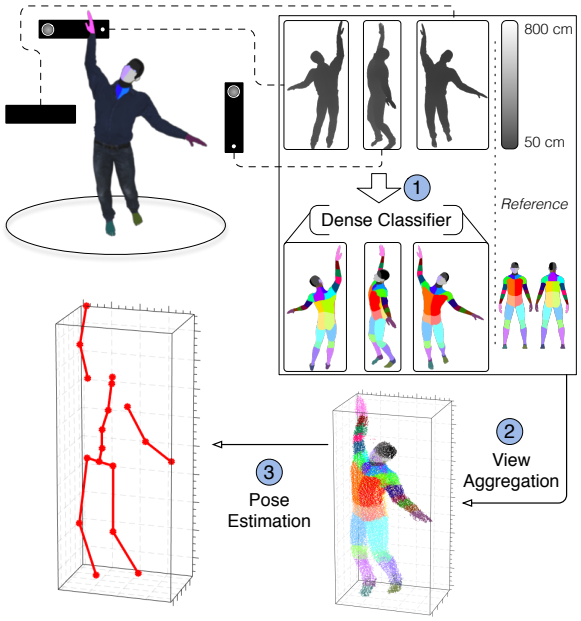


Figure 1: An overview of the problem and our pipeline. The output depth of the Kinect sensors is first passed through a dense classifier to identify the target body parts (1). We then merge all the cameras into a single 3d point cloud (2) and estimate the posture (3).

Our first contribution is a markerless human motion capture system. Our pipeline uses an intermediate representation inspired by the work of Shotton *et al.* [2]. We split the multiview pose estimation task into three subproblems of (i) dense classification, (ii) view aggregation, and (iii) pose estimation (See Fig. 1). Our approach can be distinguished from the previous work in several aspects. The context of our problem is *non-intrusive* and *real-time*. We do not assume *co-operation*, a *shape model* or a specific application such as home entertainment. As a result our method is applicable to a wider range of scenarios. We only assume availability of *multiple* externally calibrated depth cameras.

Our second contribution is three datasets of synthetic depth data for training multiview or single-view pose estimation systems. Our three datasets contain varying complexity

of shape and pose with a total of 6 million frames. These datasets are constructed in a way that enables an effective application of curriculum learning [6]. We show that a system trained on our synthetic data alone is able to generalize to real-world depth images. We also release an open-source system for synthetic data generation to encourage and simplify further applications of computer generated imagery for interested researchers[1].

II. Related Work

The previous work on depth-based pose estimation can be categorized into two classes of top-down generative and bottom-up discriminative methods.

At the core of the generative methods there is a parametric shape model of the human body. The typical approach is then to find the set of parameters that best describes the evidence. Approaches to find the best parameters vary, but the common theme is to frame it as an expensive optimization problem [1], [3], [7]–[9]. Since a successful application of generative methods require reasonably accurate shape estimates, it is common practice to estimate the shape parameters beforehand [3], [8]. This dependence on the shape estimates limits the applicability of these methods to controlled environments with co-operative users.

Bottom-up discriminative models directly focus on the current input, usually down to pixel level classification to identify individual body parts. For instance, Shotton *et al*. [2] present a fast pipeline for single view pose estimation within a home entertainment context. Their method classifies each pixel of a depth image independently using randomized decision forests. The joint information is inferred by running a mean-shift algorithm on the classification output. Girshick *et al*. [10] improve the body estimates by learning a random forest to predict joint locations as a regression problem on the classification outputs directly. It should be noted that neither the *data* nor the *implementation* of these methods are available for independent evaluation. The popular Microsoft Kinect SDK only feeds data from the sensor directly. More recently, Yub Jung *et al*. [4] presented a random-walk based pose estimation method that outperforms the previous state-of-the-art on a single view while performing at more than $1000\,\mathrm{fps}$.

In contrast, our solution is a middle ground between the discriminative and the generative approaches. Although our approach does not require a parametric shape model, we integrate high-level spatial relationships of the human body during dense classification. Specifically, by using convolutional networks one can even explicitly take advantage of CRF-like inference during classification to achieve smooth and consistent output [11].

