Department of Computer Science

Pose estimation from depth images using convolutional neural networks

Ben James Crabbe

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering

September 2015 | CSMSC-15

Declaration:

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Ben James Crabbe, September 2015

1 Summary

In this work we developed a data driven system for extracting pose descriptors from depth images. This is designed is to work in conjunction with the pre-existing system of Paiement et al. [44] for assessing the quality of human movement. The overall goal of these works is to provide an in home system for gait and movement analysis which can be used to assist doctors in monitoring patient condition and rehabilitation in an automated and non-invasive manner.

We use a deep convolutional neural network to learn a mapping from depth images of humans to a 3 dimensional manifold representation of human pose. By doing this we are able to remove the dependency of the existing system on the generic pose extraction systems implemented in commodity depth sensors, allowing the system to be applied to a wider variety of motions in the future.

We have analysed the performance of our system on the SPHERE staircase 2014 dataset [44] and show we are able to measure the pose to within a mean distance of 0.0874 in that space. We show that the level of accuracy achieved is suitable for the intended application of movement quality analysis and will be able to run at close to 100 fps.

The main contributions of this work are:

- Applying convolution neural networks to a previously untested domain: regression of human pose from depth images.
- Combining convolutional neural networks with manifold learning methods which allows us to reduce the amount of training data required.

2 Acknowledgements

I would like to thank Adeline Paiement for her dedicated assistance and supervision throughout this project, particularly in her initial suggestion of using CNNs, for finding me the hardware and space with which to work, for her dedication in responding to my questions and problems. I would also like to thank Majid Mirmehdi for buying into the work and encouraging the prospect of a publication in future. I would like to thank Sion Hannuna for very kindly allowing me to use his computer for the duration of the project. And finally I would like to thank all the people in the Vision lab for letting me share their office and their company which has given me a fantastic taste of what its like to be a PhD student, and the inspiration to pursue that end in the future.

Contents

3 Introduction

Gait analysis plays an important part in the treatment and assessment of a number of medical conditions. At present gait analysis is usually performed through a combination of visual assessment by an experienced physiotherapist, automated methods such as marker based motion capture, pressure sensitive walkways or accelerometers. It generally involves travelling to a gait assessment laboratory and this can be an issue for patients who have difficulty walking.

This problem, and a range of other healthcare challenges, are being tackled through research and development by the SPHERE (a Sensor Platform for Healthcare in a Residential Environment) group in Bristol. An automatic, in home, gait analysis pipeline has been designed [44, 68] which assesses the quality of a subjects movement using data captured by the human pose estimation (skeleton/joint tracking) systems implemented by RGB-D cameras such as the Microsoft Kinect.

The system uses data driven, machine learning methods to learn both a low dimensional representation of pose[1] and models of normal motion from which it quantifies the quality of movement. Because of this, it can be applied naturally to other types of movement quality assessment such as sports movement optimisation or physiotherapy exercise coaching. The system has been applied to a sitting-standing motion, to punching motions in boxing [45] and

[1] The points in this space are referred to as the pose vector, and the Kinect skeleton data as the body configuration or joint position vector.

to people walking up stairs [44, 68].

One issue currently limiting the effectiveness of this system is the fragility of the skeleton tracking software. This software was designed for controlling entertainment/gaming systems with the user viewed frontally, within a range of 1-4m and at a pitch angle near 0°. Outside of these conditions skeletons become noisy and unreliable. Typically only a small fraction of data recorded from a camera fixed ∼1.5m above the top of stairs is fit for use with the system. Increasing the fraction of usable data requires more intrusive camera placement which is to be avoided. The skeleton tracker also performs poorly when props are involved, for example grasping a banister or a ball often leads to erroneous joint positions for that arm. It also struggles to accurately record sitting/standing motions which is a key motion hoping to be analysed in SPHERE homes.

The aim of this project was to develop a system for determining the low dimensional pose representation used in [44] directly from RGB-D footage, with the additional aims that the system should be naturally applicable to a range of motions and able to maintain reasonable accuracy at a range of viewing angles. The methodology we found most suited to this task was a convolutional neural network (CNN). CNN's are a supervised learning method for extracting features, e.g. the pose vector, from images. After training the network is able to generalise to unseen data, producing an output inferred from the examples it has seen. This method effectively learns the feature representations that produce the most accurate regression. The advantage of this over traditional hand crafted feature extraction is that when applied to new motions the type of features extracted will be re-tuned to better capture the pose vector from the data of the new motion.

One restriction of CNN's is that they require a large amount of data to be trained effectively. In this project, as in [44, 68] we have focused on an analysis of the stair ascent motion as this was the only existing dataset of the required size. We use the joint position data captured by the Kinect to produce the ground truth pose vectors used to train the CNN. The Kinect skeletons are not ideal ground truth; by using them we are restricted to only motions which the Kinect is able to capture. Also, their errors and imprecision produce an inconsistent mapping between similar images and labelled pose vectors which can confuse the network. Despite this we find that in a majority of cases the CNN is able to match the Kinect measurement to a high degree, and in some cases predict poses which correspond better to what is seen in

the images than was produced by the Kinect. The accuracy of the CNN is measured by the distance (euclidean/L2 norm) between its predicted point and the ground truth. This is used during training as a objective/loss function and in the analysis of the final performance. The effect of using the CNN's pose measurements for the movement quality analysis of [44, 68] is also examined. Comparing it to that of the ground truth we find that in some cases the network predictions are actually more adept at producing the correct abnormality than the Kinect skeletons.

In Section 4 we present an introduction to depth imaging, human pose estimation and convolutional neural networks. In Section 5 we detail the steps used to achieve our results. In Section 6 we present and analyse our results. In Section 7 we evaluate the suitability of our chosen method and present suggested directions for future work.

4 Background and Related Work

4.1 Depth Imaging

In depth images each pixel value represents the distance of that point from the camera. Depth images are unaffected by changes in lighting or human appearance. They provide a 3D map of the scene, making background-foreground separation far easier. These features can often simplify computer vision tasks, particularly human pose estimation [10].

There are three main technologies used to produce depth images: Time of flight (ToF) cameras, Stereo imaging cameras and Structured light cameras. It was not until the last 5 years that affordable good quality and easy to use depth sensors came on the market and since then there has been an explosion in their use in the computer vision community [24].

The data in the SPHERE staircase dataset was captured using a Asus Xtion Pro Live which uses the structured light technology developed by Primesense (same as the Microsoft Kinect). It consists of an infrared laser emitter and an infrared camera, which together make up the depth sensor, and a RGB camera. An infrared laser is passed through a diffraction grating to produce a known pattern of dots that is projected onto the scene then reflected back and captured by the infrared camera. The measured pattern is compared to a reference pattern produced at a known distance of reflection, which has been stored during the calibration

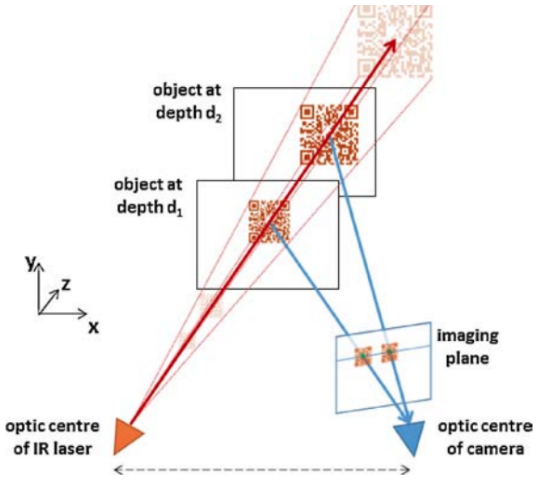


Figure 1: The process by which depth is computed from triangulation of structured light. From [24]

process. The surface of reflection being farther or nearer than the reference surface produces a shift in the pattern which is used to determine the depth value [32, 75] as shown in figure 1.

4.1.1 Sensor Performance

Most of the studies reported below have focused on the Microsoft Kinect however the Asus Xtion Pro uses the same depth sensing system. When the two were compared in [23] they were shown to perform equivalently, hence we report the findings based on Kinect performance.

The range of the depth sensor is 0.8-3.5m with increasingly noisy or incomplete readings up to 8m. It has a 43° vertical by 57° horizontal field of view [24] and captures 640 × 480 depth images.

Stoyanov et al. [65] compare the performance of the Kinect with that of two other ToF depth imaging cameras (SwissRanger SR-4000 and Fotonic B70 ToF) assessing them against a ground truth of expensive and low fps laser depth scanner measurements. They find that within a range of 3.5m the Kinect outperforms the ToF sensors and is comparable to the laser scanner, outside of this range the accuracy falls considerably.

Both Khoshelham & Elberink [32] and Smisek et al. [62] have measured this effect experimentally comparing Kinect measurements with those from high performance laser scanners. They find that temporally fluctuating noise in the depth measurements increases quadratically

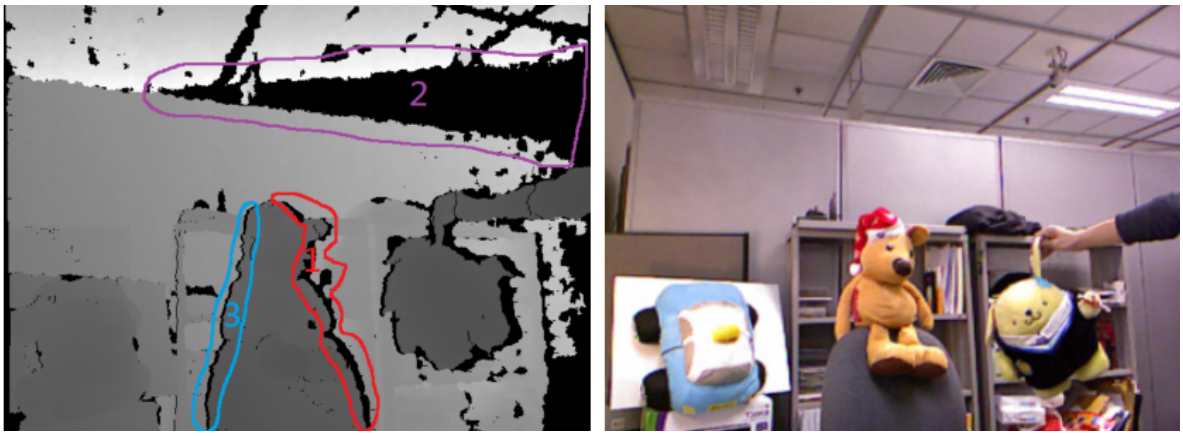


Figure 2: Shows the holes in structured light depth data due to the different perspectives of IR projector and senor (regions 1 and 3) and due to the surface of reflection being roughly 5m away and at a large angle(region 2) From [18]

with distance from the sensor so the depth precision decreases from around 0.5cm at 1m to 7cm at 7m. Nguyen et al. shows that noise increases linearly with lateral distance, and is greatly increased on surfaces at greater than 70° angles [42]. This last effect can lead to increased levels of noise around edges of humans.

As well as noise the structured light sensors often return 'unknown' depth value pixels, known as holes, when the infrared receiver cannot read the reflected pattern properly. This can occur around the sides of foreground objects due to the slightly different viewing angles between the projector and camera as in regions 2 and 3 of figure 2, or when certain surface materials, such as human hair, interfere with the infrared pattern's reflection as in region 4 in figure 3.

4.2 Human Pose Estimation

Human pose estimation (HPE) is generally considered as the task of measuring in 2D or 3D the joint positions of the human body. It is one of the most researched problems in computer vision because of its difficulty and due to its use in a variety of applications such as video surveillance, humancomputer interaction, digital entertainment, sport science and medical applications.

This is a difficult task for a number of reasons. Firstly the human body has around 20

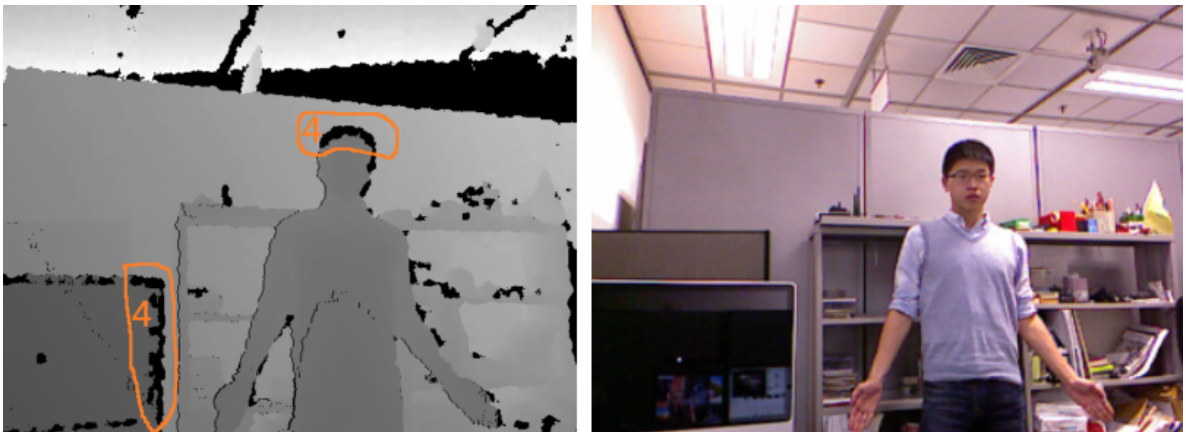


Figure 3: Shows the holes in depth data due abnormal reflections from certain glossy surfaces like the TV monitor and the subjects hair. From [18]

degrees of freedom [20], producing a huge space of possible body configurations, many of which will cause some joints to be occluded when viewed from a single camera. Additional difficulties arise from the variety in human appearance and clothing, and from left right ambiguities. Traditional motion capture methods (MoCap) rely on markers attached to the subject and multiple cameras to overcome these issues. Whilst such systems can provide highly accurate pose data, their use is restricted to controlled environments using expensive and calibrated recording equipment which renders them unsuitable in many applications.

Monocular visual pose estimation methods (reviewed in [26, 40, 41, 48, 61]) are generally divided into two approaches (e.g. by [48]); model based (or generative) and model-free (or discriminative) approaches. Model based approaches use prior knowledge of human shape and kinematics such as fixed limb lengths and defined joint angle limits to cast the image to pose transformation as a nonlinear optimisation problem or probabilistically in terms of a likelihood function, i.e. given this image data (and sometimes previous frames pose knowledge) what is the most likely valid pose. Model-free approaches instead learn a direct mapping from image data to pose, generally requiring learning/example based methods to achieve this. Some 'hybrid' approaches combine the two using model-free methods as an initialiser to model based methods.

4.2.1 Human Pose Estimation Using Dimensionality Reduction

With both of the above approaches there are significant issues posed by the high dimensionality of pose data. In model based approaches likelihood functions, which are usually multi-modal and non-Gaussian, require a randomised search [60]. Such searches in 20 dimensions are computationally expensive and often lead to super real time frame rates [26]. In model free approaches training data must account for the highly non-linear mapping between image and pose, which means that the pose space must be densely sampled in the training set. Densely sampling a 20 dimensional space, even just the parts that correspond to valid human motion, whilst also modelling all the invariant aspects such as body shapes, viewing angle etc requires an inordinate amount of data [1, 26].

Although the full pose space is very large and high dimensional it has been shown e.g. in [6, 16] that when considering only movements in a well defined activity such as walking, then the pose data can be well represented by a low dimensional latent manifold; this strategy was used in the SPHERE work [44]. In a work closely related to our own Elgammal et al. [16] use Local Linear Embedding (LLE), a dimensionality reduction method, to generate a 1D manifold representation (embedded within a 3D space), shown in figure 4, of a walking motion from single sequences of silhouette images. They use a Generalised Radial Basis Function interpolation framework [47] (a form of neural network) to learn two mappings, one from the manifold to the silhouette image space and another from the manifold to full 3D joint positions. They then invert these mappings to extract points on the manifold from silhouettes and 3D joint positions from the points on the manifold. In contrast, our work builds the manifold representation from 3D joint position data, this has the benefit of the manifold representation not being tied to subject's visual appearance. This allows it to generalise naturally to multiple subjects, which is not the case in [16] (although they do introduce an solution for this problem in [17]). It is also unlikely that this method could be used to capture abnormality in gait since defining an image to manifold transformation explicitly from the inverse constrains all input images to the poses contained in the original data. Elgammal et al. argue that learning a smooth mapping from examples is an ill-posed problem unless the mapping is constrained since the mapping will be undefined in other parts of the space. We address this issue in section 7.4.

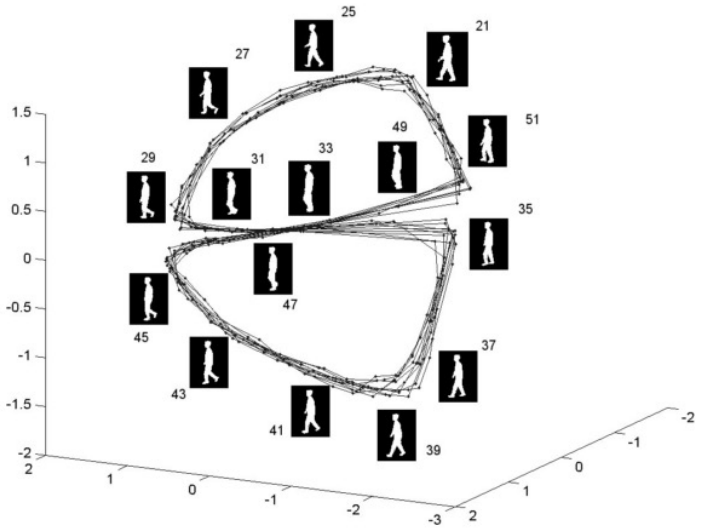


Figure 4: The 1D gait manifold produced from LLE dimensionality reduction on images of shilouttes used by Elgammal et al. to simplify joint tracking. From [16].

Brand [6], also inferred 3D pose from silhouettes using an intermediate manifold representation. He uses a maximum a posteriori estimation for mapping between the image and manifold space. This uses information across the whole input sequence to find the most likely and consistent solutions in order resolve the ambiguities in the many to many silhouette to pose mapping. A solution of this form is unacceptable in our case as one of the key features of the SPHERE system is online measurement.

Similarly Urtasan et al. [70] used Scaled Gaussian Process Latent Variable Models (SG-PLVM) [34] on single training sequences (walking and a golf swing) labelled with 2D joint positions. These models build a low dimensional manifold, and simultaneously, a continuous mapping between this and the 2D joint positions. They use the low dimensional representation to facilitate efficient maximum a posteriori based tracking in this space. Again, this is only suitable as an offline solution.

Tangkuampien and Suter [67] used Kernel Principle Component Analysis (KPCA) to learn low dimensional representations of 3D joint positions, and separately using the same method, a low dimensional representation of silhouette images. They then learn a mapping between these spaces using Locally Linear Embedding (LLE) reconstruction.

Rosales et al [49, 50] used 3D joint position data from a MoCap system to render synthetic

training data from multiple angles. Hu moments were used to extract visual features from these (and real) images. Unsupervised learning was used to cluster 3D joint position data into areas of similar pose and a neural network was trained separately on each cluster to learn the mapping from visual features to pose. Using the developments in training of deep neural networks since this work we show that it is feasible to have a single CNN learn both the features best suited and the mapping to all poses. This removes the need for clustering and separate networks leading to a simpler, easily adaptable solution. Their use of MoCap data as opposed to Kinect Skeletons for ground truth is a change we could expect to improve performance in our system. Similarly rendering synthetic training data from multiple angles would be a smart way of improving our viewing angle tolerance.

4.2.2 Human Pose Estimation From Depth Images

Challenging aspects of RGB HPE such as the variability in human appearance and scene lighting were greatly simplified with the advent of low cost commodity depth sensors. RGB-D also provides richer data for inferring 3D structure, allowing human poses which could appear identical when projected onto a 2D image plane to be distinguished. Full body HPE methods from single depth images are reviewed in [25]. With the Kinect/Xtion sensor and software packages (Kinect SDK or alternatively the open source OpenNI) low cost, flexible and reasonably accurate HPE is now available and has been employed in a huge variety of scientific applications [21, 24].

