RGBD-Dog: Predicting Canine Pose from RGBD Sensors (Supplementary Material)

Sinéad Kearney [1] Wenbin Li [1] Martin Parsons [1] Kwang In Kim [2] Darren Cosker [1]

[1]University of Bath [2]UNIST

{s.kearney,w.li,m.m.parsons,d.p.cosker}@bath.ac.uk kimki@unist.ac.kr

1. Introduction

In this supplementary material we give additional technical details on our approach.We provide details on our dog data set, which will be made available to the research community. We conduct additional experiments to test the pipeline for occlusions, exploiting depth information when solving for the shape of the dog, and compare the neural network in the pipeline with two other networks. Finally we compare the expression of our dog shape model with that of SMAL [14].

2. Method

2.1. Animal Motion Data Collection

Each recorded dog wore a motion capture suit, on which was painted additional texture information. The number of markers on the suit related to the size of a given dog, and ranged from 63 to 82 markers. We show an example of marker locations in Figure 1. These locations were based on reference to those on humans and biological study. Vicon Shogun was used to record the dogs with 20 cameras at 119.88fps, the locations of which are shown in Figure 2.

Our dataset consists of five similar motions for five dogs; walking and trotting sequences in an approximately straight line, a jump sequence, a sequence where the dog is walking over poles placed on the floor, and finally a sequence of the dog stepping or jumping on and off a table approximately 30cm in height. The props used during these sequences are shown in Figure 3. For each sequence, the dog is accompanied by its handler. This person is not wearing a motion capture suit and no skeleton data of the person is provided.

For each dog, this data is available in the form of 3D marker locations, the solved skeleton, the neutral mesh of the dog and corresponding Linear Blend Skinning weights, multi-view HD RGB footage recorded at 59.97 fps, and multi-view RGB and RGB-D images from the Microsoft Kinect recording at approximately 6 fps. The HD RGB footage will be available in 4K resolution on request. The

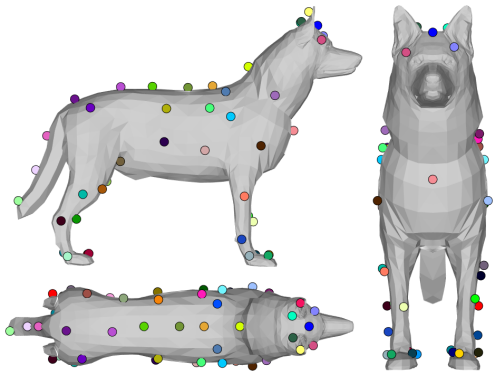

Figure 1. The locations of the markers as worn by one of the dogs in the capture session, placed on the artist-created mesh of the dog. This particular dog had 64 markers in total. Clockwise from top-left: side view, front view, top-down view.

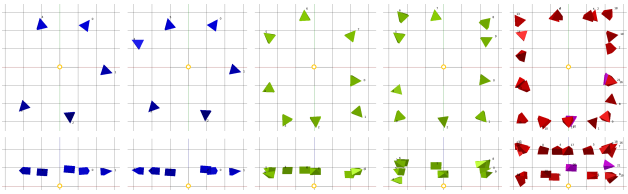

Figure 2. The layout of the different camera systems used. Each column is a top-down view (top), and a side-view of the cameras (bottom). Each system is assigned a colour: Kinect cameras are shown in blue, Sony 4K RGB in green and the Vicon cameras in red (the two Vicon witness cameras are shown in magenta). The world origin is denoted with a yellow circle and each grid is 1 metre in width/height/depth. From left: 5-Kinect setup, 6-Kinect setup, 8-Sony setup, 10-Sony setup, the Vicon setup.

number of cameras used per dog varied between eight to ten for the HD RGB cameras and five to six for the Kinects. Visualistion of this data can be seen in Figure 4. The frame count for each sequence of each dog is given in Table 1.

The number of real Kinect RGB and depth images

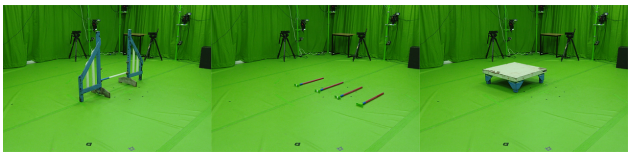

Figure 3. The props used during the acquisition of the dataset

Dogs	Average # Frames per Camera (Vicon,Kinect)					
	Walk	Trot	Jump	Poles	Table	Test
Dog1	(500,26)	(148,8)	(148,7)	(536,34)	(704,49)	(602,32)
Dog2	(300,39)	(118,7)	(220,19)	(374,50)	(330,24)	(624,52)
Dog3	(152,0)	(138,0)	(232,12)	(232,25)	(582,50)	(602,0)
Dog4	(322,0)	(290,0)	(132,0)	(642,0)	(390,0)	(602,0)
Dog5	(596,0)	(188,0)	(324,0)	(376,0)	(372,0)	(602,0)
Dog6	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(20,20)
Dog7	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(38,38)

