Abstract

This project aims to extract the pose of a single pig defined by motion capture ground truth using the Xbox Kinect sensor.

The Xbox Kinect is an advanced technology whose software was trained only on humanoid data to capture the motion of human beings. In this case, it is unclear that how well the Kinect technologies can perform on quadrupeds.

In this project a set of videos of a pig provided by the Vet School were trained based on Kinect methods for the benefit of pigs. The project implements the following approaches:

- Extract features of the pig in the depth videos by removing the background (see page 8-9) and creating a new method to normalize the pig (see page 30-34). The accuracy of the algorithm finding the location of the pig's head was 98%.
- Use IPCA (Incremental principal components analysis) to reduce the dimensionality of the features (see page 16). The method increased the speed of the whole system form 8 hours to 0.427765 second.
- Implement KNN (K-nearest neighbours) and NN (Neural network) as regression methods and RF (Random forest) as a classification method to track the motion of the pig and reached the accuracy of 85%, 93% and 93% (see page 16-25).
- Do experiments on normalization, IPCA algorithms and classifiers and make analysis by measuring the efficiency (see page 36-46).
- Work out Matlab GUI software to realize real-time motion track of the pig in the videos (see page 34-35).

After comparison between algorithms above, the results indicated that the Kinect also works well on pigs. Experiments on algorithms also proved that the whole system can effectively track the poses of the pig. After all the work above has been done in Matlab, an application of KNN was developed to capture the motion of the pig.

Contents

1. Introduction

In order to improve the welfare of animals, especially pigs, the Vet School wants to analys the behaviour of the pig. To achieve this, it is helpful to detect pigs' poses. Poses here mean the angles and the locations of the body parts of the pig (the head, the neck, the back and the bottom), as Figure 1.1 illustrates. Consequently, in this paper, experiments were launched on a single pig in depth videos recorded by the Kinect device and the pose of the pig was trained by several algorithms for classification to test how well Kinect technology can work on animals other than human.

1.1 Motivations

Compared with human behaviour analysis, animal behaviour is more difficult to detect as methods on behaviour analysis have been developed generally on human beings, which cannot work equally well when applied to other animals resulting from the problems of low efficiency and poor reproducibility [43]. As intelligent technologies are developing rapidly, in the field of biology such as genome and neuron analysis, methods of capturing the activity of animals is in high demand. One of the biggest advantages of automatic behaviour analysis is that it makes non-stop long term observation possible as computers can work 24 hours a day. Researchers will easily lose attention to motions which are too quick if detecting the pose of the pig manually [44]. Despite the fact that an increasing number of works about automatic animal behaviour analysis have been done, most of them can only solve simple problems. In Weixing Zhu et al.'s work, the sick pig was picked out by detecting its excretion time automatically [45], the accuracy of which is 78 %, while Weary et al. pointed out that different illnesses will cause different social and feed behaviours [46], which makes the approach of automatically working out the behaviours of the pig more important.

Nevertheless, motion capturing is an important part of behaviour analysis. Motion capture plays a critical role in working out what the pig's behaviour. For instance, when the pig was eating, the location of the head will be lower than usual; when it is finding food, its head will turn left and right frequently; when it is ready to turn around, it will compact its body. Hu's group [47] tried to track the motion of the pig to analyze behaviour in 2009 with no concrete result. Their method is to track the location of the whole pig in the screen and analyze the behaviour by detecting the route of the pig. The problem is that motion of the whole pig is not detailed enough to distinguish behaviours like eating and excretion. To work out behaviours correctly, more detailed motion capture analysis is necessary. As a result, in this project, the motion of four parts of the pig will be tracked (the head, the neck, the back and the button).

In most of the cases, automatic motion capture systems for animal behaviour were based on RGB cameras [43, 45, 46, 47], while others used sensors such as ultrasonic [48] and motion sensors [49] to track the motion. Depth camera introduced in Kinect technologies has never been tried.

1.2 Aims and objectives

With a view to analyzing pig behaviour, the aim of this project is to use Kinect technologies to work out a method which can extract pig's pose efficiently from depth video provided by the Vet School.

To achieve the goal, several problems should be solved so that the objectives of the project are:

(1) Understand how Kinect technologies work to track human motion.
(2) Work out the motion of the pig from depth videos.
(3) Collect and analyse experimental results of the algorithms.
(4) Evaluate and look for alternative ways to improve the algorithms.

Figure 1.1 illustrates how the system works. The inputs are depth videos of the pig. The left image shows one frame of the input image. The outputs are the joints of the pig as blue stars labeled in the right image. In the project, the outputs were worked out through classification. The joints represent four body parts: the head, the neck, the back, and button.

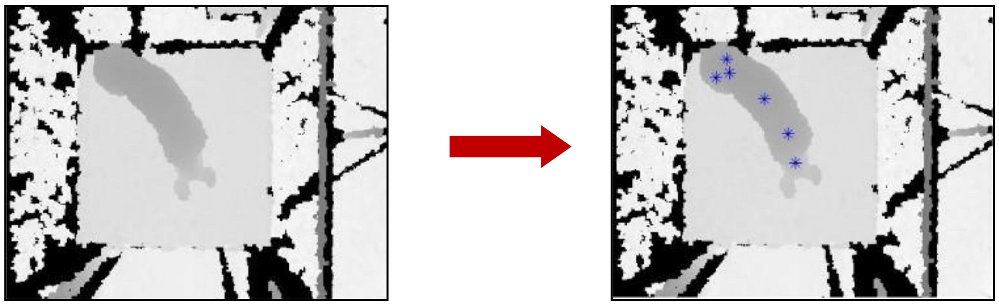


Figure 1.1: Inputs and outputs

The project also includes experiments to test the accuracy of the estimation. To calculate the accuracy, locations of reflection markers representing the real joints were recorded as MOCAP data and compared with the estimated one. The process will be discussed in the later sections.

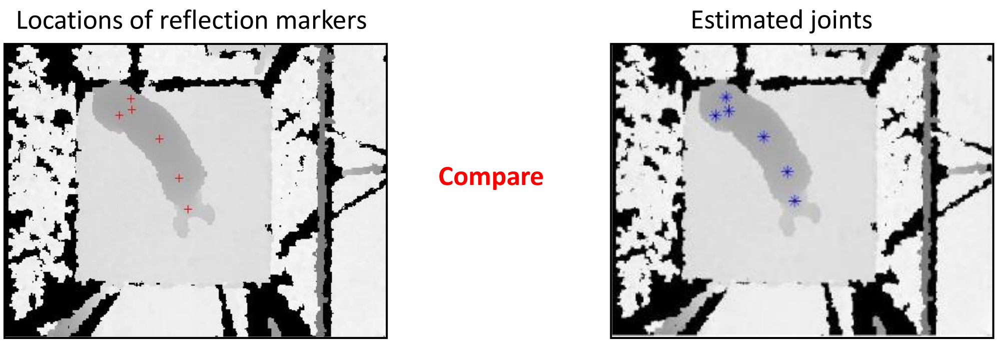


Figure 1.2: Accuracy measurement

The following 5 sections will explain how the aims were gradually achieved. In Section 2, this theoretical background used in the project will be introduced. Specifically, algorithms for speed improvement principal components analysis (PCA) and incremental principal components analysis (IPCA) and algorithms for training K-nearest neighbour (KNN), random forest (RF) and neural network (NN) will be included in the section. Section 3 will describe how the whole system is built gradually in Matlab by remove the background; normalizing images; extracting features; lowering dimensions of features; training and classification. In Section 4, experimental results with different parameters and classifications will be shown. Section 5 will make an evaluation of the work. Finally, Section 6 will give a conclusion and the future work of the project.

2. Background

2.1 Motion capture

Motion capture (MOCAP) is a key technology used widely in health care, interactive games and monitoring system. The most common steps to track the motion are to collect certain amounts of labeled data, normalize the original data, extract features, train classifiers by using training data and finally test new data through the classifiers.

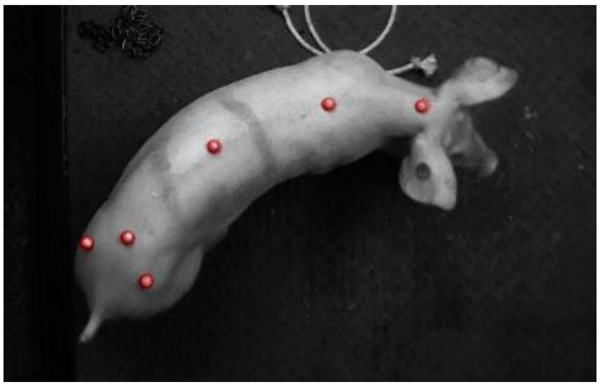


Figure 2.1: Reflection markers

A popular way to create a ground truth database is to use reflection markers. Attached to the target object's body, the markers are able to locate the object in a 3D space. In Tirakoat's research, 29 reflection markers and 10 cameras were used to create a training database [1]. Each markers works as a label so that the more cameras and reflection markers used, the more detailed motions can be captured. Meanwhile, Seiya's paper uses literal descriptions of human behaviours such as '*jump*' and '*walk*' as labels [2]. In this project, to detect the pose of the pig, six markers were put on the pig's body to track different parts of the body. Among the markers, one of which represents the head, one represents the neck, one represents the back and three represents the bottom, as the red points shown in Figure 2.1.

In the pig behaviour project, the data captured by reflection markers were called MOCAP data, which showed the pose of the pig. After the system worked out the pose of the pig, the pose data would be compared with the MOCAP data to see whether the pose estimation was right.

Normally, data generated by cameras cannot extract distinguishable features because of external factors such as lighting and drifts of the objects so that the original data will be normalized in many cases. In Manoj's paper of analyzing dance sequence, dancers with different appearances and size were rotated and transformed into a default size [3].

In the last 5 decades, training methods served for segmentation in motion capture technology have been improved dramatically. In some researches, only one classifier is applied. Wang Manjun's team classified the motion of human by kernel dynamic texture [4] by deciding a margin for each label. Zhang Zhe's paper applied KNN for segmentation [5], and others use a mixture of classifiers. In Jamie's paper [6], RF which gives a weight of each feature, was applied on labeled data. Kaustubh Kalgaonkar and Bhiksha Raj used a GMM based Bayesian classifier to recognize talking faces in their research [7]. By calculating the margins of features or similarities between features with different labels, classifiers are able to segment images into different labels.

As the size of database, number of labels and training methods used vary, accuracy resulted and time cost are quite different between systems. In some cases real-time capture is required so that additional algorithms are implemented to reduce the time of calculation mainly by lower this dimension of features before making classification. For instance, according to Zhang Zhe's paper, features were cross correlated before applying KNN to make it a real-time system [5]. Glardon's team used principal components analysis (PCA) to reduce dimensions when tracking walking people.

After training, the whole system is then ready for use. When given a new set of video data, the system will be able to capture the motion through trained classifiers.

2.2 Previous work

Before the Kinect technique using depth information, RGB cameras are used to track the motion. Thus, texture, color and shape matching technology plays a critical role in human pose estimation. While noisy pixels in depth maps have to be preprocessed, for RGB cameras, as texture and color information such as human clothing, different skin colors, lighting and shadows will confuse the system, methods should be created to solve this problem.

To detect the target object in a scene with strong lighting and shadow effect, Balan added a shadow camera which detects the lighting direction and strength in the scene [56]. To build the model, the shape of both the target object and its shadow are detected as one unit to be worked out from background through chamfer distance detection. Then the object and its shadow are segmented separately through a series of morphological operations. After that, the direction of the light and strength can be worked out by comparing pixels between an object and its shadow. This approach is not efficient as it doubles computation cost by detecting edges of objects twice. However, the light direction can be detected for further use such as rendering the scene.

In Munder's research [57], pattern features are extracted by measuring differences of pattern vectors between two frames. Munder's group designed a neuron network for pedestrian classification to locate the object first. They also found out that the texture transformations mainly occurred in the horizontal direction after crosscorrelating between two time clusters so that a vertical profile operator is applied to train texture data. Counting similarity between neighbouring frames is very common in pose detection from video and it is said from the research that crosscorrelating method worked well on walking poses. This algorithm designed especially for walking pose detection is not able to provide a qualified estimation for other motions.