The Kinect SDK and OpenNI skeleton trackers apply some inter-frame tracking algorithms to the single frame pose measurements of [57]. In this work Shotton et al. leveraged a large MoCap 3D joint position dataset which they re-targeted onto a variety of synthetic body models before rendering as if captured from a Kinect, simulating sensor noise, camera pose, and crop position. Producing synthetic depth images data is far simpler than in RGB since depth is far more invariant to subject clothing and appearance changes. Using these generated depth images and a ground truth labelling of each pixel as one of 31 body parts they trained a randomised regression forest to perform this body part classification at each pixel using simple and computationally efficient pixel wise depth comparison features

$$f(\boldsymbol{u} \mid \boldsymbol{\phi}) = z(\boldsymbol{u} + \frac{\boldsymbol{\delta}_1}{z(\boldsymbol{u})}) - z(\boldsymbol{u} + \frac{\boldsymbol{\delta}_2}{z(\boldsymbol{u})}) \tag{1}$$

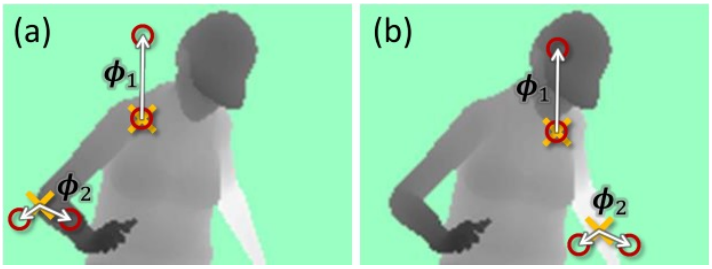


Figure 5: An example the depth comparison features from equation 1 used in [57] to perform per pixel body part classification. The yellow crosses indicate the image pixel $\boldsymbol{u}$ being classified. The red circles indicate the offset pixels as defined in equation 1. They use random forests which combine many such features to give a strong discriminative signal. From [57]

where $\boldsymbol{u}$ is a pixel location, $\boldsymbol{\phi}$ is two randomly generated 2D offsets $\boldsymbol{\delta}_1$, $\boldsymbol{\delta}_2$, $z(x, y)$ is a function which returns the depth at the input location. Examples of these features are shown in figure 5.

They then use these classifications to infer actual joint position through a simple averaging and mean shift procedure. The whole algorithm operates in real time on the computational resources allowed to them on the Xbox gaming consoles GPU. [58] adapts this work by allowing pixel classifications of a number of surrounding joints to be used when estimating the joint position, rather than the single corresponding body part pixels as in [57]. This is shown to improve the quality of the prediction for occluded joints. A similar use of MoCap joint position data for rendering synthetic images from multiple views we suggest as an ideal way for increasing the view angle and subject invariance of our system.

Other methods e.g. [**?**, 3, 73, 76] have focused on improving temporal smoothness of the measured pose by combining such discriminative methods with model based temporal tracking methods.

In a recent attempt Chan et al [8] use 3D point cloud information and propose a viewpoint and shape histogram feature based off these point clouds. This feature is then used to categorise the pose based on the action being performed using an introduced action mixed model. Each action is prescribed its own low dimensional manifold which allows a human pose database containing a limited amount of data to probabilistically infer the full pose.

4.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are biologically inspired supervised learning systems for extracting features from locally correlated inputs such as images or sound. Essentially their goal is to learn an approximation of the function

$$\boldsymbol{Y} = F(\boldsymbol{Z}, \boldsymbol{W}) \tag{2}$$

where $\boldsymbol{Z}$ is an input vector, the output $\boldsymbol{Y}$ is the extracted feature, in our case the inferred pose vector of an image, and $\boldsymbol{W}$ are the trainable parameters of the network.

They consist of a set of filter maps, essentially matrices (or tensors if applying to multiple channels), which are applied repeatedly (convolved) across the whole of the input. Each application of these filters produces an activation value which is the sum of each filter element multiplied by its corresponding input pixel value plus a shift term known as a bias. The filter is applied iteratively across the whole image, typically with some overlap. This builds up an 'activation map' for that filter, with each layer being comprised of a number of filters. The activations of each filter are stacked together, becoming the new image which is passed onto the next layer. This process is illustrated in figure 6. This type of layer, known as Convolution layers, are typically followed by a non-linear function which is what enables the CNN to learn non-linear transformations such as the image to pose transformation we require. Although it is possible to stack convolutional layers on top of each other they are often followed by a pooling layer (also called subsampling). The idea of pooling is to reduce the spatial size from the previous layer as seen in figure 7. Operating on each depth slice individually, i.e. each filter's activation map, the pooling window moves across the image taking the values of the elements in the input, conglomerating them using some operation, typically taking the max value as seen in the right hand side of figure 7. CNNs also typically contain one or more fully connected layers at the end. Being fully connected means that rather than having a small filter applied repeatedly across the input, a number of filters of the same dimensions as the input are applied to the whole volume. This then outputs a $1 \times 1 \times K$ volume where K is the number of filters. This is then identical to regular neural networks where each unit in a layer is connected to every unit in the next. In our case, fully connected layers take the final high level feature representation from the rest of the network and perform the final regression to the pose vector.

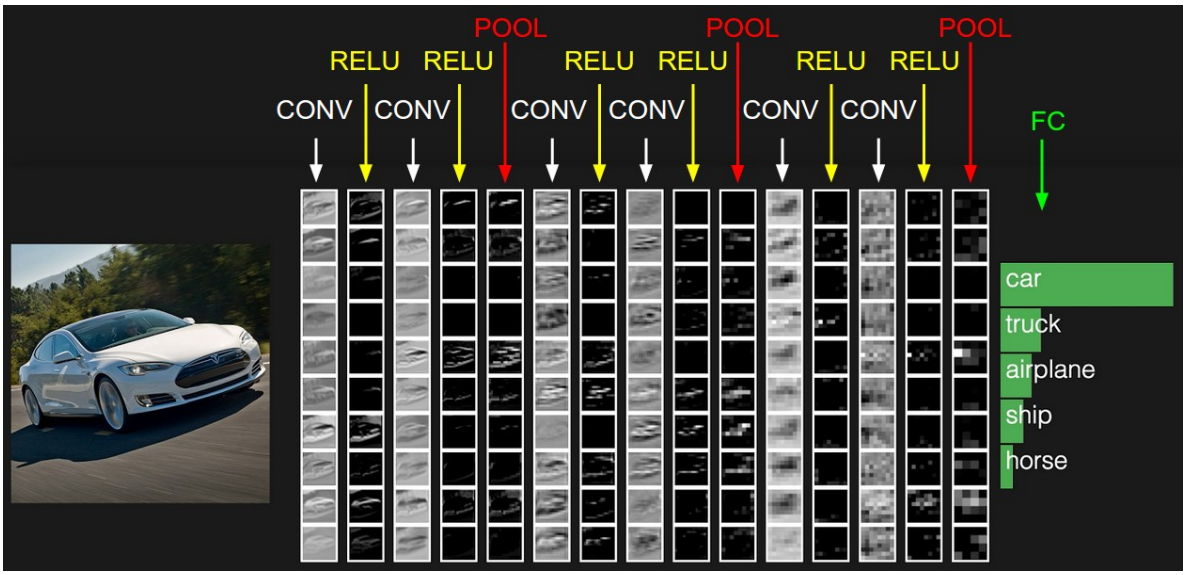


Figure 6: Shows a representation of the activations produced following a number of convolution layers, non-linearities (Rectified Linear Units, or ReLUs, are the function $max(0, x)$) and pooling layers. The network, an online demo from [31], is classifying images from the CIFAR-10 dataset using the ConvnetJS library[2].

CNNs are trained using backpropagation and gradient descent. During training an error function is defined which quantifies the difference between the networks output and the desired output. In our case, as is typical of regression tasks, we use the squared euclidean distance between the two. Using the backpropagation algorithm [51] the derivative of this error with respect to each parameter is found. Then the values of each parameter are adjusted a small amount in the direction which reduces the error. In this way, over many training examples, the network converges on a minima in the error surface across the space of all parameter values. A derivation of the backpropagation equations and extensive technical discussion of CNNs can be found in the preceding work on this project [13].

An advantage of CNNs over traditional computer vision methods is that rather than prescribing a hand-engineered feature such as the depth disparity feature in [57] or the view point and shape feature of [8], the network is responsible for learning features itself based off the data. This presents a significant advantage in our application since features which might be good for measuring pose for one type of motion may not be useful for other motions.

[2]http://cs.stanford.edu/people/karpathy/convnetjs/

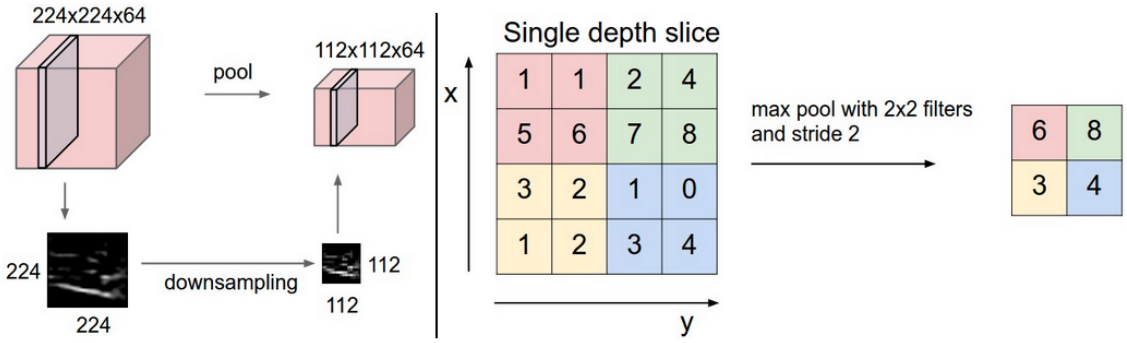


Figure 7: Shows the effect of pooling layers in a CNN. From [31]

Deep CNNs have a proven capacity to learn a huge number of different visual representations. They have achieved great success in the International Large Scale Visual Recognition Challenge (ILSVRC, or ImageNet challenge) which is conducted each year and requires identification and localisation (separately) of 1000 different types of object. The work of [33] showed how deep networks could be trained by using ReLU non-linearities in place of the previous staple of tanh or logisitic functions, this over came the vanishing gradient problem that had previously made large network infeasible. Since then CNNs have achieved the best results in each task every year running, with every single participating team using them by 2014 [52].

Training deep networks requires a large amount of data. Without this the network may begin to over fit the training data. When this occurs the errors for the network on inputs not in the training set, known as the test set, increases. It can be shown [55, 71] that the difference between the errors on test set and those on the training set is related to both the size of the training set, P, and the capacity, c, of the network i.e.

$$E_{test} - E_{training} \propto \frac{c}{P} \tag{3}$$

The capacity of the network is essentially the number of parameters being trained, which depends on the number of layers, the dimensions of the input images and the number of channels, the size and number of convolving filters and the way the filters and the pooling are applied. Increasing capacity will also decrease the size of $\boldsymbol{E}_{training}$, hence there the aim in designing the network is to find the architecture which minimises both $\boldsymbol{E}_{training}$ and $\boldsymbol{E}_{test} - \boldsymbol{E}_{training}$ as far as possible [36].

Since ImageNet contains around 15 million training examples it has enabled the training

of very large networks such as the 2014 winner GoogLeNet which had 22 layers [66]. In our case we are limited by the size of our dataset which contains a total of 6228 usable examples, many of which are practically identical due to being adjacent frames. A common method used to apply deep CNNs to tasks which lack large amounts of training data is to use a network pre-trained on a large dataset, typically ImageNet. Numerous studies [14, 22, 43, 56, 74] have shown ImageNet trained networks producing better results than randomly initialised networks in a variety of tasks, and also on depth data [2, 53]. Typically the filters in the early layers are generic edge, corner or blob detectors which can be expected to generalise well to different tasks. In higher layers, as the spatial size of the input to each filter increases (due to pooling), more task specific representations are learnt.

4.3.1 Human Pose Estimation Using CNNs

To the best of our knowledge

Jain et al. [28] were the first to apply CNNs to human pose estimation on RGB images. They trained multiple small three-layer networks to act as joint detectors with a separate network for each joint. Each network takes a small window of the full image as input. They are applied repeatedly with some overlap to produce a probability map of the joint's estimated location. They then use a spatial model which enforces consistent and valid global pose estimates from the estimated joint locations. In [29] they extend their system to videos using RGB plus motion features e.g. optical flow.

In a similar work Toshev and Szegedy [69] used a multi resolution approach to find 2D joint locations from RGB images. They used the architecture of [33] first applied to the full image at a low resolution. They then apply another network to a higher resolution image of just the area around the locations of the joints as determined by the first network. This successively refines the predictions.

Li and Chan also measure 2D joint locations from RGB images with a CNN. They use a single network with three convolution layers which they train for a pose regression and a joint detection simultaneously, sharing all convolution filters between tasks. They show that training for the extraneous task of joint detection consistently improves the accuracy of the regression task. In [38] they extend their work to 3D joint position measurements. They also compare the original multi task training with pre-training on joint detection before fine-tuning

on regression. They find little difference between the two but still show that the joint detection task consistently improves the regression performance. They show that the network produces reasonable position estimates even for completely occluded joints.

Pfister et al. [46] used the architecture of [54] (a 2013 ImageNet winner) to regress 2D upper body joint locations from RGB video. They experiment with using an ImageNet pre-trained model but find results are improved when training from scratch, a result echoed in this work. They found their accuracy improved by 5% after performing background subtraction on their data, a method we also adopt. They also experiment with using multiple frames as input but find this gives only a modest 0.3% improvement.

Recently Belagiannis et al. [4] showed that convergence rates and final accuracy in HPE tasks can be improved by replacing the common L2/Euclidean loss function with a function that reduces the effect of outliers in the objective space. They state that the L2 loss functions give a disproportionally high weight to outliers which negatively effects the training procedure, reducing the generalisation ability and increasing the convergence times. They propose a new loss function, Tukey's biweight loss function, which acts to minimise the effect of outliers and show that employing this function improves results in better accuracy and convergence rates on 2D HPE task. This is an interesting finding which is especially relevant to this work where noisy skeletons produce a significant number of outliers. Unfortunately this paper was not discovered until after most of the work was completed. A direction of future work will be to implement this loss function.

5 Methods

5.1 Dataset

The dataset used in this project (SPHERE-staircase2014 dataset [44]) includes 48 sequences of 12 individuals walking up stairs, captured by a Asus Xtion depth sensor placed at the top of the stairs in a frontal and downward-looking position.

It contains three types of abnormal gaits typical of lower-extremity musculoskeletal conditions. These include: freezing of gait referred to as stop $\times n$ where n is the number of freezes. Using a leading leg, left or right, in going up the stairs, referred to as LL and RL. All frames

have been manually labelled as normal or abnormal by a qualified physiotherapist. There are 17 sequences of normal walking from 6 individuals and 31 sequences from the remaining 6 subjects with both normal and abnormal walking. The dataset contains a reasonable variation in body shape and appearance as can be seen in figure 8.

Each example consists of the RGB image, the depth image and 3D joint position / skeleton data. Preprocessing is applied to each of these separately.

5.2 Data Preprocessing

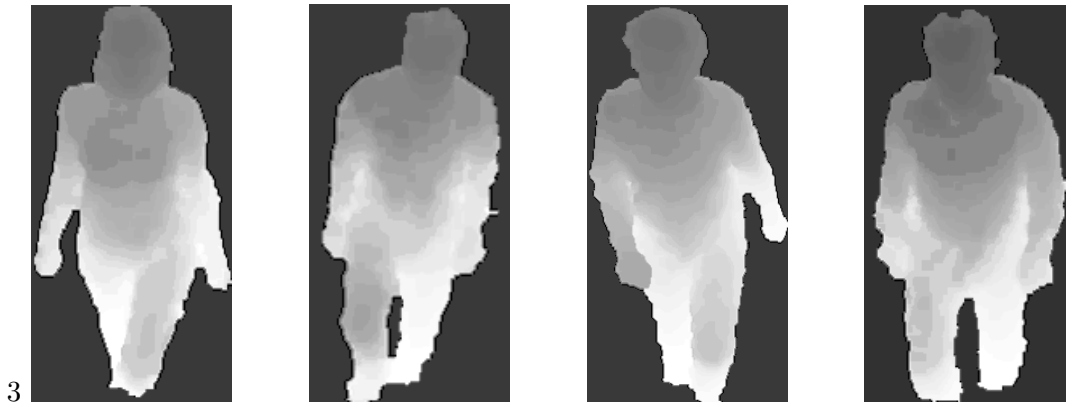
5.2.1 Skeleton Preprocessing

These processes were developed by the authors of [44] and are a pre-requisite for the manifold learning stage of that work and this. First the Skeleton data is smoothed over time to reduce the large amount of noise. Each skeleton is scaled to the same height, rotated to face forwards and translated to the origin. Each skeleton is compared with a neutral reference pose; a dissimilarity measure is computed using the sum of squared distances between corresponding joints, standardised by a measure of the scale of the reference shape. Any frames where this dissimilarity is greater than 0.1 are discarded. This threshold was determined to be as low as possible without risking removing valid but abnormal poses. Those that are removed are generally only frames from the beginning and end of sequences, where the subject is outside the sensors optimum range and skeletons become very erratic. We will show in Section 6 that despite this cut many bad quality skeletons remain in the training data.

Specifically for this project we also take the mirror image of each skeleton. Combined with a mirror image of the depth image, this effectively doubles our data which allows us to train larger capacity networks.

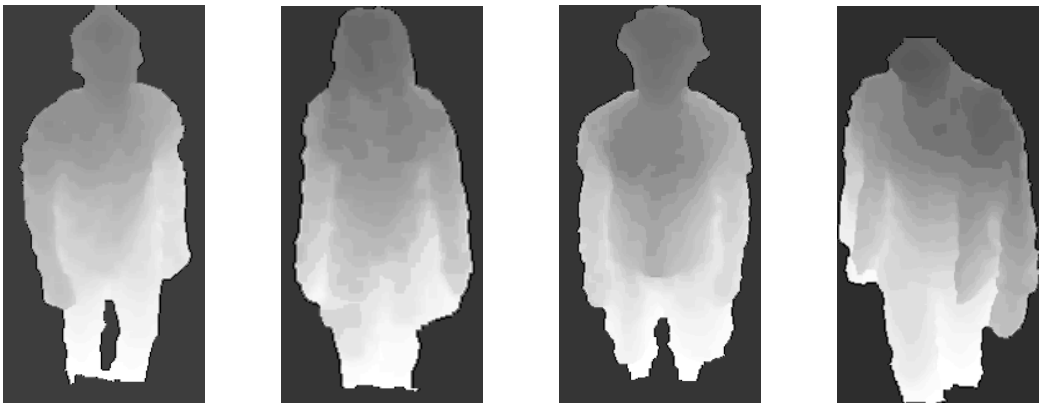
Now the collection of remaining skeletons is given to the manifold learning stage. The method used is Diffusion Maps [12] with an adaptation to better handle the remaining noise and outliers present in the skeleton data [44]. We refer to the original publications [44] and [68] for a full technical description. In essence Diffusion Maps retains the relative distances between data in the reduced space. [68] analysed the quality assessment performance using 1,2,3,4, and 5 dimensional representations and concludes that a 3D representation is the most effective.

The set of data that is used in building the manifold determines its form. In the original



(a) Subject 1. (b) Subject 2. (c) Subject 3. (d) Subject 4.

(e) Subject 5. (f) Subject 6. (g) Subject 7. (h) Subject 8.



(i) Subject 9. (j) Subject 10. (k) Subject 11. (l) Subject 12.

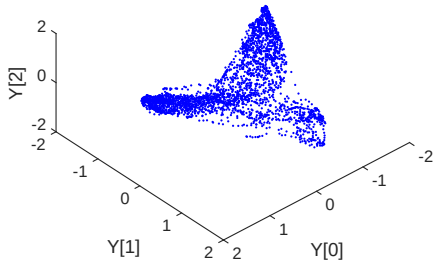
Figure 8: The subjects contained in the SPHERE staircase 2014 dataset. The depth images have been processed following the proceedure detailed in Section 5.2.2.

works [44] and [68] the authors used only the 'normal' sequences from the first 6 subjects. The manifold this produces is shown in figure 9a. We initially worked with a manifold built from all sequences abnormal and normal from each of the 12 subjects. This produces a slightly different manifold shown in figure 9c. However we found that the SPHERE normality software did not work correctly when trained with this manifold.

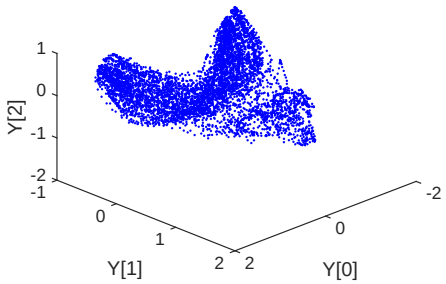
All results presented here after use the 2793 skeletons from the first 6 'normal' sequence subjects plus their horizontal flips to build the manifold, shown in figure 9b. The skeletons of the 31 sequences from the other 6 subjects (and their flips) are then projected onto the manifold (again, see [44] and [68] for details). Interestingly the flipped skeletons produce a manifold coordinate precisely equal to the non flipped version but with the first component negated. This suggests that each of the three components of the manifold are tied with the 3 regular spacial dimensions of the skeletons, although there is no guarantee of this with the Diffusion Maps method, particularly if applied to other movement types for which the left-right spatial dimension is less informative.

To illustrate the meaning of the manifold coordinates the skeletons placed at the 3 corners are shown in figure 10. Generally the normal gait sequences follow paths across the dense quarter sphere shaped surface with the frame of maximal left and right knee flex occurring at the maximum and minimum of first coordinate. The second dimension, plotted horizontally in figure 10, seems to measure the vertical extension between skeletons. The skeleton with raised/flexed knees occur near its minimal value and at its maximum we find elongated skeletons which are measured erroneously when subjects are very close to the camera but just survive the cut on skeleton dissimilarity. During processing of the images we remove these images from the dataset which results in the removal of most of the points which lie at this end of the manifold, as shown in figure 11.

