This is a post-peer-review, pre-copyedit version of an article published in **2017 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)**. The final authenticated version is available online at: https://doi.org/10.23919/SPA.2017.8166892

ANN for human pose estimation in low resolution depth images

Piotr Szczuko
Faculty of Electronics, Telecommunications and Informatics,
Gdańsk University of Technology, Multimedia Systems Department,
Gdansk, Poland
szczuko@multimed.org

***Abstract*—The paper presents an approach to localize human body joints in 3D coordinates based on a single low resolution depth image. First a framework to generate a database of 80k realistic depth images from a 3D body model is described. Then data preprocessing and normalization procedure, and DNN and MLP artificial neural networks architectures and training are presented. The robustness against camera distance and image noise is analysed. Localization accuracy for each joint is reported and application for low resolution and large distance pose estimation is proposed. A very fast regression on body joints locations in 3D space is achieved, even in case of sensor noise, large distance and reaching off the screen.**

Keywords—depth image; pose estimation; artificial neural networks; deep learning

I. INTRODUCTION

Pose estimation is a key procedure in many human-computer interfaces, particularly it is present in multimedia and gaming, with the most prominent example of Kinect sensor, applicable also in contactless motion capture in medicine [1]. Human body orientation and joints rotations define a complex, articulated 3D object, that should be captured and parameterized from a 2D grayscale or color image [2][3], or 2D depth map [4]. Depending on used methods numerous problems can occur, such as obscuration of body parts, e.g. by crossing legs or arms, front-back confusions, pose ambiguities, low accuracy of joints localizations and orientations.

Kinect sensor uses structured light to obtain depth map of size 640x480 pixels with 11-bit resolution. Algorithm based on decision trees analysing local features of the depth image is applied first to divide the scanned geometry into body parts, and then to calculate their 3D space coordinates. The method requires relatively high resolution of the depth map and high processing power (one frame is processed in 1/200s on GPU, and in 1/50s on 8 cores CPU). The localization error less than 0.1m is achieved in 75% of cases [4].

Deep neural network architecture dedicated for image processing [5][6] is usually composed of several layers, of three different types: 1) convolutional layer taking input values of rectangular areas of defined window size, performing convolution with a given set of kernels and returning filtered results acting as a feature maps; 2) pooling layers, reducing the size of processed feature sets, and introducing invariance to shifts or distortions, by e.g. choosing maximal value of a rectangular window slid with a defined step over each feature map; 3) fully connected layers to decode the internal state of the network and features into a desired output signal.

Deep architectures were used for pose estimation in a Deeppose framework [3]. 2D color natural images were used to estimate upper body pose, achieving speed of 0.1s per 220x220 image on 12-core CPU. Tompson et al. used deep convolutional network architectures to detect body joints in two sliding windows of 128x128 and 64x64 pixels over 2D images of 320x240 pixels , with processing speed or 0.051s per image [2]. Highly complex method of model-based search was implemented and joints locations were compared with motion capture ground truth data [7]. Average pose error was 0.05-0.1m (depending on a test sequence), but the processing speed was low: 0.1s per 128x128 depth image.

The aim of this work is to speed up the joint localization process and reduce the required size of input depth map, comparing to Kinect. It is achieved by applying neural networks. Multilayered perceptron network (MLP) and deep neural network (DNN) are evaluated. Low resolution depth images of 60x50 pixels (100 times less pixels than Kinect) are used. Decrease in image resolution makes the method more robust against small differences in body shapes, cloths variants, accessories, etc. A very fast regression on body joints locations in 3D space is achieved, even in case of sensor noise, large distance, and reaching off the screen.

The reminder of this paper is organized as follows: Sec. 2 documents depth images synthesis, Sec. 3 describes MLP and DNN architectures and training procedure, Sec. 4 presents results of joint localization, and Sec. 5 concludes the paper.