Meanwhile, despite the fact that pattern information will add noise in classification, unique patterns can make contribution to train data, as mentioned in Siriteerakul et al.'s research [58], which counted a similarity by calculating the sum of texture difference and mask difference. Pattern recognition is quite popular in facial recognition, as locations of five sense organs on the face are similar from person to person. Siriteerakul's group tried to find the best match of the face texture in the database through rotating the head by known angles. In the case of the whole body detection, this approach can be more difficult to recognize shapes and textures of the cloth. Thus, an efficient way to locate a human body is to detect the head or four limbs [56] first, since the color and its intensity is comparatively easier to detect.

2.3 Kinect technique

In this project, the data to work with was videos of a single pig which has been be recorded by a depth camera like Kinect technologies.

Figure 2.2: (a) Kinect motion capture (b) Kinect camera

In 2009, Microsoft established a depth capture camera called Kinect for interactive Xbox games. Meanwhile, open source Kinect tools such as facial recognition and motion capture for fingers and the whole body were also avilable to public. Different from ordinary cameras, videos recorded by depth camera will not only show the common images with patterns and RGB colors but also show the distance between objects tracked and the camera, which are called depth maps. A depth map provides depth information by plotting pixels with different grey levels on the image. Based on this advanced technique, Kinect tools have also trained a large amount of motion data of human beings. Accordingly, the applications provided by them are not only able to track the motion of humans but also recognize their behaviours.

Before the Kinect has been launched, targets are normally tracked by color changes edge detection and texture comparison. In Yoshiaki Akuzawa's research, color and edge information were collected by two cameras [9] while V Seferidis and M Ghanbari's experiments were based on texture analysis [10].

Limitations of old motion capture techniques include time consuming and high cost. To achieve adequate accuracy, it usually takes multiple cameras and complex combinations of algorithms. However, based on Kinect technologies, researchers find it more efficient and less expensive to make detection with depth camera. Experiments made by Erik demonstrate that a single depth camera works better than two web cameras with sufficient accuracy when tracking walking people [11]. In comparison, Chan-Soo Park's team found that a laser scanner is slightly more accurate than a Kinect camera but Kinect is more stable and cost less time [12].

Under this circumstance, an increasing number of Kinect-based applications have been developed recently. Lu Xia's paper using Kinect to track multiple individuals proves that the technique on human beings is quite accurate [13]. Besides human focused tracking, Kinect applications are also applied to track simple non living objects. In both Alexey Abramov [14] and Panjawee Rakprayoon's [15] papers, they used depth cameras to classify static objects.

Although depth camera is able to solve some problems, it still needs to be improved. According to Jeffrey's survey, the depth camera cannot function as usual when it meets a corner or move in a high speed [16]. Nevertheless, applications provided by Microsoft Company have only been trained on human beings so that whether it is effective on quadrupeds is unclear. In this case, analyzing Kinect-like methods on pigs will give a possible answer.

This project worked on 30 fps videos recorded by a depth camera which showed the images of a single pig. Based on the depth images, the pig's pose was captured by the Kinect technologies.

2.4 Pre-processing

2.4.1 Extract Features

Features of an image are elements which represents the uniqueness of the image. Features can be a matrix or a vector. In the pig behaviour project, features are the depth values of each frame. Besides depth values, patterns, colors and the location of the object can also be regarded as features [40]. Normally, every image contains lots of features which cost a large amount of time to compute. Therefore, it is common to discard unimportant features such as background first [6], the process of which is called feature extraction.

A. Background Removal

In most cases, image feature belonging to the target object cannot be extracted directly because very often an image will include additional objects which are not the targeted one. For example, when tracking the motion of a car, information of the road and the traffic light has to be removed first, or their features will also be extracted with that of the car, thus adding noise. To remove the background, the region of targeted objects should be worked out. In human detection, it is common to work out the whole body by locating the head. Bo and Nevatia introduced a method of get the features by estimating the proportions of the human [41]. After learning the edgelet features, the head was tracked and it was estimated that the height of the head was 0.3 times of that of the whole body. In this case, the human body was picked out of the background. To analyze pig's behaviour, information of the floor and the cage should be removed first, which was resolved by setting two thresholds, which will be introduced, in Part B

B. Threshold

Setting a threshold is an important method in terms of segmentation. The main idea of thresholding is to distinguish the target object from the background by deciding a color range. It is assumed that pixels within the range belong to the object while pixels out of the range will be the background. Normally, the segmented images are binary ones where two figures representing the target and the background [42]. In this project, grey level depth images were recorded so that thresholding method can be effective in this case.

C. Depth Error

In terms of depth camera, estimation error will occur caused by invalid return of the camera. For instance, when a depth camera fails to estimate the distance between the camera and an object, it will return zero in certain pixels, which adds estimation errors in the depth image. Figure 2.3 is a depth map recorded by the depth camera. There're some black areas in the frame indicating that the camera failed to detect the depth of those areas. Those areas are called depth error. Depth error is common in the Kinect detection so that several methods have been developed to deal with the error.

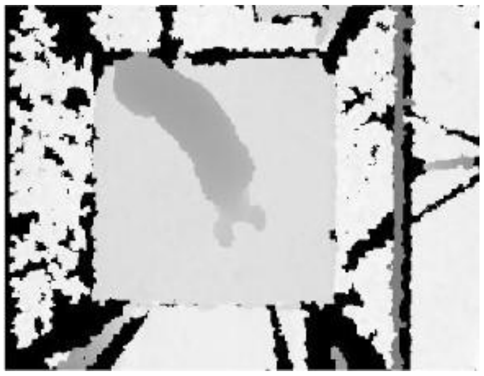


Figure 2.3: A depth map

To solve this problem, Xia's group invented an algorithm by gradually smoothing the shape edges [13]. They detected zero pixels and edges of the zero areas. Then they gave zero pixel a new value by calculate the mean depth of pixel around them started from the edge. While Xia laid emphasis on surrounding pixels, Matyunin's algorithm paid more attention on the relationship between frames. In Matyunin et al.'s paper, instead of detecting the surrounding pixels, they gave zero the pixel a new value by detecting the values of this pixel in different frames [39]. They used a Gaussian to work out a weight for the same pixel in different frame and multiple it with the value of the pixel. Then the sum of the products is the new value of the zero pixels. In practice, working out the value by tracking the pixel in different frames might be not credible when no regulation is found between frames (i.e. the value of the pixel is always zero in the whole video).

2.4.2 Normalization

Usually, the given videos are quite different from each other in size, location, color or lighting, which makes it hard to find the regulation of features. In the pig pose problem, despite the fact that there's no need to take color and lighting issues into consideration in the case of depth images, the pig moves around with the head towards various directions. Thus, the raw data needs to be normalized according to some rules. In this project, all the pigs were rotated to the horizontal axis with their head facing right, and the centers of the images were moved to the center of the pig.

A. Transformation and Rotation

Moving objects in videos can be appear anywhere in any direction in the frame making it difficult to make classification so that methods should be applied to locate the object and move and rotate them to the same coordinates.

In human motion detection, the one tracked by a camera needs to be rotated to the front view and reshaped into a default size. To normalize the figures, Baak et al. computed a least-square plane with a radius of 0.15 m from the center of the sphere when doing human pose estimation [40]. While Baak rotated the body by detecting the angle from the center to the whole body, in Lu et al.'s work, they normalize the depth image according to the depth information of the head [13]. By comparing the deviation between the edge and the pixels, the head was tracked and the depth of the whole image was thus normalized.

Unlike humans whose head is always on top of the body, when observing the pig through the camera on the top, location of the pig can be anywhere in the image which make it difficult to track the head. Under this circumstance, to rotate and transform the pig's body, one essential step is to locate the head of the pig so that all the pig can be rotated to the same direction. Normally, to detect a part of the body in a video, frames need to be constant, or the normal methods will lose the target. Algorithms such as shape context are often used in this case. For instance, In Bo Chen's research, by comparing neighbours of edge pixels, the operator is able to work out a deformation vector to determine the transformation of a certain region. When the center of the head was detected, the algorithm will search the new location of the head in the next frame around its $4\times12\times4$ neighbours [56]. Despite the fact that the algorithm is able to track the head to a high accuracy, it will not efficiently recognize the head when frames are randomly picked up from different videos.

In the pig behaviour project, in order to rotate the pig, the head and the bottom need to be distinguished. However, as the training data were randomly collected from database, algorithms based on the continuity between frames are not ideal here. Instead, after observation, it can be seen that the shape and the depth of the pig's head and the bottom is quite different. Specifically, the shape of the bottom is usually round and the depth value of the bottom is normally smaller than that of the head. In this case, a new algorithm based on the shape and the depth values were invented in this project, which will be discussed in the Section 3. The algorithm is able to track the head randomly picked frames.

B. Least square line fitting

In this project, the pig is rotated by the arctangent of a slope from a linear equation computed by least square line fitting. The aim of the algorithm is to work out a linear equation representing the axis of symmetry of the pig. Figure 2.4 demonstrates the least square fitting function. The arctangent of the slope of the red line is the rotation angle.

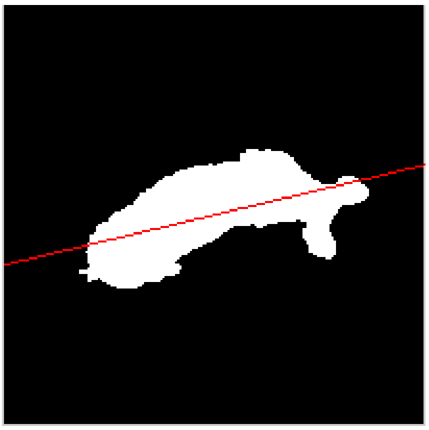


Figure 2.4: Polynomial curve fitting

Given the pixel matrix of the pigs $(\mathrm{X,Y})$. Each row $(\mathrm{x,y})$ is the coordinate of a pixel in the image and there are n pairs of coordinates in the image. Let the least square curve fitting be:

$$y=\mathrm{a}x+\mathrm{b} \tag{2.1}$$

Here the residual sum of squares of all points should set minimum:

$$I=\sum_{k=1}^{n}[y-\bar{y}]^2=\sum_{k=1}^{n}[y-\sum_{k=1}^{n}a_k P_i^{k\text{-}1}]^2 \tag{2.2}$$

$$\frac{1}{2}\frac{\partial I}{\partial a_k}=\sum_{k=1}^{n}[y-\sum_{k=1}^{n}a_k P_i^{k\text{-}1}]P_i^{k\text{-}1}=0 \tag{2.3}$$

The formula can be finally described as:

$$A=\begin{bmatrix} a \\ b \end{bmatrix}=\begin{bmatrix} n & \sum_{\mathrm{k}=1}^{n} x_k \end{bmatrix}^{-1} \bullet \begin{bmatrix} \sum_{k=1}^{n} y_k \\ \sum_{k=1}^{n} T_k \end{bmatrix} \tag{2.4}$$

The formula fits every two distinct coordinates. The linear slope a and the intercept b can be computed by the formula above. The method was able to work out a linear operation giving an axis of symmetry of the pig. However, when the difference between the head and the bottom was not distinguishable, the angle will be the complement angle of the desired one, methods of solving this problem will be discussed later in Section 3.

C. Edge Detection

Edge detection method is widely used during the pre-processing period when doing segmentation. By working out the shape of the target, according to shape information, different parts of the target can be recognized easily.

In Bo Wu's paper [41], silhouette oriented method called Edgelet is raised. The algorithm combined with Bayesian function is designed especially for human pose estimation, which conquers problems such as body shape matching and cloth pattern recognition. By taking the body shape and the intensity of detected edge into consideration, the algorithm is able to discard some noise automatically and thus reducing the cost. The results proved that it can deal with shape and texture problems.

Other edge detection methods include a Prewitt operator applied in facial detection launched by Asano [51], which used a four-direction filter to check the edges of characteristic regions instead of the whole image; In Rahman's research, he applied a Sobel operator with two Gaussian distributions to detect the edge of human based on a series of images [52]. While Prewitt works well on finding detailed edges, Sobel operates quicker.