Multiview Depth. Many of the existing methods for single view pose estimation, especially the top-down methods, can be naturally extended for the multiview case. Michel *et al*. [12] show that through calibration of the depth cameras one can reconstruct a detailed 3d point cloud of target subjects to do offline pose estimation by applying particle swarm optimization methods over the body parameters of a puppet. The shape parameter of each subject is manually tuned before the experiments. Phan and Ferrie [13] use optical flow in the RGB domain together with depth information to perform multiview pose estimation within a human-robot collaboration context at the rate of $8\,\mathrm{fps}$. They report a median joint prediction error of approximately $15\,\mathrm{cm}$ on a T-pose sequence. Zhang *et al*. [9] combine the depth data with wearable pressure sensors to estimate shape and track human subjects at $6\,\mathrm{fps}$. In this work we demonstrate how the traditional building blocks can be effectively merged to achieve the state-of-the-art performance in multiview depth-based pose estimation.

Dense Image Classification. Most current competitive approaches make use of deep convolutional networks [11], [14]–[17]. Long *et al*. [16] introduced the notion of fully convolutional networks. Their method applies deconvolution layers fused with the output of the lower layers in the network to generate fine densely classified output. Zheng *et al*. [11] show that it is also possible to integrate the mean-field approximation on CRFs with Gaussian pairwise potentials as part of a deep convolutional architecture. This approach enables end-to-end training of the entire system to get state-of-the-art accuracy in image segmentation. The takeaway message is that under certain assumptions and modelling constraints the machinery of convolutional networks can be interpreted as reasoning within the spatial domain on dense classification tasks. This proves useful when there are long distance dependencies in the spatial domain, such as the case when it is necessary to infer the side of the body to produce correct class labels, something that can not be decided locally.

Synthetic Data. Shotton *et al*. [2] augment the training data with synthetic depth images for training and evaluation of their method. They also note that synthetic data in this case can be even more challenging than real data [2]. Park and Ramanan [18] synthesize image frames of video to improve hand pose estimation. Rogez *et al*. [19] use depth synthesis to estimate hands' pose in an egocenteric camera setting. Gupta *et al*. [20], [21] use synthetically generated trajectory features to improve cross-view action recognition by feature augmentation. Since we take a fully supervised approach, dense labels are required to train the part classifier. The use of synthetic data saved us a huge data collection and labelling effort.

Curriculum Learning. Bengio *et al*. [6] describe curriculum learning as a possible approach to training models that involve non-convex optimization. The idea is to rank the training instances by their difficulty. This ranking is then used for training the system by starting with the simple

[1]All the data and source codes are accessible at the first author's homepage at shafaei.ca.

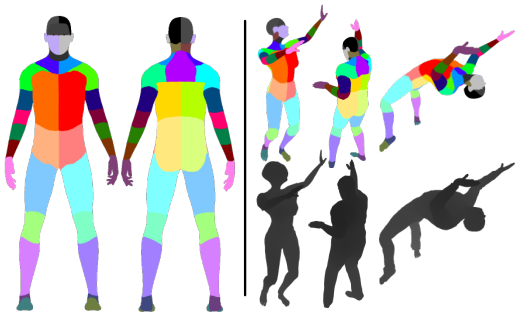


Figure 2: Color-coded body texture to identify regions of interest (left), and two random samples of depth and groundtruth from our data (right). We define 43 regions and distinguish left and right side of the body.

instances and then gradually increasing the complexity of the instances during the training procedure. This strategy is hypothesized to improve the convergence speed and the quality of the final local minima [6]. Our experiments suggest that a controlled approach to training deep convolutional networks can be *crucial* for training a better model, providing an example of curriculum learning in practice.

III. Synthetic Data Generation

We build a pipeline to generate samples of synthetic depth images of human body shapes using a variety of poses and viewpoints. Our sampling process is described in Alg. 1.

Algorithm 1 Sample data

$\mathcal{C}$: Pool of characters.
$\mathcal{L}$: Range of camera locations.
$\mathcal{P}$: Pool of postures.
n: Number of cameras.

procedure SAMPLE($\mathcal{C}, \mathcal{L}, \mathcal{P}, n$)
  $c \sim \text{Unif}(\mathcal{C})$ ▷ select a random character
  $l_{1:n} \sim \text{Unif}(\mathcal{L})$ ▷ select n random locations
  $p \sim \text{Unif}(\mathcal{P})$ ▷ select a posture
  $S \leftarrow$ Render depth image and groundtruth
  return S ▷ $S = \{(D_i, G_i)\}_{i=1}^{n}$

The character set $\mathcal{C}$ is the set of 3d models with variations in shape, age, clothing, and gender, generated using the free open-source project Make Human[2]. We create a special skin texture for the characters to color-code each region of interest (see Fig. 2). We found the 43 body regions shown in Fig. 2 to give sufficiently good results.