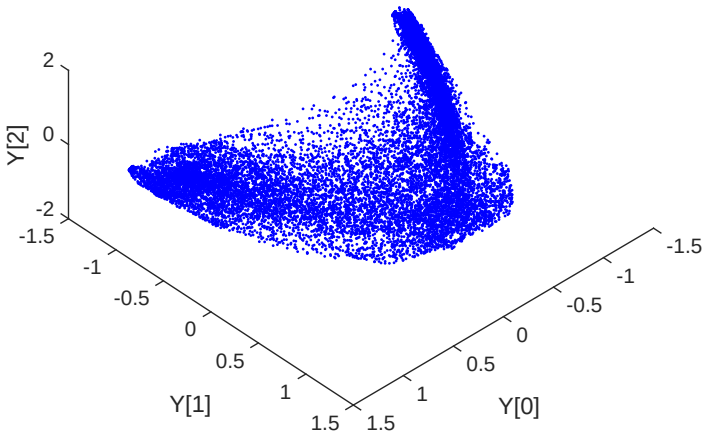
The meaning of the 3rd manifold coordinate is less clear. Figure 12 compares skeletons of minimum and maximum values for points of equal Y[0], Y[1], we are able to find no discernible pattern. This is reflected by the CNN as we find it far less accurate in this coordinate than the others. However this coordinate is somewhat subject specific, as seen in figure 13, with each subject found to traverse the manifold with a Y[2] range smaller than that of the full dataset. We find that fine-tuning our networks on the spare sequences of the subject being tested improves results by 28%.



(a) The manifold produced by normal sequences of subjects 1-6. This was the form used in [44]. We connect points from neighbouring frames for better visualisation of the structure.



(b) The manifold produced by normal sequences of subjects 1-6 including their horizontal flips. This was the form used in this work.



(c) The manifold produced by all sequences including their flips. This manifold was found to produce incorrect normality analysis.

Figure 9: Shows the pose manifolds used in this work.

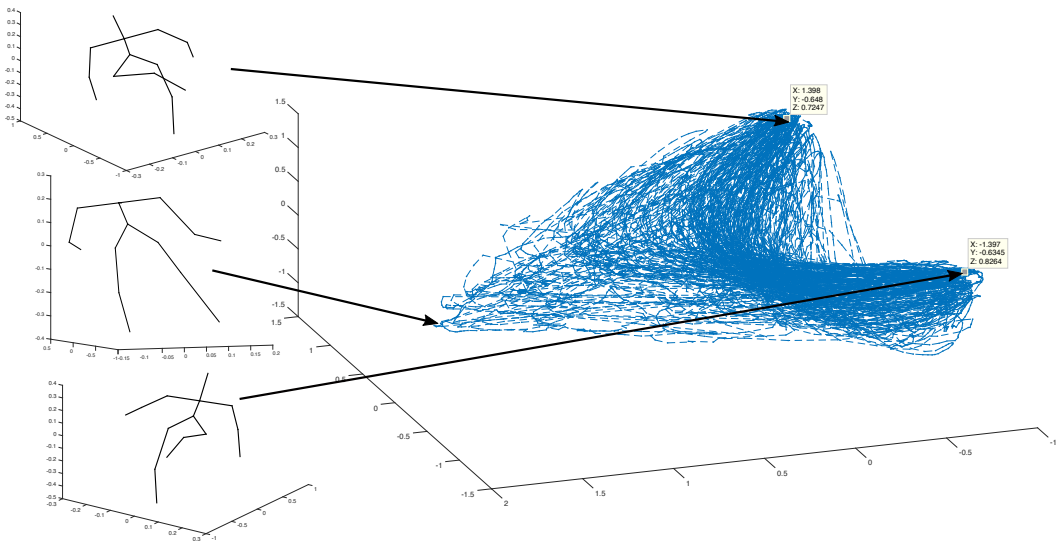


Figure 10: Shows the skeleton to manifold mapping. Normal gait sequences trace paths between the two corners on the right, with the position of maximal knee flex the turning point. The left most corner of the manifold consists of elongated skeletons which tend to be measured when the subject is too close to the sensor.

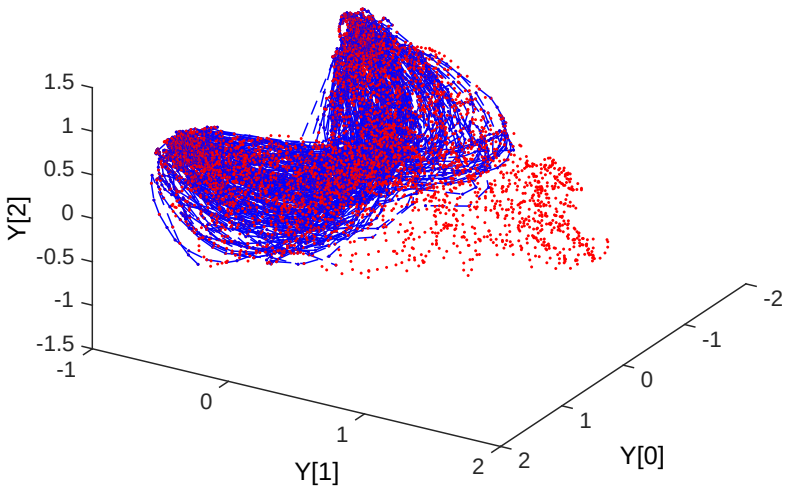
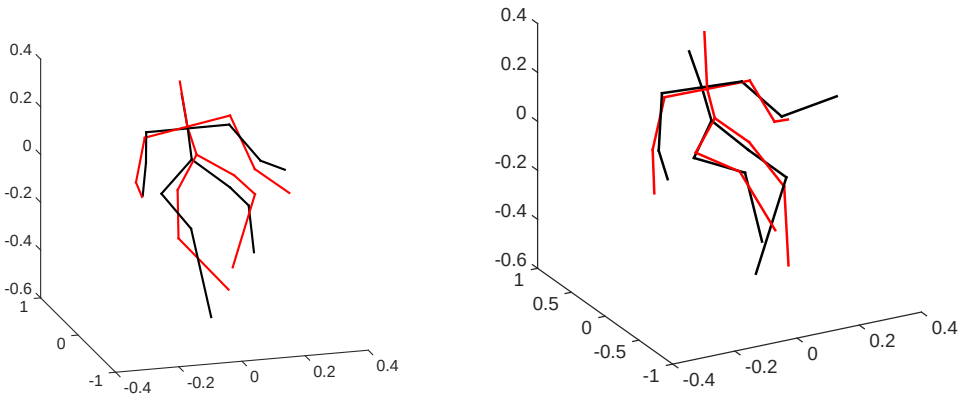


Figure 11: Shows all manifold points in red. After image pre-processing we retain only those in the blue area.



(a) Difference of 0.4099

(b) Difference of 0.503.

Figure 12: Shows skeletons within 0.02 of the same Y[0],Y[1] position but with maximum difference in Y[2]. Red skeletons are minimum Y[2], black are maximum. We find no clear difference between the Skeleton of minimum and maximum Y[2].

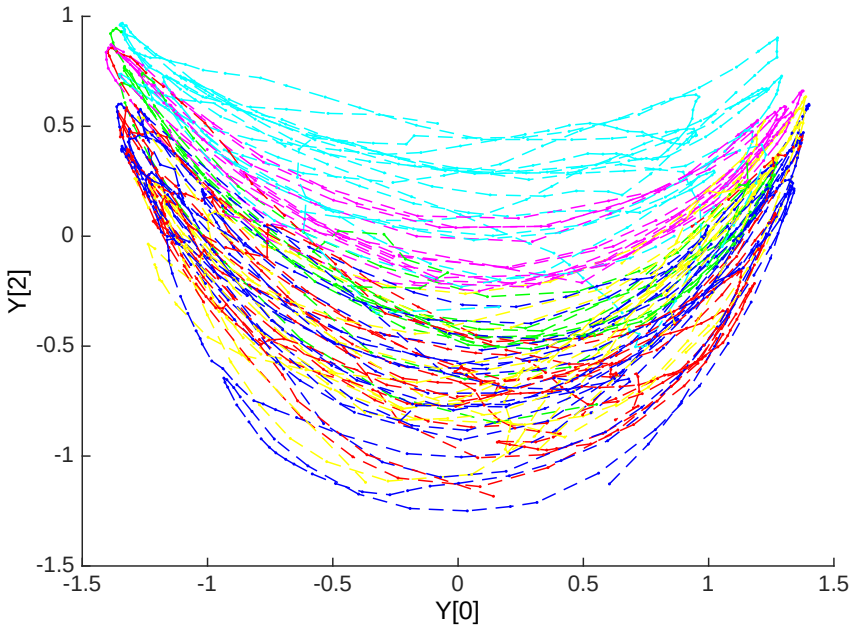


Figure 13: Each normal sequence of the first 6 subjects plotted in a different colour. We observe that Y[2] coordinates for each subject tend to remain within a sub-region of full range. This finding promoted the use of subject fine-tuning which improves the accuracy of Y[2] predictions. This effect is analysed in Section 6.1

(a) Typical image before hole filling.

(b) Filtered and filled using a simplifed version of [7].

(c) Filled using max fill method.

Figure 14: A comparison of hole filling methods. We use the max fill method.

5.2.2 Depth Preprocessing

As described in section 4.1.1 depth images are generally incomplete and noisy. We first fill holes in the data using a simple method which iteratively paints in the maximum pixel values from the neighbouring cells. We initially experimented with using the method of [7] which combines noise filtering and hole filling. However the version that we accessed did not use the motion detection and segmentation components. In practise we found that the quality of the filled and filtered depth maps produced by this version were inferior to those produced by the simple method. [7]'s method also required far longer processing times, 1 week for the full set as opposed to a 20 minutes for the simple method.

We then perform background subtraction on the filled depth images. We employed the C++ BGS library [63] and tested each of the provided methods on our data. For all methods

some initial set of background frames at the beginning of the sequence were required to achieve satisfactory results. The best results were achieved using the dp/DPZivkovicAGMMBGS method which implements [77].

We found that the foreground masks produced often mistook sensor noise for moving objects. To fix this problem we apply a simple cleaning procedure to the masks which takes only the largest collection of positive pixels. This discards the small noise related blobs. We also apply a small erosion and dilation to get rid of noise effects connected to the largest blob. There are also problems correctly identifying the feet of the subjects since the floor is at approximately equal depth and therefore the same colour in the depth image. The masks generally only extend to around the mid point of the shin. For some subjects feet can appear mid stride when raised high enough from the ground to be distinguishable. We also attempted using the RGB images to improve extraction of the feet. However most subjects are wearing dark coloured shoes which did not differentiate well from the colour of the stairs, meaning the results were just as poor as for the depth, with additional issues in other parts of the images where clothing matched the colour of the walls. Additionally on some frames the subtraction from the depth images fails badly. Sometimes noise cause large sections of background to be included in the mask, to remove these we place a limit on the change in size of the mask between consecutive frames. Sometimes towards the end of sequences the algorithm [77] can think that the subject is actually the background causing a human shaped hole to appear in the centre of the mask. These and other miscellaneous errors were removed by hand. Unfortunately we do find that a small amount of bad images remain in the dataset which causes errors for the network, this is discussed in Section 6.2.

We then take the cleaned mask and extract the foreground containing the human figure from the depth image. In some cases patches of background get included around the edges of the mask. We remove any pixels which have values greater than two standard deviations from the mean of the foreground. Due to the perspective of the camera the depth values of the foreground increase towards the feet of the subject, therefore if this cut is too strict we begin to loose valid pixels near the feet. To avoid this whilst still removing all background pixels from the rest of the image we select only the top 75% of the figure and apply a more stringent cut.

Since all skeleton data was scaled and transformed to a common point we do the same

to the depth data. We normalise the mean depth values of each image by finding the mean depth value of a small region near the subjects waist. Using only this region gives an accurate indication of the distance of the subject from the camera rather than transient effects such as arm swing. This value is subtracted from all foreground pixels before all being scaled by 0.5.

Next we crop the background area from each image so that each is left with the average width to height ratio of masks in the full dataset which is 0.504.

We then try to normalise the position of the figure within the image. There are two issues with the data which make this a necessary step, firstly the length of lower leg included in the masks varies from frame to frame, secondly there are a number of sequences in which the head of the subject leaves the camera's field of view during the middle of the ascent. Without fixing this problem the scale of the subjects in the images changes depending on how much of them was captured in the mask, which would mean the network would be required to learn an invariance to scale. In general we developed these pre-processing procedures to reduce the amount of invariance the network is required to learn. To fix this we try and identify the position of the shoulders. The width of the mask as a function of height in the image was found across the full dataset, the point of greatest curvature was determined to lie at a mask width of 0.65 of the full image width (after initial cropping). The point at which the width of the mask exceeds this value is defined as the shoulders. From this we find the position of the shoulders in each image. The mean position of the shoulders in all non-headless images was found to be 0.22 of the full image height. Then each image was rescaled, adding additional pixels of background so that the position of the shoulders occurs at exactly 0.22. If the initial shoulder position was less than 0.22 then we assume the image is missing part of its head and rows are added to the top, if it is greater than 0.22 we assume it misses more leg than usual and rows are added to the bottom. Whilst this doesn't recreate the actual foreground pixels of the legs or head, it does mean that the foreground pixels for each intact body part do occupy roughly the same position in the images. One limitation of this method is that it is difficult to handle images which lack both head and legs. We attempt to deal with this by storing the mean number of rows added to the bottom of the image across the sequence. Then if we find on the next frame that we are missing rows from the top (i.e. part of the head is missing) then we continue to add the mean number of rows to the bottom.

Next we adjust the contrast of the image. Initially we used the basic scaled depth values

and a black background. Analysis of the activations produced by the first layer filters of an ImageNet pre-trained network on this type of input, shown in figure 16b, seemed to indicate that depth information was not being extracted effectively. ImageNet first layer filters, shown in figure 16a, generally respond to edges. It is clear that the very small differences in depth between parts of the body will produce far smaller activations than the huge difference between the figure and the black background. To encourage the network to focus on the smaller edges we tried setting the background of the images to the mean depth value around the waist of the subject, as shown in figure 15b. However after training until convergence the network seemed to have discarded most of the filters activations rather than adjusting to them as is seen in 16c. Finally we used histogram equalisation to increase the differences in depth values within the figure. We use a background value of 0.82 of the mean value at the subjects waist as this was slightly less than the minimum value of any valid foreground pixels seen in the dataset. This results in the largest spread possible in the foreground depth values without ever having a foreground value at a colour darker than the background. This scheme seems to achieve the desired result of allowing interior depth values and edges to be preserved through the first convolution layer as seen in figure 16.

Finally, we remove images from the beginning and end of sequences where the subject is rotating or is very close to the camera as these are generally inaccurate and also because there are so few of these frames that they represent rare outliers in the dataset.

A common pre-processing step when working with CNNs is to subtract the mean image of the training data (i.e. the mean value of each pixel) from each image. This is supposed to speed up training for reasons detailed in [5, 35, 37]. After testing the accuracy of the networks trained on such mean subtracted data we found that this operation actually decreased the performance as shown in figure 17. However this test was only conducted on an ImageNet pre-trained AlexNet (see section5.4), we have not studied the effect when training from scratch.

5.2.3 RGB Pre-processing

For RGB images we applied the foreground masks retrieved from the corresponding depth images and then applied clipping and scaling in the same manner as the depth images. The result is shown in 15d.

We measured the accuracy of an ImageNet pre-trained AlexNet (see section5.4), which

(a) Black background scheme. (b) Mean background method. (c) Final scheme. (d) Pre-processed RGB

Figure 15: Shows the input image colour schemes tested with the CNN. We found (c) produced the largest responses in the first layer filters of a pre-trained network, shown in figure 16

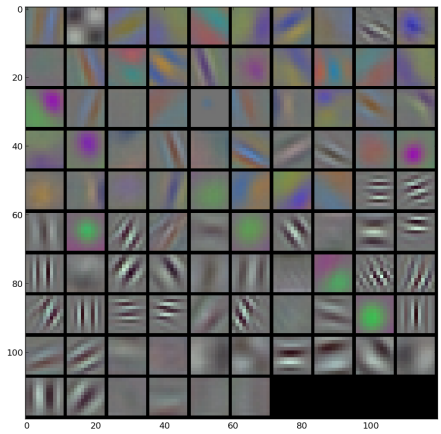
requires a 3 channelled input image, when using the RGB images only, the depth image replicated for each of the 3 channels only, and a combination of the two which consisted of 1 channel of the average of red and green, 1 channel of green and blue averaged and the depth in the final channel. As is shown in figure 18 the replicated depth was shown to produce the most accurate regression. All other results reported here use a pure depth input image.

5.3 CNN Library

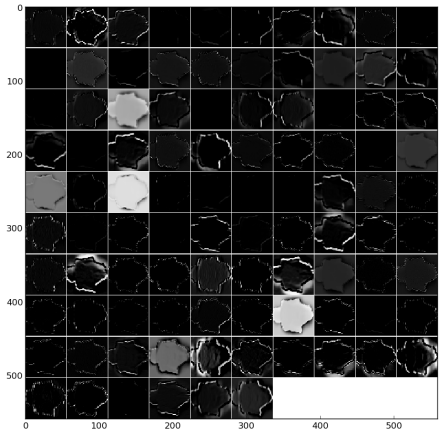
We use the open source CNN library Caffe [30] which is common among researchers including [11, 22, 46, 53, 59, 64, 72].

Caffe requires datasets to be in certain formats for training. When using multi-dimensional labels, as in our case, one is constrained to using a HDF5 formatted [19] dataset.

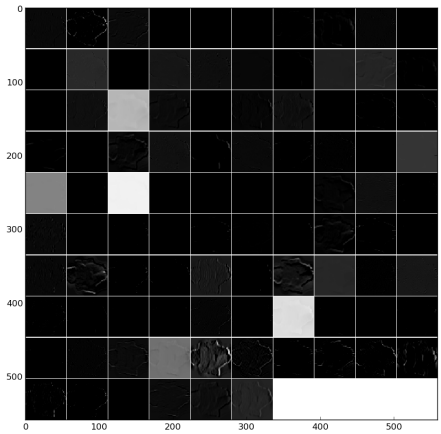
One disadvantage of this is that automated shuffling options are not available as in the other dataset formats. Shuffling training examples is important for achieving the best training results in the shortest times. When training, each adjustment to the network's parameters is proportional to the error on the last example. If the network is shown each ascent sequence in order the errors between each consecutive frame will be small since they contain very similar images and poses. Therefore we will waste lots of iterations making small updates. Since the



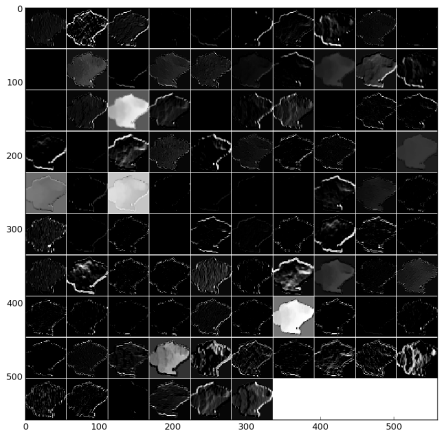
(a) The filters in the 1st convolution layer of a pre-trained then fine-tuned AlexNet [33].



(b) Black background scheme



(c) Mean background method



(d) Final scheme

Figure 16: The activations produced by the first layer filters (a) for each of the colour schemes of figure 15.

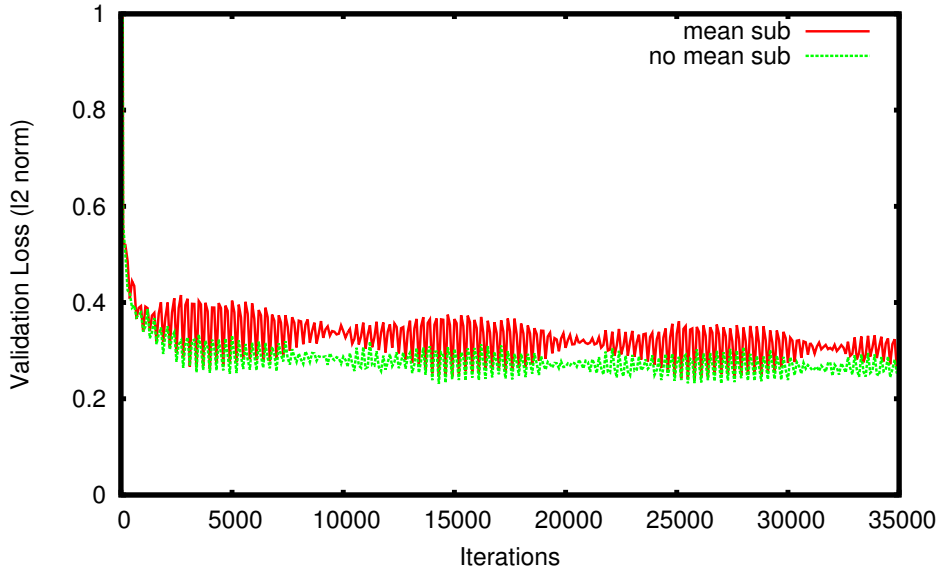


Figure 17: Shows the network accuracy against number of training examples when using mean subtracted data and non mean subtracted data. The accuracy of network predictions was seen to decrease when using mean subtracted data, hence we abandon this common practise.