A gradient algorithm Canny Edge Detector established by Canny John [53] is implemented as an OpenCV tool for free use. It is one of the most traditional ways to draw the edge. The main idea of the algorithm is to detect the edge through a Gaussian smooth firstly and applying a distance transfer (DT) secondly. In Xia's research [54], the algorithm is used to detect edges from a 2D depth array. In addition to finding all the edges in the depth map, Xia's group also added a method to reduce the noise. When the class is smaller than a certain size, it will be replaced by its neighbour class.

Chakraborty, I. [55] also applied Canny Edge Detector for human detection in videos. To improve the detection, 8 neighbourhoods of edge pixels are checked, the minimum intensity of which is set to be 3 to delete the noisy edge pixels. Canny Edge algorithm is ideal to extract edge features because of its high accuracy.

In this project, to distinguish the head and the bottom of the pig, Canny Edge algorithm was used. Canny Edge Detection mainly has three steps: applying a Gaussian filter on the image, doing first deviation and computing the intensity gradient.

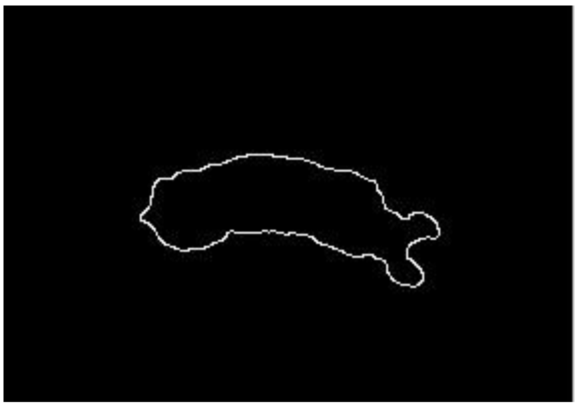


Figure 2.5: Canny edge detection

In the first step, the Gaussian filter was applied on the image. A Gaussian filter is normally used to smooth the image and eliminate the noise. Despite the fact that images will lose some features after doing Gaussian, it is helpful to work out the edge. Let the size of the Gaussian filter be m by n. For each pixel in the image, Figure 2.5 illustrates pixels included in the filter.

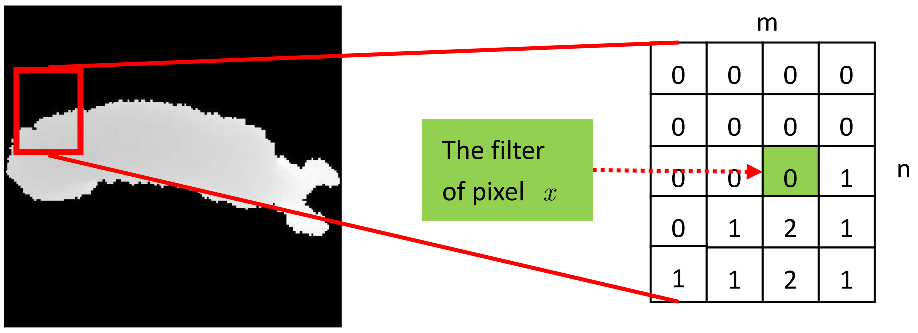


Figure 2.6: a Gaussian filter

As Formula (1) indicates, θ and σ are the mean and the standard deviation of the elements in the filter, and the new x can be calculated by the Gaussian function named as $g(\mathrm{x})$:

$$g(\mathrm{x})=\frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot e^{-\frac{(x-\theta)^2}{2\sigma^2}} \tag{2.5}$$

In the second step, the first deviation is applied on the new image generated by a Gaussian filter in both horizontal and vertical axis. Figure 2.7 illustrates the first deviation done on a row in the horizontal direction by Formula (2.6) through substraction.

$$G_x = \frac{\partial g}{\partial x} = g(\mathrm{x}+1)\text{-}g(\mathrm{x}) \tag{2.6}$$

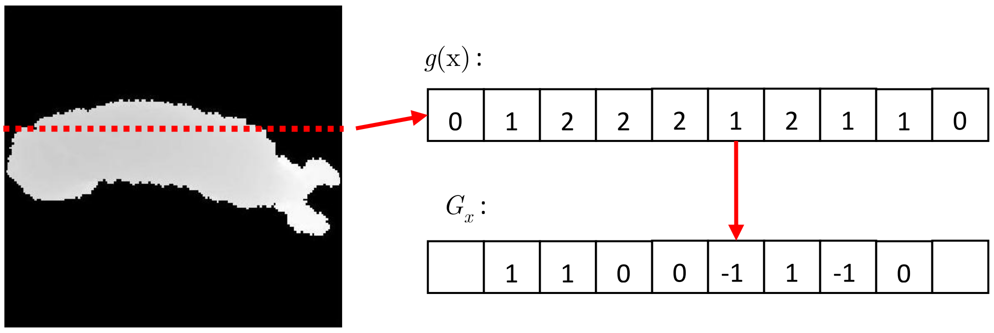


Figure 2.7: the first deviation

After the outcomes of first deviation from both horizontal and vertical axis G_x and G_y have been computed, the third step is to do the intensity gradient. Formula (2.7) and Formula (2.8) shows the gradients determined by the values of the first deviation:

$$G = \sqrt{G_x^{\,2} + G_y^{\,2}} \tag{2.7}$$

$$\theta = ac\tan\left(\frac{G_y}{G_x}\right) \tag{2.8}$$

2.5 Algorithms for Lower Dimensions

As the size of the depth image from the original videos recorded by depth camera is 240 by 320 pixels, which means each depth image has 76800 pixels and 2500 pixels after normalization, doing classification directly on them will be a time consuming work. Besides, as features are normally not equally important, training all features will add more noise into the system. In this case, to reduce the time complexity, certain methods should be done to lower the dimensions of the image. This project used PCA and IPCA to reduce the dimensions of features, that is to say, when given a set of data which has n features, PCA or IPCA output the same data with k features, where $n >> k$.

2.5.1 PCA

PCA (Principal component analysis), established in Pearson's paper [17], is able to create new projection coefficients, the number of which is smaller than the original variables by orthogonal transform them into a new coordinate system. The algorithm was then widely used in features compression. Glardon successfully test PCA on tracking walking people with different size in his research [18]. Similar works have also been done by Tsuruta [19] and Razali's team [20]. Their results showed that PCA can largely reduce the time while keeping the accuracy.

Suppose the original data matrix is X, where n rows are the number of pixels and f columns are the number of samples in the training database. The matrix of training data X is then:

$$X=\begin{bmatrix} x_{11} \ x_{12} \ \cdots \ x_{1f} \\ x_{21} \ x_{22} \ \cdots \ x_{2f} \\ \cdots \\ x_{n1} \ x_{n2} \ \cdots x_{nf} \end{bmatrix} \tag{2.9}$$

And the algorithm can be described as follows:

(1) Calculate the zero mean matrix X_d by the following formula:

$$\mathrm{X}_d = \mathrm{X}\text{-}\overline{\mathrm{X}} \tag{2.10}$$

where each row of $\overline{X}$ is the mean value of the pixels.

(2) The correlation coefficient matrix U is a n by n matrix and u_{ij} is an element in U, which is defined as:

$$u_{ij}=\frac{1}{f\text{-}1}\sum_{k=1}^{f} x_{ik}x_{jk} \\ =\frac{\sum_{k=1}^{f}(x_{ik}\text{-}\overline{x})(x_{jk}\text{-}\overline{x})}{\sqrt{\sum_{k=1}^{f}(x_{ik}\text{-}\overline{x})^2(x_{jk}\text{-}\overline{x})^2}} \tag{2.11}$$

(3) Given the inner product U computed by covariance $X_d{}^T X_d$, eigenvalues, λ and eigenvectors L, the covariance matrix can be calculated by:

$$E=X_dL \tag{2.12}$$

(4) The features of columns EV in the new eigenspace can be obtained by:

$$EV=X_dE \tag{2.13}$$

In this case, k values with largest eigenvalues are needed and the rest of $n\text{-}k$ eigenvectors are discarded.

(5) Finally, the new eigenvectors of each sample can be calculated by the following formula:

$$X_i^{new}=EV^T \times (\mathrm{X}_i\text{-}\overline{X}) \tag{2.14}$$

Despite the fact that PCA improves the efficiency of training by the large, it still have some shortcomings. Danijel Skočaj and Aleš Leonardis mentioned in their paper that all data should be given in advance when processing PCA, which is not ideal for large database [21]. In practice, when doing PCA in Matlab, the problem of out of memory which resulted from the high dimension vectors of the training data, often occurs.

In order to solve this problem, rather than doing PCA directly, in improved algorithm called IPCA was implemented in the system.

2.5.2 IPCA

A well-known computational approach to PCA involves solving an eigensystem problem by computing the eigenvectors and eigenvalues of the sample covariance matrix. For PCA method, all the training images need to be available before computing principal components. Approach of this kind is called a batch method. However, batch PCA is no longer suitable for new type of research in which a large number of online real-time video streams need to be estimated, which is motivated by the development of animal vision systems [22]. Further, if the image has many features or there are many training images, both the computation and storage complexity grow dramatically.

To solve the problem of developed learning mode and reduce the computation complexity, IPCA (Incremental principal component analysis) was raised three decades ago in machine learning field. Instead of doing transformations for only one time, IPCA divides sample data into several groups and does PCA on each group sequentially. Yuyang Weng et al. in 2003 invented a covariance-free IPCA in 2003 [22], which can work quite well when the sample database is large enough. However, for those with insufficient samples, its' performance was not stable. In this project, the ratio between the number of samples and that of eigenvectors is sometimes as small as 0.052 (130/2500) so that the algorithm is not suitable in this case. Danijel Skočaj and Aleš Leonardis's IPCA algorithm took the training images sequentially and got new eigonspace from the subspace gradually [21]. It is said in their paper that the results of IPCA is as good as standard PCA and it is flexible by treating images differently. To analysis pig's behaviour, both one single clip and a set of videos were treated as training database so that the algorithm can be ideal in this case. Steps of IPCA can be described as follows:

(1) Given training data matrix X and decide m_0 sample images to be processed in the batch mode. Let current mean value be $\overline{X}^{(\mathrm{n})}$, current eigenvectors be $L^{(\mathrm{n})}$, current coefficients be $F^{(\mathrm{n+1})}$ and the new input sample is x.

(2) When putting in a new sample, coefficient of the image can be calculated as:

$$a{=}L^{(\mathrm{n})\mathrm{T}}(\mathrm{x}\text{-}\overline{X}^{(\mathrm{n})}) \tag{2.15}$$

(3) The new image is then reconstructed as x':

$$x' = L^{(n)}a + \overline{X}^{(n)} \tag{2.16}$$

(4) The orthogonal product of $L^{(n)}$, r, is then:

$$r = x' - x \tag{2.17}$$

(5) For samples $m_i(\mathrm{i}=1,2,....\mathrm{f})$, work out a new eigenspace $\mathrm{L}' = [\mathrm{L}\ \frac{r}{\|r\|}]$, where r is the residuum vector orthogonal to $\mathrm{L}^{(n)}$.

(6) Features in the new basis can be determined as:

$$X' = \begin{bmatrix} X^{(n)} & a \\ 0 & r \end{bmatrix} \tag{2.18}$$

(7) After that, by repeating PCA on X', new mean value $\overline{X}''$, new eigenvectors L'' and eigenvalues Ev'' can be obtained.

(8) According to the input parameter k, discard vectors which are not important:

$$k^{(n+1)} = k^{(n)}, L'' = [\mathrm{e}_1'', ..., \mathrm{e}''_{k(f+1)}] \tag{2.19}$$

(9) Repeat Step (1) by $X^{(n+1)} = L''^T(\mathrm{X}' - \overline{X}''\mathrm{Ev}_{1\mathrm{x}f+1})$.

(10) Update the eigenvector and the mean by:

$$L^{(n+1)} = \mathrm{L}'\mathrm{L}'' \tag{2.20}$$

$$\overline{X}^{(n+1)} = \overline{X}^{(n)} + L'\overline{X}'' \tag{2.21}$$

(11) The new eigenvalues are: $Ev^{(n+1)} = [\mathrm{Ev}_1'', ..., \mathrm{Ev}_{k(f+1)}'']$.