II. DATASET

A new dataset was created consisting synthetic low resolution depth images of a human figure. Blender3D animation software was used for image rendering. The virtual camera was positioned at coordinates (0,0,0), with no tilt, pointed along Z axis, with horizontal field of view set to 57° (same as Kinect). Depth value, calculated as a distance of pixel from the camera, was rendered in a frame of 60x50 pixels in 8-bit greyscale (8 times lower depth resolution than Kinect). To reduce jagged lines on the figure outline a 16 sample anti-aliasing was used.

The human figure was positioned in an empty space, without background. To obtain the same conditions in a real application a separation of foreground object based on depth information should be performed, for example by assuming the body to be the closest object to a camera. A particular method for such a separation if out of scope of this paper.

To simplify posing of the figure, the skeleton was using inverse kinematics (IK) for left and right hand-forearm-arm-spine chains, with IK targets influencing positions of wrists, and other bones in the chains adapting by solving the IK to reach targets. The joints had skeletal constrains defined, such as the elbow rotation allowed only along one axis and within the range of (0,180) degrees. This approach simplified generation of subjectively correct body poses.

IK targets were positioned along a 3D grid, with small random offsets added, to uniformly cover the working area in front of the figure. The absolute position of the whole body was changed randomly to introduce shifts in x, y and z directions relative to the stationary camera. The movement ranges allowed only far reaching hands to leave the camera frame, and the rest of the figure always remained in the frame (Table 1). In total 47,628 sample images were created, randomly split in 60/40 proportion into training and testing sets, counting 28,576 and 19,052 images respectively.

TABLE I. JOINTS LOCATIONS RANGES FOR POSING THE HUMAN FIGURE

	min [m]	max [m]
R Hand X[a]	-0.2	1.2
L Hand X[a]	-1.2	0.2
R, L Hand Y[a]	-0.3	1.5
R, L Hand Z[a]	0.15	1.3
Root X	-1	1
Root Y	0	0.5
Root Z	1.6	2.2

[a.] Hand coordinates relative to root joint location

A Python script was created for posing the figure following the procedure mentioned above, and logging all ground truth joints coordinates to a file. For each rendered depth image the figure state was obtained: 3D location in meters of the root joint (reflecting the figure absolute location in front of the camera), 3D coordinates in meters of both hands, elbows, arms, relative to the root, and 2D coordinates in pixels of hands, elbows, and arms in the image plane. Hands locations are uniformly spread in the space in front of the figure, but elbows and arms movements are limited due to dependence on IK solutions, thus their values are characterised by non-uniform distribution.

The pose state is defined as a set joints 3D coordinates:

$$\mathbf{P}=\{(j_x, j_y, j_z)\} \tag{1}$$

where a joint $j \in$ {root, right arm, right elbow, right hand, left arm, left elbow, left hand}. Pose estimation error is defined twofold, as an error along particular axis $\varepsilon^j_{\text{loc}}$, and as a distance in space between true and estimated values $\varepsilon^j_{\text{dist}}$ for particular joint j:

$$\varepsilon^j_{\text{loc, dim}} = |j_{\text{dim}} - j^*_{\text{dim}}| \tag{2}$$

$$\varepsilon^j_{\text{dist}} = \|(j_x, j_y, j_z), (j^*_x, j^*_y, j^*_z)\| \tag{3}$$

where $dim \in \{x, y, z\}$, and j^*_{dim} is estimated joint coordinate along axis specified by dim, and $\|\cdot,\cdot\|$ is a distance metric between two points in space. Here, an Euclidean distance is used.

In the 2D image plane the localization of joints is a set **L** of pixel coordinates:

$$\mathbf{L}=\{(i_x, i_y)\} \tag{4}$$

and error is expressed as:

$$\varepsilon^i_{\text{loc, dim}} = |i_{\text{dim}} - i^*_{\text{dim}}| \tag{5}$$

where $dim \in \{x, y\}$ are image plane coordinates in pixels.

III. TRAINING OF MLP AND DNN ARCHITECTURES

The depth images dataset was preprocessed by adding noise to the depth value: for a half of training images a noise with mean=0.5cm and +/-0.5cm standard deviation, and for the other half a noise with mean=1cm and +/-1cm standard deviation. A noise was added to all testing images with mean=0, 0.5, 0.75cm and 1cm, and deviations of 0, 0.5, 0.75 and 1 respectively, composing four sets with the same test images but varying in noise level. Next, an average image over training subset was calculated, and subtracted from every training and testing sample (Fig. 1).