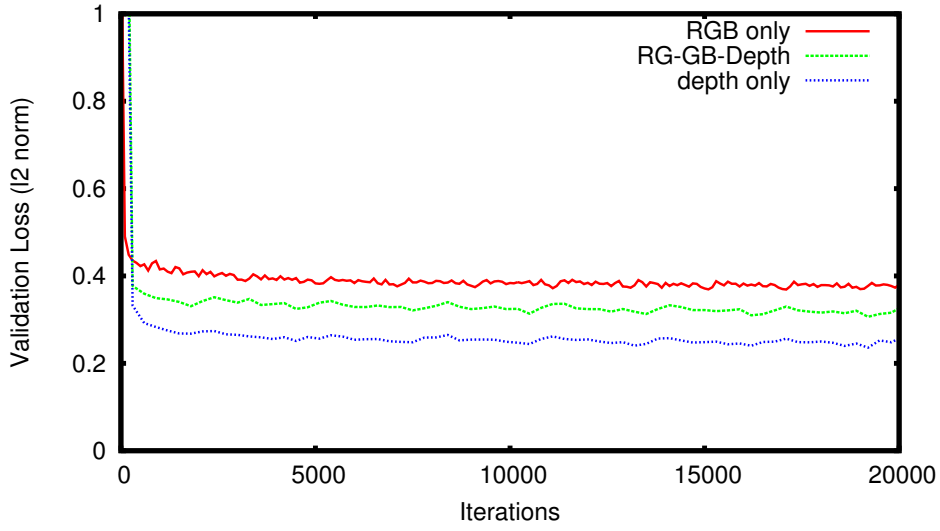


Figure 18: Shows the network accuracy when using the RGB images only, the depth image replicated for each of the 3 channels only, and a combination of the two which consisted of 1 channel of the average of red and green, 1 channel of green and blue averaged and the depth in the final channel. We use depth only for final results.

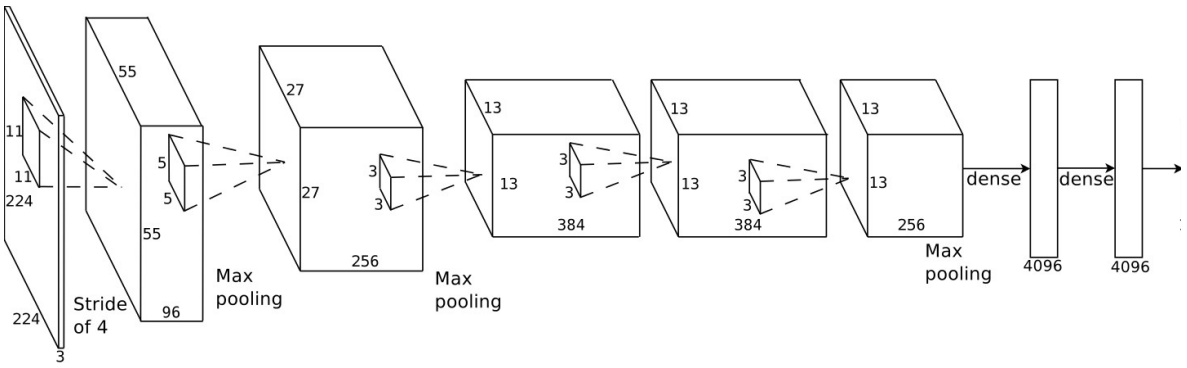


Figure 19: The architecture of [33] is used.

HDF5 input layer in Caffe doesn't provide an automated shuffling we have to pre-shuffle the images and labels before storing them.

5.4 Network Architecture

The architecture used, shown in figure 19, is a version of that which won the 2012 ILSVRC [33]. It consists of 5 convolutional layers, the 1st, 2nd and 5th of which are followed by max pooling layers.

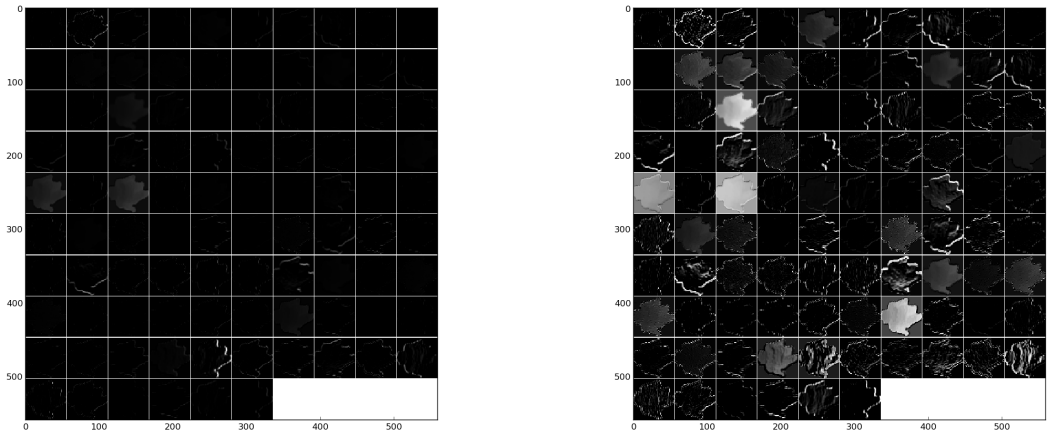
This network also features Local Response Normalisation (LRN) layers after the 1st and 2nd convolution layers. These normalise the values of each filter's activation with respect to the others in the same spatial position. Each activation is divided by $(1 + \frac{\alpha}{n} \sum_i^n x_i^2)^\beta$ where x_i are the activations of filter i at the same position. n, α and β are adjustable parameters, which we leave unchanged from those used by [33] at 5, 0.0001 and 0.75 respectively. The effect of this operation on our data is shown in figure 20.

To adapt this network for regression we replace the final fully connected layer with a 3-element layer which produces our final predicted pose vector. The Softmax loss function is replaced with

$$Loss = \frac{1}{2N} \sum_{i=1}^{N} \|Y_i^p - Y_i^l\|_2^2 \tag{4}$$

where Y^p is the network prediction and Y^l is the label.

One advantage of using this architecture is that we can begin training from filters trained on ImageNet before fine-tuning on our smaller dataset. We tested the networks performance



(a) The activations before LRN.

(b) And after

Figure 20: Shows the effect of Local Response Normalisation on 1st layer activations.

under 3 of these initialisation schemes with all other parameters fixed: 1) using the pre-trained weights for all layers, 2) using the pre-trained weights only on the first two convolution layers with a random initialisation on the others and 3) using a full random initialisation. We found that using ImageNet weights in all layers significantly hindered the final performance as shown in 21. Schemes 2 and 3 produced very similar results, with scheme 2 having slightly better performance at some points earlier in the training. We use scheme 2 for our final results.

We also experimented with other architectures including VGG-S [59] the 2014 ILSVRC 2nd place model which has been shown to generalise better to other tasks than any other architecture [9]. However this deep 19 layer network was too memory intensive. It could only run on our 2GB GPU with a batch size of 1 which produced extremely noisy losses and would not converge.

We also experimented with various custom architectures including: all convolutional architectures which forego pooling layers, using convolutional layers with 2 pixel strides instead, with the idea that this would increase the networks spatial precision since max pooling layers discard precise spatial information. Simple 2 and 3 layer architectures, based on the idea that our data is all very similar in appearance and therefore does not require very high level features. Architectures with filters sized to match the features we hoped them to extract

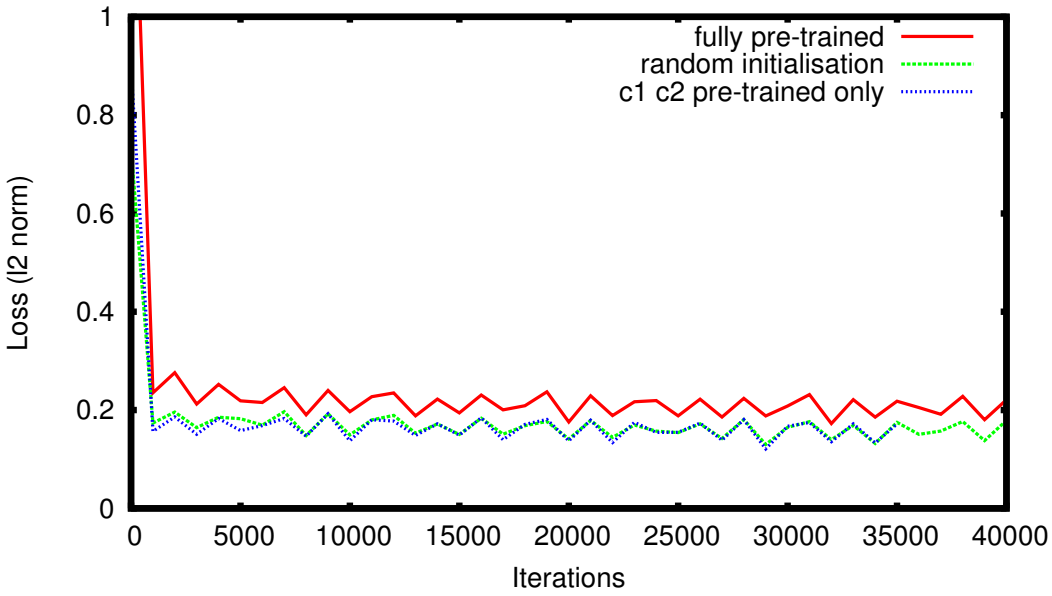


Figure 21: Shows a validation set loss for various pre-training schemes. We find that beginning training from ImageNet pretrained weights throughout the network hinders performance. Using pre-trained weights only in the first two layers produced the best results. We use this strategy for our final tests.

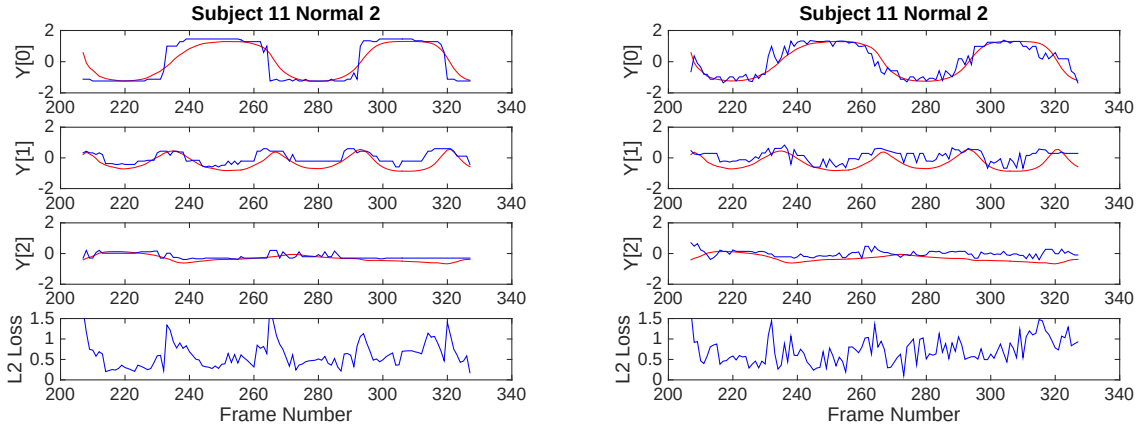
e.g. initial layer filters were slightly wider than the edge effects in the image to avoid the noisy activations we see in figure 20b, 2nd layer filters were sized to match arm and leg widths and 3rd layer filters to match shoulders and head sizes etc. However, we found none of these architectures to improve accuracy over that of [33].

5.4.1 Regression Vs. Classification

Regression tasks are generally harder to optimise CNNs for than classification tasks [31]. With a regression task we require one specific output from the network for each example. With a classification task, using a Softmax Loss, the exact output value is not important, only the relative values. Another problem with regression is the effect of outliers on training as we discussed with reference to [4] in section 4.3.1. An additional benefit of classification is that you get a distribution over the outputs which gives an indication of confidence not possible with pure regression.

To this end we experimented with posing our problem as a classification task by binning each component of the pose vectors into a class. We experimented with a range of numbers of classes. Figure 22 plots the predicted pose vectors after being converted back to the mid value of the predicted class, as well as the L2 norm between this value and the labels (note this was not the loss used in training the network, but is calculated afterwards for a measure of accuracy). We find that 51 classes produces a mean loss of 0.60 and 500 classes a mean loss of 0.71. Our final regression network outperforms both of these with a loss of 0.13, this result is displayed in figure 19o. These results suggest that this approach is far less suitable than pure regression.

Inspired by the results of Li and Chan [39] who found that regression accuracy was improved by joint training with a joint detection task; we also attempted joint training of classification and regression tasks in various forms. These included: performing the classification off of early layers in an attempt to force gradient terms to the lower layers. A sub-net which classified the first component and shared features with the regression network. Both shared and separate fully connected sections for regression and classification. However, across all these studies we found no improvement in accuracy over pure regression. We also found that the more complicated architectures were prone to diverging during training.



(a) With 51 classes per component. Mean loss = 0.6035 std = 0.3154.

(b) With 500 classes per component. Mean loss = 0.7141 std = 0.3054.

Figure 22: The top 3 graphs plot the 3 components of the pose vector for the labels (red), and the network prediction (blue) for a classification network. The 4th plot shows the error measured as the distance between the labels and prediction. A mean error of 0.13 for this sequence was achieved using pure regression.

5.5 Training Details

Following skeleton selection and depth pre-processing we are left with 6228 distinct examples, plus their horizontal flips. So as to best utilise our data we cross validate training 6 networks with pairs of adjacent subjects withheld for testing in each i.e. 1 and 2, 3 and 4, 5 and 6 etc.

Each of these networks was trained for 50000 iterations with a batch size of 25 examples on a GeForce GTX 750 GPU taking almost 7 hours each. We observed no overfitting effects even after 700000 iterations. However results did not improve beyond 50000 iterations. The batch size was dictated by the 2GB memory limit of the GPU however it seemed reasonably suitable with only a small amount of noise in the loss curves as seen in figure 21.

Additionally we present subject fine-tuned results where each of these trained networks is further trained for 10 epochs (10 times through the training data) on the remaining non-test sequences of that same subject. We found that overfitting did occur for these small training sets. The optimum results required tailoring to the size of the training set rather than a fixed number of iterations.

In both cases we use the adaptive learning rate method AdaGrad [15] which scales the learning rate for each parameter based off the size of its previous updates. We found in practise that this method achieved a roughly equal loss to that of stochastic gradient descent with momentum and to Nesterov's accelerated gradient method. AdaGrad has the benefit of removing the need to tune for hyper parameters such as the learning rate and momentum. We also apply a weight decay of 0.005. We found that altering this had a negligible effect on the accuracy of the network so chose to keep it at this commonly used value.

For each subject we present results for their longest sequence since longer sequences make the SPHERE software's gait cycle time analysis more accurate. In the case of subjects 9, 11, and 12 we test one normal and one abnormal sequence taking the longest one of each.

6 Results

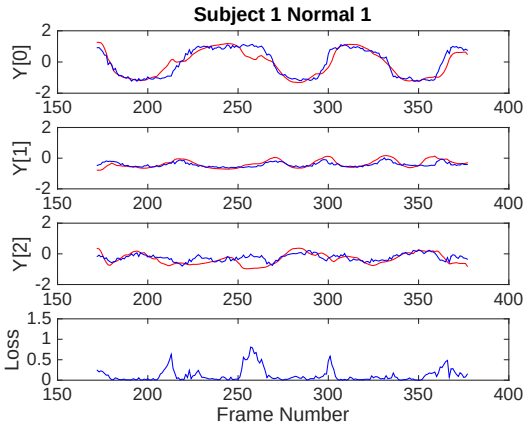
We measure the accuracy/error by the same loss function used to train the network i.e. the distance (L2 norm) between the predicted pose vectors, always shown in blue, and the ground truth labels shown in red. Across all tested sequences we find an average error of 0.1565 for the non subject fine-tuned models which is reduced to 0.1129 after fine-tuning. The percentage of the total error per component is [37.86%,25.66%,36.48%].

For the first 6 subjects we present only accuracy data, shown in figure 21, since these sequences are used by the SPHERE software to build the normality model.

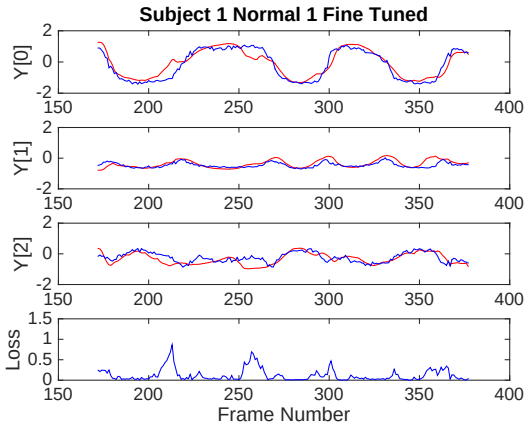
For subjects 7-12 we present both accuracy data and pose and dynamics quality scores, shown in figure 17. These quality scores were computed for us by the authors of [44] using the same methods and models (trained from the skeleton data) of that work.

6.1 Fine-Tuning

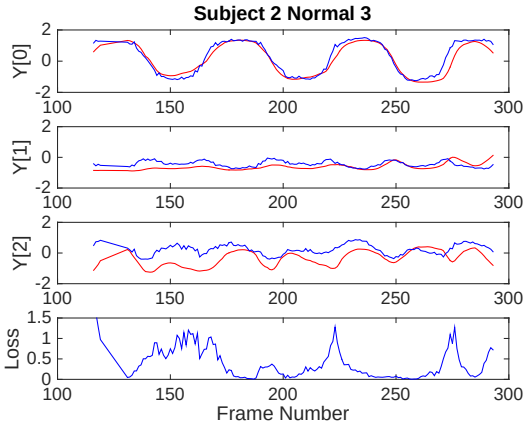
For 13 of the 15 tested sequences fine-tuning reduced the mean error. The greatest decrease was seen in that of subject 2 (figures 23c and 23d) with the error decreasing by 0.1579. In the non fine-tuned case the predicted $Y[2]$ component appears to follow the form of the labels but with a slight offset producing a large and consistent error across the sequence. Looking at plots of the label data for these sequences in figure 18a we see that the tested sequence Normal 3 contains the minimum $Y[2]$ points in the whole training set. This explains why the network



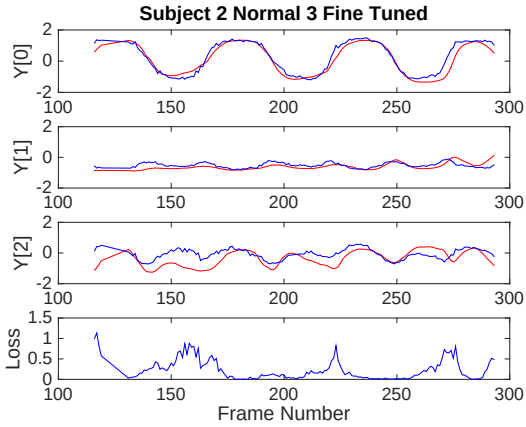
(a) mean loss = 0.1207, std = 0.1604.



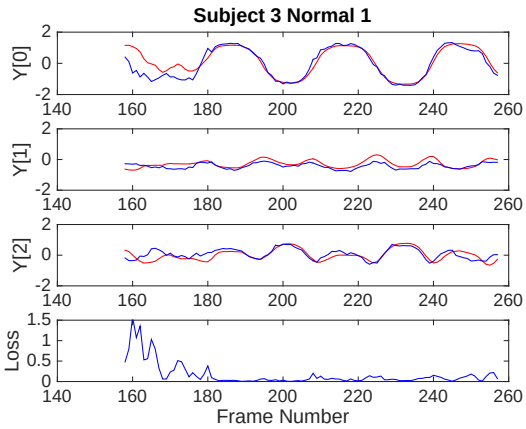
(b) mean loss = 0.1227 std = 0.1497.



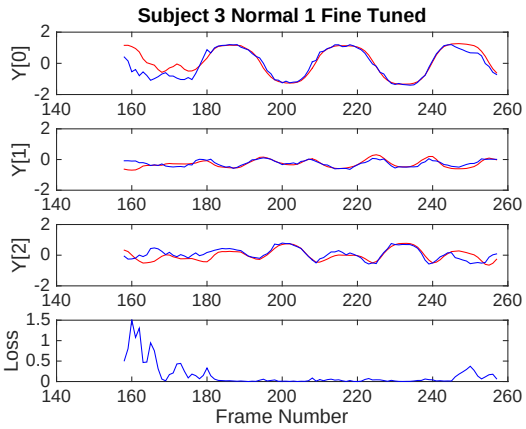
(c) mean loss = 0.3859, std = 0.3678.



