# Feature Descriptors for Gait Analysis From Depth Sensors

August 30, 2015

## Contents

# 1 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 requires patients to travel to a gait assessment laboratory which is far from ideal for patients who have difficulty walking.

This problem, and a range of other healthcare challenges, is 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 [**?**, **?**] which assesses the quality of a subjects movement using inexpensive RGB-D cameras such as the Microsoft Kinect.

Currently this system uses joint position information captured by the OpenNI skeleton tracking software, based on the algorithm of [**?**]. This skeleton tracking software infers the 3D coordinates of each of the body's relevant joints producing a $n_{joints} \times 3$ dimensional vector. This data is then processed using a manifold learning method, Diffusion maps [**?**], to reduce the dimensionality of the data. This method builds up a 3D representation of the types of body configurations displayed in a training dataset containing footage of the motion being measured. New skeleton data is then projected onto this manifold. This effectively parameterises the motion, removing the redundant information contained in the skeleton data, and enabling simple comparison of poses. [1] Finally, a statistical model of normal gait is built up from the training data using these pose vectors. New data is compared with this model producing a quality score for both pose and dynamics on a frame-by-frame basis.

[1] We will refer to the projected points in this space as the pose vector, and to the skeleton data as the body configuration or joint position vector.

Since this system uses data driven, machine learning methods to learn both the manifold representation of pose and the model of normal motion, it can be applied to other types of movement quality assessment such a sports movement optimisation or physiotherapy exercise coaching. The system has been applied to a sitting-standing motion, to punching motions in boxing $^{howtositehttp://www.irc-sphere.ac.uk/work-package-2/movement-quality?}$ and to people walking upstairs.

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 attached to the ceiling above the stairs is fit for use with the system. Increasing amount of usable data requires more intrusive camera placement which is to be avoided. The skeleton trackers also perform poorly when props are involved in the scene, 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.

The aim of this project is to develop a tailor made system for determining the reduced pose vector directly from RGB-D footage. To enable the flexibility of the rest of the system we require this new component to exhibit the same flexibility as the rest of the system by being able to record a wide range of motions. It should also work with an effective accuracy under the kinds of viewing angles produced by practical, unobtrusive, in home camera placements. This requires a data driven approach since the pose representation we wish to infer is not fixed, differing based on the body configurations presented in training data.

The methodology we find most suited to this task is a convolutional neural network (CNN). CNN's are a supervised learning method for extracting features, e.g. the pose vector, from images. Given training images labelled with the expected output the network extracts progressively higher level features representations leading to the final pose vector. Following training the network is then able to generalise to unseen data, producing an output inferred from the examples it has seen.

CNNs have been effectively applied to 2D human pose estimation from RGB images [**?**, **?**, **?**, **?**, **?**, **?**, **?**] where the positions of joints in the image plane were inferred. In [**?**] they were also applied to 3D joint position estimation from RGB, where they were shown to have reasonable

accuracy from a range of viewing angles when trained with data captured by 4 cameras placed around the subjects. They have also been shown to benefit from depth depth data in the tasks of object detection [**?**], object pose estimation [**?**] and object recognition [**?**].

For assessing the effectiveness of our solution we focus on the staircase ascent motion, as this is the motion for which we possess the largest dataset. Refered to as the SPHERE staircase 2014 dataset [**?**], this 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. Each Frame provides RGB, depth and 3D joint position data. The accuracy of our predicted pose vectors are measured by computing the mean squared error (MSE) of the networks predicted pose vectors against the label values. We also measure the change in overall system performance (how well the measured gait quality score matches the score labelled by a trained physiotherapist).

To the best of our knowledge this project will be the first time that CNNs will be applied to a 3D human pose estimation task on RGB-D images. It will also be a novel combination of CNNs and manifold learning methods. We find that this makes the CNN easier to train and more effective overall since it has far less outputs to specify. If this is proved to be the case it could potentially be applied to other tasks that have been attempted with CNNs such as human action recognition.

# 2 Background and Related Work

## 2.1 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 [**?**].

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 RGB-D sensors came on the market, since then there has been an explosion in their use in the computer vision community [**?**].

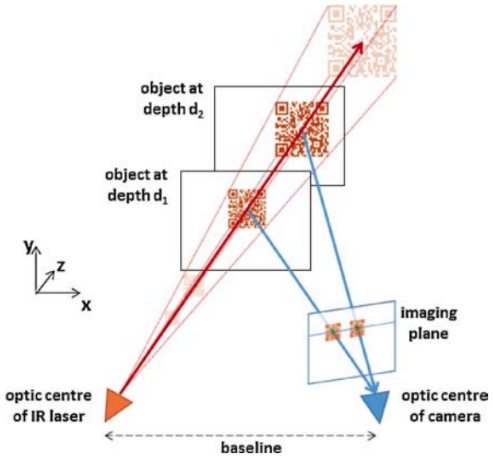



Figure 1: The process by which depth is computed from triangulation of structured light. From [?]

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, an infrared camera, which together make up the depth sensor, and an 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 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 [?, ?] as shown in figure 1.

### 2.1.1 Sensor Performance

Most of the studies reported below have focused on the Microsoft Kinect however both sensor contain the same depth sensing system and have been shown to perform equivalently when compared [?], hence we report the findings based off of the 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 [?].

Stoyanov et al. [?] 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

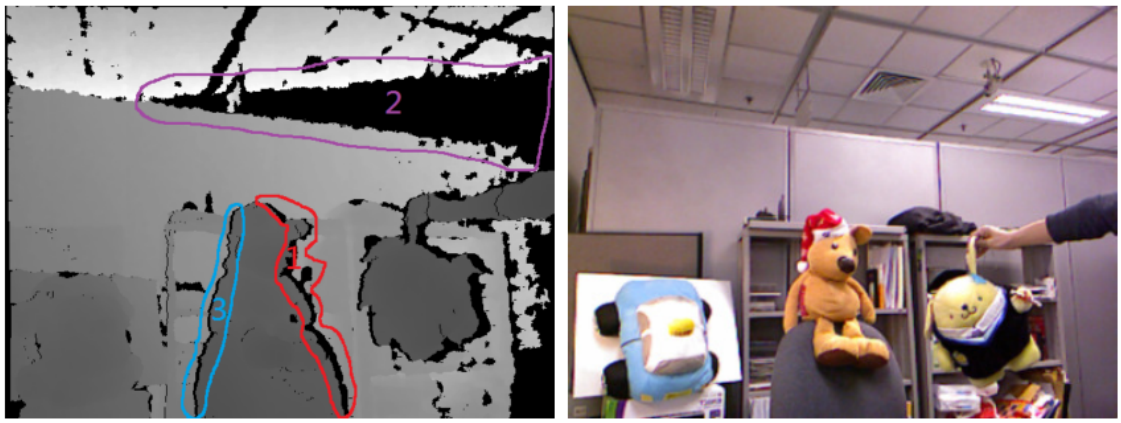



Figure 2: Shows the holes in structured light depth data due to the different perspectives of IR projector and senor (regions 1 and 3) and due to the surface of reflection being roughly 5m away and at a large angle(region 2) From [?]

a ground truth of expensive and low fps laser depth scanner measurements. They find that within a range of 3.5m the Kinect outperforms that of the ToF sensors and is comparable to the laser scanner, and that outside of this range the accuracy falls considerably.