(12) Turn to Step (1) until all the images have been processed.

Follow the steps above, there’s no need to include the whole training database when doing transformation, for the eigenspace is gradually rotated according to new input images one by one, thus reducing calculation time and saving space.

2.6 Algorithms for classification

Many effective methods were invented for classification. SVM (Support vector machine) invented by Cortes and Vapnik in 1995 [31] makes segmentation by working out the margins between classes. The algorithm works well on simple linear classification problems such as discriminating Pakinson symptoms [32]. When it comes to non-linear problem, a kernel function should be added in SVMs to keep high accuracy [33]. In Sharma’s paper [56], a CRF-based probabilistic framework is built to generate detect accurate body parts. For the first frame, a rough body shape is tracked. After comparing features of its neighbouring frames, more concrete body parts are work out based on the classification of the first frame.

When it comes to the training and testing, as introduced in Section 1, either classification or regression algorithms can be used. In this paper, three algorithms will be used and compared. The function of a classifier is to observe which class the testing data belong to by comparing it with the classified sample data. When it comes to a regression algorithm, rather than giving an exact class to the testing data, it will work out the exact values according to the similarity between the testing and training data. For example, in pig behaviour project, KNN was a classifier method when K was one and a regression when K was more than one. RF was used as classifiers and the testing data were given a class by the algorithms and then the locations of the pig's joints in testing data was just the same as that of the class, while NN was used as a regression algorithm worked out the locations of pig's joints directly.

2.6.1 KNN

KNN (K-nearest neighbours) is one of the most basic algorithms for segmentation proposed by D.O. Loftsgaarden and C.P. Quesenbery in 1965 [23] is an instant learning method which does not need to build a training model before testing. One big advantage of KNN is that it is easy to understand and develop. By simply calculating the Euclidean distance between testing and sample data, the testing data will be classified by analyzing the classes of K samples with shortest distances to the testing one. Zhe Zhang et al. mentioned in their research that KNN can be quite accurate and fast based on a small database with few features [24]. On the other hand, KNN can be quite time consuming when given a larger database or more features. Accordingly, researchers have tried to find ways to accelerate the method for decades [25, 26].

In the pig behaviour analysis experiments, as the number of features is small enough and it takes no more than 0.5 second to classify each depth image, the standard KNN can work well so that there's no need to apply improved KNN algorithms. Besides, KNN is a basic algorithm so that the result of which can provide other classifiers with a convinced reference.

Figure 2.8 is an example illustrating how KNN works in a two dimensional axis when K is four. The blue stars are the nearest neighbours of the red star. To apply regression method on KNN, features of the red star will be the mean of features of its four neighbours.

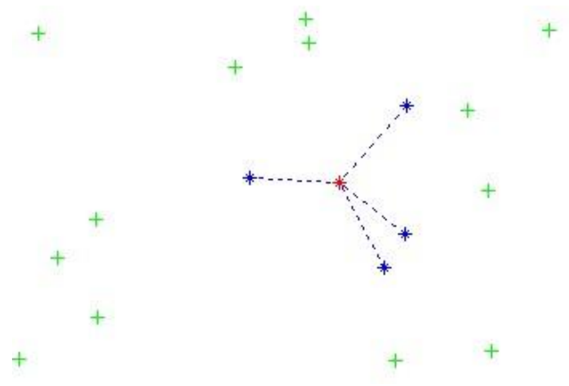


Figure 2.8: KNN (K=4)

Given features of testing data X, sample data $Y=\{Y_i|Y_i \in Y \cap i=1,2...f\}$ and its labels $L=\{l_i|Y_i \in Y \cap i=1,2,...f\}$ and the number of nearest neighbours k, where f is the number of samples in the database and Y_i is the features of one sample, the major steps of KNN are as follows:

(1) For each sample Y_i, calculate the distances between X and Y_i: $D=\{d_i|d_i=\text{distance}(X,Y_i) \cap i=1,2,...f\}$.

(2) Pick up k Y_i with the smallest d_i: $Y'=\{Y_i|d_i=\text{shortest}(D,k)\}$.

(3) Get the labels of Y': $L'=\{l_i|Y_i \in Y' \cap i=1,2,...f\}$.

(4) The mean location of K nearest neighbours L' is regarded as the location of testing data X.

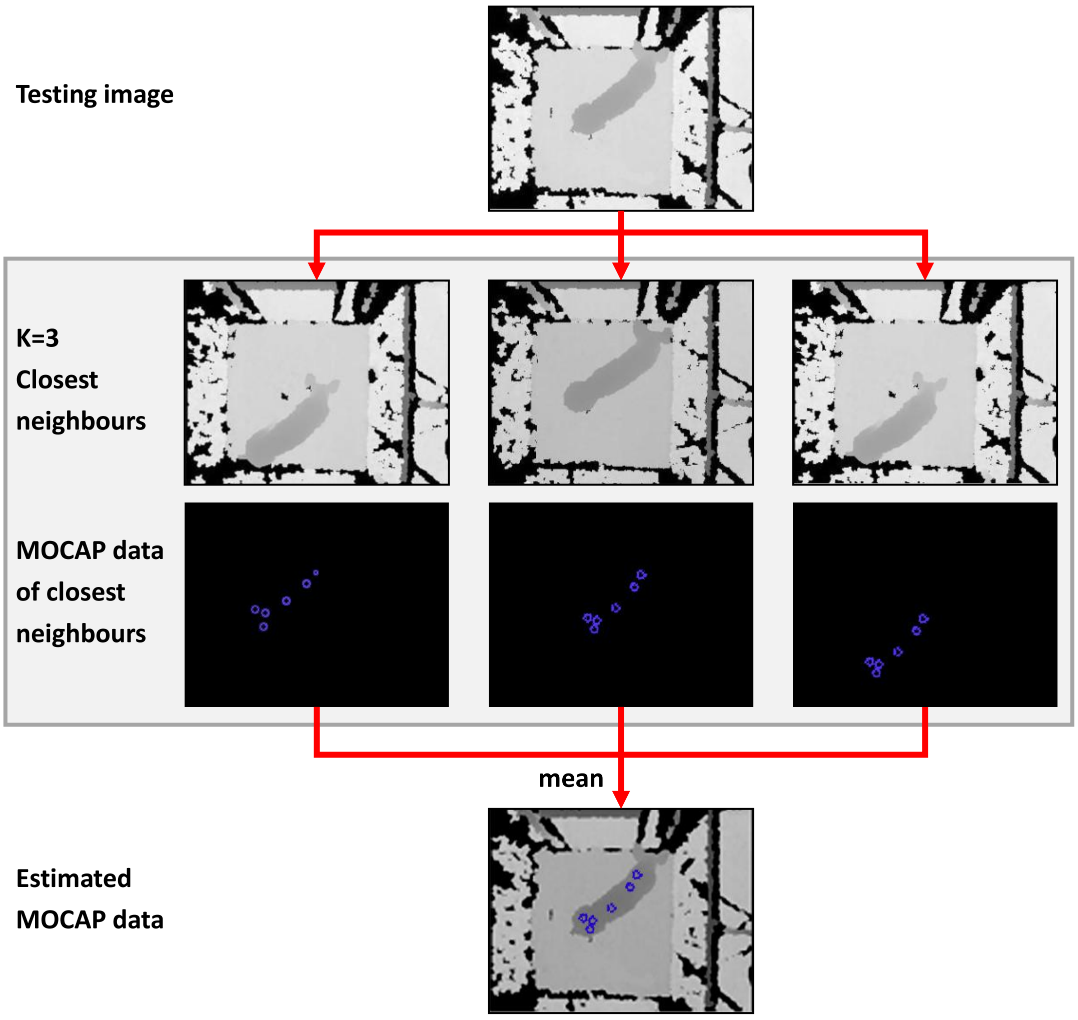


Figure 2.9: KNN regression

When K is one, the estimated MOCAP data is exactly the same one as the closest neighbour of the testing data; when K is more than one, the estimated data is the mean of the image's closest neighbours' MOCAP data. Accordingly, KNN was used as a regression method here. It is reasonable and easy for KNN to make segmentation by Euclidean distance. The distances indicate the difference between depth values between frames. But whether every feature has the same weight is questionable. Besides, the problem of long running time should be solved in the project.

2.6.2 Random forest

The classifier RF (Random forest) proposed by Breiman [27] makes segmentation according to the weight of classifications of every decision tree in the forest. Each tree is considered as a weak classifier. When putting a testing data into the forest, each tree gives the data a classification and the final class of testing data is the one with the highest votes. According to Khan et al., RF has a reputation of high accuracy and short training time [28]. On the other hand, Hong et al. pointed out that RF gives poor results when too much noise appear resulting from a large number of features so that they created a treeing-weighting random forest [29]. Similarly, Shotton et al. also introduced a weighting system in RF algorithm [6], while Singh et al. introduced a highbred RF with weak classifiers [30].

In this project, despite the fact that the number of features is small enough (less than 50), no systematic classification has been made on training data, which means each piece of training data is considered as a single class so that there can be thousands of classes. Under this circumstance, two elements will be important. The first one is the size of the training database. With insufficient data, if the classifiers failed to choose the most similar sample, the final class of the testing data will be quite different from the real one. Another element is the size of the random forest. The more trees in the forest, the more probably that the best training data will be picked up. The whole system can be described as:

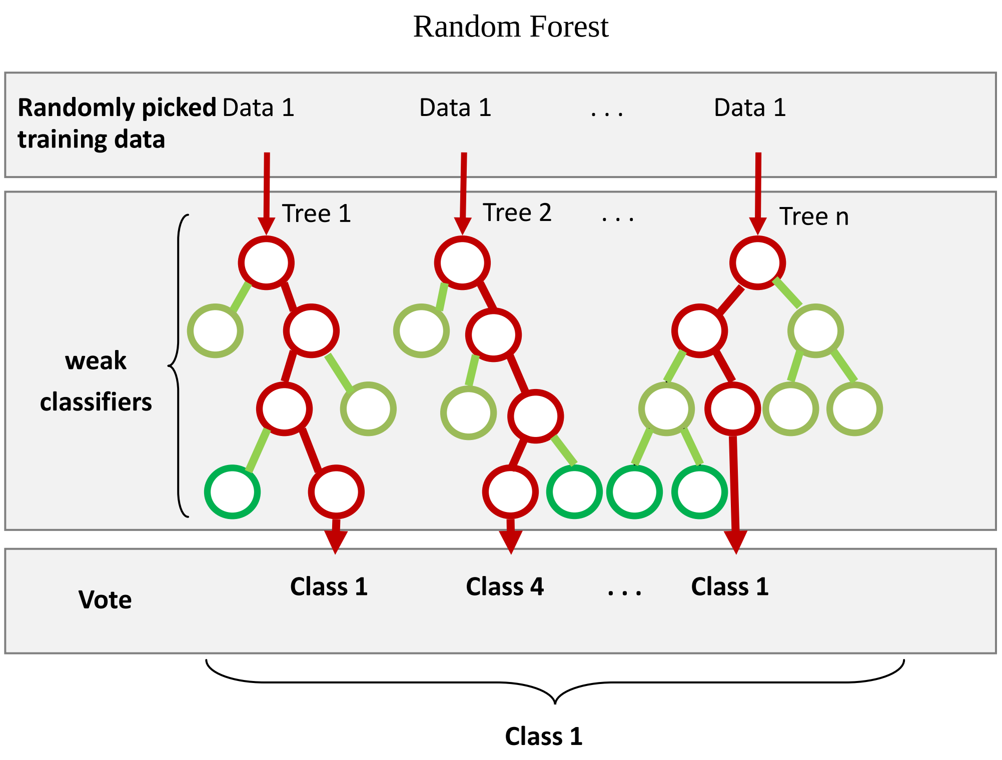


Figure 2.10: Random forest

A. Randomized decision forest

A forest consists of several trees with split nodes (grass green) and leaf nodes (jade green). Let k be the number of features of the images in the training database, n be the number of training samples chosen by every tree to make segmentation on from the whole database and d is the maximum depth of the tree. Steps of RF can be concluded as follows:

(1) For each tree, choose n samples from the training database to train a classifier and use the rest of the samples to calculate the precision of the classifier.