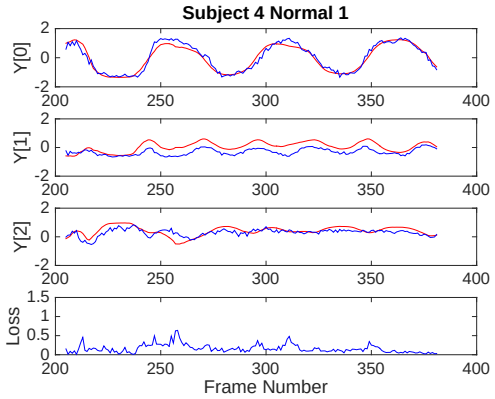
(d) mean loss = 0.2280 std = 0.2457.



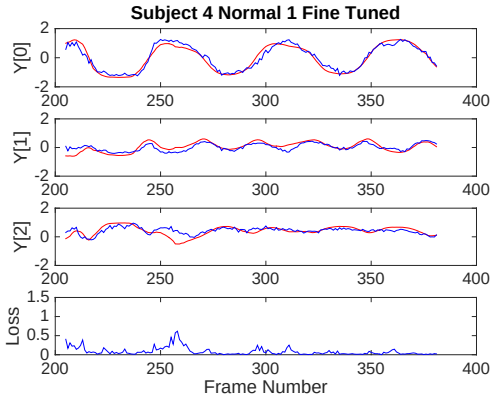
(e) mean loss = 0.1648, std = 0.2772.



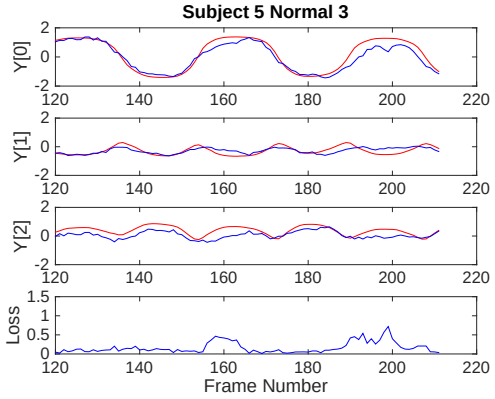
(f) mean loss = 0.1465 std = 0.2713.



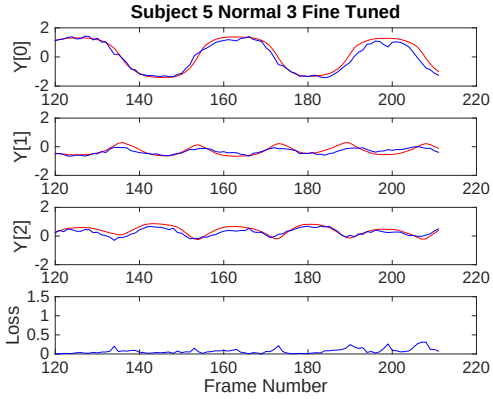
(g) mean loss = 0.1613, std = 0.1109.



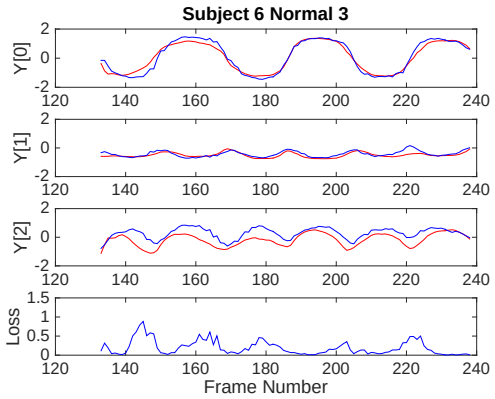
(h) mean loss = 0.0852 std = 0.1030.



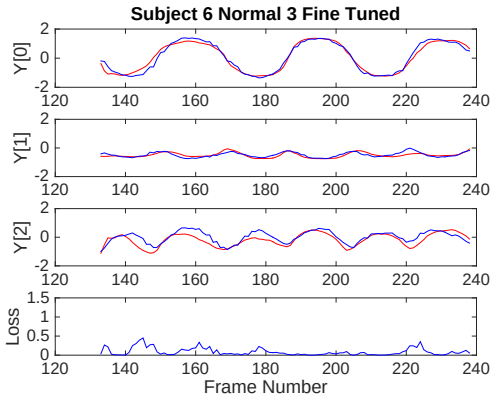
(i) mean loss = 0.1640, std = 0.1450.



(j) mean loss = 0.0751 std = 0.0751.

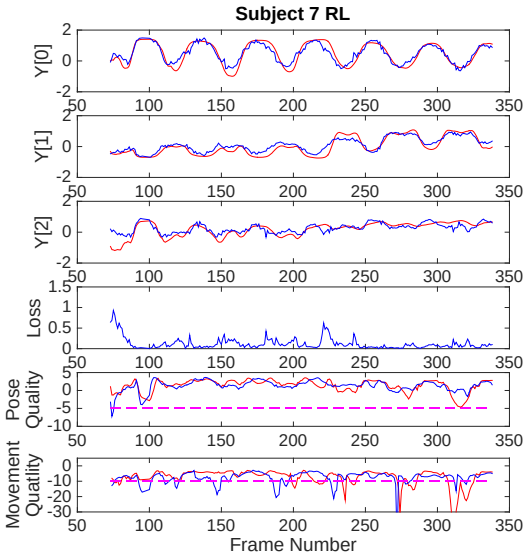


(k) mean loss = 0.1952, std = 0.1884.

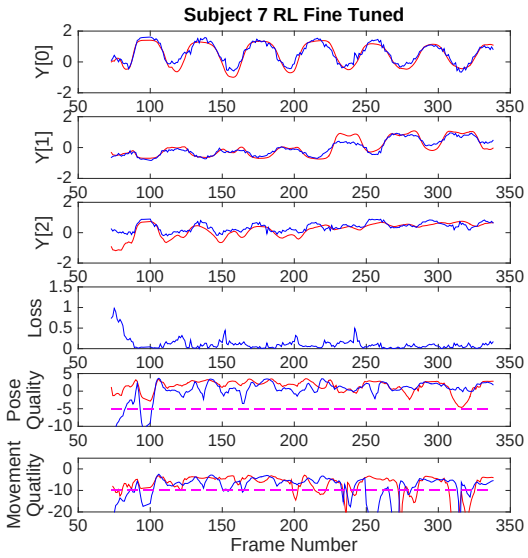


(l) mean loss = 0.0873 std = 0.0970.

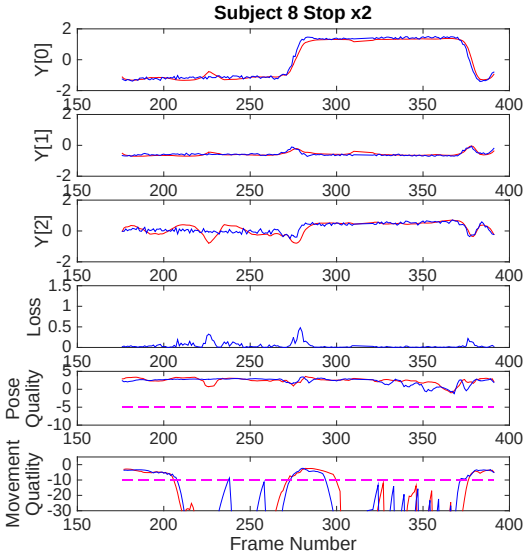
Figure 21: Top 3 graphs show the 3 components of the pose vector for the labels (red), and the network prediction (blue). The 4th plot shows the error measured as the distance between the labels and predictions. Fine-tuned results are produced by networks which have been trained with spare sequences of the subject being tested.



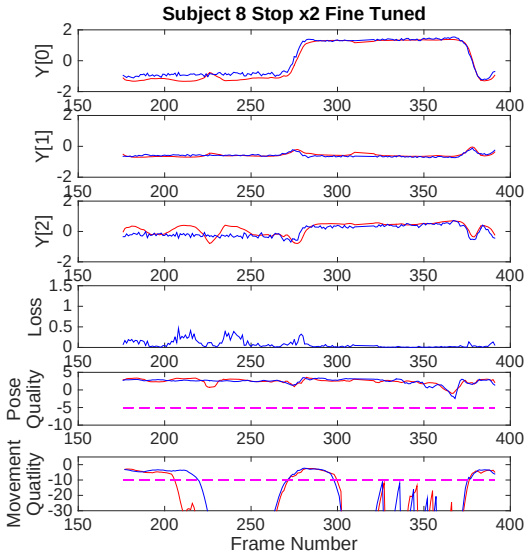
(a) mean loss = 0.1208, std = 0.1388.



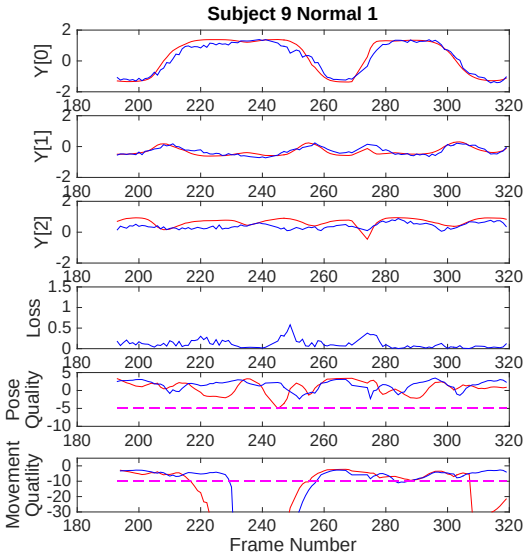
(b) mean loss = 0.1133 std = 0.1439.



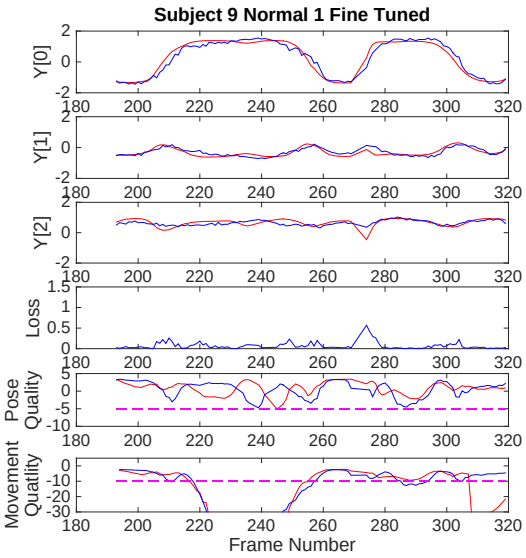
(c) mean loss = 0.0457, std = 0.0720.



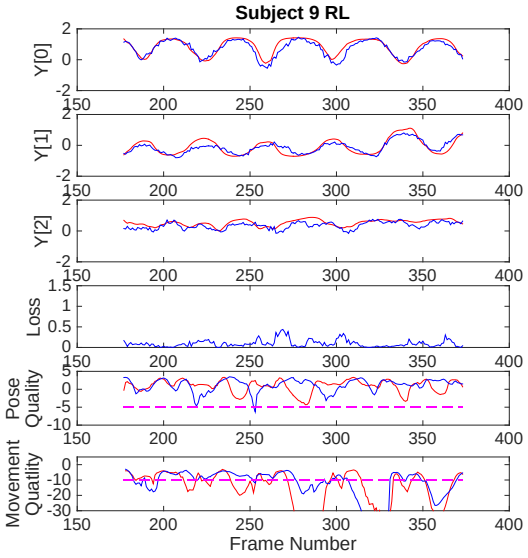
(d) mean loss = 0.0795 std = 0.0950.



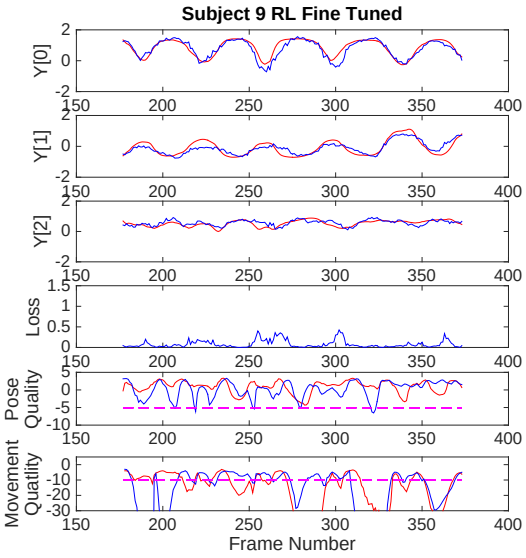
(e) mean loss = 0.1110, std = 0.0995.



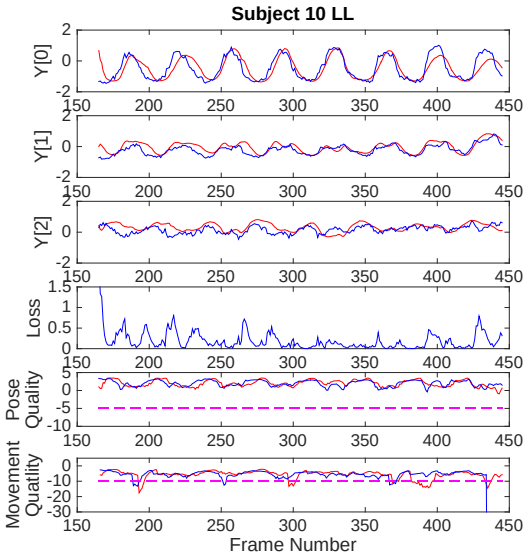
(f) mean loss = 0.0692 std = 0.0852.



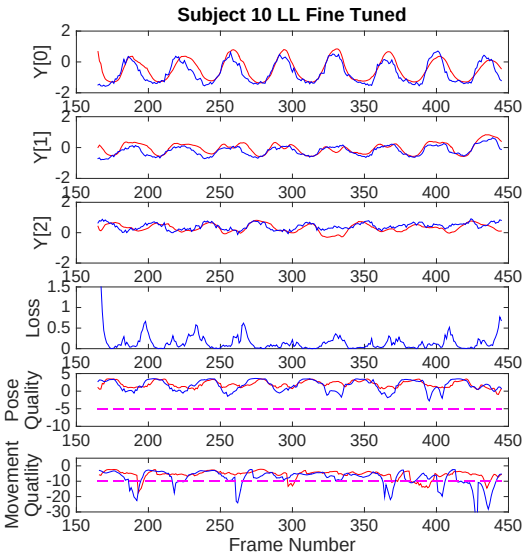
(g) mean loss = 0.0944, std = 0.0875.



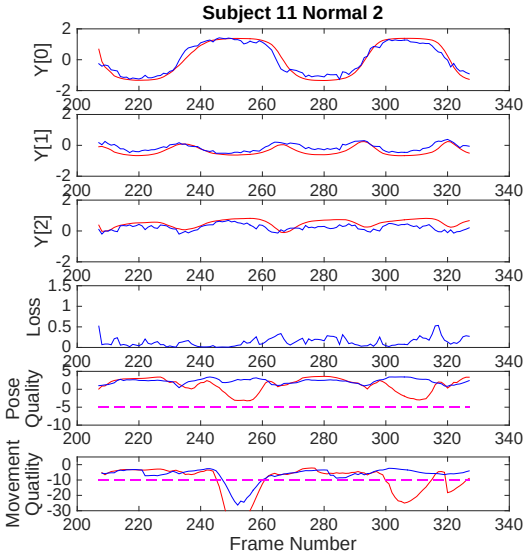
(h) mean loss =0.0821 std = 0.0917



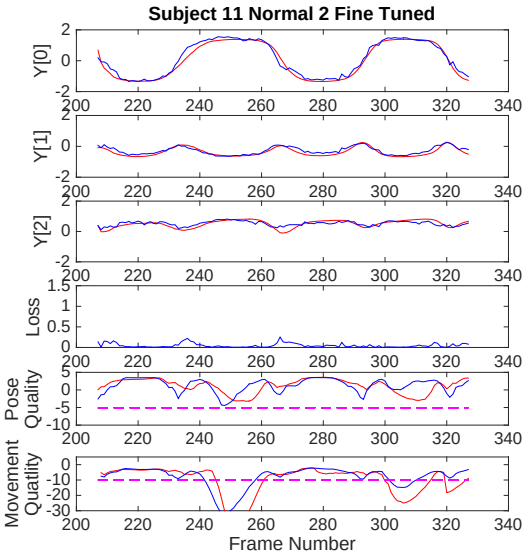
(i) mean loss = 0.1937 std = 0.2353.



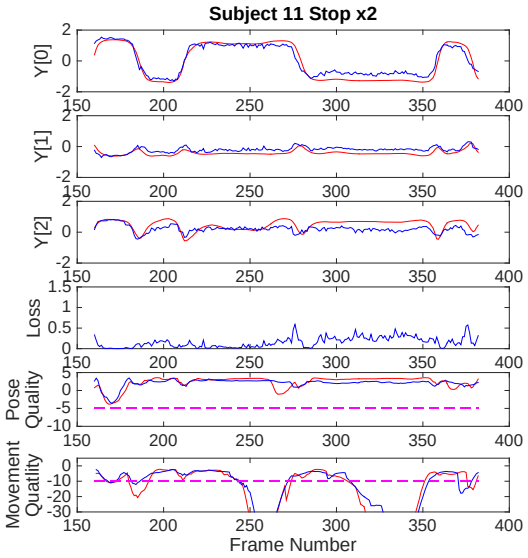
(j) mean loss =0.1575 std = 0.2448



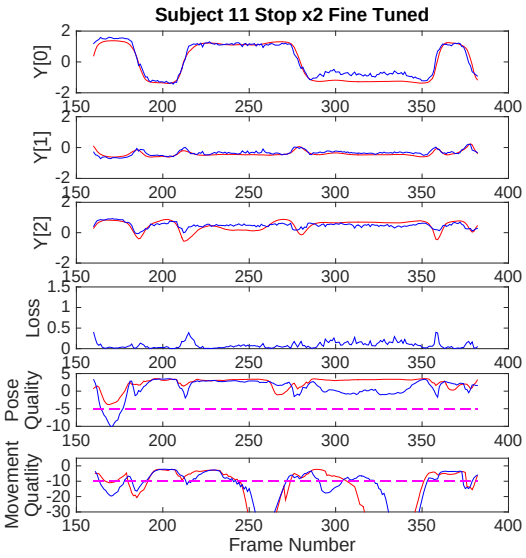
(k) mean loss = 0.1337, std = 0.1083.



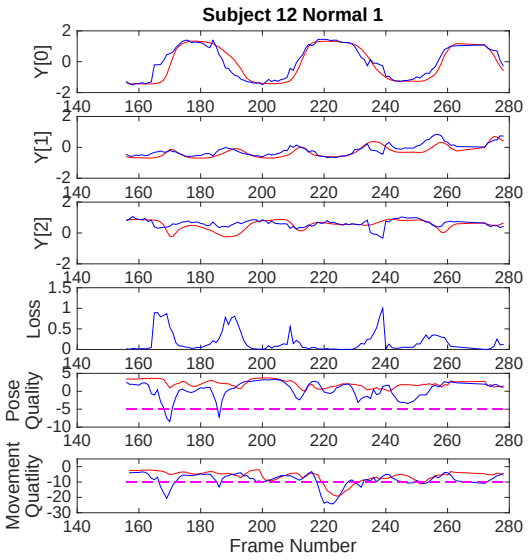
(l) mean loss = 0.0474 std = 0.0496



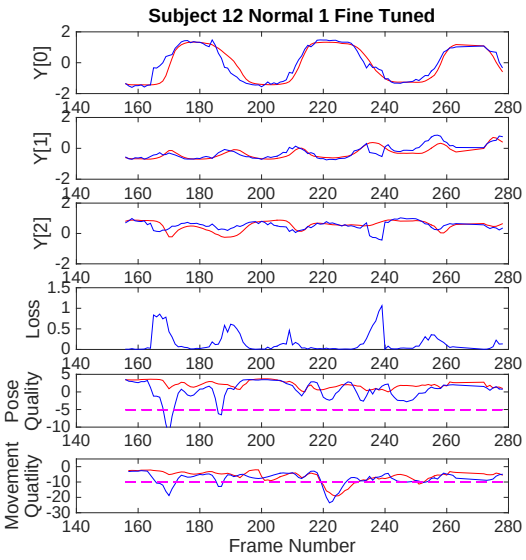
(m) mean loss = 0.1547, std = 0.1231.