The camera location $\mathcal{L}$ is an interval description for camera placement. The azimuth spans the $[-\pi, +\pi)$ range. The camera distance from the character is within $[1.5, 4]$ m. Note that this distance constraint is only on the data generation process and not on the pose estimation pipeline. We also need real human postures $\mathcal{P}$ to pose our characters. We prepare this set by clustering all the human postures of the CMU motion capture[3] dataset and picking the 100,000 most dissimilar poses.

Table I: We generate three datasets as our curriculum. There are a total of 100,000 different postures. The simple set is the subset of postures that have the label 'walk' or 'run'. Going from the first dataset to the second would require pose adaptation, while going from the second to the third dataset requires shape adaptation.

Dataset	Postures	Characters	Samples
Easy-Pose	simple (~10 K)	1	1 M
Inter-Pose	all (100 K)	1	1.3 M
Hard-Pose	all (100 K)	16	300 K

The final parameter n is the number of cameras for each sample. The function `sample` returns a set of n pairs (D_i, G_i), where D_i is the quantized depth image and G_i is the groundtruth image as seen from the i-th camera. While the depth image *includes* the clothing items, in the groundtruth image we only render the textured skin. Rendering is done with Autodesk Maya[4]. We set the camera parameters such that the generated data resembles the Kinect 2 output.

We define three datasets with varying complexity as our curriculum for training, as described in Table I. The easiest dataset has only one character with the subset of postures that are labeled with 'walk' or 'run' (`Easy-Pose`). We then increase the difficulty by varying the set of possible postures (`Inter-Pose`), requiring the classifier to handle larger pose variations. The final dataset includes the set of all characters with different physical shapes as well as all the postures from our dataset (`Hard-Pose`). A transition from `Inter-Pose` to `Hard-Pose` would require learning shape variations. The entire data is generated with $n = 3$ cameras and each dataset has `train`, `validation`, and `test` sets with mutually disjoint postures. The datasets and the Python script to render customized data are made publicly accessible[5].

IV. Multiview Pose Estimation

Our framework consists of three stages (see Fig. 1). In the first stage we process the input of each sensor independently to generate an intermediate representation. Each depth image is passed through a Convolutional Neural Network (CNN) to generate densely classified depth. Having an intermediate representation gives us the flexibility to *add* and *remove* sensors from the environment with minimal reconfiguration.

[2] http://www.makehuman.org/

[3] http://mocap.cs.cmu.edu/

[4] http://www.autodesk.com/products/maya/

[5] Available at shafaei.ca.

To merge all the data we use the extrinsic camera parameters and reconstruct a 3d point cloud in the reference space. This 3d point cloud is then labeled with the classification probabilities and used for pose estimation. In the following sections we look at each step in detail.

A. Dense Depth Classification

We use CNNs to generate densely classified depth. Our architecture is motivated by the work of Long *et al*. [16]. More specifically we use deconvolution outputs fused with the information from the lower layers to generate fine densely classified depth. With this approach we are taking advantage of the information in the neighboring pixels to generate densely classified depth. This is in contrast to random forest based methods such as [2] where each pixel is evaluated independently.

Preprocessing. The input to our network is a 250×250 pixel depth image with 30 pixels of margin. The depth values are quantized to 255 levels spanning the $[50, 800]\,\mathrm{cm}$ range. We translate the average depth of each person to approximately $160\,\mathrm{cm}$ and spatially rescale to fit inside our window while preserving the aspect ratio.

Classification. The preprocessed image is then fed to the CNN of Fig. 3. The output of the network is $250 \times 250 \times 44$, representing a 44 dimensional vector of probabilities on each pixel for all the 43 classes and the background. The deconvolution kernel size of 19×19 at the final stage enforces the spatial dependency between the adjacent pixels.

Training. We initially attempted to train our network directly on `Hard-Pose`, however, in all of the trials with different optimization settings the accuracy of the network did not go above 50% in average per-class classification. Resorting to the curriculum learning idea of Bengio *et al*. [6] we simplified the task by defining easier datasets that we call `Easy-Pose` and `Inter-Pose` (see Table I).