(2) Each node contains k features. To create subnodes, a best margin should be worked out according to the chosen features. Here weak classifiers were used to get the margin, which will be discussed in later in Section B.

(3) Repeat step (2) until all n samples have been classified.

(4) Every tree has to be fully grown trees.

According to the steps, RF can be quite accurate if the forest is large enough. An individual feature only is not strong enough to tell the pig's pose so that weak classifiers applied in each tree make segmentation by analyzing individual features gradually and ensemble the whole information of all features.

B. Training

Figure 2.10 demonstrates a linear weak classifier surface in a 3-dimensional axis. As can be seen, two classes shown in red and blue are classified by the green surface. Although there're some points which are wrongly classified, the error rate of linear classifier is respectively low in this case.

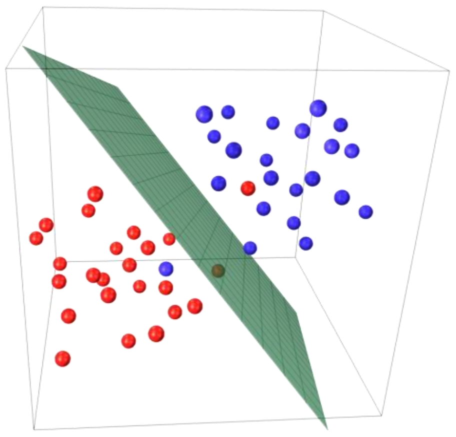


Figure 2.11: Weak classifier in 3-dimensional spaces

In step (2) of random forest, a weak classifier is used to find the best split. Specifically, each node only has two subnodes. To put it in another way, training samples in each node are divided into two classes by the weak classifier. For each tree, the training algorithm can be described as follows.

(1) Randomly pick n samples from training database making a dataset A and let τ be a threshold vector. Candidates of in the nodes can be described as:

$$\phi=(\mathrm{x},\tau) \tag{2.22}$$

x in the formula is the feature vector and τ is the threshold vector. Both of two vectors have the same length.

(2) Then the classifier can be determined as:

$$f(\mathrm{a})=\mathrm{mx}+\mathrm{n} \tag{2.23}$$

In the formula above, x are the features of sample a.

(3) For samples in dataset A, choose the left or the right node by the following formulas:

$$A_l(\phi)=\{\mathrm{a}|\mathrm{a} \in \mathrm{A}\ \&\ \mathrm{f(a)}<\tau\} \tag{2.24}$$

$$A_r(\phi)=\mathrm{A}\backslash\mathrm{A}_l(\phi) \tag{2.25}$$

(4) For each node, the best ϕ with the largest gain is stored.

$$\phi' = \arg\max_{\phi} P(\phi) \tag{2.26}$$

$$P(\phi)=\mathrm{H(A)}-\sum_{n\in\{\mathrm{l,r}\}}\frac{|A_n(\phi)|}{|\mathrm{A}|}H(\mathrm{A}_n(\phi)) \tag{2.27}$$

In Formula (6), $H(\mathrm{A})$ is the Shannon entropy computed on the number of the class. In the pig behaviour project, each depth image was considered as an individual class.

(5) If the largest gain $P(\phi')$ is sufficient, and the depth of the tree is below the maximum depth d, then recurse the subsets $A_l(\phi)$ and $A_r(\phi)$.

C. Testing

In the process of testing data, it is put into every decision tree to be classified by the weak classifiers trained in training database. When the testing data reaches a node, the value vector is calculated by Formula (2). By judging whether the values are larger than the thresholds, the testing data chooses either left of right subnode. After repeating the choosing steps by calculating the value until reaching the leaf node where contains the final class. Each tree makes a classification for the testing data so that there are n classifications in all. Those results computed by tree were voted for their own classes. When all decision trees have segmented the testing data, votes are sum up and the class with the most votes is assumed to the right class of the testing data.

2.6.3 Neural Network

In 1873, Alexander Bain proposed NN (Neural network) in the field of brain memory study [35]. Yet in 1969, David Marr proposed the system as a computational model of cerebellum [36]. This is the first time the brain model was described as a computational function. Instead treating every feature equally, NN gives a weight for all the features when doing training. One obvious advantage of NN is that it treate features differently. Yeqin Wang found that NN worked better than a single network the nearest neighbour algorithm when doing pattern recognition between wood [37]. Burke et al. also mentioned in their research that the precision of NN was higher than other statistical classifiers but still, it has drawback of overfitting. That is to say, NN sometimes cannot recognize the class of the testing data as it fails to fit any training data. They thus suggested some methods to solve the problem by either penalize large weights or stopping iterative algorithm [38].

In this project, a Matlab function of neural network was used. The function solves the overfitting problem by adding a parameter called radial basis to reduce the weights of inputs. Here NN was used as a regression method rather than a classifier, the inputs of which were feature vectors and the outputs were the value MOCAP data.

A. *Training*

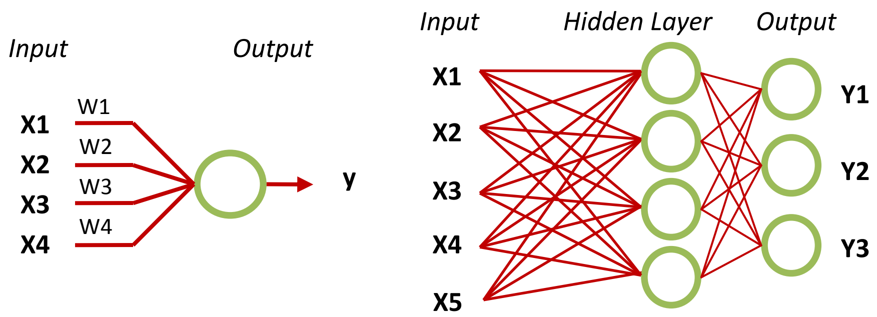


Figure 2.12: (a) NN for a single neuro **(b) NN for multi-labels**

In terms of training, a neural model is built. As Figure 2.12 illustrates, the green circle is the neuron inside which stores the weight of each feature. In pig behaviour project, the inputs are features while the outputs are the coordinates of the labels. The new output is calculated according to the weights of each feature. And the training steps of the algorithm are:

(1) Let X be n features of one piece of training data, Y be the labels and the weights of every features and nodes in the hidden layers are initially the same: $w_i=\frac{1}{n}$,i=1,2,...,n .

(2) The output can be calculated as:

$$Y'=\sum_{i=1}^{n} w_i x_i \qquad (2.28)$$

(3) Then for each feature, the new weight is adjusted by the following formula:

$$\Delta w_i=\eta(\text{Y-Y'})\text{x}_i \qquad (2.29)$$

(4) Repeat Step (2) until the weights do not change. The weights of the features are then stored.

(5) Repeat Step (1) to Step (4) for every training data.

The algorithm above is only for one layer, normally, as Figure 2.11 illustrates, there's one or more hidden layers between input and output layer. In Matlab function, only one hidden layer is used and the number of neurons in the hidden layer is the same as that in the input layer. To conclude, in this project, there're 15 neurons in both input and hidden layer and 12 in output layer. For each neuron in both hidden layer and output layer, 15 pairs of inputs x and their weights w are stored so that there are 405 pairs of neuron information $(\mathrm{x,w})$ in the whole system.

B. Radial basis

The parameter radial basis was used to solve the over fitting problem by reduce the weights. The figure below shows the architecture of radial basis in neural network regression. Here the Radial Basis Layer is the input layer and the Special Linear Layer is the hidden layer.

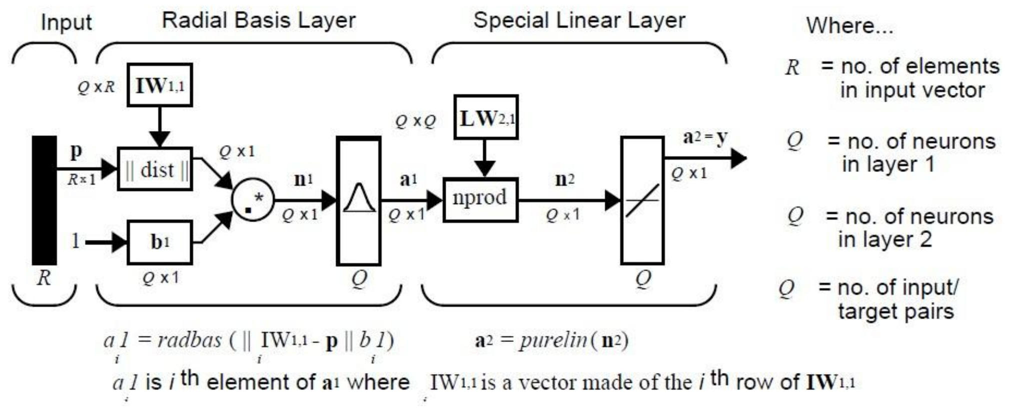

Input
Radial Basis Layer
Special Linear Layer
Where...
R = no. of elements in input vector
Q = no. of neurons in layer 1
Q = no. of neurons in layer 2
Q = no. of input/ target pairs
Q x R
IW1,1
p
R x 1
|| dist ||
Q x 1
1
b1
Q x 1
n1
Q x 1
a1
Q x 1
Q x Q
LW2,1
nprod
n2
Q x 1
a2 = y
Q x 1
R
Q
Q
a1i = radbas (|| iIW1,1 - p || b1i)
a2 = purelin(n2)
a1i is i th element of a1 where iIW1,1 is a vector made of the i th row of IW1,1

Figure 2.13: Radial basis

Besides the input feature vectors and their weights, a bias $\mathrm{b1}$ is computed by the input spread with the value of 0.8326/SPREAD. $\|dist\|$ is each neuron's weighted input, which is the distance between the input vector and its weight vector. Each neuron's net input is the product of its weighted input with its bias, calculated with net product. Each neuron's output is its net input passed through radial basis.

The larger the spread is, the larger area around the input vector where neurons in layer 1 will respond with significant outputs will be. Therefore if spread is small the radial basis function is very steep, leading neurons with large weights having larger outputs than other neurons. As a result, contributions of neurons with small weights will be so small that to be recognized. In another word, small spread is very likely to cause overfitting problem.

When the value of spread grows, the slope of radial basis function becomes smoother so that more neurons can respond to input features. The network then acts as if it is taking a weighted average between target vectors whose design input vectors are closest to the new input vector. A large spread value making most of neurons response to the input features will result in a set of complete outputs, which is desired in the pig behaviour project. Otherwise, if neurons don’t work when calculating a feature, data worked out by NN will lose pose information.

C. Testing

For testing data X_{test}, work out the value of the output labels for every training data according to Formula (4). Then the similarity of between the estimated labels and the real labels will be calculated. Labels with the largest similarity are considered as the labels of the testing data.

3. System Model

3.1 Basic Structure

This section will introduce how the system was built and the following figures illustrate the structure of the system. Figure 3.1 gives the task of the project, which is to estimate the pose of the pig as defined by the locations of the pig’s joints in the video. The inputs are depth images of the video and the outputs are the estimated joints (bottom, back, neck and head) of the pig.

Figure 3.2 shows how the whole system works. Major steps of the system include image normalization, dimension reduction, training, testing and error estimation.

The major task in the normalization period is to extract features by removing the background, eliminating the noise, reshaping the image, finding the pig’s head and rotating the pig to the horizontal axis.

In the training stage, three classifiers (KNN, RF and NN) are tested separately based on the same training data. For the three algorithms, KNN and NN were used as regression methods and RF as a classification one.

When it comes to experiments, the efficiency and accuracy of normalization method, IPCA and three classifiers were analyzed and discussed.