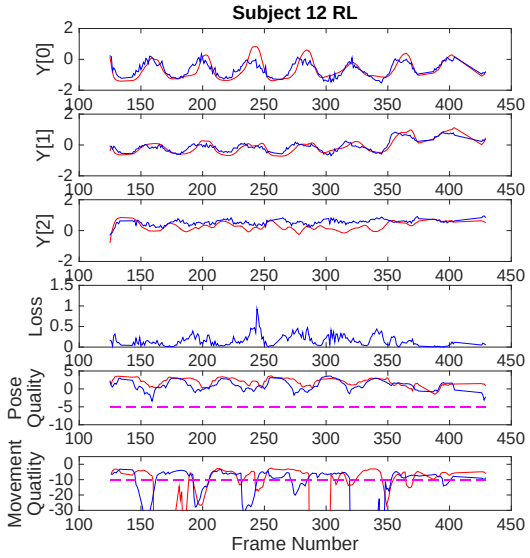
(n) mean loss =0.0867 std = 0.0792



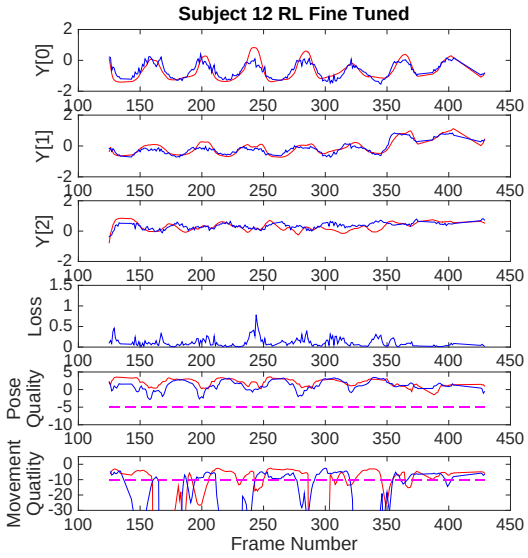
(o) mean loss = 0.1893, std = 0.2492.



(p) mean loss = 0.1673 std = 0.2368

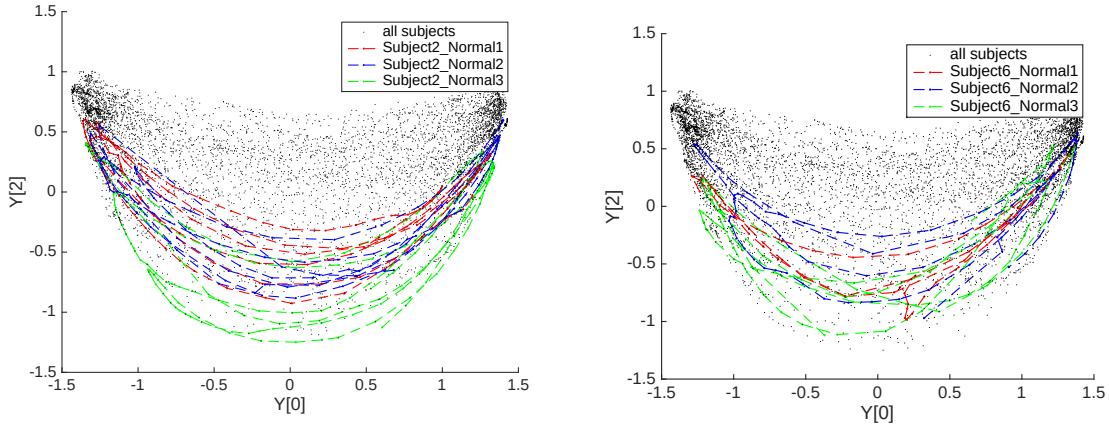


(q) mean loss = 0.1611, std = 0.1334.



(r) mean loss = 0.1097 std = 0.1040

Figure 17: Top 3 graphs show the 3 components of the pose vector for the labels (red), and the network prediction (blue). The 4th plot shows the error measured as the distance between the labels and predictions. Fine-tuned results are produced by networks which have been trained with spare sequences of the subject being tested. 5th and 6th plots show the movement quality measurements of [44] when given the network's prediction (blue) and the original results of the labels (red). The magenta dashed line shows the empirically determined abnormality thresholds.



(a) Subject 2. Fine-tuning on Normal 1 and 2 produced a decrease in error of 0.1579 in 3.

(b) Subject 6. Fine-tuning on Normal 1 and 2 produced a decrease in error of 0.1079 in 3.

Figure 18: Shows a plot of the skeleton/label data for each sequence of the subjects in which fine-tuning produced the greatest decrease in the error of the network's predictions.

was not able to measure this low $Y[2]$, since it has not been trained with any points with these values of $Y[2]$. The other subject 2 sequences used for fine-tuning also have paths below the mean $Y[2]$ of the rest of the training data which explains why the fine-tuning helps to reduce the predicted $Y[2]$. This same explanation also seems to apply to the second most effective fine-tuning results, shown in 18b; that of Subject 6 for which the error was reduced by 0.1079. These results suggest that with a more even spread of training data we could expect improved accuracy, with or without fine-tuning.

For two sequences: Subject 1 Normal 1 and Subject 8 Stop $\times 2$ the error increased after fine-tuning.

For Subject 8 (figures 23c and 23d) fine-tuning produced a decrease in the predicted $Y[2]$ and an increase in $Y[0]$ particularly for the first frozen pose (the tested sequence was a Stop $\times 2$). Both these changes took the predictions away from the ground truth, increasing the error. Looking at a plot of the training/label data, shown in figure 19 it seems the spread in $Y[2]$ for this subject was larger than for most other subjects. However the accuracy both before and after fine-tuning was well below the mean of the rest of the data.

In the case of Subject 1 it seems the error is actually coming from a miss-measured skeleton in the label data meaning improved network accuracy actually increases the measured

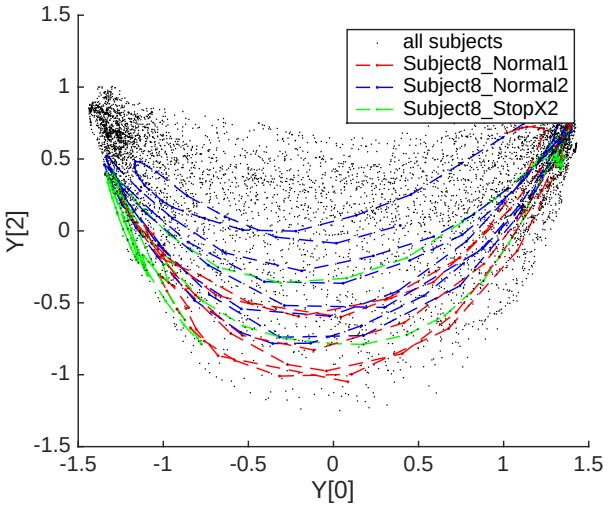


Figure 19: Shows a 3D plot of the skeleton data for Subject 8. Fine-tuning on Normal 1 and 2 produced an increase in error of 0.0338 in the Stop x2 sequence.

error. The results plots (figures 23a and 23b) show that the label data, particularly the $Y[0]$ component, changes sharply at the points of greatest error which indicates a jump in the Kinect skeleton. Figure 20 shows the image and labelled skeleton for frame 257 of this sequence which is the position of the second peak in the loss graph. We see that the Kinect's algorithm has been confused by the occlusion of the left leg causing it to measure the left foot in front of the right which we see from the depth image is not the case. The network is not fooled by this occlusion, predicting a point near the maximum $Y[0]$, the skeleton nearest to this point shows the right foot correctly in front of the left. This was also the case around 213 and 303 where we have other peak in loss.

6.2 Network Errors Vs. Label Errors Vs. Image Errors

Erroneous skeletons in the labels for tested data are common. The largest losses across all the tested data occur at the beginning of sequences Subject 2 Normal 3 (results figure 23d, skeleton example figure 21a), Subject 3 Normal 1 (results figure 23f, skeleton example figure 21b), Subject 7 RL (results figure 22b, skeleton example figure 21c) and Subject 10 LL (results figure 20i, skeleton example figure 21d). As mentioned in section 5.2.1 skeleton data which is too dissimilar from a skeleton with neutral pose are discarded, which typically removes the frames at the beginning and end of the sequence where the subject is out of the optimum

(a) Labelled skeleton.

(b) Depth image.

(c) Skeleton nearest to network's prediction.

Figure 20: Shows the image label and prediction for frame 257 of subject 1 normal 1. The results (figure 23b) showed a large error at this frame which is in fact due to a miss-measured skeleton.

range of the sensor. However this cut cannot be too stringent without excluding potential valid but abnormal poses from the data. Also poses that are well within this cut, but still far from the true pose are not excluded.

The examples displayed in 21a, 21b and 21d prove that the network can be accurate at distances greater than the Kinect algorithm, provided that the pose in the image is similar to some in the rest of the training data which do have accurate skeletons. This could be due to the difference in methods of the Kinect skeleton algorithm [57] and the network. We believe that the depth comparison features (presented in equation 1 and figure 5) used for pixel body part classification in that work are are sensitive to range from the sensor. At increased range the values of these features must change since the width of body parts in pixels will be smaller producing different values for the same body location and offsets at different ranges. Additionally, at longer range there will be fewer body pixels meaning that the statistics become noisier and any errors have a larger effect on the final joint position estimation. Although this is not discussed in their work, this would seem to make their trained random forests become inaccurate with range. In contrast our method scales the subject's body to a fixed size. At increased ranges we merely have less information due to the reduction in depth precision at longer ranges and due to pixilation on up scaling. These effects will likely cause a loss in precision, put poses should still be reasonably accurate provided there

are similar poses represented in the training data.

Whilst these erroneous skeletons are particularly common at the beginning of sequences they can also occur mid sequence when the Kinect algorithm is fooled by a challenging pose/image, particularly occlusions. This is the case in the Subject 1 Normal 1 sequence as discussed in section 6.1 with an example shown in figure 20. Other examples are displayed figure 22.

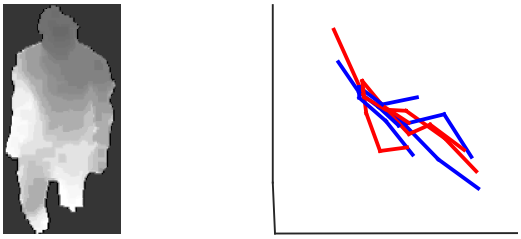
There are also frames with large error where the skeletons do appear accurate. Some examples of these are shown in figure 23. These can occur where there are poses which receive a strange mapping when compared to similar images in the training data such as in figure 23b where a right leg forward pose receives a negative $Y[0]$, or figure 23c where a close to the camera lent in image, which normally receive large $Y[1]$, is instead mapped to a regular $Y[1]$.

Another type of error is caused by bad images which were not removed during pre-processing. Typically these are when the background subtraction failed, the subject had not fully faced the sensor or the subject is too close to the sensor. They should have been removed during pre-processing but were missed. Examples are shown in figure 24

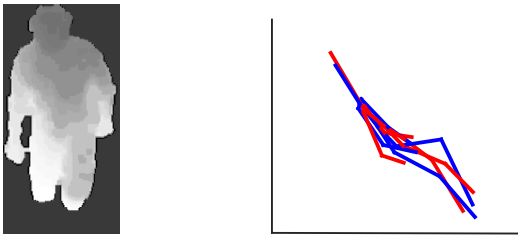
In figure 23 we analyse every error further than the 1σ from the mean for all fine-tuned sequences. Each of these 255 frames is assigned 1 of 5 classes: a network error (red), a skeleton error (green) or an image error (magenta).

In each frame we studied the images and then examined the Kinect skeleton and compared the position of the prediction and label on the manifold. Only if the skeleton is clearly inaccurate is a skeleton error assigned. This has generally only been determined when one leg is measured in front of the other and the depth values of the image tell us otherwise or there is a clear double measurement of one foot as in figure 22. In this case if the network's prediction in $Y[0]$ agrees with the image; i.e. right leg forwards positive $Y[0]$, left leg forwards negative $Y[0]$, legs at equal depth $Y[0] \sim 0$, then we deem the network accurate otherwise we determine that both network and skeletons are erroneous. We do not use the nearest skeleton method to determine network accuracy because it is not particularly robust; by nature of the skeleton to manifold mapping some noisy skeletons can exist close to other skeletons which would appear a reasonable prediction. We try to give an objective classification however this can be difficult; figure 24 presents some borderline cases.

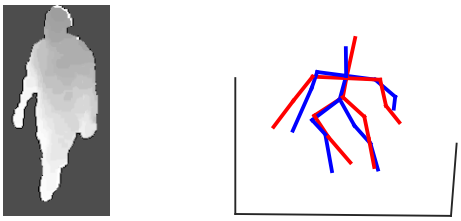
If we exclude skeleton and image errors the mean error on the tested sequences drops to



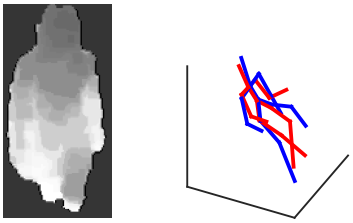
(a) Subject 2 Normal 3 frame 117 loss = 1.13. Labelled skeleton has legs at a similiar depth, network's prediction has right leg far in front of left.



(b) Subject 3 Normal 1 frame 162 loss = 1.30. Labelled skeleton has right leg in front of left, network measures the opposite which is correct.

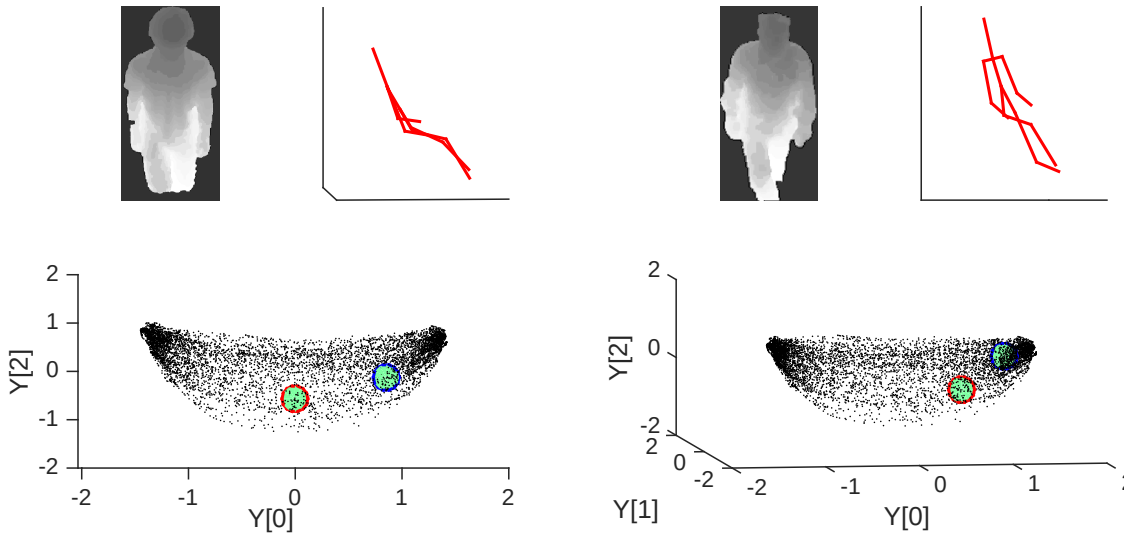


(c) Subject 7 RL frame 75 loss = 0.97. Labelled skeleton is not accurate. The network also fails since this is a strange pose where the subject appears to be turning on to the stairs. Since there are few examples of poses of this kind in the training data, and the ones that are have bad skeletons we can not expect better performance.



(d) Subject 10 LL frame 165 loss = 2.63. Labelled skeleton legs crossed over right above left, network measures left leg forward correctly.

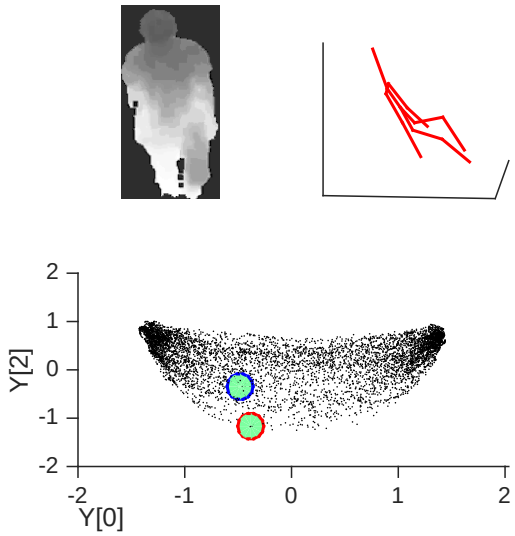
Figure 21: Out of range skeleton errors: shows labelled skeletons in red and the skeletons nearest to the network's predicted pose vector in blue for each of the images. These are some of the tested images which produced the greatest errors, occurring at the beginning of sequences where the subject is outside of the Kinects optimum range. We find that these errors can be attributed to poor ground truth with the network able to produce a more accurate measure provided that similar poses with correct labels do exist in the training data.



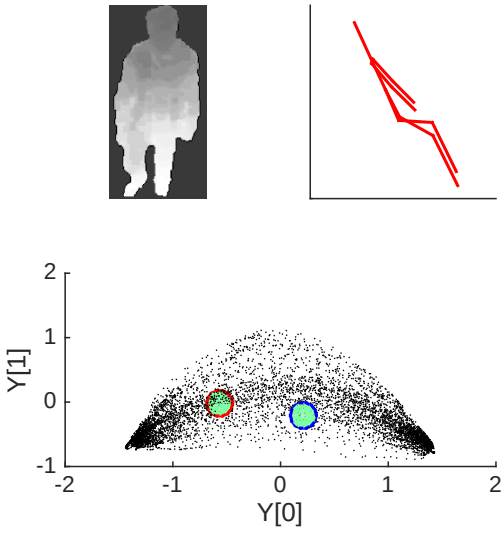
(a) Subject 2 Normal 3 frame 275. Loss = 0.5178. The skeleton of this frame is determined to be inaccurate since relative leg positions disagree considerably with the depth image.

(b) Subject 4 Normal 1 frame 205. Loss = 0.5738. The skeleton of this frame is determined to be inaccurate since there is a clear error in the left foot position being measured at the same position as the right due to occlusion.

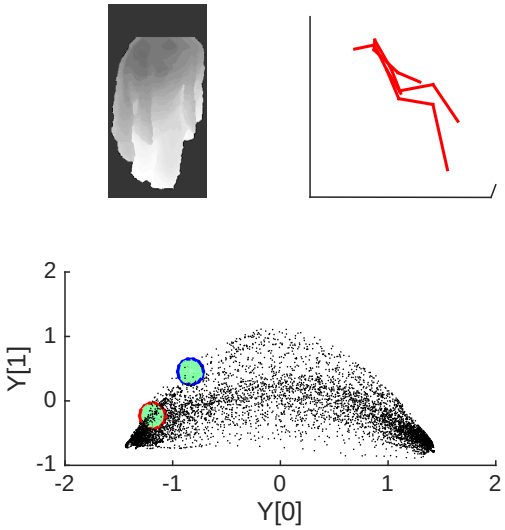
Figure 22: Skeleton errors: shows for each image the labelled skeleton and corresponding pose vector circled in red, and the network's prediction in blue.



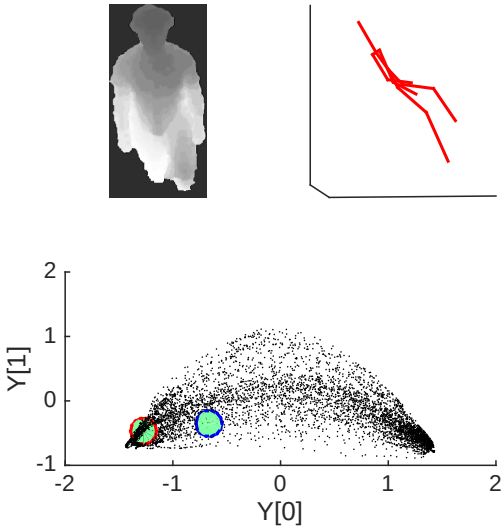
(a) Subject 2 Normal 3 frame 143 loss = 0.4268. The cause of errors in this sequence is discussed in section 6.1.



(b) Subject 12 RL frame 167 loss = 0.3274.



(c) Subject 12 Normal 1 frame 253 loss = 0.3105.



(d) Subject 11 Stop x2 frame 311 loss = 0.2779.

Figure 23: Network Errors: shows label skeleton and corresponding pose vector in red and the network's predicted pose vector in blue for each of the images. In these cases the Kinect skeletons are deemed accurate. By design, the network predicts a pose vector typical of images similar to those it has seen in training. Network errors are caused by under represented pose vectors as in (a), or by inconsistencies with the training data as in (b) and (c).

(a) Subject 12 Normal 1 frame 239 loss = 1.06. A poor foreground mask slipped through the cuts due to its roughtly human shape and size.

(b) Subject 10 LL frame 444 loss = 0.77. Subject is too close to the sensor; legs are not visible.

Figure 24: Image Errors: shows labelled skeletons in red and the skeletons nearest to the network's predicted pose vector in blue for each of the images. These are some of the tested images which produced the greatest errors. These errors can be attributed to images that were meant to be removed during pre-processing, either because of bad masks or because the subject was too close to the camera or they were not fully turned towards the camera.