The output values of **P** and **L** (root location, joints relative locations, and image plane coordinates) were normalized to a range ⟨0,1⟩. Normalization factors were obtained from the training dataset, individually per each output and are used again to scale the network output back to original ranges for correctly expressing errors ε^j_{loc}, ε^j_{dist}, ε^i_{loc}. The normalization helps each output value and gradient to influence the network weights in the same degree, increasing the speed of training [8].

Two architectures are considered, a Convolutional Deep Neural Network (DNN), and Multi-Layered Perceptron (MLP) treated as a multiple regression networks, trained to return continuous values of predicted coordinates **P** and **L**, minimizing $\varepsilon^j_{loc,\,dim}$ and ε^i_{loc}.

The input for DNN is an image of 60x50 pixels, first convoluted with 20 different kernels of 5x5, resulting in 20 feature maps of size 60x50. Next, for each map the max pooling is performed within a sliding window of 2x2 with a stride of 2x2, resulting in 20 feature maps of size 30x25. After another convolution and pooling 15x13x50 feature maps are obtained. This results in 9750 values passed to fully connected layer of 500 neurons, and a final fully connected layer of 33 output neurons (Fig. 2). In each neuron a rectified linear activation function was used, following the suggestion that Rectified Linear Units (ReLU), are able to learn faster than *tanh* or sigmoid due to lower computational complexity and lack of saturation [9]. MLP is composed of the input layer with 3000 neurons, then 500 neurons in a hidden layer (the same number as in fully connected layer in DNN), and 33 in output layer (Fig. 3). ReLU function was used as the activation in all neurons.

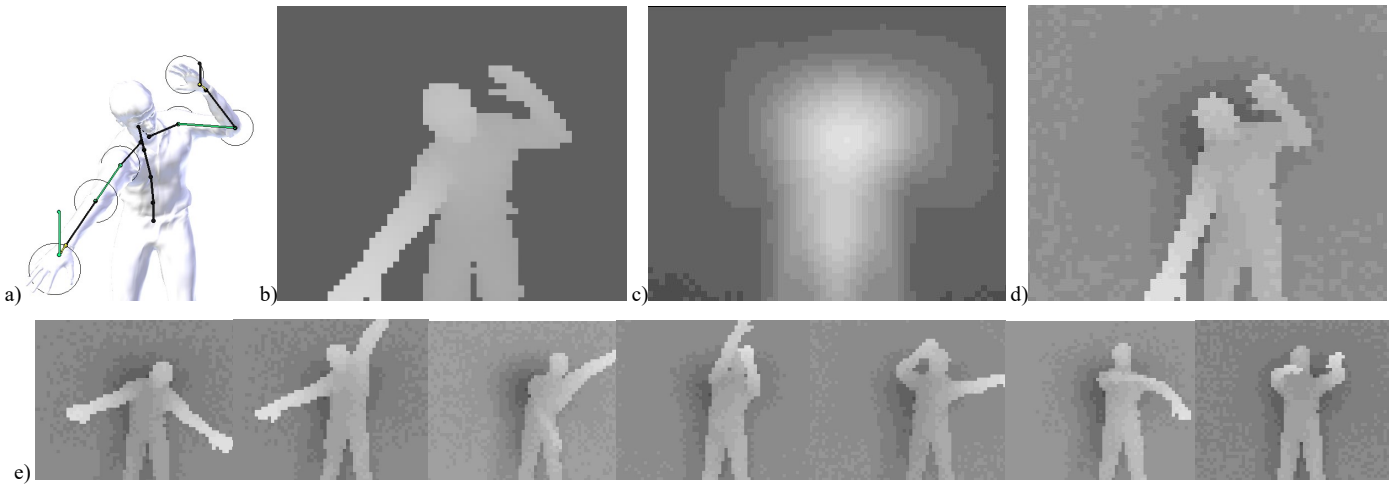


Fig. 1. Depth image preparation: a) human model and upper body skeleton (circles mark joints of interest), b) rendered depth image, c) mean image of the training set, d) noised input with mean removed.