Table 1. A table displaying the average number of frames per camera per motion. The first value in each pair refers to the frames from the Vicon system while the second value refers to the Kinect system. For each of the dogs in Dog1-Dog5, the same 5 motions are provided. A separate arbitrary test sequence is also provided if available. For the test dogs, Dog6 and Dog7, only a test sequence is provided.

recorded from all cameras for all five motions of the five dogs is 1,950. The number of 4K RGB images recorded from all cameras for all five motions of the five dogs is 73,748. In total, 8,346 frames of skeleton motion data were recorded using the Vicon Shogun software.

In comparison with other available datasets of keleton-annotated dog images, Biggs et al. [3] provide 20 landmarks for 218 frames from video sequences. The size of the images are either 1920x1080 or 1280x720 pixels. Cao et al. [4] provide 20 landmarks for 1,771 dogs in 1,398 images of various sizes.

2.2. Data Augmentation

Our synthetic dataset is generated from the result of applying the processed skeleton motion to the neutral mesh using linear blend skinning. The same 20 virtual cameras were used to generate synthetic images for all five dogs, along with cameras using the extrinsic parameters of the 8 to 10 Sony RGB cameras used to record each dog. For each motion, two sets of images were generated. In the first set, the root of the skeleton contains the rotation and translation of the dog in the scene, and the second set of images are generated where the root has fixed rotation and translation. Another version of the two sets was created by mirroring the images, giving our final synthetic dataset approximately 834,650 frames.

2.3. Network Architecture

We use the network of Newell et al. [10] based on the implementation provided by Yang [13]. We provide a diagram of the network in Figure 5 and direct the user to the paper by Newell et al. [10] for full details of the network components.

2.4. Data Normalisation for the Generation of Training Heatmaps

3D Joint locations of the skeletons are defined in camera space J_{3Dcam} and 2D joint locations, J_{2Dfull} are their projected values in the synthetic image. Only images where all joints in J_{2Dfull} are within the image bounds were included in the dataset. The images are shaped to fit the network inputs by following the steps outlined in Algorithm 1, producing images that are 256x256 pixels in size.

The bounding box of the transformed 256x256 image, and the bounding box of the original mask are used to calculate the scale and translation required to transform the dog in the 256x256 image back to its position in the original full-size RGBD image. J_{2Dfull} are also transformed using Algorithm 1, producing J_{2D256}. Finally, the z-component in J_{3Dcam} is added as the z-component in J_{2D256}, giving J_{3D256}. The x- and y- components of J_{3D256} lie in the range [0,255]. We transform the z-component to lie in the same range by using Algorithm 2. In Algorithm 2, we make two assumptions:

1. The root joint of the skeleton lies within a distance of 8 metres from the camera, the maximum distance detected by a Kinect v2 [1]

2. Following Sun et al. [12], the remainder of the joints are defined as offsets from the root joint, normalised to lie within $\pm$ two metres. This is to allow the algorithm to scale to large animals such as horses, etc.

Algorithm 1: Transform RGBD image for network input

1 Calculate dog bounding box from binary mask;
2 Apply mask to RGBD image;
3 Crop the image to the bounding box;
4 Make the image square by adding rows/columns in a symmetric fashion;
5 Scale the image to be 256x256 pixels;
6 Add padding to the image bringing the size to 293x293 pixels and rescale the image to 256x256 pixels;

2.5. Pose Prior Model

We use a Hierarchical Gaussian Process Latent Variable Model (H-GPLVM) [7] to represent high-dimensional