We start training the network with the `Easy-Pose` dataset. Similar to Long *et al*. [16] we learned during our experiments that the entire network could be trained densely *end-to-end* from *scratch* without any class balancing schemes. Each iteration processes eight depth images and we stop the initial phase of training at $250\,\mathrm{K}$ iterations reaching dense classification accuracy of 87.8% on `Easy-Pose`. We then fine-tune the resulting network on `Inter-Pose`, initially starting at an accuracy of 78% and terminating after $150\,\mathrm{K}$ iterations with an accuracy of 82%. Interestingly the performance on the `Easy-Pose` is preserved throughout this fine-tuning. Finally we start fine-tuning on the `Hard-Pose` dataset and stop after $88\,\mathrm{K}$ iterations. Initially this network evaluates to 73% and by the termination point we have an accuracy of 81%. In all of the above steps we test on the `validation` sets of each dataset as the `test` sets are reserved for the final pose estimation task. The evolution of our three networks is shown in Table II. Notice how the final accuracy improved from 50% to 81% by applying curriculum learning. The transition of training from `Net 1` to `Net 2` demands generalization of posture, while the transition from `Net 2` to `Net 3` requires shape invariance. Our experiments demonstrate a real application of curriculum learning in practice.

Table II: The dense classification accuracy of the trained networks on the `validation` sets of the corresponding datasets. `Net 2` and `Net 3` are initialized with the learned parameters of `Net 1` and `Net 2` respectively.

Dataset	Easy-Pose		Inter-Pose		Hard-Pose	
	Start	End	Start	End	Start	End
Net 1	0%	87%	–	–	–	–
Net 2	87%	87%	78%	82%	–	–
Net 3	87%	85%	82%	79%	73%	81%

B. View Aggregation

At this step we have collected n densely classified outputs from our cameras. We wish to generate a single labeled 3d point cloud. Using the intrinsic parameters of each Kinect we can reconstruct a local point cloud. We can merge all the point clouds using the extrinsic camera parameters to transform each point cloud to a reference space.

We then collect a set of statistics, namely: (i) the median of each dimension, (ii) the covariance matrix, (iii) the eigenvalues of the covariance matrix, (iv) the standard deviation of each dimension, and (v) the minimum and the maximum of each dimension for each class and use it as features for the later stage. Since we are interested in real-time performance we limit ourselves to features that can be extracted quickly. The redundancy of information in the feature space, such as including the eigenvalues of the covariance matrix as well as the covariance itself, is to ensure linear models can take advantage of all the information that is available. We can devise filtering approaches for the merge process or perform temporal smoothing when the data is a sequence. However, we have found that our computationally inexpensive approach is sufficient to achieve state-of-the-art results. Our simple approach gives a feature vector $f \in \mathbb{R}^{1032}$, that is, concatenation of all 24 features per class.

C. Pose Estimation

We treat the problem of pose estimation as regression. For each joint j in our target skeleton we would like to learn a function $F_j(\cdot)$ that predicts the location of joint j based on the feature vector f. After examining a few real-time performing design choices such as *linear regression* and *neural networks* we learned that simple linear regression gives the best trade-off between complexity and performance. We experimented with various neural network architectures, but the simple linear regression always performed better *on average*. Our linear regression is a least squares formulation

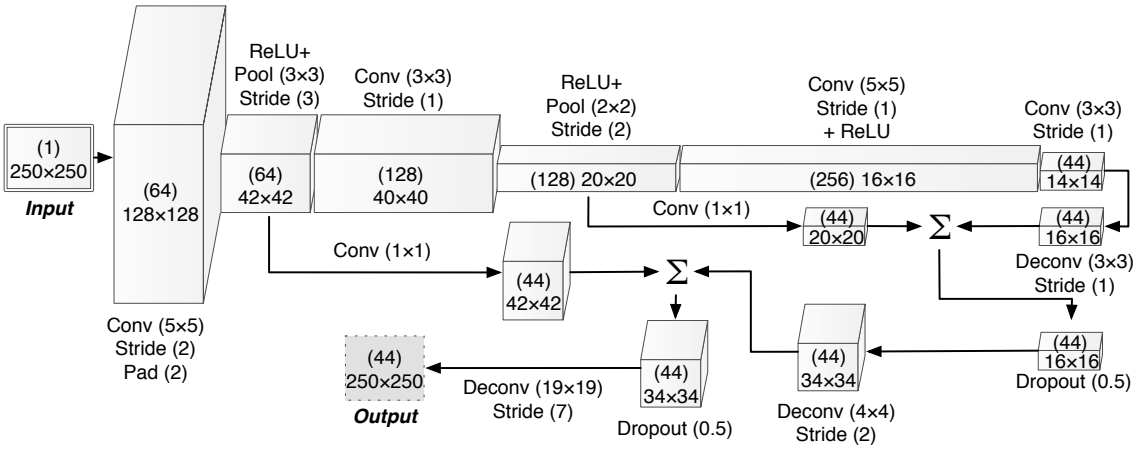


Figure 3: Our CNN dense classifier. The input is a 250×250 normalized depth image. The first part of the network generates a $44 \times 14 \times 14$ coarse densely classified depth with a high stride. Then it learns deconvolution kernels fused with the information from lower layers to generate fine densely classified depth. Like [16] we use summation and crop alignment to fuse information. The input and the output blocks are not drawn to preserve the scale of the image. The number in the parenthesis within each block denotes the number of the corresponding channels.