Both Khoshelham & Elberink [?] and Smisek et al. [?] have measured this effect experimentally comparing Kinect measurements with those from high performance laser scanners. They find temporally fluctuating noise in the depth measurements increases quadratically with distance from the sensor so the depth precision decreases from about 0.5cm at 1m to 7cm at 7m. Nguyen et al. shows there to also be linearly increasing noise with lateral distance, and greatly increased noise on surfaces at greater than 70° angles [?]. 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.

It should be noted that each of these studies mentioned above [?, ?, ?] fail to consider the


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

environmental factors in the quality of the measurement. Fiedler & Muller [?] show that air draft can cause changes of the depth values up to 21mm at a distance of 1.5m, and temperature variations cause changes up to 1.88mm per 1° C. They also find a temperature dependant drift in the position of objects captured by the RGB camera.

## 2.2 Human Pose Estimation

Human pose estimation (HPE) is generally considered 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 due to the difficultly of the problem and the variety of applications such as video surveillance, humancomputer interaction, digital entertainment and sport science as well as medical applications.

This is a difficult task for a number of reasons. Firstly the human body has around 20 degrees of freedom [?], 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) methods 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 [?, ?, ?, ?, ?]) are generally divided into two approaches (e.g. by [?]); 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.

### 2.2.1 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 [1]. Such searches in 20 dimensions are computationally expensive and often lead to super real time frame rates [?]. 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 only 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 [?, ?].

Although the full pose space is very large and high dimensional it has been shown e.g. in [?, ?] that if considering only the movements in a well defined activity e.g. walking, then the pose data can be well represented by a low dimensional latent manifold. In a work closely related to our own Elgammal et al. [?] use a Local Linear Embedding method to generate a 1D manifold representation (embedded within a 3D space)$^{(FIGURE?)}$ of a walking motion from single sequences of silhouette images. They use a Generalised Radial Basis Function interpolation framework [?] (a form of neural network) to learn the nonlinear mappings from the manifold to silhouette image space and 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 that the same manifold representation is not tied to subject appearance and generalises naturally to multiple subjects, which is not the case in [**?**] (although they do introduce an solution for this problem in [**?**]). 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 sequence. 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. However we show that having unseen (in the training set) poses mapped to points away from the manifold, which is precisely what an undefined mapping causes, is the a good indication of ab $^{Sortthislater}$

Brand [**?**], 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 on of the key features of the SPHERE system is online measurement.

Similarly Urtasan et al. [**?**] used Scaled Gaussian Process Latent Variable Models (SGPLVM) [**?**] 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 [**?**] 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. This has some similarity with using a CNN, where the image is transformed in to progressively more concise feature representations, before the regression takes place. The disadvantage of using LLE to perform the final transformation is that it is contained to within the manifold space contained in pose training data.

Rosales et al [**?**, **?**] 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 both learn 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. Although 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.

### 2.2.2 Human Pose Estimation From Depth Images

With the advent low cost commodity depth sensors challenging aspects of RGB HPE such as the variability in human appearance and scene lighting were greatly simplified. RGB-D also provides richer data for inferring 3D structure; human poses which could be appear identical when projected onto the 2D image plane can be distinguished. Full body HPE methods from single depth images are review in [**?**]. With the Kinect sensor and its bundled 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 [**?**, **?**].

The Kinect SDK and OpenNI skeleton trackers apply some inter-frame tracking algorithms to the single frame pose measurements of [**?**]. In this work Shotton et al. leveraged a large mocap 3D joint position dataset which they re-targetted 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 forrest to perform this body part classification at each pixel using simple and computationally efficient pixel wise features. 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. [?] 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 [?]. 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 is suggested as an ideal way for increasing the view angle and subject invariance of our system.

Another discriminative method is [?] where they find geodesic extrema (which are expected to correspond to the feet, hands and head) from Dijkstras algorithm on a graph produced by connecting all depth pixels in the image into a map. These points are identified (as hands feet or head) by applying local shape descriptors around the area.

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

In a recent attempt Chan et al [?] 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.

## 2.3 Convolutional Neural Networks

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

$$\boldsymbol{Y} = F(\boldsymbol{Z}, \boldsymbol{W}) \tag{1}$$

where $\boldsymbol{Z}$ is an input image, the output $\boldsymbol{Y}$ is the inferred pose vector for this 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 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 such a 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 4. 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 5. Operating on each depth slice individually, i.e. each filters activation, 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 5. 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.

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 euclidean distance between the two. Using the backpropagation algorithm [**?**] 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 further technical discussion of CNNs in general can be found in the preceding work on this project [**?**].

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 [**?**] or the view point and shape feature of [**?**], the network is responsible for learning features its self based off