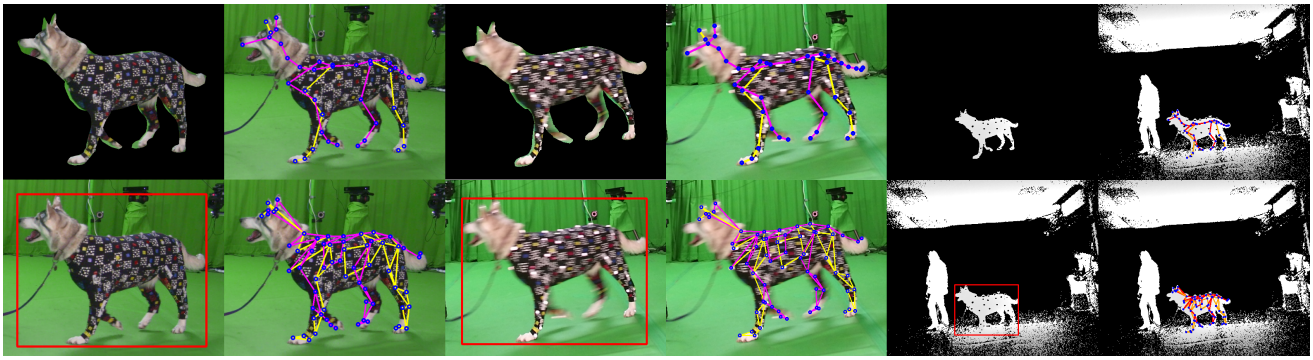

Figure 4. Image data included in this dataset is, from left, 4K RGB frames, 2K RGB frames from a Microsoft Kinect, and the depth information from a Kinect. Here, the RGB footage is cropped near to the dog bounding box, and the depth image is shown as the full frame. Clockwise from the top left image of each format, we show the image where the silhouette mask has been applied, the projected skeleton of the dog, the projected marker positions of the dogs with connecting lines for ease of identification, and the dog bounding box. For the projection of skeleton and markers in RGB images, yellow denotes limbs on the left side of the body and magenta on the right. For depth images, these colours are orange and red respectively.

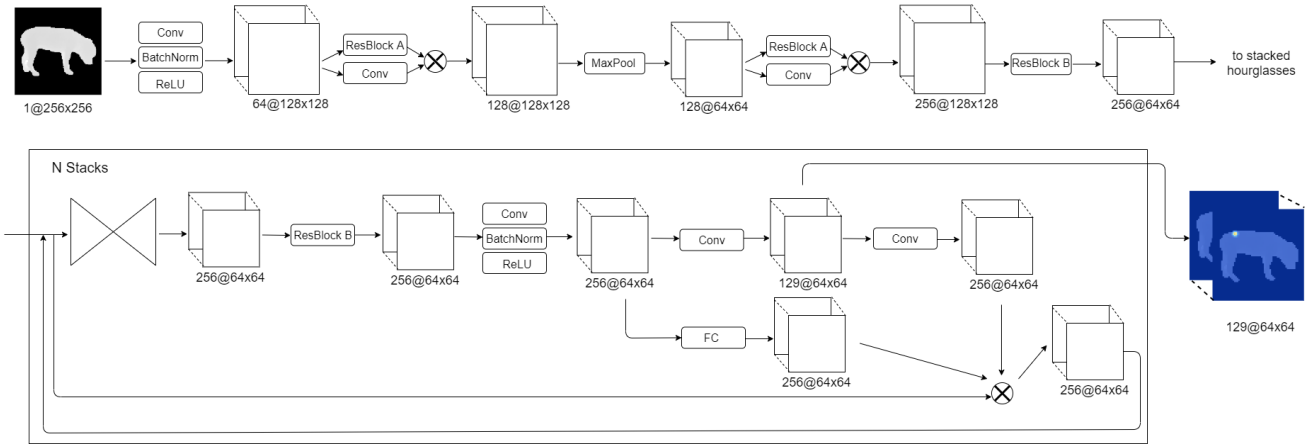

Figure 5. We use the stacked-hourglass network of Newell et al. [10]. In our experiments a stack of two hourglasses is used."Conv" stands for convolution and "FC" for fully connected. For full details on the network implemented, we direct the user to the paper of Newell et al. [10].

Algorithm 2: Normalising joint depth for network input

$rootJoint = J_{3D256}[0]$;
for $j \in J_{3D256}$ **do**
  if $j == rootJoint$ **then**
    $rootJointDepth = j[3]$;
    $j[3] = (min(j[3], 8000)/8000) * 255$;
  else
    $j[3] = (j[3] - rootJointDepth)/2000$;
    $j[3] = max(min(j[3], 1), -1)$;
    $j[3] = (j[3] * (255/2)) + (255/2)$;

skeleton motions lying in a lower-dimensional latent space. Figure 6 shows how the structure of the H-GPLVM relates to the structure of the dog skeleton: The latent variable representing the fully body controls the tail, legs, spine, and head variables, while the four legs are further *decomposed* into individual limbs. Equation 1 shows the corresponding joint distribution.