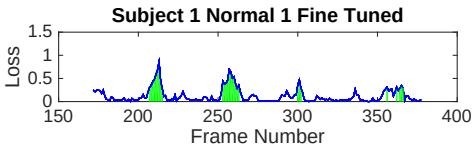
0.0786, with a standard deviation of 0.0693.

6.3 Movement Quality

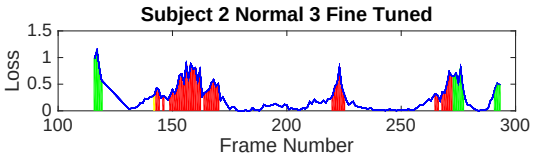
[44] uses two measures: pose quality and dynamics quality for assessing the quality of human movement from the pose vectors of the stair ascent. These quantities were computed for us by the authors of [44] using the network predictions. These results are presented in figure 17.

The pose quality is an estimate of the likelihood of an instantaneous pose vector being a normal pose. It is a probability density function of pose vector learnt form those normal poses used to train the model. In this case these models are trained for the labels of the 17 normal sequences of subjects 1-6. For all sequences, normal and abnormal, the pose quality is generally expected to remain roughly within normal limits since the abnormality of these sequences is in the dynamics e.g. freezing of gait rather than in the pose. A threshold for abnormality was determined empirically at -5, this is shown in figures 17 as a dashed magenta line.

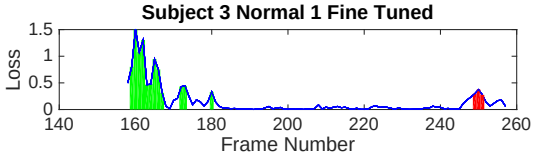
Generally the lower the loss is, the better the pose quality tracks that of the labels. There are however some regions around the edges of the manifold where small changes in pose vector



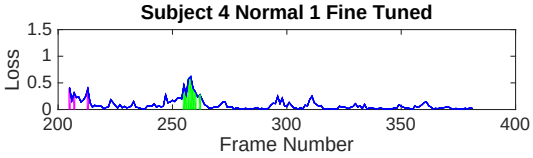
(a) adjusted mean loss = 0.0738 std = 0.0698.



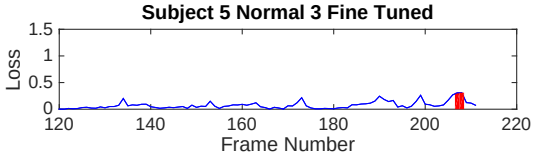
(b) adjusted mean loss = 0.1932 std = 0.2126.



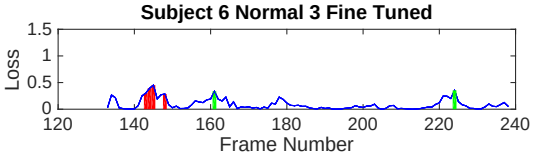
(c) adjusted mean loss = 0.0650 std = 0.0902.



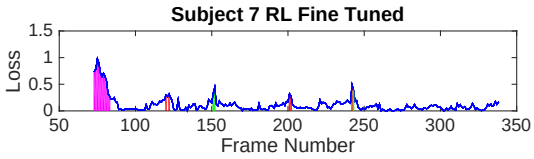
(d) adjusted mean loss = 0.0669 std = 0.0617.



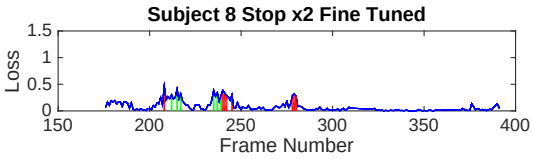
(e) adjusted mean loss = 0.1214 std = 0.1167.



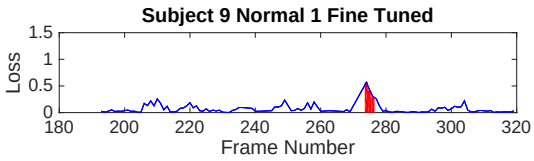
(f) adjusted mean loss = 0.0817 std = 0.0895.



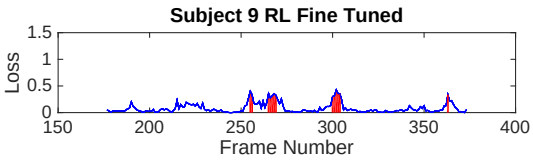
(g) adjusted mean loss = 0.0875 std = 0.0698.



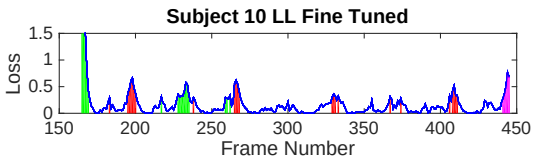
(h) adjusted mean loss = 0.1932 std = 0.2126.



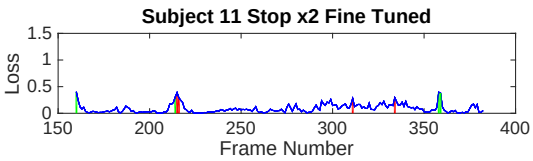
(i) adjusted mean loss = 0.0650 std = 0.0902.



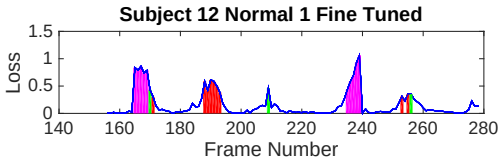
(j) adjusted mean loss = 0.0942 std = 0.1061.



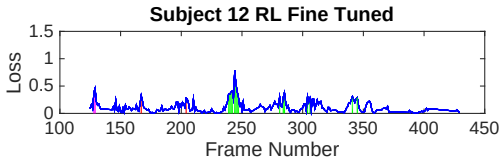
(k) adjusted mean loss = 0.0941 std = 0.1085.



(l) adjusted mean loss = 0.0786 std = 0.0693.



(m) adjusted mean loss = 0.0815 std = 0.0821.



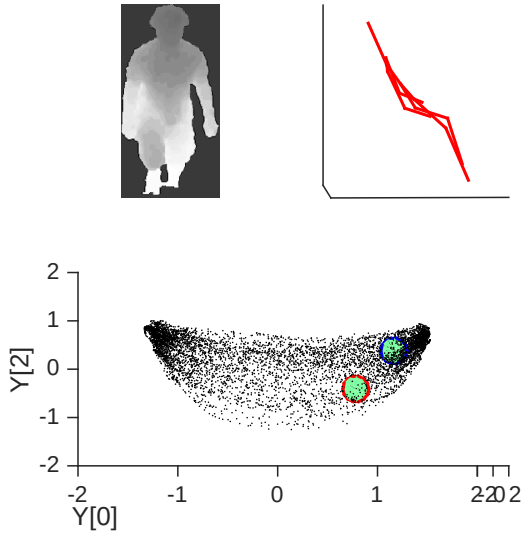
(n) adjusted mean loss = 0.0784 std = 0.0819.

Figure 23: Plots of the loss i.e. the distance between the labelled pose vector and the network's predicted pose vector for the subject fine-tuned models. Every frame with loss greater than 1σ from the total mean is categorised as a network error (red), an image error (magenta) or a skeleton error (green). Examples of each of these errors are shown in figures 23, 24 and 22 respectively. The mean for each sequence is recomputed after excluding skeleton and image errors to give a more reliable estimate of the network's true precision. The collective mean after excluding these errors drops from 0.1565 to 0.0874.

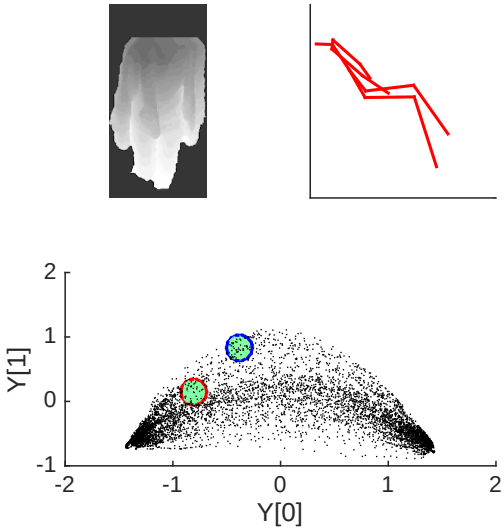
produce large changes in quality. For example around frame 100 in Subject 7 RL (figure 22b) we find a discrepancy of the pose quality between labels and predictions of ∼-7 despite the loss being only ∼0.02. A similar case occurs in Subject 11 Stop x2 around frame 160. We believe that ideally such small changes in pose should not be producing such large differences in quality. Given a greater collection of normal pose training data the model may be more robust and this effect could be lessened. Alternatively some sort of smoothing may need to be applied to the model.

There are abnormal poses measured in Subject 12 Normal 1 (figure 19p) are due to

For the movement quality score each sequence is separated into gait cycles and each frame is assigned a coordinate equal to its position in the cycle. Again a probability density function on pose vector and its stage of the movement cycle is learnt from the normal sequences of subjects 1-6. This is displayed on a heat map in figure 25. As with pose quality there are some areas, such as around $Y[0] = -1, X = 0.3$ the likelihood can fall sharply with $Y[0]$ which can be seen on this figure. The range of $Y[2]$ values which are deemed normal is much broader than that of $Y[0]$ or $Y[1]$ which means that the fact the network tends to measure this with lower accuracy should have less effect on the final results. Again an empirically determined threshold of -10 is used to define abnormality. Generally it is expected that normal sequences



(a) Subject 11 Stop x2 frame 211 loss = 0.3881. A difficult case where the image and network's prediction seem plausible but the skeleton does too. In this case we give the skeleton the benefit of the doubt and assign a network error.



(b) Subject 12 Normal 1 frame 256 loss = 0.3511. Here the skeleton says the right knee is behind the left. Image seems to show them at equal depth so we assign a network error.

Figure 24: Shows some borderline cases when attributing the causes for errors in figure 23. We try to be as objective as possible only ascribing a skeleton error if there is a clear error with the ground truth. These cases are determined to not be clear enough so we assign network errors in both cases.

should remain above this threshold, left/right leg lead (L/RL) sequences should dip below every cycle and freeze (Stop × N) sequences should go below at every freeze of gait. However even for the labels there is a tendency for normal sequences to produce abnormal scores in places when stride times are not completely consistent as in the case of those presented in figure 17. As with pose quality smaller loss in the networks predictions should produce a better agreement with label dynamics quality scores. However small miss-measurement can again produce large differences in the score.

In the R/LL sequences (figures 20j and 22b) we find that the network predictions tend to produce greater abnormality in the correct places than the labels did. For these sequences dynamics abnormality is increased considerably when the pose vector does not go beyond $Y[0] = 0$ for the frames where both legs are on the same step. When this occurs the path for the gait cycle enters the dark blue regions of the normality model for this component (shown in figure 25). Looking at the sequences Subject 10 LL and Subject 7 RL (20j and 22b) its seems that the network tends to measure $Y[0]$ below the label value at this point in the gait cycles. Comparing the predicted pose vectors, the images and the skeletons of frame 116 of Subject 7 RL, shown in figure 26a, it does seem that the Kinect has over-measured the position of the left knee. This leads it to produce a pose vector in the negative region of $Y[0]$ which in turn produces a normal dynamics quality score, where as the network's prediction is much closer to $Y[0] = 0$ producing an abnormal dynamics score. We find the same situation in the Subject 10 LL sequence with the right foot measured slightly in front of the left in these frames as shown in figure 26b.

However, the other tested L/RL sequence Subject 9 RL (figure 21h) exhibits the opposite behaviour at some of these cycle points i.e. predicts slightly negative $Y[0]$ when both feet should be on the same step. Looking at the images for these frames (frame 256 shown in figure 26c) a reason for this may be that the subject's left arm swings down across his knee which the network may be mistaking for a knee at increased depth. In the other points of this kind, where the network prediction's $Y[0]$ is equal or less than the skeleton's we find that the arm does not cover the knee as seen in figure 26d. This is an understandable error for the network to make. There were probably too few training examples where the arm covered the knee/hip in this manner for the network to learn to gauge this effect.

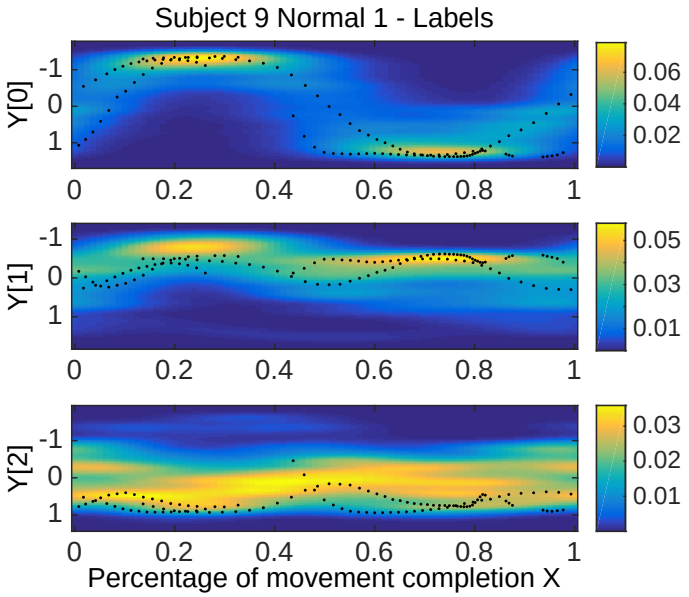
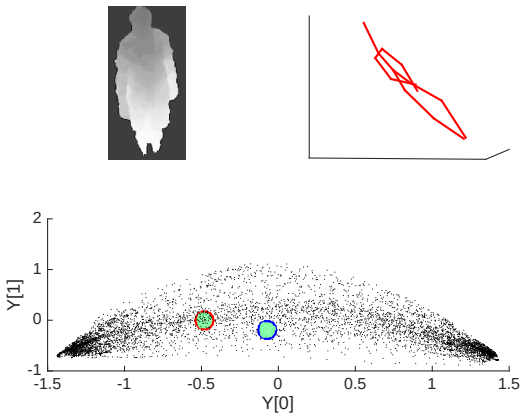
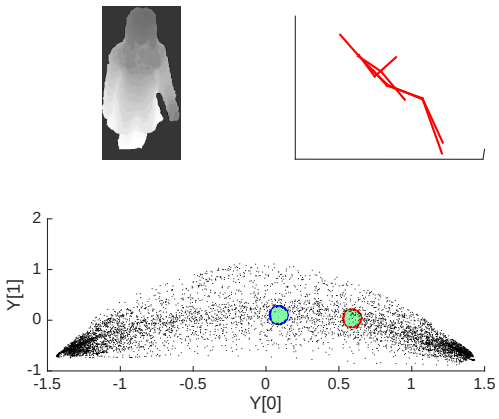


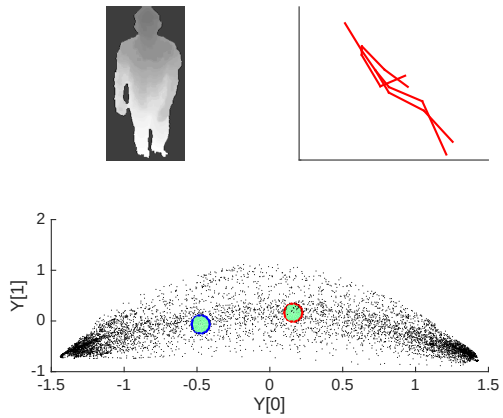
Figure 25: Shows the dynamics quality model (colour) as a function of pose vector and movement stage. The black dots represent frames from the Subject 9 Normal 1 sequence after its division into gait cycles. The strongest indication of normality is the value of Y[0]. Y[2] has a much broader range of normal values.



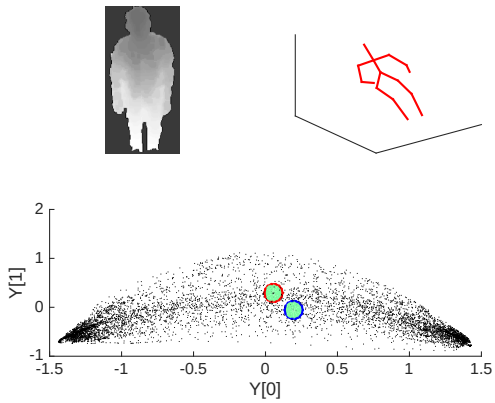
(a) Subject 7 RL frame 116. Kinect skeleton measures left knee slightly in front of right but image shows to them at equal depth.



(b) Subject 10 LL frame 365. Kinect skeleton measures right foot slightly in front of the left.



(c) Subject 9 RL frame 256. In this frame the network measurement is worse than the Kinect's. We think the presence of the subject's arm over the left knee causes the network to think the knee is raised.



(d) Subject 9 RL frame 188. In the same sequence when the arm is not covering the knee the network measurement is in line with the other R/LL sequences.

Figure 26: Shows the images, label skeletons and their corresponding points on the pose manifold (red) and the networks prediction (blue) for frames in the 3 tested L/RL sequences where both feet are on the same step. In LL sequences we expect the pose vectors to remain in the negative $Y[0]$ region of the manifold and RL sequences the positive. We find the Kinect tends to over measure this phase of the movement which causes these sequences to not always be properly classified as abnormal. In the majority of cases the network measurements do a better job at identifying the abnormality.

6.4 Processing Time and Memory Requirements.

On the GeForce GTX 750 GPU the average forward pass time is 9.68827 ms. This is the time required to pass one image through the network and produce the pose vector. Even allowing some additional time for the image pre-processing on such hardware this system could operate in real time at near 100 fps. On a single CPU the average forward pass time is 265.556 ms, a frame rate of around 4. This is comparable to the speed of Shotton et al.'s algorithm [57] which runs at ∼200 fps on the Xbox 360 GPU.

A single image and its activations require 8.68 MB of memory. The network's weights require 227.5 MB.

7 Discussion

7.1 Data Limitations

Due to inaccuracy of the ground truth provided by the Kinect skeletons there is not a completely consistent mapping between image and the pose vector. The effect the inclusion of erroneous skeletons has had on the final accuracy of the network is difficult to predict. In [4] it was shown that when reducing the loss weighting given to outliers by using Tukey's biweight function as a measure of loss instead of the L2 norm, which was used in this work, final network performance was improved by between 8 and 10%. This was for 2D joint position data produced through hand labelling, with a CNN extracting these positions from RGB images. We imagine that this ground truth is likely to be more accurate than that produced by the Kinect skeletons which would indicate that we could stand to make even greater gains in accuracy than those found in [4]. Whilst the implementation of this loss function could have improved results a more effective solution would have been creating a new dataset using motion capture data as ground truth. This would have probably reduced the number of erroneous labels to a negligible amount. However, it was decided early on that creating a new dataset of the size and variety required was probably not possible given limited timescale and availability of motion capture equipment. We instead focused on using the pre-existing SPHERE staircase dataset so as to verify the suitability of CNNs for this task before investing in producing new datasets.

Another alternative considered was using publicly available datasets. The most suitable was

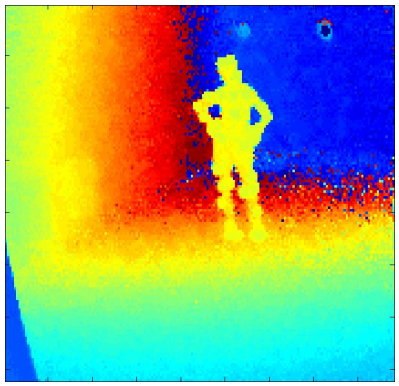


Figure 27: An example of the depth images contained in the Human3.6M dataset [27]. This was produced by a Mesa SwissRanger SR4000 ToF depth sensor. We deemed these images too different from the kind produced by the depth sensors employed in SPHERE to be useful for augmenting training data or assessing system performance.

Human3.6M [27] which contains depth images and accurate 3D joint position measurements from a motion capture system. The depth images in Human3.6M were a captured using a Mesa SwissRanger SR4000 ToF depth sensor, an example is shown in figure 27. These sensors are an order of magnitude more expensive than the Kinect. The images they produce were deemed too dissimilar to those from those produced by the sensors to be installed in the SPHERE homes to provide a valid assessment of performance.

7.2 Measuring accuracy

Despite the issues with the ground truth it has been shown that the network is able measure the pose vector with an accuracy close enough to be compatible with the SPHERE movement quality analysis system in the majority of frames. However, due to the abstract nature of the pose vector it is difficult to present results with an obvious and meaningful measure as is done in the literature concerned with full joint position measurements. Such papers usually present results in terms of the percentage of correctly located joints within varying accuracy limits. In our case the distance between poses is measured not in meters or pixels but in the distance in the space of the pose manifold. Therefore, we believe the clearest evidence of accuracy is found from visually comparing the ground truth with the predictions as was presented in figures 21 and 17. However even this comparison can be misleading in cases where the ground

truth is itself not accurate. We have attempted rectify this by identifying these cases through comparing the Kinect skeletons with the images annotating the errors in each sequence with the correct cause as was shown in figure 23.