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

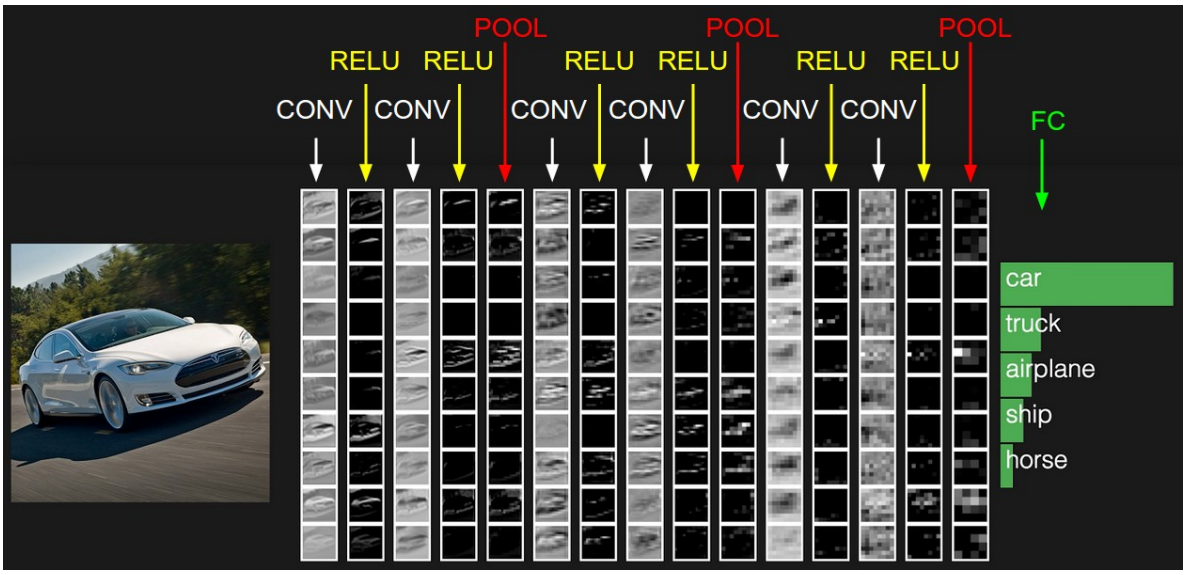



Figure 4: Shows a representation of the activations produced following a number of convolution layers, non-linearities (Rectified Linear Units, or ReLUs, are the function $max(0, x)$ and pooling layers. The network, an online demo from [**?**], is classifying images from the CIFAR-10 dataset using the ConvnetJS library[2]. )

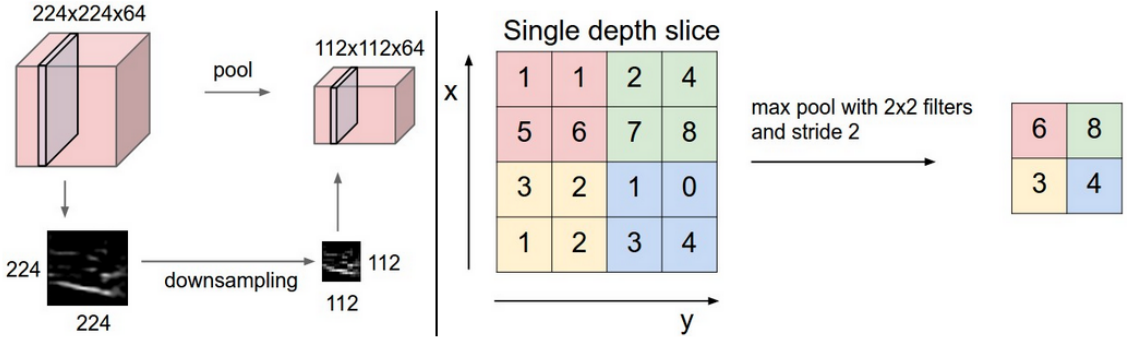



Figure 5: Shows the effect of pooling layers in a CNN. From [**?**]

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.

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 [**?**] 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 [**?**].

Training deep networks requires a large amount of data. Without this the network may begin to overfit 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 [**?**, **?**] 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.

$$E_{test} - E_{training} \propto \frac{c}{P} \tag{2}$$

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 [**?**].

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 [**?**]. 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 [**?**, **?**, **?**, **?**, **?**] have shown ImageNet trained networks producing better results than randomly initialised networks in a variety of task, and also on depth data [**?**, **?**]. 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.

### 2.3.1 Human Pose Estimation Using CNNs

Jain et al. [**?**] were the first to apply CNNs to human pose estimation. 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 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 [**?**] they extend their system to videos using RGB plus motion features e.g. optical flow.

In a similar work Toshev and Szegedy [**?**] used a multi resolution approach to find 2D joint locations. They used the architecture of [**?**] 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 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 [**?**] 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. [**?**] used the architecture of [**?**] (a 2013 ILSVRC 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. The found a 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,

finding this gives only a modest 0.3% improvement.

Tompson et al. [**?**]

# 3 Methods

## 3.1 Data Preprocessing

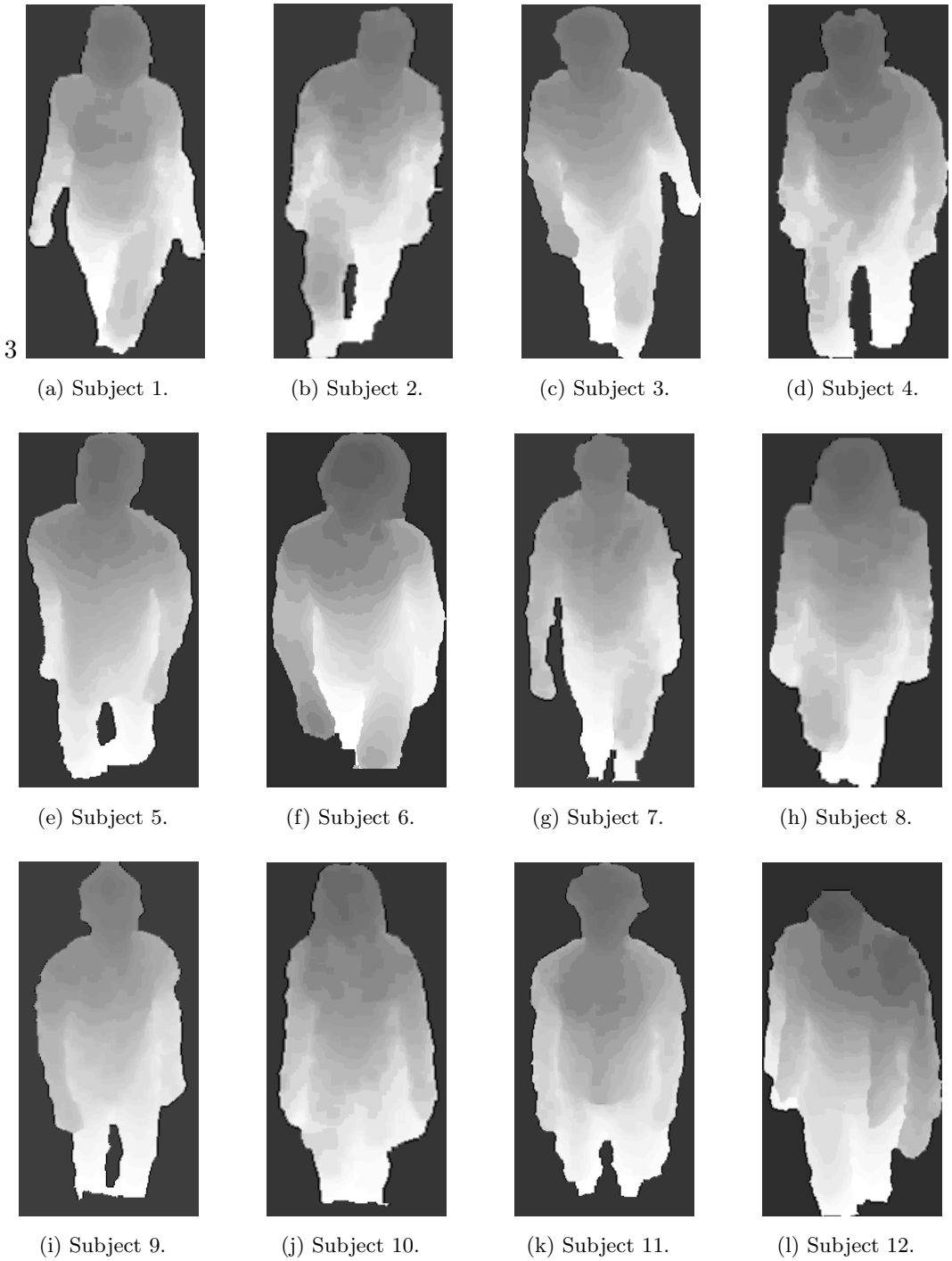
The dataset used in this project (SPHERE-staircase2014 dataset [**?**]) includes 48 sequences of 12 individuals walking up stairs, captured by a Asus Xtion depth sensor placed at the top of the stairs in a frontal and downward-looking position. It contains three types of abnormal gaits with lower-extremity musculoskeletal conditions, including freezing of gait and using a leading leg, left or right, in going up the stairs. 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 6.