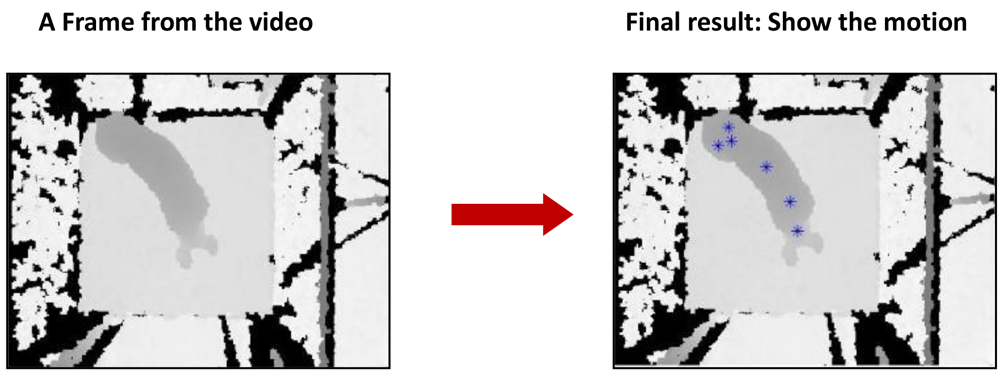


Figure 3.1: Inputs and outputs

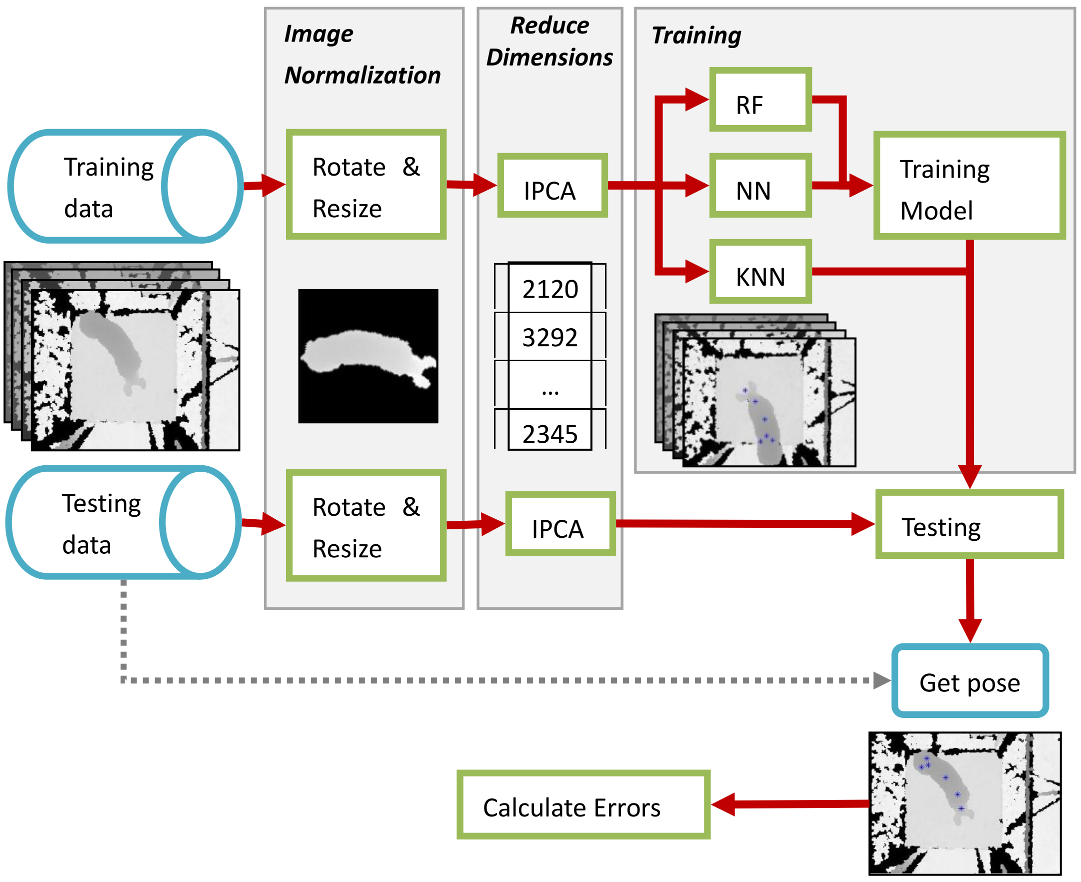


Figure 3.2: Structure of the whole system

3.2 Data Prepared

3.2.1 Depth Map

All data used in this project was provided by the Vet School. Totally 75 videos were recorded by both RGB and Kinect depth camera in the rate of 30 fps. In this project, only depth images will be used. The depth images whose size is 240 by 320 were saved as binary files. The rows and columns of the file represent coordinates (x, y) and the value is the depth of the pixel in millimeter. Files of one video were saved in a folder. Figure 3.3 (a) is shows a depth map recorded by the depth camera. The depth images were stored as a 240 by 320 matrix containing the value of each pixel in the image.

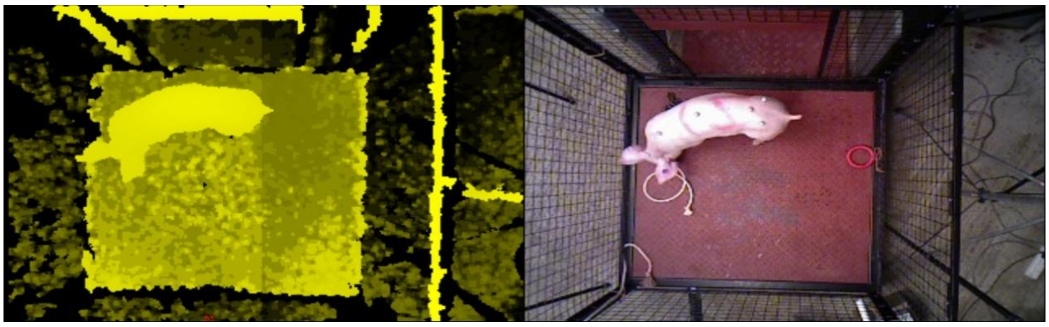


Figure 3.3: (a) A frame recorded by depth camera (b) A frame recorded by RGB camera

3.2.2 MOCAP data

The pose of the pig is detected by 6 markers, three of which are the labels of bottom, one back, one neck and one head, as Figure 3.3 (b) illustrates. Then the locations of the markers in a three-dimensional space were recorded. For each video, the locations were saved as a mat file named as 'mocap_data.mat', where each row represents a depth map and the 18 columns are the coordinates of the markers. As table 1 indicates:

Table 1: Structure of the mocap data (size: n*18)

	X1	Y1	Z1	…	X6	Y6	Z6
Frame 1	-108.95	446.35	279.76	…	259.37	-245.34	571.679
…	…	…	…	…	…	…	…
Frame n	-110.43	445.82	280.72	…	257.02	-245.84	572.70

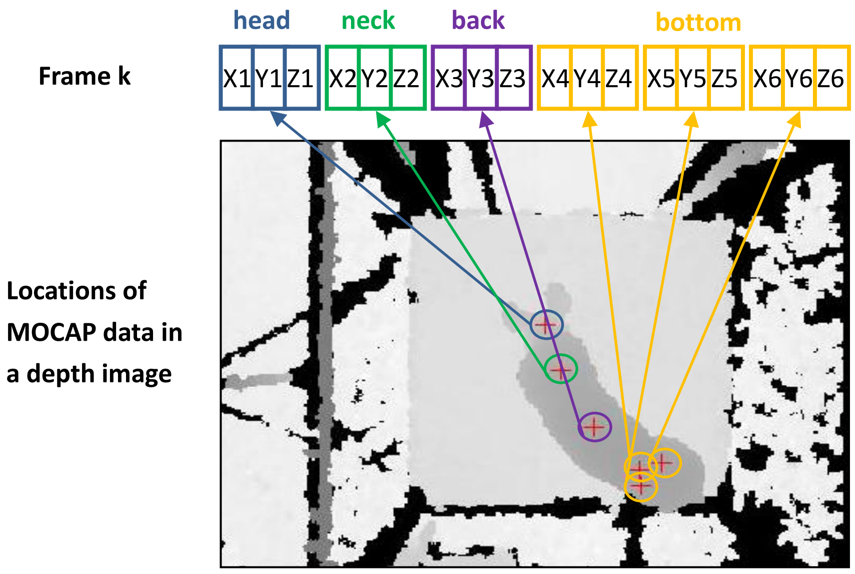


Figure 3.4: MOCAP data

Since the pig stretched and struggled for some time, locations of the markers were not able to be detected correctly in some depth maps. In this case, by checking the locations of the markers, the last 5 videos were not included in the project as too many errors appear in localization. In the test, Video 3 to Video

42 are used as training data and the rest of them are testing ones. Figure 3.5 shows frames with wrong MOCAP data.

While doing training, 130 depth images were randomly picked from each training video so that there're totally 5200 samples in the training database.

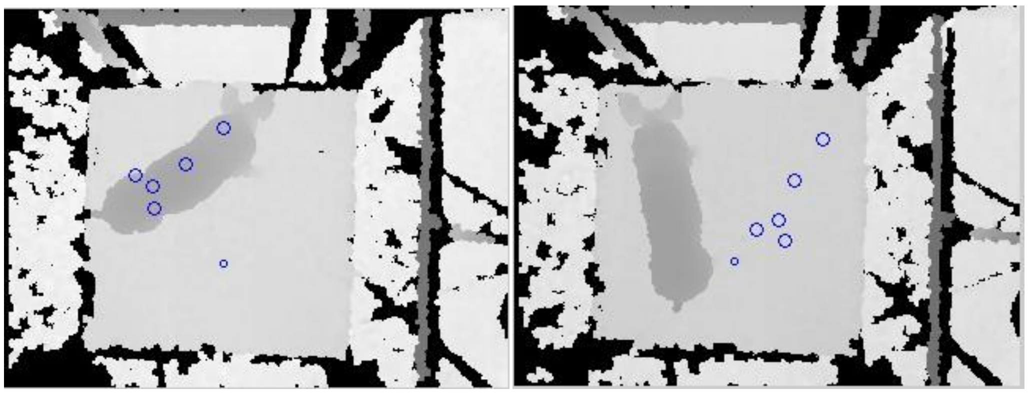


Figure 3.5: Wrong MOCAP data

3.3 Extract Feature

To extract features of the pig, background should be removed first. It takes three steps to achieve this process.

The first step is to remove the background where the pig will never reach such as the right part of the cage, the regions of which are estimated manually. Blue regions in Figure 3.6 are areas that the pig is not able to access.

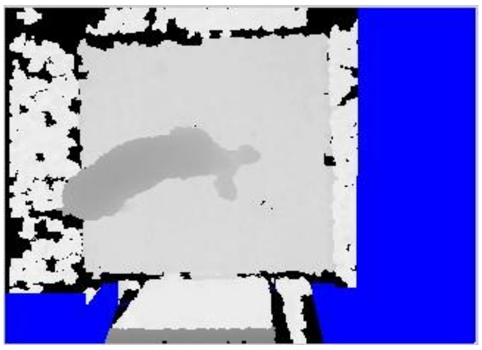


Figure 3.6: Removed areas

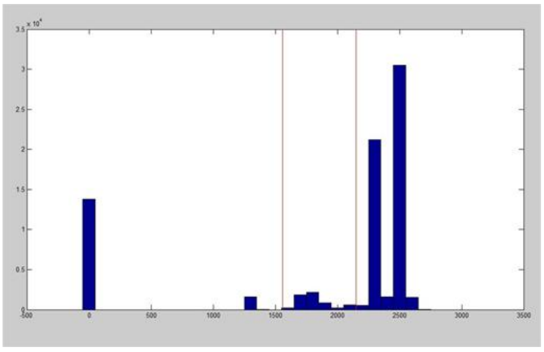


Figure 3.7: Threshold

The next step is to work out the histogram of the depth image and decide thresholds to keep the information of the pig and remove the background. As can be seen, distance between the pig and the camera is always smaller than that between the floors and the camera but larger than that between the camera and the cage. In this case, thresholds, which are 1560 mm and 2150 mm, were worked out manually by observing all the depth images. Figure 3.7 shows the histogram of one depth map. The red lines are the thresholds. The pig's features can recognize and extracted within the range between 1560 and 2150 mm.

After remove the background by thresholding, there are still some small noise dots as blue areas shown in Figure 3.8 (a), which need to be eliminated in the third step. Specifically, only the largest consistent region is kept for every depth map. This process can be achieved by connected components method.