Errors that the network does make seem to be explainable based on aspects of the dataset. For the least accurate sequence, Subject 2 Normal 3, the labelled pose vectors were found to exist in a sparse region of the manifold such that there were no other examples from which the network could be expected to infer the labelled pose. Such under-represented poses could be remedied by capturing more data. Alternatively, a reduction in the number of dimensions in the pose representation would reduce the variability needed to be modelled by the network and therefore the variability of data needed. In [68] the accuracy of the normality assessment measurements was analysed as a function of degree of dimensionality for the pose representation. There was found to be very little difference (a 0.01 reduction in probability of correct identification of abnormal frames) between using 2 and 3 dimensions. The reason for this is evident in the dynamics model's normality distribution which was shown in figure 25; almost the entire range of the third component for points on the manifold is given an equal probability of normality at all points in the movement cycle. This also has the implication that the $Y[2]$ component, which is responsible for 36.48% of the total error, has little to no effect on the pose quality measurement. Although this analysis would not necessarily hold if the system were being applied to other movement types.

7.3 Subject Specific Networks

We found that the accuracy of the predicted pose vectors could be improved from 0.1565 to 0.1129 by fine-tuning the networks on data of the subjects being measured. Since identification is required for personalised movement quality results to be stored, it seems reasonable that a personalised network could be selected by the system on the fly. However, for this to be implemented in SPHERE homes it will require training data from the residents. Whilst this may be possible to automate when using Kinect skeleton data, if, as planned, we switch to using motion capture data in the future then it probably becomes infeasible. Mixing data is also probably not possible since the differences in the form of the joint positions captured by the two would likely produce incompatible manifolds, or strange mappings if projected to

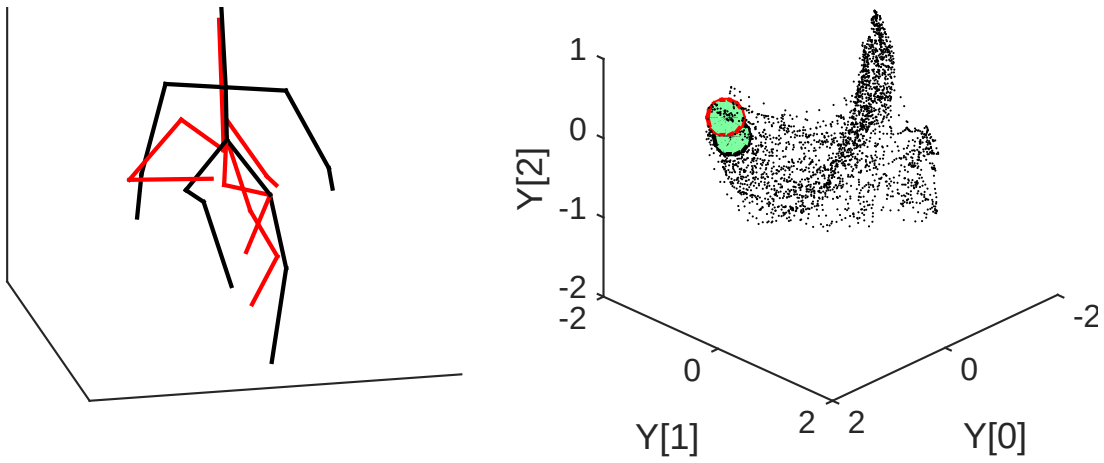


Figure 28: Shows a regular skeleton in black and a irregular one in red. The pose vectors for these two when projected onto the manifold are plotted on the right. They recieve very similar mappings. This shows that we are not neccessarily guaranteed that pose vectors that are classified as normal actually come from normal skeletons.

the same one.

7.4 Abnormal Poses

There is one fundamental issue with the use of the manifold pose representation as it currently stands. This is in its handling of abnormal poses which can be defined as any pose not contained in the 'normal' training data which is used to build the manifold representation and the models of normal pose and dynamics in the SPHERE system.

As it currently stands the manifold gives no way to identify abnormal poses. For example in figure 28 we take a skeleton, shown in black, and multiply it by random values, producing the highly irregular skeleton shown in red. We then compute the manifold projection for this irregular skeleton which is plotted against the original point. We see that this highly irregular pose is placed very close to normal poses and even, in this case, to the original pose. In general, all skeletons mapped on to the manifold are placed somewhere close to a pre-existing point. This means that when using the original Kinect system there is no guarantee that a pose that falls in the region of normality is actually normal.

The CNN by itself does not prove a natural fix to this problem. We made a collection

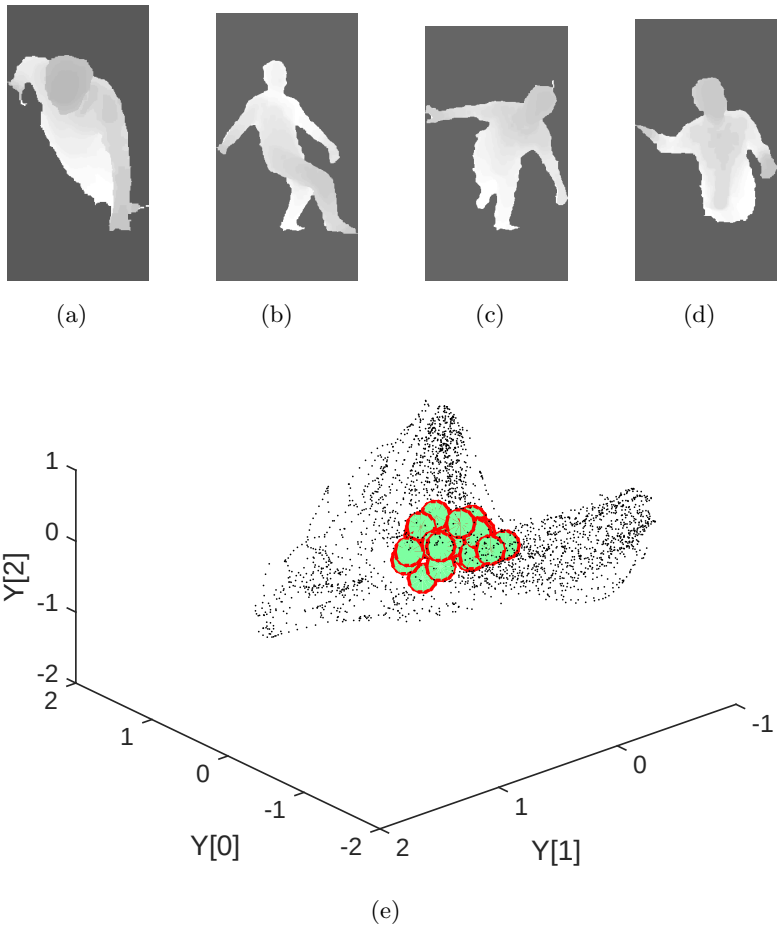


Figure 29: (a)-(d) shows some examples of the 31 abnormal poses that were tested with the CNN. We find that, given images far from those in the training data, the network predicts outputs clustered around the origin.

of images containing abnormal poses, some of which are shown in figure 29, and ran them through the CNN. The outputs were clustered randomly around the origin as shown in 29e. Whilst this is better than having them distributed around the manifold as with the skeleton projection method, it still does not allow identification of abnormality based off the pose vector.

The obvious way to solve this problem is to expand the training data to include abnormal poses. This would mean that there would be specific regions of the manifold which contained abnormal poses, and the CNN could be trained to map the images to them. This however would require the manifold to model much more variation which would probably require more

dimensions making the quality analysis and the CNN mapping to the space more complicated. It would also require vastly more data, avoiding this was one of the main attractions of using the reduced representation in the first place. There is also the problem that the Kinect will not be able to measure many abnormal poses.

A better solution might be to find some way for the CNN to distinguish between normal and abnormal poses itself. One way to do this may be to add a 4th dimension to the CNN's objective which encodes abnormality. This would have to be computed from full joint position data perhaps in a manner similar to the skeleton similarity cut used for excluding noisy skeletons from our dataset in Section 5.2.1. Another possibility may be to encode a similar abnormality measure in the regular 3 components by scaling the distance of the projected points from the manifold by the skeleton abnormality. We intend to work with the authors of [44] to find a solution to this problem in future work.

7.5 Directions of Future Work

Firstly, we aim to find a solution to the problem of representing abnormal poses as discussed in the previous section.

Another main aim for future work will be to test the CNN's ability to handle poses which the Kinect cannot, mainly the sitting-standing motions that are hoped to be analysed in the SPHERE homes. This will require creating a dataset using motion capture for joint position ground truth.

Also we would like to build a dataset of a regular gait and/or stair ascent motion viewed from multiple angles. One challenge in doing this is that depth sensors typically suffer from interference when multiple sensors are viewing the same scene. Repeating the motion and filming from multiple angles is not an ideal solution. It would be better to capture the same sequence from multiple angles simultaneously so that analysis of accuracy against viewing angle is not dependent on the recorded motion. The most efficient way to do this may be to render synthetic depth images from the motion capture data in a manner similar to [57]. This would have the added benefit that if the type of depth sensor to be used in the SPHERE houses were to change in the future, a new set of training could be re-rendered using the characteristics of that particular sensor.

8 Conclusion

The main aim of this project was to develop a flexible system for extracting pose descriptors from depth images to work as a component of the SPHERE movement quality analysis system. We have achieved this goal through a novel application of convolutional neural networks to pose estimation from depth images. We have analysed the performance of our system on the SPHERE staircase 2014 dataset and show we are able to measure the pose with a mean error distance of 0.0874 in that space. We showed that the level of accuracy achieved is suitable for the intended application of movement quality analysis and will be able to run at close to 100 fps. We believe that given the correct data this system will be able to meet the two further aims of the project in being able to work for a range of movements, and at a range of viewing angles. To test this belief we intend to create a dataset of a sitting-standing motion, and a second dataset of a stair ascent motion using motion capture joint position data for ground truth.

References

[1] Ankur Agarwal and Bill Triggs. Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58, 2006.

[2] Luís a Alexandre. 3D Object Recognition using Convolutional Neural Networks with Transfer Learning between Input Channels. *13th International Conference on Intelligent Autonomous Systems, Springer*, 2014.

[3] Andreas Baak, Meinard Muller, Gaurav Bharaj, Hans Peter Seidel, and Christian Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1092–1099, 2011.

[4] Vasileios Belagiannis, Christian Rupprecht, Gustavo Carneiro, and Nassir Navab. Robust Optimization for Deep Regression.

[5] León Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. *Proceedings of COMPSTAT'2010*, pages 177–186, 2010.

[6] M. Brand. Shadow puppetry. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2, 1999.

[7] Massimo Camplani and Luis Salgado. Adaptive spatio-temporal filter for low-cost camera depth maps. *2012 IEEE International Conference on Emerging Signal Processing Applications, ESPA 2012 - Proceedings*, pages 33–36, 2012.

[8] Kai-chi Chan, Cheng-kok Koh, and C S George Lee. A 3-D-Point-Cloud System for Human-Pose Estimation. 44(11):1486–1497, 2014.

[9] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the Devil in the Details: Delving Deep into Convolutional Nets. *arXiv preprint arXiv: ...*, pages 1–11, 2014.

[10] Lulu Chen, Hong Wei, and James Ferryman. A survey of human motion analysis using depth imagery. *Pattern Recognition Letters*, 34(15):1995–2006, 2013.

[11] Xianjie Chen and Alan Yuille. Articulated Pose Estimation by a Graphical Model with Image Dependent Pairwise Relations arXiv : 1407 . 3399v1 [cs . CV] 12 Jul 2014. pages 1–10.

[12] Ronald R. Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[13] Ben Crabbe. Feature Descriptors for Gait Analysis from Depth Sensors: Litrature Review. pages 1–37, 2015.

[14] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *International Conference on Machine Learning*, pages 647–655, 2014.

[15] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[16] A. Elgammal and Chan-Su Lee Chan-Su Lee. Inferring 3D body pose from silhouettes using activity manifold learning. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, 2, 2004.

[17] A. Elgammal and Chan-Su Lee Chan-Su Lee. Separating style and content on a nonlinear manifold. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, 1, 2004.

[18] Litong Feng, Lai Man Po, Xuyuan Xu, Ka Ho Ng, Chun Ho Cheung, and Kwok Wai Cheung. An adaptive background biased depth map hole-filling method for Kinect. *IECON Proceedings (Industrial Electronics Conference)*, pages 2366–2371, 2013.

[19] Mike Folk, Gerd Heber, and Quincey Koziol. An overview of the HDF5 technology suite and its applications. *Proceedings of the EDBT/ . . .*, pages 36–47, 2011.

[20] David a. Forsyth, Okan Arikan, Leslie Ikemoto, James O'Brien, and Deva Ramanan. Computational Studies of Human Motion: Part 1, Tracking and Motion Synthesis. *Foundations and Trends in Computer Graphics and Vision*, 1(2/3):77–254, 2005.

[21] Matheus Giovanni and Soares Beleboni. A brief overview of Microsoft Kinect and its applications. pages 1–6.

[22] Ross Girshick, Jeff Donahue, Trevor Darrell, U C Berkeley, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, pages 2–9, 2014.

[23] H. Gonzalez-Jorge, B. Riveiro, E. Vazquez-Fernandez, J. Martínez-Sánchez, and P. Arias. Metrological evaluation of Microsoft Kinect and Asus Xtion sensors. *Measurement: Journal of the International Measurement Confederation*, 46(6):1800–1806, 2013.

[24] Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with Microsoft Kinect sensor: A review. *IEEE Transactions on Cybernetics*, 43(5):1318–1334, 2013.

[25] Thomas Helten, Andreas Baak, Meinard Müller, and Christian Theobalt. Full-body human motion capture from monocular depth images. *Lecture Notes in Computer*

Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8200 LNCS:188–206, 2013.

[26] Yap Wooi Hen and Raveendran Paramesran. Single camera 3D human pose estimation: A Review of current techniques. *2009 International Conference for Technical Postgraduates (TECHPOS '09)*, pages 1–8, 2009.

[27] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2014.

[28] Arjun Jain, Jonathan Tompson, Mykhaylo Andriluka, Graham W. Taylor, and Christoph Bregler. Learning Human Pose Estimation Features with Convolutional Networks. *arXiv preprint arXiv: ...*, pages 1–10, 2013.

[29] Arjun Jain, Jonathan Tompson, Yann LeCun, and Christoph Bregler. MoDeep: A Deep Learning Framework Using Motion Features for Human Pose Estimation. pages 1–15, 2014.

[30] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *ACM Conference on Multimedia*, 2014.

[31] A. Karpathy. Stanford cs231n: Convolutional neural networks for visual recognition. University Lecture. Available at `http://cs231n.github.io/` as of May 5th, 2015, 2015.

[32] K. Khoshelham. Accuracy Analysis of Kinect Depth Data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII(5):133–138, 2012.

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pages 1097–1105, 2012.

[34] Neil D Lawrence. Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data. *Advances in Neural Information Processing Systems 16*, pages 329–336, 2004.

[35] Y. LeCun, L. Bottou, G. Orr, and K. Muller. *Neural Networks: Tricks of the Trade*. 1998.

[36] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2323, 1998.

[37] Yann a. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus Robert Müller. Efficient backprop. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7700 LECTU:9–48, 2012.

[38] S. Li and Antoni B. Chan. 3D Human Pose Estimation from Monocular Images with Deep Convolutional Neural Network. *Asian Conference on Computer Vision (ACCV)*, 2014.

[39] Sijin Li and Antoni B Chan. Heterogeneous Multi-task Learning for Human Pose Estimation with Deep Convolutional Neural Network. *CPVR2014*, 2014.

[40] Zhao Liu, Jianke Zhu, Jiajun Bu, and Chun Chen. A survey of human pose estimation: The body parts parsing based methods. *Journal of Visual Communication and Image Representation*, 32:10–19, 2015.

[41] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104:90–126, 2006.

[42] Chuong V. Nguyen, Shahram Izadi, and David Lovell. Modeling kinect sensor noise for improved 3D reconstruction and tracking. In *Proceedings - 2nd Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization and Transmission, 3DIMPVT 2012*, pages 524–530, 2012.

[43] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In *CVPR*, pages 1717–1724, 2014.

[44] A. Paiement, L. Tao, S. Hannuna, M. Camplani, D. Damen, and M. Mirmehdi. Online quality assessment of human movement from skeleton data. *BMVA press*, 2014.

[45] A. Pavement. Online quality assessment of human movements from skeleton data. Online. Available at `http://www.irc-sphere.ac.uk/work-package-2/movement-quality` as of September 14th, 2015, 2015.

[46] Tomas Pfister, Karen Simonyan, James Charles, and Andrew Zisserman. Deep Convolutional Neural Networks for Efficient Pose Estimation in Gesture Videos. *Asian Conference on Computer Vision (ACCV)*, 2014.

[47] Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[48] Ronald Poppe. Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding*, 108(1-2):4–18, 2007.

[49] R. Rosales, M. Siddiqui, J. Alon, and S. Sclaroff. Estimating 3D body pose using uncalibrated cameras. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 1:1–8, 2001.

[50] Rómer Rosales and Stan Sclaroff. Inferring body pose without tracking body parts. *Cvpr*, 2(June):721–727, 2000.

[51] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors, 1986.

[52] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, C. V. Jan, J. Krause, and S. Ma. ImageNet Large Scale Visual Recognition Challenge. *arXiv preprint arXiv:1409.0575*.

[53] Max Schwarz, Hannes Schulz, and Sven Behnke. RGB-D Object Recognition and Pose Estimation based on Pre-trained Convolutional Neural Network Features. *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[54] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. OverFeat : Integrated Recognition , Localization and Detection using Convolutional Networks. *arXiv preprint arXiv:1312.6229*, pages 1–15, 2013.

[55] H. Seung, H. Sompolinsky, and N. Tishby. Statistical Mechanics of Learning from Examples. Physical Review A, 1992.

[56] Ali Sharif, Razavian Hossein, Azizpour Josephine, Sullivan Stefan, and K T H Royal. CNN Features off-the-shelf : an Astounding Baseline for Recognition. *Cvpr'2014*, 2014.

[57] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Real-time human pose recognitiom in parts from single depth images. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[58] Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, and Alex Kipman. Efficient human pose estimation from single depth images. *Decision Forests for Computer Vision and Medical Image Analysis*, pages 175–192, 2013.

[59] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recoginition. *ICLR*, 2015.

[60] C. Sminchisescu and B. Triggs. Kinematic jump processes for monocular 3D human tracking. *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, 1, 2003.

[61] Cristian Sminchisescu. 3D Human motion analysis in monocular video techniques and challenges. *Proceedings - IEEE International Conference on Video and Signal Based Surveillance 2006, AVSS 2006*, 2006.

[62] Jan Smisek, Michal Jancosek, and Tomas Pajdla. 3D with Kinect. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1154–1160, 2011.

[63] Andrews Sobral. {BGSLibrary}: An OpenCV C++ Background Subtraction Library. In *IX Workshop de Visão Computacional (WVC'2013)*, Rio de Janeiro, Brazil, June 2013.

[64] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. *ICLR*, 2015.

[65] A Stoyanov, T Louloudi, A Andreasson, H Lilienthal. Comparative evaluation of range sensor accuracy in indoor environments. *Proceedings of the 5th European Conference on Mobile Robots, ECMR 2011*, pages 19–24, 2011.

[66] Christian Szegedy, Scott Reed, Pierre Sermanet, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, pages 1–12, 2014.

[67] T. Tangkuampien and D. Suter. Real-Time Human Pose Inference using Kernel Principal Component Pre-image Approximations. *Procedings of the British Machine Vision Conference 2006*, pages 62.1–62.10, 2006.

[68] Lili Tao, Adeline Paiement, Dima Damen, Majid Mirmehdi, Sion Hannuna, Massimo Camplani, Tilo Burghardt, and Ian Craddock. A Comparative Study of Pose Representation and Dynamics Modelling for Online Motion Quality Assessment. *Computer Vision and Image Understanding - SI: Assistive Computer Vision and Robotics, Under review Submitted in March 2015.*

[69] Alexander Toshev and Christian Szegedy. DeepPose: Human Pose Estimation via Deep Neural Networks. *CVPR*, 2014.

[70] Raquel Urtasun, David J. Fleet, Aaron Hertzmann, and Pascal Fua. c. *Proceedings of the IEEE International Conference on Computer Vision*, I:403–410, 2005.

[71] Vladimir Vapnik, Esther Levin, and Yann Le Cun. Measuring the VC-Dimension of a Learning Machine. Neural Computation, 1994.

[72] Pichao Wang, Wanqing Li, Zhimin Gao, Jing Zhang, Chang Tang, and Philip Ogunbona. Deep Convolutional Neural Networks for Action Recognition Using Depth Map Sequences. 2015.

[73] Mao Ye, Xianwang Wang, Ruigang Yang, Liu Ren, and M Pollefeys. Accurate 3D pose estimation from a single depth image, 2011.

[74] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How Transferable are Features in Deep Neural Networks? *Nips14*, 27, 2014.

[75] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE Multimedia*, 19:4–10, 2012.

[76] Youding Zhu, Behzad Dariush, and Kikuo Fujimura. Controlled human pose estimation from depth image streams. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops*, 2008.

[77] Z. Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, 2(2), 2004.