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

### 3.1.1 Skeleton Preprocessing

These processes were developed by the authors of [**?**] 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. These are generally frames from the beginning and end of sequences, where the subject is outside the sensors optimum range, and any particularly noisy periods in the middle.

Specifically for this project we also take the mirror image of each skeleton. Combined with a mirror image of the depth image, this doubles our data.



(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 6: The subjects contained in the SPHERE staircase 2014 dataset.

Now the collection of remaining skeletons is given to the manifold learning stage. The method used is Diffusion Maps [?] with an adaptation to better handle the remaining noise and outliers present in the skeleton data. We refer to the original publications [?] and [?] for a full technical description. In essence Diffusion Maps retains the relative distances between data in the reduced space of the data. [?] 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 which is used in building the manifold determines its form. In the original works [?] and [?] the authors used only the 'normal' sequences from the first 6 subjects. The manifold this produces is shown in figure 7a. 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 7c. Once we switched to using the final manifold we observed an increase in accuracy of around 20%. This shows that the form of the manifold can have a significant effect on the accuracy of the CNN, and therefore that these results for gait analysis may not be a reliable indication of performance on other movements and manifolds.

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 7b. The skeletons of the 31 sequences from the other 6 subjects (and their flips) are then projected onto the manifold (again, see [?] and [?] 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.

To illustrate the meaning of the manifold coordinates the skeletons placed at the 3 corners are shown in figure 8. 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 8, seems to measure the vertical distance between skeletons. The skeleton with raised 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 disimilarity. 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 **??**.

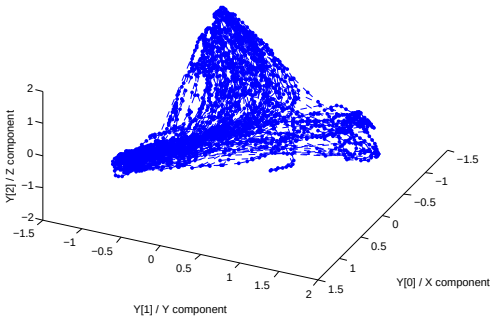
The meaning of the 3rd manifold coordinate is less clear. Figure 10 compares skeletons of minimum and maximum values for points of equal y[0], y[1], we are able to find no discernible pattern. This is reflected by the CNN as we find it far less accurate in this coordinate than the others. There are two observations which we use to improve the accuracy: 1) this coordinate can reasonably well modelled by a polynomial function of Y[0] and Y[1] as shown in 12. 2) This coordinate is found to be somewhat subject specific as seen in figure 11.

### 3.1.2 Depth Preprocessing

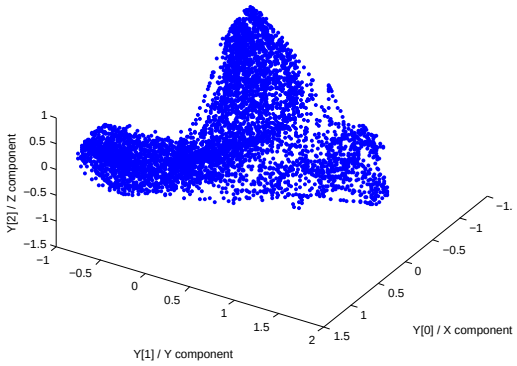
As described in section 2.1.1 depth images are generally incomplete and noisy. We first fill holes in the data using a simple method which iteratively paints in the maximum pixel values from the neighbouring cells. We initially experimented with using the method of [**?**] which combines noise filtering 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. [**?**]'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 [**?**] 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 [**?**].