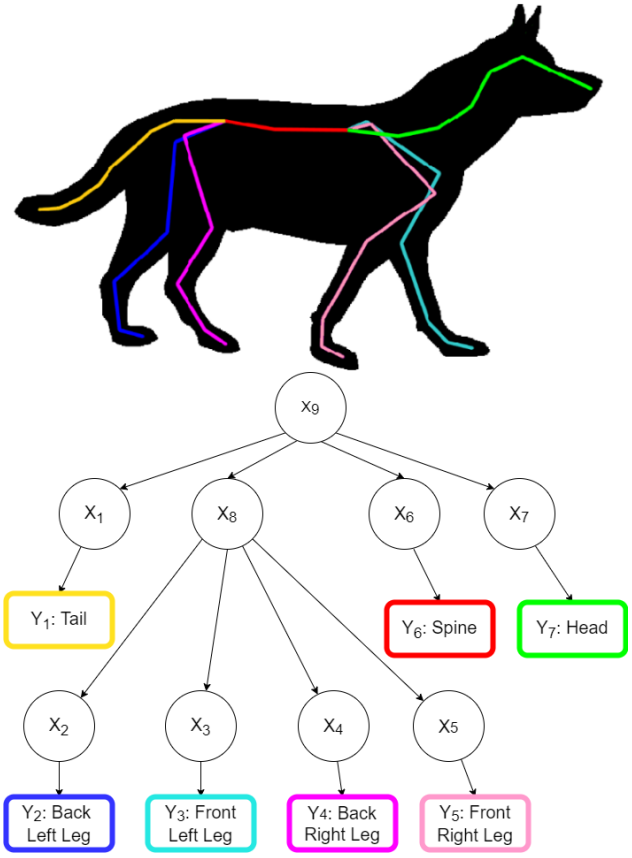

Figure 6. The structure of our H-GPLVM. Each node X_i produces joint rotations (and translation, if applicable) Y_i for the bones with the corresponding colour.

$$
\begin{aligned}
p(Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7) = \\
&\int P(Y_1|X_1)\dots \\
&\times \int p(Y_2|X_2)\dots \\
&\times \int p(Y_3|X_3)\dots \\
&\times \int p(Y_4|X_4)\dots \\
&\times \int p(Y_5|X_5)\dots \\
&\times \int p(Y_6|X_6)\dots \\
&\times \int p(Y_7|X_7)\dots \\
&\times \int p(X_2, X_3, X_4, X_5|X_8)\dots \\
&\times \int p(X_1, X_8, X_6, X_7|X_9) dX_9 \dots dX_1, \quad (1)
\end{aligned}
$$

where Y_1 to Y_7 are the rotations (and translations, if applicable) of the joints in the tail, back left leg, front left leg, back right leg, front left leg, spine and head respectively and X_1 to X_7 are the nodes in the model for each respective body part, X_8 is the node for all four legs, and X_9 is the root node.

Let Y be the motion data matrix of f frames and dimension d, $\mathbb{R}^{f \times d}$, containing the data of Y_1 to Y_7. K_{x_i} is the radial basis function that depends on the q-dimensional latent variables X_i that correspond to Y_i. $[s_i, e_i]$ define the start and end index of columns in Y that contain the data for Y_i. N is the normal distribution. Then,

$$p(Y_i|X_i) = \prod_{j=s_i}^{e_i} N(Y_{i[:,j]}|0, K_{x_i}), \quad (2)$$

where $Y_{i[:,j]}$ denotes the j-th column of Y_i.

2.5.1 Joint-specific Weights When Fitting the Model

When fitting the H-GPLVM to the network-predicted joints, each of these joints has an associated weight to guide fitting. This is a elementwise-multiplication of two sets of weights, W_1 and W_2. W_1 is user-defined and inspired by the weights used by the Vicon software. Specifically, these are [5,5,5,0.8,0.5,0.8,1,1,1,0.8,0.5,0.8,1,1,1, 0.8,0.5,0.5,0.8,1,1,0.1,0,0.1,0,0.8,1,1,1,1,0.8,1,1,1,1,1,1,1,1, 1,1,1,1]. This has the effect of giving the root and spine the highest weight (5), the end of each limb has a higher weight (1) than the base of the limb (0.8). Each joint in the tail is given equal weights (1). As the ears were not included in the model, a weight of 0 was given to the ear tips, and 0.1 given to the base of the ears, in order to slightly influence head rotation.

Prior to the fitting stage, the shape and size of the dog skeleton has either been provided by the user or generated by the PCA shape model. The bone lengths L of this skeleton can be calculated. For the current frame, we calculate the length of the bones in the skeleton as predicted by the network, L_N. The deviation from L is then calculated as $abs(L - L_N)/L$. W_2 is calculated as the inverse of this deviation, capped to be within the range [0,1].