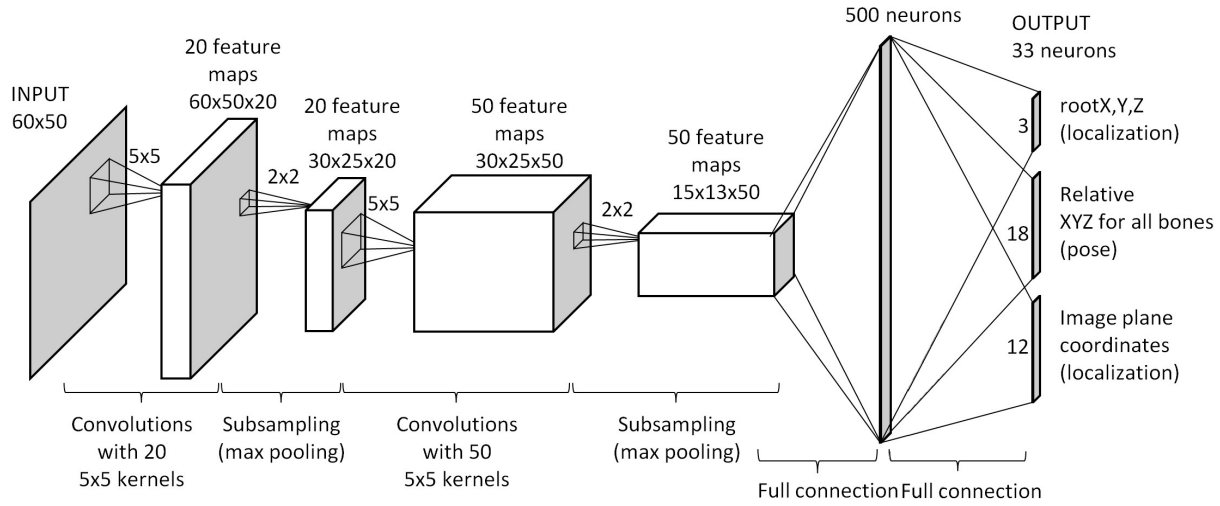


Fig. 2. Achitecture of the DNN used in the experiment

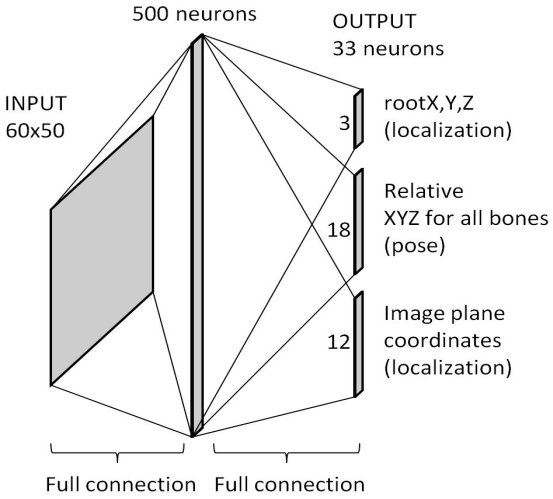


Fig. 3. Achitecture of the MLP network

The training was conducted in R environment on a 4 core i5@3.2GHz CPU, with 16GB RAM. For both networks the learning rate was set to 0.05, and momentum to 0.3. The validation error was observed, and the training was stopped once the mean absolute error decrease became lower than 0.00001 per epoch.

One epoch of MLP training took approx. 6 seconds, and one DNN epoch took 180 seconds. MLP was trained for 3600 epochs, and DNN for 1000 epochs.

The mean absolute error (MAE) for training data and evaluation data during training is presented on Fig. 4, visualised for particular epoch number, and for elapsed time. It can be noticed that DNN achieves accuracy similar to MLP after ca. 24 hours of training, meaning 4x more time is required for DNN. After this period a further training for 24 hours improves the accuracy only by 0.008 MAE.