with an ℓ_2 regularizer which is also known as Ridge Regression and has a closed-form solution. We also experimented with the LASSO counterpart to obtain sparser solutions but the improvements were negligible while the optimization took substantially more time. If the input data is over a sequence, we further temporally smooth the predictions by calculating a weighted average over the previous estimates, *i.e.*, $\tilde{Y}_i = \sum_{j=0}^{K} \lambda_j Y_{i-j}$ s.t. $\sum_{j=0}^{K} \lambda_j = 1$. The regularizer hyper-parameters and the optimal smoothing weights are chosen automatically by cross-validation over the training data.

V. Evaluation and Discussion

In this section we present our evaluation results on three datasets. Since each dataset has a specific definition of joint locations we only need to train the regression part of our pipeline on each dataset. The CNN depth classifier of Sec. IV-A is trained *once* and *only* on the synthetic data. Our experiments are run with Caffe [22] on a Tesla K40 GPU. Each forward pass of the CNN takes $6.9\,\mathrm{ms}$ on the GPU or $40.5\,\mathrm{ms}$ on the CPU and requires $25\,\mathrm{MB}$ of memory per frame. The entire pipeline can operate at $30\,\mathrm{fps}$ on a single machine while communicating with up to four Kinects.

Evaluation Metrics. There are two evaluation metrics that are commonly used for pose estimation: mean joint prediction error, and mean average precision at threshold. Mean joint prediction error is the measure of average error incurred at prediction of each joint location. Mean average precision at threshold is the fraction of predictions that are within the threshold distance of the groundtruth. Because of the errors in groundtruth annotations it is also common to just report the mean average precision at $10\,\mathrm{cm}$ threshold [1], [3], [4].

Datasets. We provide quantitative evaluations on the Berkeley MHAD [5] and our synthetic dataset. Although for *single view* depth images there are a few datasets such as EVAL [1], and PDT [8], the only publicly available dataset for multiview depth at the moment of writing is the Berkeley MHAD [5].

A. Evaluation on UBC3V Synthetic

For evaluation we use the `Test` set of the `Hard-Pose`. This dataset consists of $19{,}000$ postures with 16 characters from three cameras placed at random locations. These $19{,}000$ postures are not present in the training set of our dataset and have not been observed before. The groundtruth and the extrinsic camera parameters come from the synthetic data directly and there are no errors associated with them. Having groundtruth for body part regions and the posture helps us separate the evaluation of the dense classifier and the pose estimation technique. That is, we can evaluate the pose estimation technique assuming perfect dense classification is available separately from the case where classification comes from our CNN. This separation gives us insight on how improvement on dense classification is likely to affect pose estimation, and whether one should spend time on improving the classifier or the pose estimator.

For training we have the multi-step fine-tuning procedure as described in Sec. IV-A. We refer to the final fine-tuned network as `Net 3` throughout our experiments.

Dense Classification. The `Test` set of the `Hard-Pose` includes $57{,}057$ depth frames with dense class annotations that are synthetically generated. Figure 4 demonstrates a few random classification samples and the corresponding groundtruth. The CNN *correctly* identifies the direction of

Figure 4: The groundtruth body part regions versus the output of `Net 3` classifier on the `Test` set of the `Hard-Pose`. Each pixel of the `Net 3` output is the color of the most likely class.

the body and generates true classes for the *left* and the *right sides*, but seems to be ignoring sudden discontinuities in the depth. For instance in the middle column of Fig. 4 parts of the right shoulder are mixed with the head classes. The overall accuracy of `Net 3` on the `Test` set is 80.6%, similar to the reported accuracy on the `Validation` set in Table II.

Pose Estimation. We evaluate our linear regression on the groundtruth class and the classification output of our CNN. The estimate derived from the groundtruth serves as a lower bound on the error for the pose estimations algorithm. The mean average joint prediction error is shown in Fig. 5. Our system achieves an average pose estimation error of 2.44 cm on groundtruth, and 5.64 cm on the `Net 3`. The gap between the two results is due to dense classification errors. This difference is smaller on easy to recognize body parts and gets larger on the hard to recognize classes such as hands or feet. It is possible to reduce this gap by using more sophisticated pose estimation methods at the cost of more computation. In Fig. 6 we compare the precision at threshold. The accuracy at 10 cm for the groundtruth and the `Net 3` is 99.1% and 88.7% respectively.