3. Evaluation and Results

3.1. Ground Truth for BADJA Comparison

In order to compare our results with BADJA [3], we need to calculate the ground truth joints positions of the SMAL skeleton, S_{SMAL}. Using WrapX [2], an off-the-shelf mesh registration software package, the neutral mesh of the SMAL model is registered to the neutral mesh of each of the 5 dogs N_{dog}, producing the mesh N_{SMAL}. We can then represent N_{SMAL} as barycentric coordinates of N_{dog}. Using these barycentric coordinates, given N_{dog} in a pose,

Dog	Method	Metric	All	Head	Body	Tail
Dog1	Ours	MPJPE	**9.430**	**12.788**	**8.810**	**8.006**
		PCK	**0.443**	0.210	**0.495**	**0.514**
	BADJA[3]	MPJPE	19.532	21.619	17.915	22.527
		PCK	0.196	**0.214**	0.225	0.089
Dog2	Ours	MPJPE	**11.333**	**9.098**	10.703	**15.536**
		PCK	**0.424**	**0.645**	**0.448**	**0.128**
	BADJA[3]	MPJPE	12.154	13.163	**8.553**	22.458
		PCK	0.296	0.393	0.337	0.073
Dog3	Ours	MPJPE	**9.063**	**9.152**	**8.400**	11.044
		PCK	**0.450**	**0.354**	**0.492**	**0.415**
	BADJA[3]	MPJPE	10.839	15.203	10.597	**7.235**
		PCK	0.392	0.276	0.430	0.387
Dog4	Ours	MPJPE	**11.757**	**12.968**	**11.700**	**10.723**
		PCK	**0.296**	0.269	**0.354**	0.142
	BADJA[3]	MPJPE	24.936	20.964	29.222	15.439
		PCK	0.168	**0.347**	0.105	**0.189**
Dog5	Ours	MPJPE	**14.561**	**14.414**	**10.523**	27.329
		PCK	**0.230**	**0.189**	**0.273**	0.136
	BADJA[3]	MPJPE	20.188	15.321	21.340	**21.436**
		PCK	0.168	0.184	0.169	**0.150**

Table 2. 2D error results comparing our pipeline and that used in BADJA [3] on each of the 5 dogs. Errors are reported relating to the full body or focussed body parts, as shown in Figure 6 of the main paper.

P_{dog}, we compute the corresponding P_{SMAL}. The BADJA joint regressor then produces S_{SMAL} from P_{SMAL}.

The renderer of BADJA [3] mirrors the projection of the predicted skeleton S_{BADJA}. This means that for the 2D result, the identity of joints on the left side of S_{BADJA} are swapped with their corresponding paired joints on the right. For 3D comparison, we mirror S_{SMAL} with respect to the camera. Next, we find the scales, sc_{SMAL} and sc_{BADJA}, such that when applied to S_{SMAL} and S_{BADJA} respectively, the head length of both skeletons is 2 units. We apply these scales and also apply sc_{SMAL} to S_{GT}, the ground-truth skeleton that is in our configuration. Finally, our predicted skeleton S_{PRED} is scaled to have the same head length as S_{GT}.

3.2. Comparision to BADJA

We include the 2D results when comparing the results of our pipeline with that of Biggs et al. [3] in Table 2.

3.3. Applying the Pipeline to Real Kinect Footage

Running the network on real-world data involves the additional step of generating a mask of the dog from the input image. Two pre-trained networks are used to generate the mask: Mask R-CNN [6] and Deeplab [5]. Both were trained on the COCO dataset [8] and implemented in Tensorflow. During testing, it was found that although Deeplab provided a more accurate mask than Mask R-CNN, it would at times fail to detect any dog in the image, both when the dog is wearing a motion capture suit and when not. It would also fail to reliably separate the dog from its handler. In our experiments, Mask R-CNN detected the dog in the vast majority of images, although the edge of the mask was not as accurate as that provided by Deeplab. Therefore, the image is first processed by Mask R-CNN and the bounding box produced is then used to initialise the input image to Deeplab where it is refined, if possible. A comparison of the masks is shown in Figure 7. A homography matrix is automatically generated from the Kinect which, when applied to the RGB mask, produces the mask for the depth image.

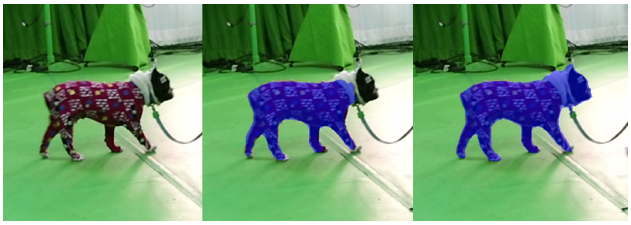

Figure 7. An example where Deeplab failed to detect a dog in the image (left), the mask as detected by Mask R-CNN (center) and the mask created by Deeplab initialised by the bounding box from Mask R-CNN (right).