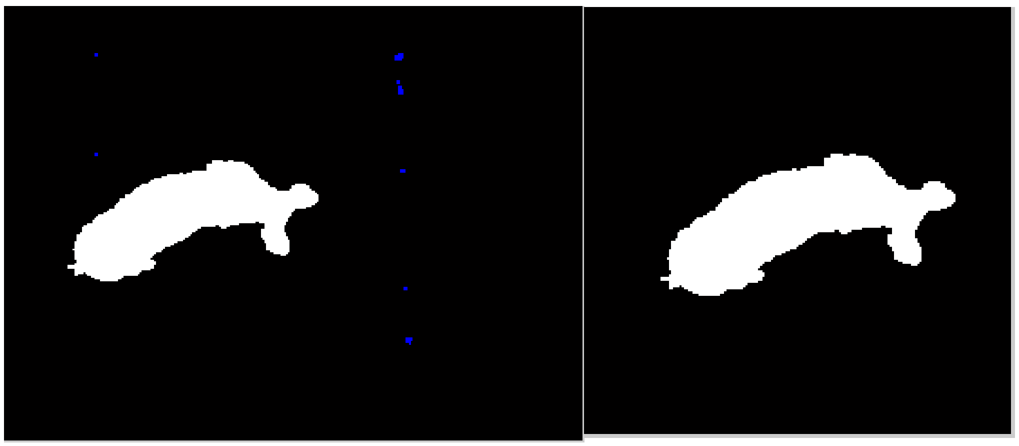


Figure 3.8: (a) Noise **(b) Background removed**

3.4 Normalization

3.4.1 Image normalization

In this project, the original depth image will be rotated and resized before training. The center of the pig is detected by the median algorithm and the images are accordingly resized. Least square line fitting algorithm is used to rotate the pig to the horizontal axis. Meanwhile, as the sizes of the pigs in the video are different, the images need to be resized.

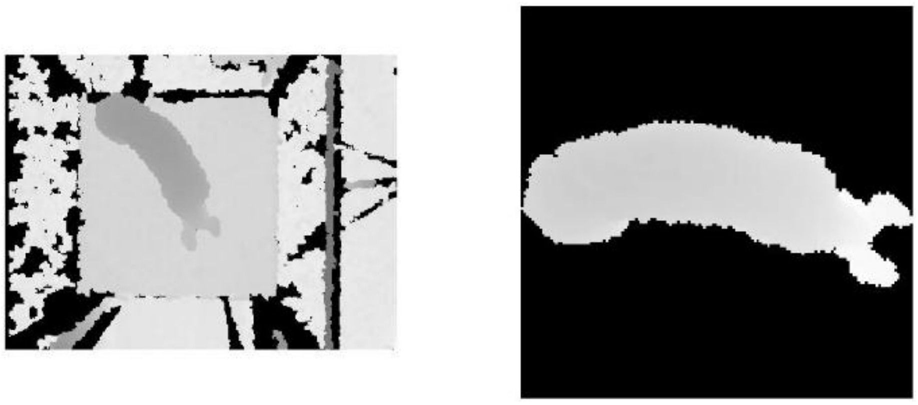


Figure 3.9: (a) 320x240 Depth image **(b) 50x50normalized image**

Figure 3.9 (b) illustrates the output of normalization. In this step, the inputs are 240 by 320 depth images recorded by depth camera. The outputs are 50 by 50 images, where non-zero pixels storing the depth values of the points are the features of the pig. Therefore, after normalization stage, features of each image have been reduced to 2500.

A. Resize the image

The original image was cut into a 201 by 201 image according to the center of the pig first. The square image can then be rotated towards the center point (101, 101), as the image below shows. Then center of the pig was detected by using median method, the idea of which is to sort the coordinates of the pixels representing the depth of the pig according to their locations in one column and let the coordinate in the middle of the column be the center.

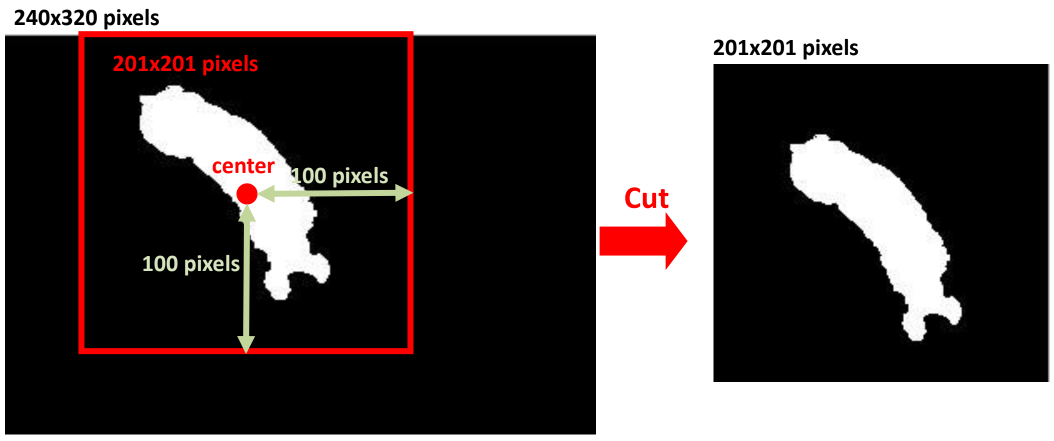


Figure 3.10: Resize

B. Rotation

In terms of rotation, all depth images will be rotated to the horizontal axis, the angle of which was computed by doing arctangent on the straight slope of the linear equation as Figure 3.10 illustrates.

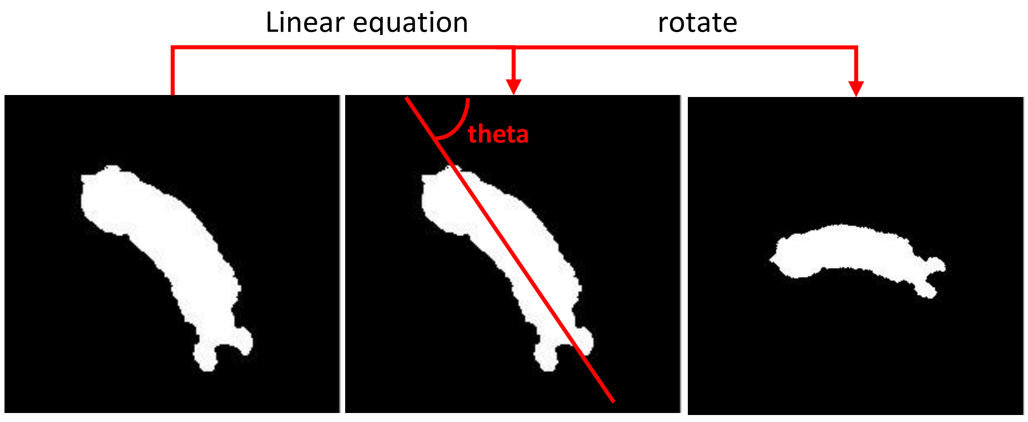


Figure 3.11: Rotation

As can be seen from Figure 3.11, normally, after rotation by least square fitting, the size of the pig will be changed so that the images need to be resized again.

C. *Resize the image*

As can be seen from Figure 3.11, after rotation, the size of the pig changed and there was lots of background information which was useless. In this case, the image should be resized again to a default size to keep the maximum useful features. As Figure 3.12 demonstrates, the maximum height Dy and Dx width of the pig were detected, among which the larger value was assumed to be the size of the new image. In this example, the image was cut into a Dx by Dx new image.

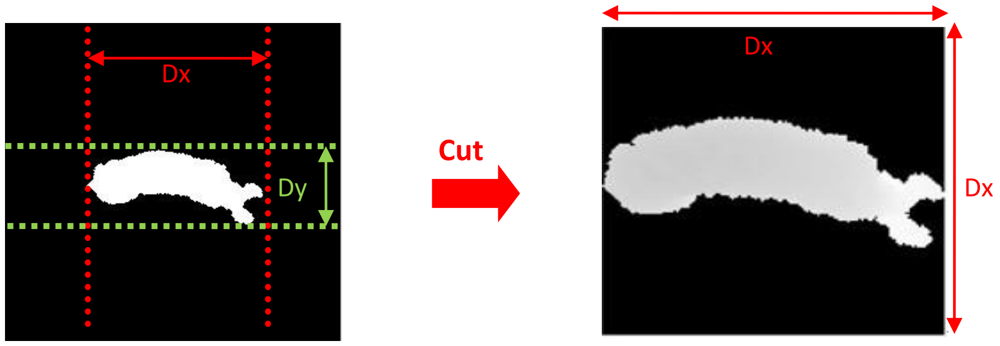


Figure 3.12: Resize the image

D. *Find the Head*

In order to keep the head in the right side, a simple recognition method based on depth information and standard deviation was applied after rotation.

(1) Firstly, calculate the total length of the pig and equally cut the pig into three parts, which are the right side, the middle and the left side. Since the image has been already rotated to the horizontal axis, the task is to work out whether the head is on the right side or the left side.

(2) In most cases, the linear equation worked out the least square fitting algorithm is not the desired one when the straight slope of the equation is larger than 1 (45 degree) as Figure 3.14 (b) demonstrates. So the image will be rotated by 90 degree if the slope is over 1.

(3) Get the centers of both sides and calculate the mean distances between the centers and edge of the pig. Canny edge detection algorithm was used to work out the edge.

(4) Based on the centers of two sides and mean distances as radius, draw a circle in both sides and collect pixels that are in the circle.

(5) Calculate the deviation of the distance between the center and those pixels in the circle and work out the standard deviation.

(6) In the case shown in Figure 3.14 (c), when the gap between two standard deviations is large enough (>2), the side with bigger one is the head. If the head is on the left side, then rotate the image by 180 degree.

(7) Else if the gap is small, then calculate the standard deviation of the depth change in the same way, the side with the smaller value is the head.

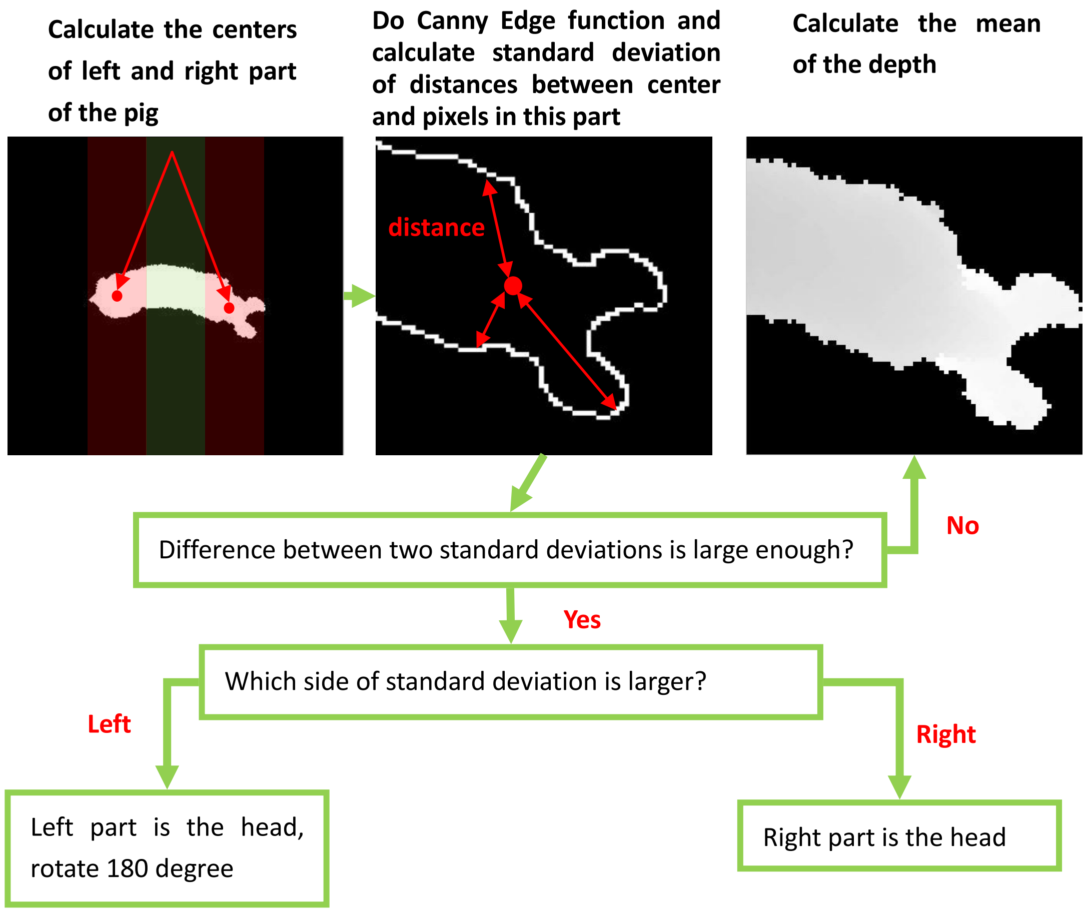

Calculate the centers of left and right part of the pig
Do Canny Edge function and calculate standard deviation of distances between center and pixels in this part
Calculate the mean of the depth
distance
Difference between two standard deviations is large enough?
No
Yes
Which side of standard deviation is larger?
Left
Right
Left part is the head, rotate 180 degree
Right part is the head