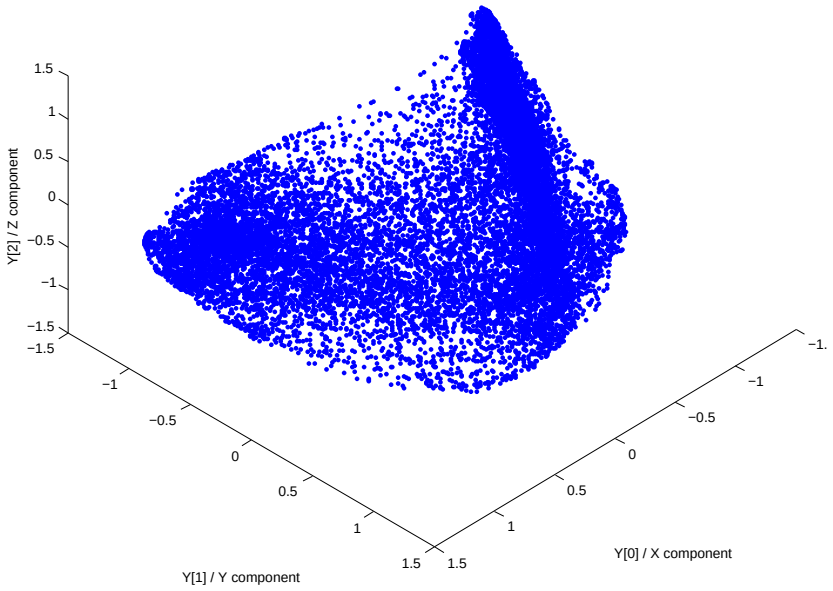
We found that the foreground masks produced often mistook sensor noise for moving objects. To fix this problem we apply a simple cleaning procedure to the masks which takes only the largest collection of positive pixels. This discards the small noise related blobs. We also apply a small erosion and dilation to get rid of noise effects connected to the largest blob. There is 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




(a) The manifold produced by normal sequences of subjects 1-6. This was the form used in [**?**]. 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. The larger variation in the z component (the vertical axis in the figure) led to lower network accuracy and incorrect normality analysis.

Figure 7

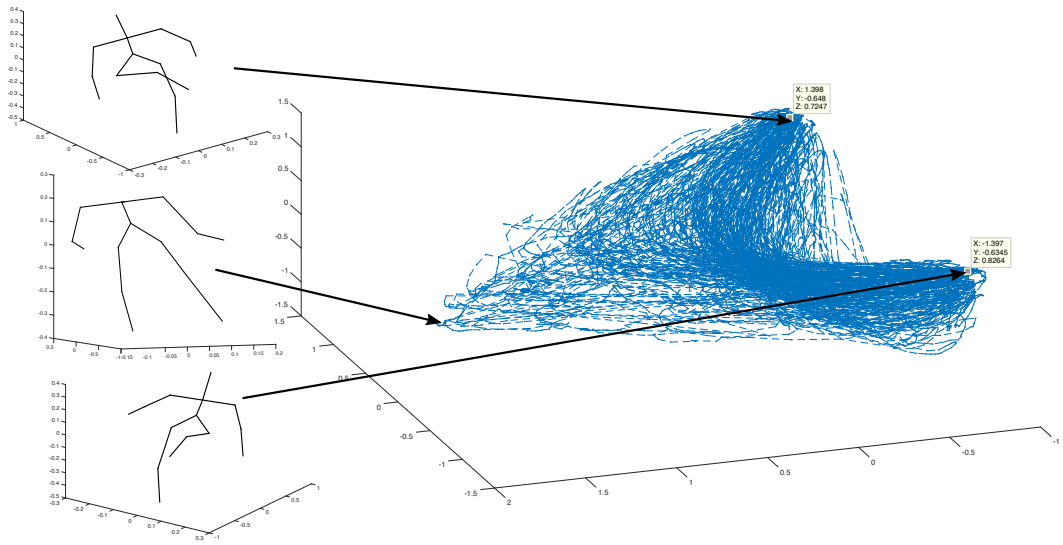


Figure 8: Normal gait sequences trace paths between the two corners on the right, with the postion of maximal knee flex the turning point. The left most coner of the manifold consists of elongated skeletons which tend to be measured when the subject is too close to the sensor.

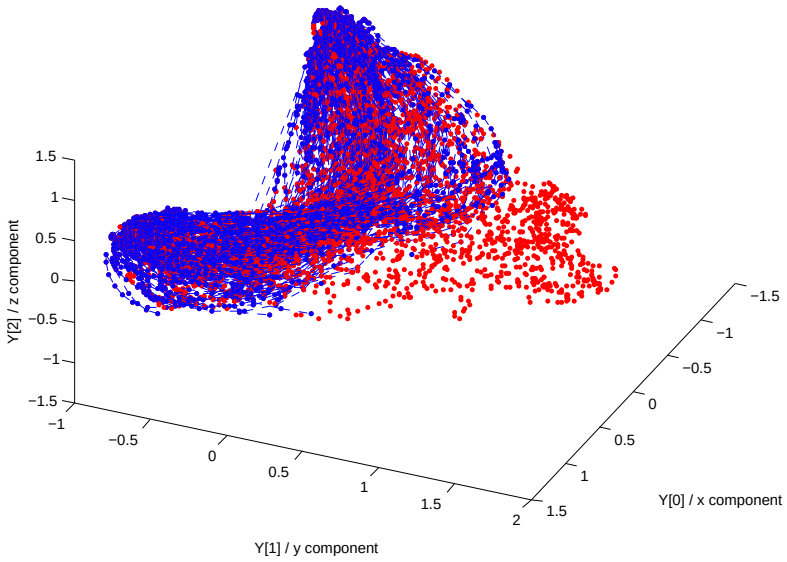
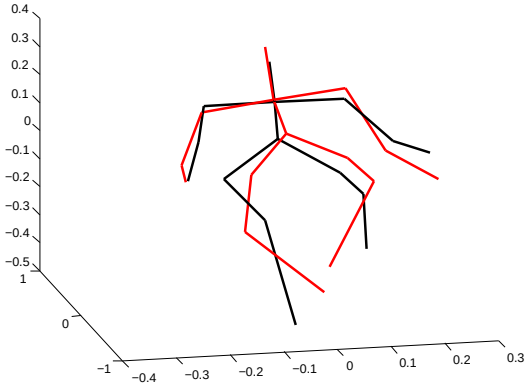


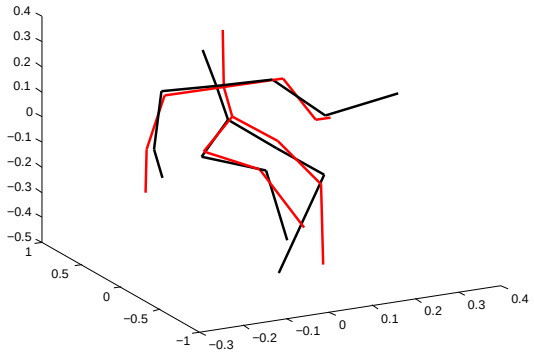

Figure 9: After image pre-processing we retain only those points shown in blue.




=-0.2211, black has Y[2] = 0.1888 ] =-0.0499 ,

(a) (b)

Red skeleton has Y[2 Red skeleton has Y[2




black has Y[2] = 0.4803.]