Dog	Method	Metric	All	Head	Body	Tail
Dog6	CNN	MPJPE	14.754	**7.496**	10.099	36.559
		PCK	0.285	0.225	0.358	0.119
	H-GPLVM	MPJPE	13.996	12.239	10.475	26.757
		PCK	0.268	0.200	0.330	0.144
	H-GPLVM (known shape)	MPJPE	**6.375**	7.667	**7.764**	**0.743**
		PCK	**0.545**	**0.344**	**0.528**	**0.800**
Dog7	CNN	MPJPE	**8.758**	**6.461**	**5.811**	20.390
		PCK	**0.456**	**0.523**	0.552	0.089
	H-GPLVM	MPJPE	9.533	11.383	6.391	**17.501**
		PCK	0.426	0.138	**0.576**	**0.243**

Table 3. 2D Error results when using real Kinect images , showing the error result of the network prediction (CNN) and the final pipeline result (H-GPLVM). For Dog6, we also show the error where the shape of the dog mesh and skeleton is known when fitting the H-GPLVM.

Table 3 contains the 2D results when applying our pipeline to real Kinect footage.

3.4. Exploiting Depth Information to Solve Shape

In this section, different methods for fitting the shape will be described. In all cases, the shape is represented as model parameters to the PCA shape model. The results of each method are displayed in Table 4. Each entry in the "Method" column is described below.

In general, our pipeline method of solving for shape by referring to bone lengths over a sequence (Original) provided the best results. This has the effect of keeping the

shape constant for all frames. We compare the accuracy of the pipeline when the shape of the dog is allowed to change on a per-frame basis, during the H-GPLVM refinement stage (Method1).

Additionally, we compare the accuracy with our minimisation function takes into account the distance between the mesh produced by the model-generated skeleton to the reconstructed depth points. When fitting the model-generated skeleton to the network-predicted joints, we have a one-to-one correspondence as we know the identity of each joint predicted. This is not the case for the vertices on the generated mesh and the reconstructed depth points. Matches are made from the generated mesh to the Kinect points, and vice versa using Algorithm 3, where the angle threshold is set to 70 degrees, giving the two sets of matches m_1 and m_2. Two tests are performed: the first creates the matches only once during fitting the model (Method2), and the second repeats the matching stage after minimisation up to 3 times provided that the error between the two set of joints reduces by at least 5% (Method3). Finally, two tests were performed with mutual matches only, ie, the matches that appear in both m_1 and m_2. This match is performed only once (Method 4) or repeated up to 3 times provided that the error between the two set of joints reduces by at least 5% (Method5).

Algorithm 3: Creating matches from vertices in the source mesh to the target mesh

validMatch = [];
for $i = 0$ **to** $length(sourceMesh)$ **do**
    vertexLoc = sourceMesh[i];
    vertexNormal = sourceNormals[i];
    nearestMatchInTarget = $knnsearch$(vertexLoc, targetMesh);
    targetLoc = targetMesh[nearestMatchInTarget];
    targetNormal = targetNormals[nearestMatchInTarget];
    angDiff = $DifferenceInAngles$(vertexNormal, targetNormal);
    if *angDiff* $<$ *angleThreshold* **then**
        validMatch.$append$([i, nearestMatchInTarget]);

3.5. Robustness to Occlusions

The training data for the network is free from occlusions. To test the pipeline for robustness to occlusions, we apply a mask of a randomly located square. This square is 75 pixels in size which is approximately 30% of the image width/height. As expected, Table 5 shows that the results of the pipeline perform worse with the masked image as opposed to the original image. However, the H-GPLVM is able to reduce the error of the joint locations.

Dog	Method	Metric	All	Head	Body	Tail
Dog6	Original	MPJPE	0.667	**0.466**	0.627	0.993
		PCK	**0.873**	**0.969**	**0.938**	0.575
	Method1	MPJPE	0.727	0.538	0.671	1.094
		PCK	0.804	0.887	0.848	0.581
	Method2	MPJPE	**0.655**	0.527	**0.599**	**0.958**
		PCK	0.850	0.900	0.916	0.594
	Method3	MPJPE	0.704	0.516	0.675	0.985
		PCK	0.798	0.906	0.822	**0.613**
	Method4	MPJPE	0.666	0.480	0.619	1.000
		PCK	0.843	0.938	0.892	0.594
	Method5	MPJPE	0.721	0.523	0.689	1.019
		PCK	0.787	0.912	0.816	0.569
Dog7	Original	MPJPE	**0.557**	**0.494**	**0.471**	**0.888**
		PCK	**0.922**	**0.947**	**0.982**	**0.711**
	Method1	MPJPE	0.902	0.740	0.784	1.436
		PCK	0.706	0.803	0.778	0.385
	Method2	MPJPE	0.874	0.706	0.741	1.457
		PCK	0.725	0.819	0.806	0.378
	Method3	MPJPE	0.937	0.767	0.837	1.421
		PCK	0.655	0.763	0.704	0.395
	Method4	MPJPE	0.885	0.705	0.771	1.422
		PCK	0.716	0.809	0.411	0.783
	Method5	MPJPE	0.925	0.770	0.817	1.417
		PCK	0.673	0.780	0.723	0.408

Table 4. 3D error results as calculated using PA MPJPE and PA PCK 3D using the original pipeline and the various methods where dog shape can change on a per-frame basis. In general, the best result is achieved when the dog shape is based on bone length of the predicted skeleton and held constant throughout the sequence. A description of each method is provided in Section 3.4.