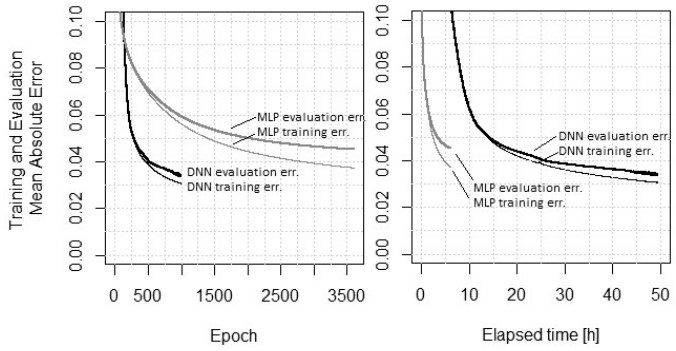


Fig. 4. Mean absolute errors during the training for given number of epochs and given time.

IV. ACCURACY EVALUATION

The MLP and DNN regression was evaluated for all four testing sets (each with the same 19,052 poses, with depth values disturbed by 4 types of additive noise). Joint localization errors expressing a difference between network output and actual joint location $\varepsilon^{j}_{\text{loc, dim}}$ (2) were calculated for both network architectures (Fig. 6).

A. Noise robustness

The dependence of pose **P** estimation errors on camera distance was verified by calculating Pearson correlation (Table 2). It was observed that a small positive correlation is present and errors subtly increase with distance, with the strongest dependencies for 2D coordinates in MLP scenario (up to 0.172 for noise of 0.01m). On the other hand lack of noise for MLP for hands localization lowers the accuracy, as the network was not trained on noise free images.

TABLE II. MAXIMAL CORRELATION BETWEEN LOCALIZATION ERRORS AND DISTANCE FROM THE CAMERA DEPENDING ON DEPTH NOISE

	Mean of depth noise [m]:			
	0	***0.005***	***0.0075***	***0.01***
Localization scenario:	***Correlation between localization errors and camera distance:***			
MLP for all joints in 2D	0.065	**0.115**	**0.145**	**0.172**
MLP for all joints in 3D	0.076	0.062	0.055	0.052
DNN for all joints in 2D	0.066	0.053	0.066	0.121
DNN for all joints in 3D	0.075	0.087	**0.105**	**0.104**
MLP for hands in 2D	0.044	0.024	0.016	0.020
MLP for hands in 3D	0.076	0.062	0.055	0.048
DNN for hands in 2D	0.013	0.051	0.065	0.055
DNN for hands in 3D	0.041	0.087	**0.105**	**0.104**

Mean absolute errors of joint 3D localization $\varepsilon^{j}_{\text{loc}}$ over all poses **P** achieved by MLP and DNN are similar regardless of noise levels (Fig. 6b). For the subsequent analysis of localization errors a test set with mean noise of 0.0075m was used.

B. Localization accuracy

It is evident that global localization of the root bone is the most accurate, the absolute error $\varepsilon^{root}_{\text{loc, x,y,z}}$ rarely exceeds 2cm. Arms and elbows joints are precisely located as well (absolute error less than 10cm). The lowest accuracy is achieved for hands (median error less than 5cm, but in worst cases reaching 20cm), supposedly due to a large variation in area and shape of the hand in the depth image.

It should be stressed here that each actual joint is immersed deep inside the 3D geometry, few centimetres below the observable surface. The network is expected to reason about the joint location under the surface based on outside geometry, and this makes the location problem hard to some extent, especially for hands.

Absolute 2D localization error $\varepsilon^{j}_{\text{loc, x,y}}$ (5) in the pixels coordinates is below 6 pixels, so this must be taken into account for applications relying on 2D positioning.

The joints localization error expressed as an Euclidean distance $\varepsilon^{j}_{\text{dist}}$ (3) for DNN has a median of 10cm for hands, reaching 30cm in worst cases. Other joints are located with higher precision. Generally, MLP architecture performs worse than DNN in all experiments (Fig. 5). Sample frames with low and high errors are presented in Fig. 7.