Figure 10: Shows skeletons within 0.02 of the same Y[0],Y[1] postition but with maximum difference in Y[3]. We find no clear difference between the Skeleton of minimum and maximum Y[2]. This is reflected by the CNN which measures this coordinate with a lower accuracy than the first two.

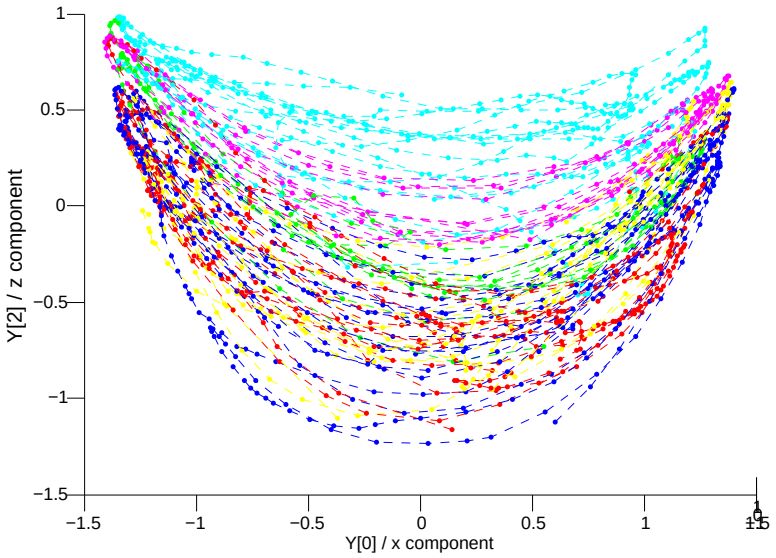



Figure 11: Each normal sequence of the first 6 subjects plotted in a different colour. We observe that Y[2] coordinates for each subject tend to ramain within a sub-region of full range. This finding promoted the use of subject finetuning which improves the accuracy of Y[2] predictions.

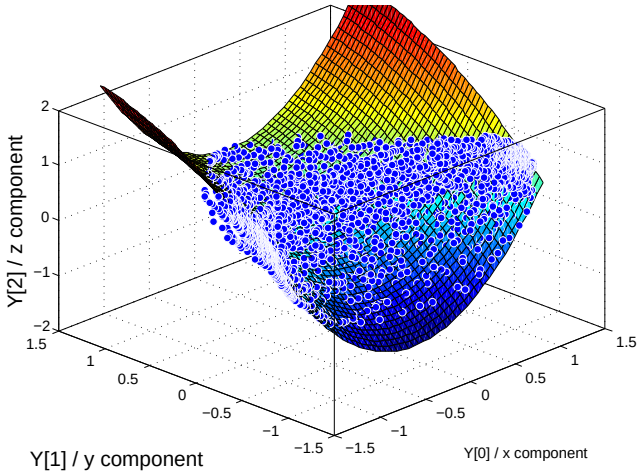



Figure 12: The gait manifold is reasonably well modelled by a polynomial function of the first two coordinates. We attempt to use this to improve the networks accuracy for Y[2] predictions.

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 we 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 the subtraction fails badly on some frames, particularly towards the end of a sequence. These images are discarded.

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 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 the 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

(a) Typical image before hole filling.

(b) Filter and filled using a simplifed version of [**?**].

(c) Filled using max fill method.

Figure 13: A comparison of hole filling methods. We use the max fill method.

arm swing. This value is subtracted from all foreground pixels before all being scaled to 0.5.

Next we crop the background area from each image so that each is left with the average width to height ratio of 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 preprocessing 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 greatest curvature was determined to lie at a width of 0.65 of the full image width (after initial cropping). We therefore define the point at which the the width of the mask exceeds this value as the shoulders. From this we find the position of the shoulders in each image. We found the mean position of the shoulders in all non-headless images to be 0.22 of the full image height. Then each image is 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 colour balance 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 15b, seemed to indicate that depth information was not being extracted effectively. ImageNet first layer filters,

(a) Initial colouring scheme.

(b) Mean background method.

(c) Final scheme.

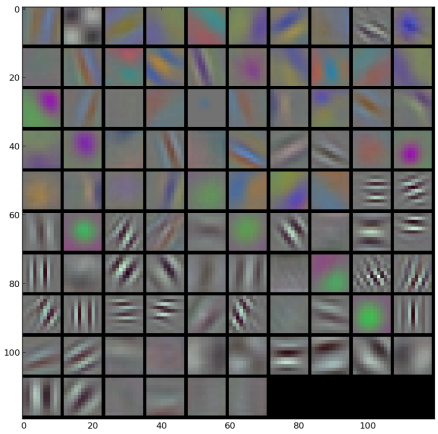
(d) Pre-processed RGB

Figure 14

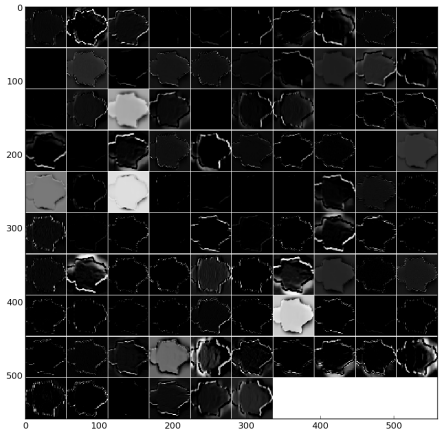
shown in figure 15a, generally respond to edges. It is clear that the very small differences in depth between parts of the body will produce far smaller activations than the huge difference between the figure and the black background. To encourage the network to focus on the smaller edges we tried setting the background of the images to the mean depth value around the waist of the subject, as shown in figure 14b. 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 15c. 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 minimum value 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 **??**.

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 their are so few of these frames that they represent rare outliers in the dataset. The presence of such rare examples in the training data has been shown to hinder final network performance by [**?**].