3.6. Comparison With Other Networks

We compare the network result of our pipeline, which uses the stacked-hourglass network of Newell et al. [10], with the networks of Sun et al. [12] and Moon et al. [9]. The networks were given the given the same training, validation and test data and trained for the same number of epochs. Tables 6 and 7 show that the network of Newell et al. [10] produced more accurate predictions in both 2D and 3D.

The method of Moon et al. [9] predicts 3D joint positions based on the voxel representation of the depth image. The author's pipeline first uses the DeepPrior++ network of Oberweger and Lepetit [11] to predict the location of a reference point based on the centre of mass of the voxels. This reference point used to define the other joints in the skeleton and is more feasible to predict than the root of the skeleton itself. Due to memory and time constraints, the training data

Dog	Method	Metric	All	Head	Body	Tail
Dog6	CNN	MPJPE	1.100	0.811	1.085	1.436
		PCK	0.584	**0.819**	0.554	0.444
	H-GPLVM	MPJPE	**1.005**	**0.746**	**1.027**	**1.193**
		PCK	**0.606**	0.794	**0.596**	**0.450**
	Original	MPJPE	0.667	0.466	0.627	0.993
		PCK	0.873	0.969	0.938	0.575
Dog7	CNN	MPJPE	0.814	**0.609**	0.760	1.189
		PCK	**0.769**	**0.868**	0.791	0.602
	H-GPLVM	MPJPE	**0.781**	0.673	**0.711**	**1.110**
		PCK	0.768	0.816	**0.801**	**0.618**
	Original	MPJPE	0.557	0.494	0.471	0.888
		PCK	0.922	0.947	0.982	0.711

Table 5. 3D Error results of PA MPJPE and PA PCK 3D when using real Kinect images that have been randomly masked, where each skeleton is scaled such that the head has length of two units. We give the errors of the two stages of the pipeline, showing that the H-GPVLM can improve the network result. The original errors of the pipeline are shown for ease of comparison and are not highlighted if more accurate results were achieved.

Dog	Network	Metric	All	Head	Body	Tail
Dog6	Newell et al.	MPJPE	**14.754**	**7.496**	**10.099**	36.559
		PCK	**0.285**	**0.225**	**0.358**	0.119
	Sun et al.	MPJPE	30.219	37.329	27.602	29.513
		PCK	0.078	0.050	0.076	0.119
	Moon et al.	MPJPE	16.791	14.148	14.779	**26.383**
		PCK	0.155	0.160	0.031	**0.192**
Dog7	Newell et al.	MPJPE	**8.758**	**6.461**	**5.811**	20.390
		PCK	**0.456**	**0.523**	**0.552**	0.089
	Sun et al.	MPJPE	11.904	10.381	7.870	26.412
		PCK	0.364	0.345	0.411	**0.243**
	Moon et al.	MPJPE	14.693	10.593	15.479	**17.358**
		PCK	0.239	0.321	0.245	0.115

Table 6. 2D MPJPE and PCK error results when using real Kinect images as produced by the networks of Newell et al. [10], Sun et al. [12] and Moon et al. [9]. In general, Newell et al. [10] performs best.

for this network contained the synthetic jump sequence of a single dog as seen by 28 cameras.

To test the result of this network, we calculate the mean euclidean distance from the reference point to the root of the ground-truth skeleton across all frames . We compare this to the distance from the center of mass of the voxels to the root. First we test the network on a single camera of a synthetic trot sequence of the training dog. The mean distance for the reference point was 302.64mm and mean distance for the center of mass was 302.55mm. Next we tested the network on two real Kinect sequences where again the reference point increased the error of the center of mass point by approximately 0.1mm. As a result, the center of mass was used as the reference point for each image when training the network of Moon et al. [9], rather than that predicted by DeepPrior++.

Dog	Network	Metric	All	Head	Body	Tail
Dog6	Newell et al.	MPJPE	**0.866**	**0.491**	**0.776**	1.523
		PCK	**0.745**	**0.956**	**0.780**	0.425
	Sun et al.	MPJPE	1.594	1.561	1.723	1.341
		PCK	0.279	0.300	0.340	0.250
	Moon et al.	MPJPE	0.896	0.879	0.912	**0.867**
		PCK	0.715	0.685	0.714	**0.756**
Dog7	Newell et al.	MPJPE	**0.563**	**0.364**	**0.507**	0.939
		PCK	**0.907**	**0.993**	**0.943**	0.707
	Sun et al.	MPJPE	0.889	0.698	0.810	1.372
		PCK	0.734	0.821	0.743	0.595
	Moon et al.	MPJPE	0.901	0.667	1.017	**0.832**
		PCK	0.715	0.834	0.649	**0.770**

Table 7. 3D Error results of PA MPJPE and PA PCK 3D when using real Kinect images, where the ground-truth skeleton is scaled such that the head has length of two units. We show the errors for the networks of Newell et al. [10], Sun et al. [12] and Moon et al. [9], with Newell et al performing best.