B. Evaluation on Berkeley MHAD

This dataset includes 12 subjects performing 11 actions while being recorded by 12 cameras, two Kinect ones, an Impulse motion capture system, four microphones, and six accelerometers. The motion capture sequence with 35 joints is the groundtruth for pose estimation on this dataset (for a list of joints see Fig. 9a). We only use the depth information of the two opposing Kinect ones for pose estimation.

At the moment of writing there is no protocol for evaluation of pose estimation techniques on this dataset. The leave-

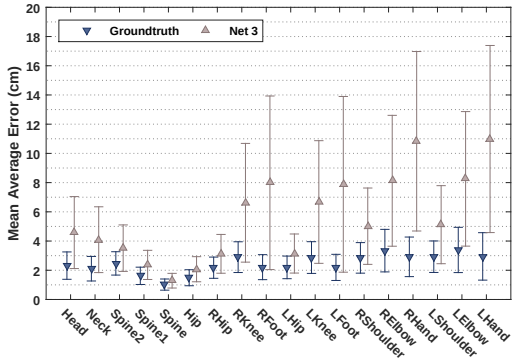


Figure 5: Mean average joint prediction error on the groundtruth and the `Net 3` classification output. The error bar is one standard deviation. The average error on the groundtruth is 2.44 cm, and on the `Net 3` is 5.64 cm.

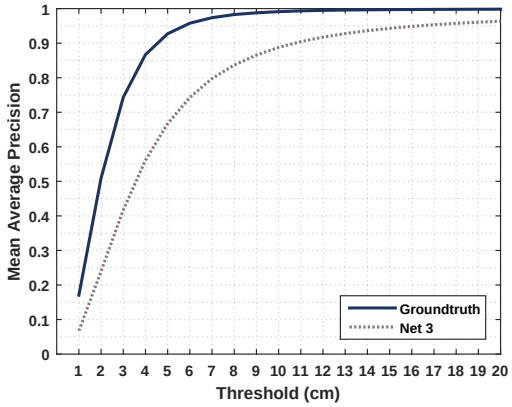


Figure 6: Mean average precision of the groundtruth dense labels and the `Net 3` dense classification output with accuracy at threshold 10 cm of 99.1% and 88.7% respectively.

one-out approach is a common practice for single view pose estimation. However, each action has five repetitions and we can argue in general that it may not be a fair indicator of the performance because the method can adapt to the shape of the subject of the test on other sequences to get a better result. Furthermore, we are no longer restricted to only a few sequences of data as in previous datasets.

To evaluate the performance on this dataset we take the harder leave-one-subject-out approach, that is, for evaluation on each subject we train our system on all the other subjects. This protocol ensures that no extra physical information is leaked during the training and can provide a measure of robustness to shape variation.

Dense Classification. To use the CNN that we have trained on the synthetic data we need to rescale the depth images of this dataset to match the output scale of Kinect 2 sensors. After this step we simply feed the depth image to the CNN to get dense classification results. Figure 7 shows

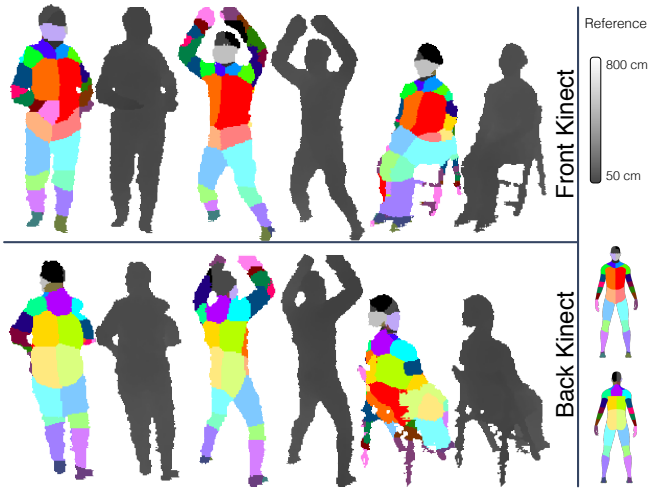


Figure 7: The dense classification result of `Net 3` together with the original depth image on the Berkeley MHAD [5] dataset. `Net 3` has been trained only on synthetic data. Each pixel is colored according to the most likely class.

the output of our dense classifier from the two Kinects on a few random frames. Even though the network has been only trained on synthetic data, it is *generalizing* well on the real test data. As demonstrated in Fig. 7, the network has also successfully captured the long distance spatial relationships to correctly classify pixels based on the orientation of the body. The right column of Fig. 7 shows an instance of high partial classification error due to occlusion. On the back image, the network mistakenly believes that the chair legs are the subject's hands. However, once the back data is merged with the front data we get a reasonable estimate (see Fig. 8).