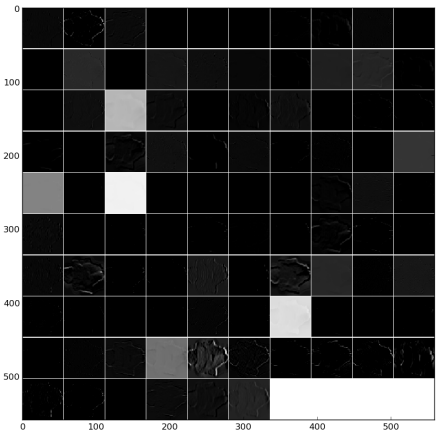
A common pre-processing step when working with CNNs is to subtract the mean image of



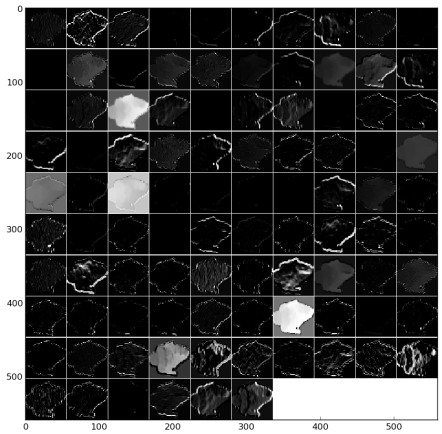

(a) The filters in conv1 of a imagenet pre-trained then finetuned AlexNet [**?**].




(b) Initial colouring scheme




(c) Mean background method




(d) Final scheme

Figure 15: The activations produced by the first layer filters for each of the colour schemes of figure 14.

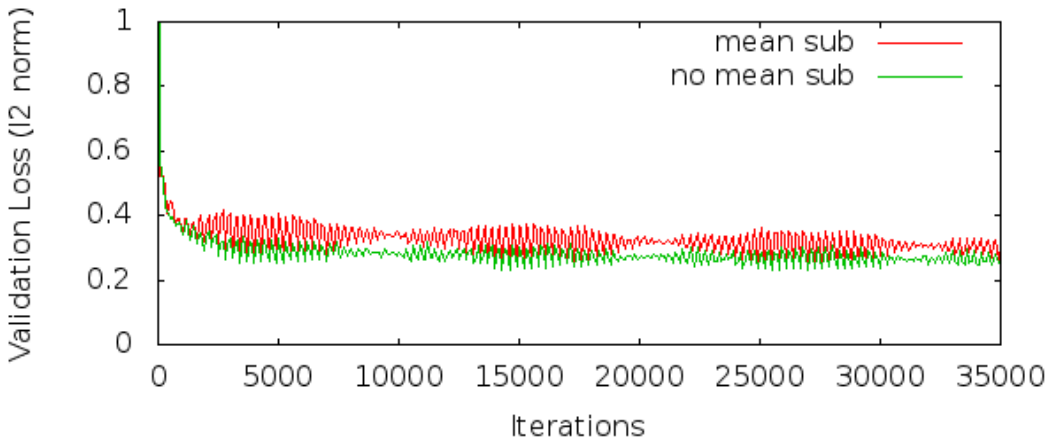



Figure 16: The accuracy of network predictions was seen to decrease when using mean subtracted data, hence we abandon this common practise.

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 [?, ?, ?]. After testing the accuracy of the network's trained on such mean subtracted data we found that this operation actually decreased the performance as shown in figure 16. However this test was only conducted on an ImageNet pre-trained AlexNet (see section3.3) , we have not studied the effect when training from scratch.

### 3.1.3 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 14d.

We measured the accuracy of an ImageNet pre-trained AlexNet (see section3.3), which 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 17 the replicated depth was shown to produce the most accurate regression. All other results reported here use a pure depth input image.

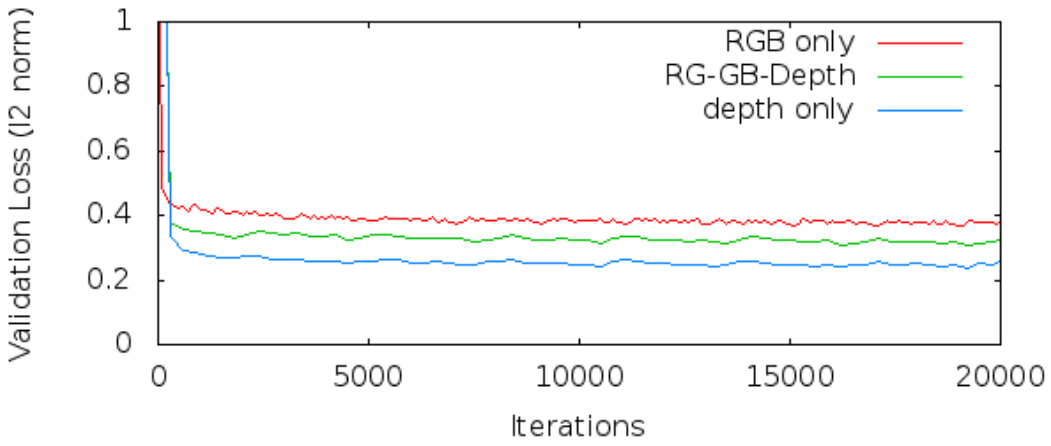


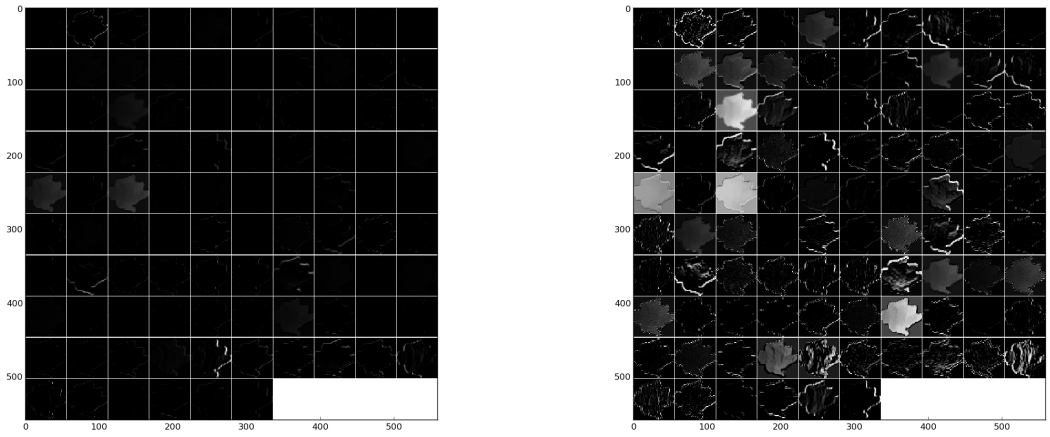

Figure 17: Network accuracy when using the RGB images only, the depth image replicated for each of the 3 channels only, and a combination of the two which consisted of 1 chanel of the average of red and green, 1 channel of green and blue averaged and the depth in the final channel.

### 3.2 Software

We use the open source CNN library Caffe [**?**] which is common among researchers including [**?**, **?**, **?**, **?**, **?**, **?**, **?**].