3.7. Comparison of Our Shape Model with SMAL

As the skeleton configuration of the two shape models are different, the SMAL model cannot be directly fit to network-predicted joints. Instead, to compare the models, we fit each model to the neutral dog mesh and skeleton of each dog in the set of Dog1-Dog5. For each dog, the average SMAL mesh is registered to the original dog mesh and the corresponding joint locations are calculated using the SMAL joint regressor. A different version of our shape model is created for each test where the information for the test dog is removed from the shape model.

We aim to find the shape parameters for each model that produces the mesh that most accurately represents each dog. As the scale of the SMAL model differs to the dog meshes, the overall scale of both models is also optimised in this process along with the shape parameters. For each model, we report the error result as the mean euclidean distance from each joint in the skeleton as produced by the model and the ground-truth joint in millimetres. We report the same error for each vertex in the meshes. These are shown in each row of Table 8. We perform tests where the models fit to only joint information (the first row of Table 8), fit to only vertex information (the second row) and both joint and vertex information (the third row).

This assumes that the pose of the model and that of the test dog are identical, which may not the case. As such, we then performed tests where the pose can change, i.e. we now solve for scale, shape parameters and pose parameters when fitting the model. The steps described above are repeated, and the results are reported in the final three rows of Table 8.

	Model Fit To	Errors - Ours		Errors - SMAL	
		joints	mesh	joints	mesh
Fixed pose	joints	45.458	26.819	**37.824**	**23.182**
	mesh	44.050	**69.923**	**23.221**	72.220
	joints & mesh	45.190	26.915	**37.636**	**23.242**
Solved pose	joints	**18.331**	**10.430**	23.582	17.925
	mesh	**7.225**	**25.649**	11.138	56.058
	joints & mesh	**17.255**	**10.1585**	22.175	14.689

Table 8. Given the corresponding configuration of ground-truth mesh and joint locations, for each dog in the set Dog1-Dog5, we find the global scale and model parameters of our shape model and the SMAL model that best fits to just the joint locations, just the mesh, or both the joints and mesh (rows 1-3). This test is repeated when finding the global scale, model parameters and skeleton pose parameters (rows 4-6). Errors are reported as the mean euclidean distance in millimetres for either each joint in the skeleton or each vertex in the mesh. SMAL achieves better results for a fixed pose, and our model achieved better results when the pose of the skeleton was allowed to change.

In general, the SMAL model achieved better results when the pose of the dog was fixed, whereas our model achieved better results when the pose was allowed to move. We believe this is due to each animal in the SMAL model having a similar neutral pose to each other whereas the neutral pose in our model is dog-specific.

References

[1] Kinect api overview. https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn782033(v=ieb.10), 2014. Accessed: 2019-08-07. 2

[2] Russian3dscanner. https://www.russian3dscanner.com/, 2019. Accessed: 2019-08-07. 4

[3] Benjamin Biggs, Thomas Roddick, Andrew Fitzgibbon, and Roberto Cipolla. Creatures great and smal: Recovering the shape and motion of animals from video. In *Asian Conference on Computer Vision*, pages 3–19. Springer, 2018. 2, 4, 5

[4] Jinkun Cao, Hongyang Tang, Hao-Shu Fang, Xiaoyong Shen, Cewu Lu, and Yu-Wing Tai. Cross-domain adaptation for animal pose estimation, 2019. 2

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 5

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 5

[7] Neil D Lawrence and Andrew J Moore. Hierarchical gaussian process latent variable models. In *Proceedings of the 24th international conference on Machine learning*, pages 481–488. ACM, 2007. 2

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 5

[9] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *Proceedings of the IEEE conference on computer vision and pattern Recognition*, pages 5079–5088, 2018. 6, 7

[10] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016. 2, 3, 6, 7

[11] Markus Oberweger and Vincent Lepetit. Deepprior++: Improving fast and accurate 3d hand pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 585–594, 2017. 6

[12] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 529–545, 2018. 2, 6, 7

[13] W. Yang. A pytorch toolkit for 2d human pose estimation. https://github.com/bearpaw/pytorch-pose, 2019. Accessed: 2019-08-07. 2

[14] Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 3d menagerie: Modeling the 3d shape and pose of animals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6365–6373, 2017. 1