Pose Estimation. We use the groundtruth motion capture joint locations to train our system. For each test subject we train our system on the other subjects' sequences. The final result is an average over all the test subjects.

Figure 9a shows the mean average joint prediction error. The total average joint prediction error is $5.01\,\mathrm{cm}$. The torso joints are easier for our system to localize than hands' joints, a similar behavior to the synthetic data results. However, it must be noted that even the groundtruth motion capture on smaller body parts such as hands or feet is biased with a high variance. During visual inspection of Berkely MHAD we noticed that on some frames, especially when the subject bends over, the location of the hands is outside of the body point cloud or even outside the frame, and clearly erroneous. The overall average precision at $10\,\mathrm{cm}$ is 93%.

An interesting observation is the similarity of performance on Berkeley MHAD data and the synthetic data in Fig. 5. This suggests that the synthetic data is a reasonable proxy for evaluating performance, as has been suggested by Shotton *et al*. [2]. Figure 9b shows the accuracy at threshold for joint location predictions.

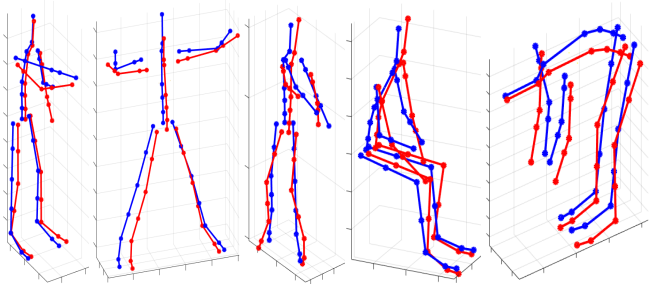


Figure 8: The blue color is the groundtruth of Berkeley MHAD [5] and the red color is our pose estimate.

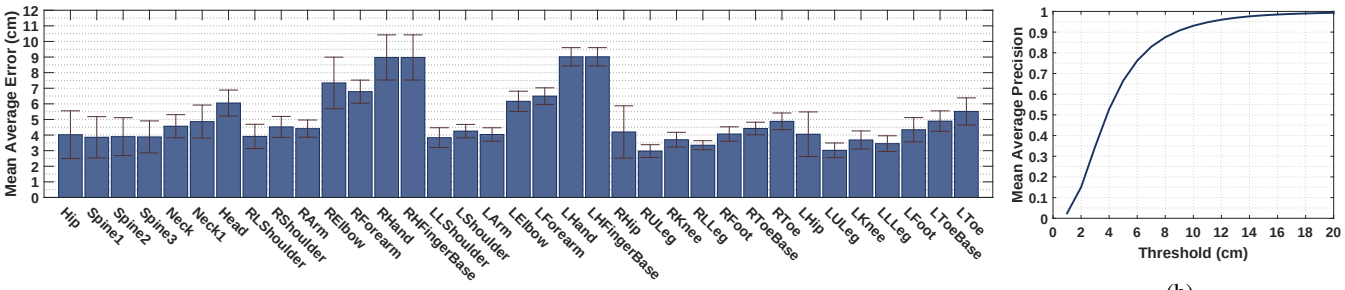
Table III: Mean and standard deviation of the prediction error by testing on subjects and actions with the joint definitions of Michel *et al*. [12]. We also report and compare the accuracy at $10\,\mathrm{cm}$ threshold.

	Subjects			Actions		
	Mean	Std	Acc (%)	Mean	Std	Acc (%)
OpenNI [12]	5.45	4.62	86.3	5.29	4.95	87.3
Michel *et al*. [12]	3.93	2.73	96.3	4.18	3.31	94.4
Ours	**3.39**	**1.12**	**96.8**	**2.78**	**1.5**	**98.1**

We also compare our performance with Michel *et al*. [12] in Table III. Since they are using an alternative definition of skeleton that is derived by their shape model, we only evaluate over a subset of the joints that are closest with the locations presented in Michel *et al*. [12]. Note that the method of [12] uses predefined shape parameters that are optimized for each subject *a priori* and does not operate in real-time. In contrast, our method does not depend on shape attributes and operates in real-time. Following the procedure of [12] we evaluate by testing the subjects and testing the actions. Our method improves the previous mean joint prediction error from 3.93 to 3.39 (13%) when tested on subjects and 4.18 to 2.78 (33%) when tested on actions.