Caffe requires datasets to be in certain formats for training. When using multi-dimensional labels, as in our case, you are constrained to using a HDF5 formatted [**?**] 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 HDF5 input layer in Caffe doesn't provide an automated shuffling we have to pre-shuffle the images and labels before storing them.




(a) The activations before LRN.

(b) And after

Figure 18

### 3.3 Architecture

The architecture used, shown in figure {*addLater*}, is a version of that which won the 2012 ILSVRC [**?**]. 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. These normalise the values of each filter's activation with respect to the others in the same spatial position. Each activation is divided by

$$(1 + \frac{\alpha}{n} \sum_{i}^{n} x_i^2)^{\beta} \tag{3}$$

where $x_i$ are the activations of filter $i$ at the same position. $n$, $\alpha$ and $\beta$ are adjustable parameters, which we leave unchanged from those used by [**?**] at 5, 0.0001 and 0.75 respectively. The effect of this operation on our data is shown in figure 18.

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 an Euclidean loss which computes the l2-norm between this final vector and the label.

One advantage of using this architecture is that we can begin training from the weights

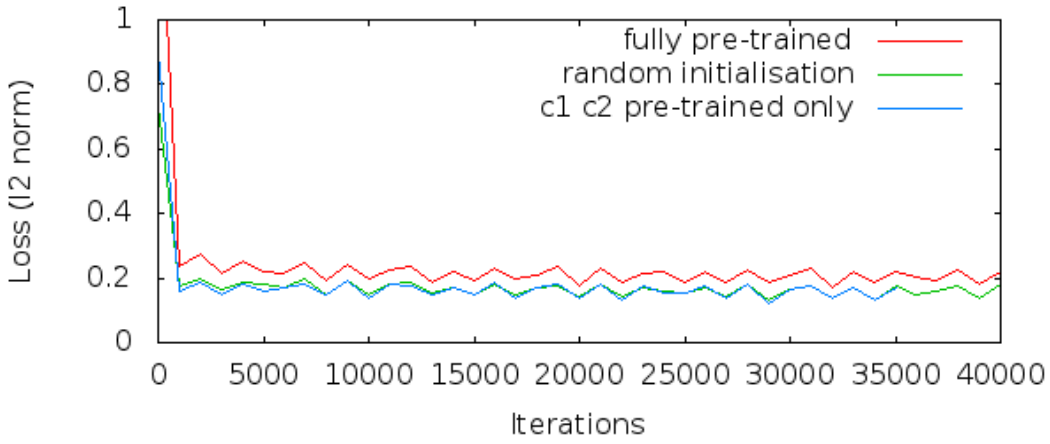



Figure 19: We find that begining training from ImageNet pretrained weights throughout the network hinders performance. Using pre-trained weights only in the first two layers produced the best results. We use this strategy for our final tests.

trained on ImageNet before fine tuning on our smaller dataset. We tested the networks performance 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 19. 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.

# 4 Results

## 4.1 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 train 6 networks with pairs of adjacent subjects withheld for testing in each i.e. 1 and 2, 3 and 4, 5 and 6. For each subject we present results for their longest sequence. This is because longer sequences make the

SPHERE softwares 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.

## References

[1] 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.

# Appendices

## A Pre-existing SPHERE System For Movement Quality Analysis

In this section we will cover the existing system (as originally described in [**?**] and [**?**]) highlighting the points to be considered for our work.

As mentioned in section **??** the aim of the system is to quantify the quality of movement. This is achieved through comparing the recorded motion to a taught reference model of perfect motion. Although it has been evaluated specifically for gait measurements, the system aims to be widely applicable. Provided with suitable training data demonstrating perfect motion it could be applied to physiotherapy exercises, or in a sports movement optimisation application with little adaptation. This has been demonstrated with the system being applied to boxing and sitting-standing motions presented on the SPHERE web page[3].

The gait analysis system has been trained using the SPHERE-staircase2014 dataset [**?**]. This dataset includes 48 sequences of 12 individuals walking up stairs, captured by a Kinect camera placed at the top of the stairs in a frontal and downward-looking position. It contains three types of abnormal gaits with lower-extremity musculoskeletal conditions, including freezing of gait and using a leading leg, left or right, in going up the stairs. All frames have been manually labelled as normal or abnormal by a qualified physiotherapist. 17 sequences

[3] www.irc-sphere.ac.uk/work-package-2/movement-quality

of normal walking from 6 individuals were used for training purposes and 31 sequences from the remaining 6 subjects with both normal and abnormal walking were kept for testing of the system. An example with the skeleton overlaid is shown in **??**.

The system can be considered as a pipeline, as represented in **??**. Once deployed in real home environments , another software component being developed in SPHERE will be responsible for human detection and recognition, hence we can presume input depth images similar to those contained in the dataset, and also knowledge of the person. The skeleton tracking packages from section **??** extract the joint positions of the body in the image. The skeletons are averaged over a temporal window to smooth noise before being scaled, rotated and translated to normalise the pose (these steps will be detailed further in section$^{insertreferencelater}$).

Next the low-level feature extraction stage builds a feature vector out of the skeleton data to encode the pose. Tao et al. tests and compares a number of viable feature representations for the skeleton data. They find that using a vector of the each joint coordinate concatenated performs best overall [**?**].

This joint position vector is then processed using the non-linear dimensionality reduction method Diffusion Maps [**?**]. The stage is first trained offline, building the manifold representation from the training subset of the data. The characteristic of manifolds produced by Diffusion Maps is that the data retain the relative euclidean distances between point in the reduced space. This representation is then extended to new data in the testing phase by projecting new skeletons onto the existing manifold.

Next the reduced pose vectors $\mathbf{Y}$ of the training data is used to learn two probabilistic continuous HMM (Hidden Markov Model) models, one of instantaneous pose, and one of dynamics. The normality of a pose is then the likelihood of the of $\mathbf{Y}$ being described by the pose model. Similarly the dynamical model computes a quality based on the likelihood of $\mathbf{Y}$ given the proceeding frames and the model of normal dynamics. Thresholds on these two normality scores are used to classify each frame normal or abnormal.

The system is 'online' in that it measures abnormality on a frame-by-frame basis, rather than processing a recorded sequence offline and measuring across the full sequence. This enables it to produce data on the parts of the motion that are deviating from normality, which is a benefit.

# B Networks

## B.1 AlexNet

name: "gaitAlexNet"