For MLP network the pose estimation and joints location speed reaches over 32,000 images per second, and for DNN over 530 images per second (averaged over all 19k testing samples, measured on a 4 core i5@3.2GHz CPU). Thus the achieved speed exceeds the value of 200 images reported for Kinect [4]. Taking into account the very low resolution of depth image (60x50 pixels) the performance of both architectures is satisfactory. Depending on considered applications, and requirements for speed and accuracy the MLP or DNN can be used.

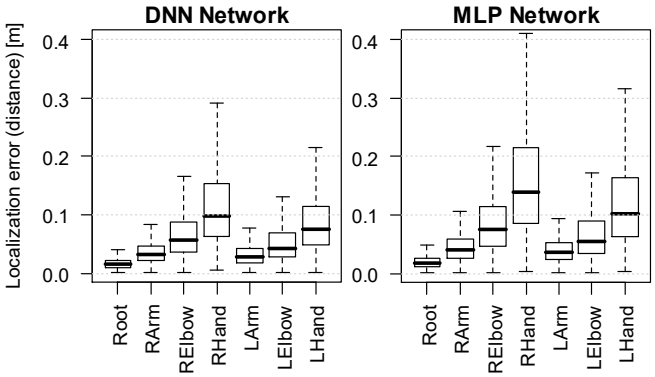


Fig. 5. Localization errors (distances) for joints

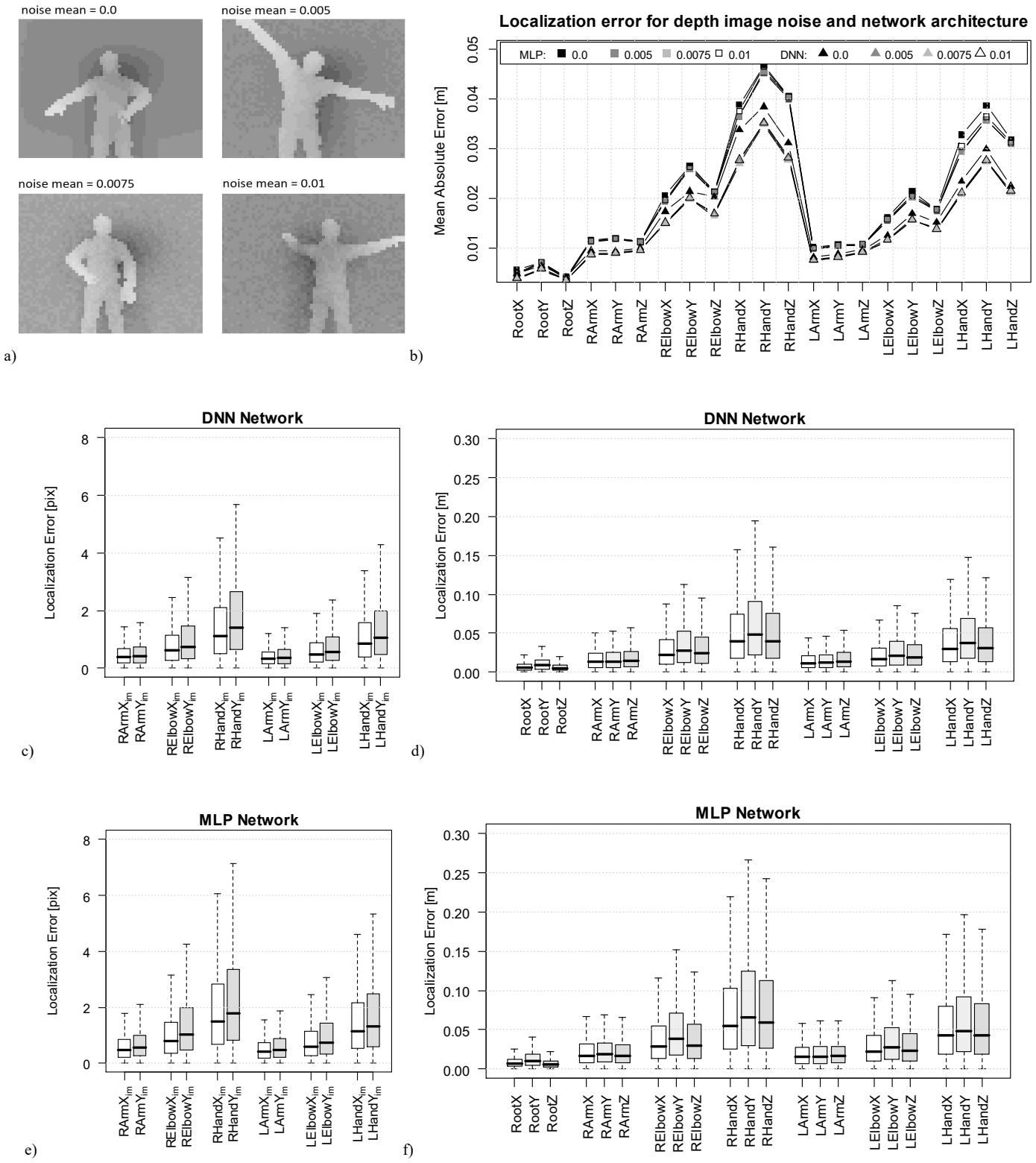
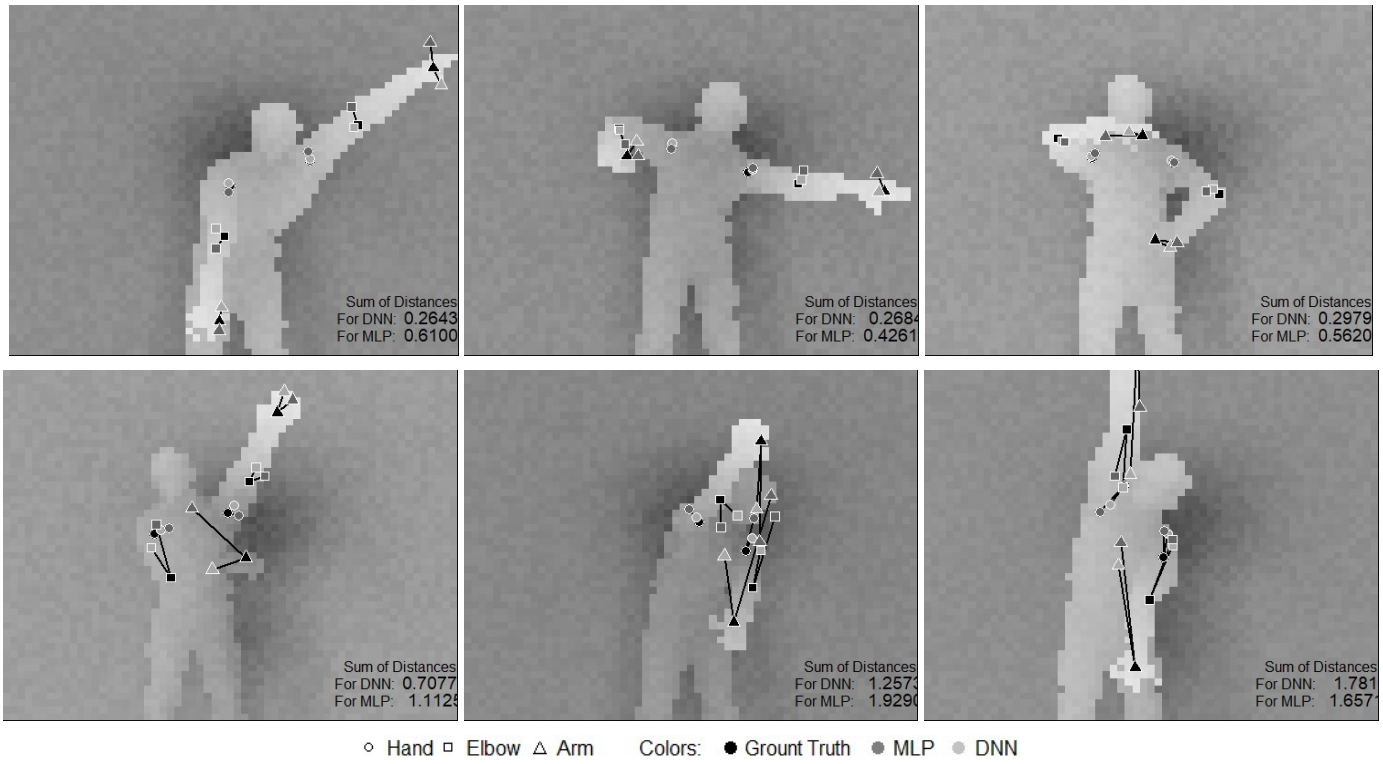