VI. Conclusion

We presented an efficient and inexpensive markerless motion capture system that uses only a few Kinect sensors. Our system only assumes availability of calibrated depth cameras and is capable of real-time performance without requiring an explicit shape model or cooperation. We further presented a dataset of ~6 million synthetic depth frames for pose estimation from multiple cameras. Our experiments demonstrated an application of curriculum learning in practice and our system exceeded state-of-the-art multiview pose estimation performance on the Berkeley MHAD dataset.



(a) (b)

Figure 9: (a) Joint prediction mean average error on the Berkeley MHAD [5] dataset. The error bar is one standard deviation. The total mean average error per joint is $5.01\,\mathrm{cm}$. (b) Mean average precision at threshold for the entire skeleton on the Berkeley MHAD dataset.

ACKNOWLEDGEMENTS

We would like to thank Ankur Gupta for helpful comments and discussions. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research. This work was supported in part by NSERC under Grant CRDPJ 434659-12 and the ICICS/TELUS People & Planet Friendly Home Initiative at UBC.

REFERENCES

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

[2] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time Human Pose Recognition in Parts from Single Depth Images," *Communications of the ACM*, vol. 56, no. 1, 2013.

[3] M. Ye and R. Yang, "Real-time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera," in *CVPR*, 2014.

[4] H. Yub Jung, S. Lee, Y. Seok Heo, and I. Dong Yun, "Random Tree Walk Toward Instantaneous 3D Human Pose Estimation," in *CVPR*, 2015.

[5] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy, "Berkeley MHAD: A Comprehensive Multimodal Human Action Database," in *WACV*, 2013.

[6] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum Learning," in *ICML*, 2009.

[7] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real Time Motion Capture Using a Single Time-of-flight Camera," in *CVPR*, 2010.

[8] T. Helten, A. Baak, G. Bharaj, M. Müller, H.-P. Seidel, and C. Theobalt, "Personalization and Evaluation of a Real-time Depth-based Full Body Tracker," in *3rd joint 3DIM/3DPVT Conference (3DV)*, 2013.

[9] P. Zhang, K. Siu, J. Zhang, C. K. Liu, and J. Chai, "Leveraging Depth Cameras and Wearable Pressure Sensors for Full-body Kinematics and Dynamics Capture," *TOG*, vol. 33, no. 6, 2014.

[10] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient Regression of General-activity Human Poses from Depth Images," in *ICCV*, 2011.

[11] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr, "Conditional Random Fields as Recurrent Neural Networks," in *ICCV*, 2015.

[12] D. Michel, C. Panagiotakis, and A. A. Argyros, "Tracking the Articulated Motion of the Human Body with Two RGBD Cameras," *Machine Vision and Applications*, vol. 26, no. 1, 2015.

[13] A. Phan and F. P. Ferrie, "Towards 3D Human Posture Estimation Using Multiple Kinects Despite Self-Contacts," in *15th IAPR International Conference on Machine Vision Applications*, 2015.

[14] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs," in *ICLR*, 2015.

[15] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for Object Segmentation and Fine-grained Localization," in *CVPR*, 2015.

[16] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *CVPR*, 2015.

[17] G. Papandreou, L. Chen, K. Murphy, and A. L. Yuille, "Weakly- and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation," in *ICCV*, 2015.

[18] D. Park and D. Ramanan, "Articulated Pose Estimation with Tiny Synthetic Videos," in *CVPR*, 2015.

[19] G. Rogez, J. S. Supancic III, and D. Ramanan, "First-Person Pose Recognition using Egocentric Workspaces," in *CVPR*, 2015.

[20] A. Gupta, J. Martinez, J. J. Little, and R. J. Woodham, "3D Pose from Motion for Cross-view Action Recognition via Non-linear Circulant Temporal Encoding," in *CVPR*, 2014.

[21] A. Gupta, A. Shafaei, J. J. Little, and R. J. Woodham, "Unlabelled 3D Motion Examples Improve Cross-View Action Recognition," in *BMVC*, 2014.

[22] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," in *ACM International Conference on Multimedia*, 2014.