Fig. 6. Localization results: a) samples of testing images with noise 0, 0.005, 0.0075 and 0.01 m, b) mean absolute errors (in m) achieved by MLP and DNN architectures for given depth noise (in m). Detailed results for and noise with mean=0.0075: c) absolute errors in image plane localization of joints (in pixels) for DNN architecture, d) absolute errors in 3D localization of joints (in m) for DNN architecture, e) absolute errors in image plane localization of joints (in pixels) for MLP architecture, f) absolute errors in 3D localization of joints (in m) for MLP architecture



Poses with low and high localization errors (sums of distances between Ground Truth and estimated locations).

V. CONCLUSIONS

Presented method of human figure pose estimation by localization of joints in 3D space achieved satisfactory accuracy, and very high processing speed. The method operates on low resolution depth image (100 less pixels than Kinect sensor). Decrease in image resolution makes the method more robust against small differences in body shapes, cloths variants, accessories, etc.

Presented results can be improved by introducing tracking of joints over time, relying on prediction or Kalman filtration of moving limb performing a gesture. However, such a strategy will not be explored, as it would be time consuming to recover from the tracking errors, and the focus of this work is the fast localization in low resolution depth images.

The accuracy is sufficient for pose estimation and the method can be used for low resolution depth sensors in gesture interfaces, i.e. onscreen menus with items pointed by the user by hand movements in space, similar to many Kinect applications. The method is robust against sensor noise, allowing imprecision of depth of +/- 1cm.

The observed small differences in accuracy between elaborated DNN dedicated to image processing and simple MLP will be explored. Longer network training exploiting GPU acceleration is planned, as expected to improve DNN accuracy.

The ongoing work focuses on introducing larger variation of body poses, including front and side orientations and legs movement.

REFERENCES

[1] N. Hesse, G. Stachowiak, T. Breuer, M. Arens, “Estimating body pose of infants in depth images using random ferns”. In Proceedings of the IEEE International Conference on Computer Vision Workshops,pp. 35-43, 2015.

[2] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler, “Joint training of a convolutional network and a graphical model for human pose estimation”. In Advances in neural information processing systems, pages 1799–1807, 2014.

[3] A. Toshev and C. Szegedy, “Deeppose: Human pose estimation via deep neural networks”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1653–1660, 2014. https://arxiv.org/pdf/1312.4659.pdf

[4] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, “Real-time human pose recognition in parts from single depth images”. Communications of the ACM, 56(1):116–124, 2013. doi:10.1145/2398356.2398381

[5] Y. LeCun, Y. Bengio, G. Hinton, “Deep learning”. Nature 521, 436–444 (28 May 2015) doi:10.1038/nature14539

[6] A. Krizhevsky, I. Sutskever, G. Hinton, “Imagenet classification with deep convolutional neural networks”. Advances in Neural Information Processing Systems. 1: 1097–1105, 2012.

[7] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, “Real time motion capture using a single time-of-flight camera”. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 755–762. IEEE, 2010.

[8] Y. LeCun, L. Bottou, G. B. Orr, K.-R. Müller, “Efficient BackProp”, Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science 1524, pp 9-50, 2002.

[9] V. Nair, G.E. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines”. Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp. 807-814, 2010.