Figure 3.13: Find the head

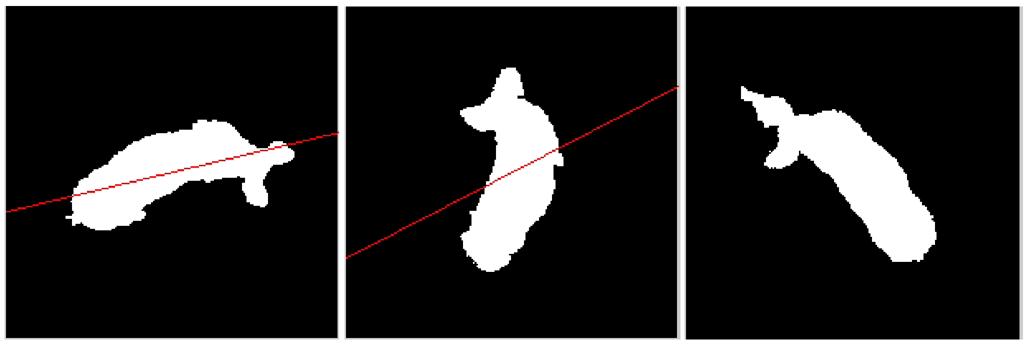


Figure 3.14: (a) Right slope (b) Wrong slope (c) Reversed head

As Figure 3.14 (c) shows, if the head is on the left side, then the system has got a wrong rotation result. To see the efficiency of the algorithm, a method to detect the error rate of the algorithm was introduced, which will be discussed in Section 4. After checking the frequency when the head is wrongly positioned by comparing the head with MOCAP data, the accuracy of this method is 98.14%.

D. Reshape the image

The final step in the normalization stage is to reshape all the images into a 50 by 50 one to ensure that each image has 2500 features.

3.4.2 Dimension Reduction

IPCA was applied instead of PCA to reduce the number of features, as PCA would result in a problem of out of memory in Matlab when doing singular value decomposition on a large size of training data (2500 by 5200).

The inputs of IPCA are 2500 features of 5200 depth images and the outputs are 12 features per each image. The number of features chosen was 12 became it gave the best results which will be discussed further in Section 4.

3.5 Training

Both RF and NN need to build a training model before testing the data while KNN can be called directly. Training data were picked up from Video 3 to Video 43 and 130 depth images were randomly picked up from each video. The process of choosing training data was done when doing IPCA and saved as ‘trainingDATA.mat’ to make sure that all the classifiers were learned from the same set of data.

3.6 Testing

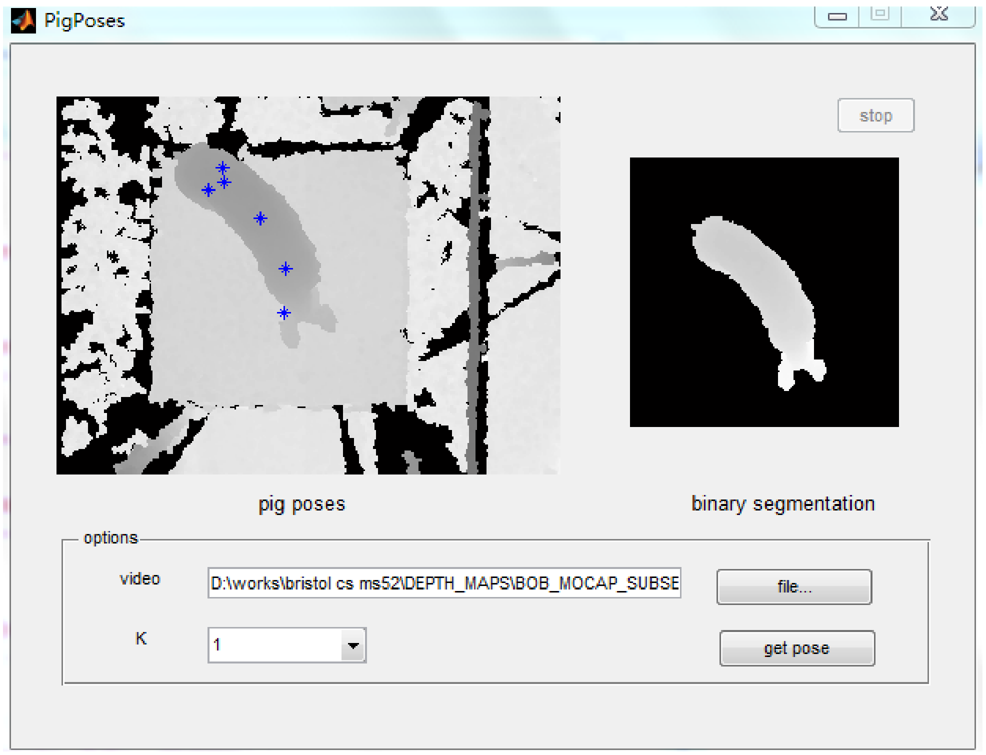
The first 220 frames per a video were chosen to be tested since the markers are very likely to be mistakenly positioned in the later period of the video when pig intended to go out of the cage. As a result, those frames were discarded to avoid noise. To test a single clip, Video 2 was used while Video 44 to Video 69 were tested when analyzing the overall accuracy.

3.7 Experiments

The distances and the angles between the location of estimated markers and the real markers were compared. As mentioned before, sometimes the location of the markers was lost while recording the data because of the technical limitation of the hardware. In specific, coordinates given to the receiver are zero. While doing comparisons, a function was created to remove those coordinates.

3.8 Software

Based on training data provided as mentioned before, software with IPCA and KNN was developed by Matlab GUI. The reason why KNN was chosen as a classifier is that it does not need to train a model so that users can directly see the results of the detection. In other words, the software is able to detect the pose in 0.952336 second per each frame, which will be discussed further in the next section. It took about 1 minute for the other two classifiers to build a model before determining the pig's pose the pig (see Section 4.5). As shown in Figure 3.15, the route of the video and parameter K are adjustable.



video: D:\works\bristol cs ms52\DEPTH_MAPS\BOB_MOCAP_SUBSE

K: 1

Figure 3.15: Software Interface

4. Experiments

To find out how well the system worked, the MOCAP data estimated by different classifiers and algorithms were compared with the real MOCAP data of the depth image. In the following sections, all the real MOCAP data were represented by red crosses and the estimated ones were blue stars.

In this approach, two errors will be analyzed, the distance and the angle errors between the estimated markers and the real markers. And the accuracy of the algorithms was also computed. If the distance error is larger than 6.5 cm, which is one fourth of the length of the pig's head, or the error of the angle is larger than 15 degree, then it is assumed that the pose is wrongly estimated. In this case, accuracy of angles was considered more important than that of distance, as behaviour analysis depend largely on poses of the pig rather than its location. On the other hand, distance error was still a good reference when measuring the efficiency of the system. Experiments were based on two sets of data as shown in the table below. In the sections of IPCA and Comparison, Set I was used while data of Set II was tested in KNN, RF and NN sections.

Table 2: Testing data

Set I	Video 2
Set II	Video 44 – Video 69

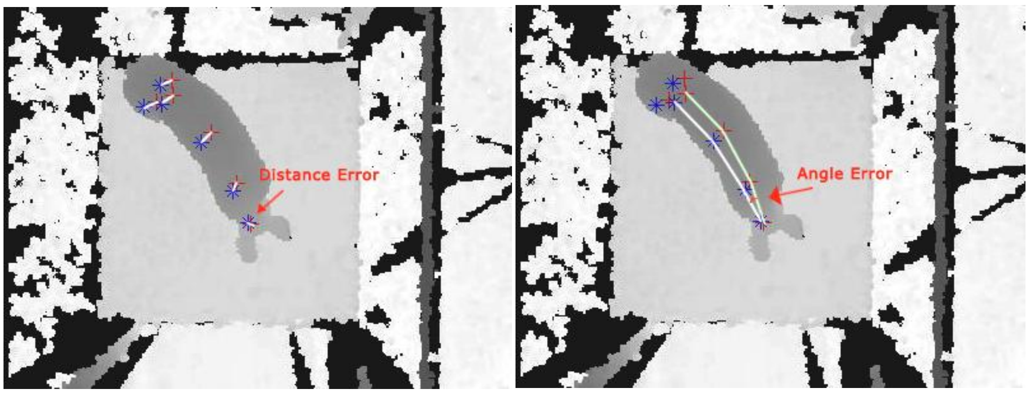


Figure 4.1: (a) Distance error **(b) Angle error**

4.1 Rotation

A system was written to check the accuracy of rotation. In the depth image, the head can be either on the left or the right part of the image. By collecting the location of the head in the motion capture data, the system is able to determine whether the head is in the right location. After comparing images in both Set I and Set II, the accuracy of which was 98.26%. Figure 4.2 shows the process of checking system.

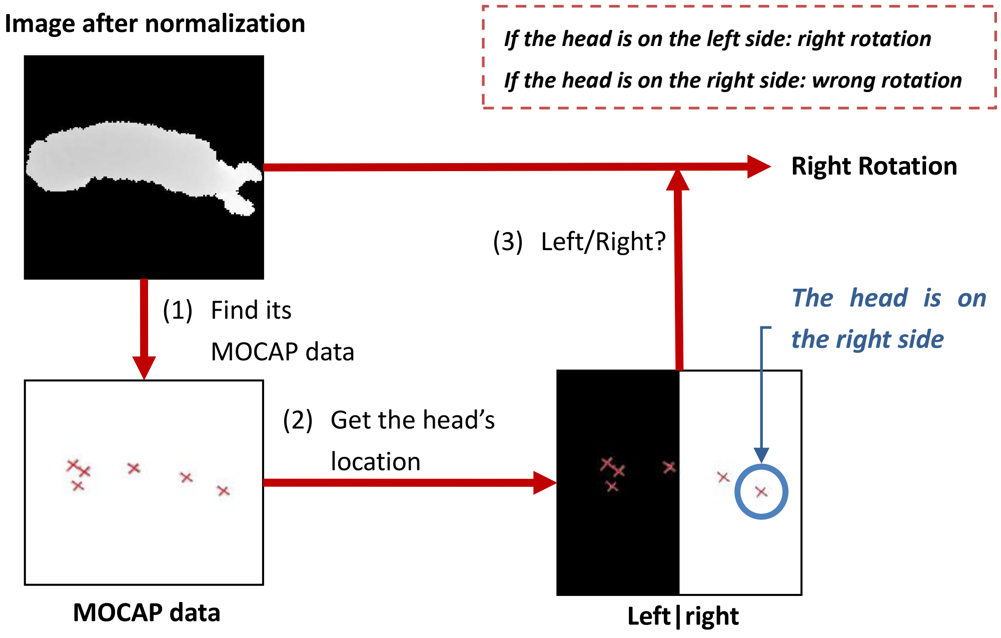


Image after normalization
If the head is on the left side: right rotation
If the head is on the right side: wrong rotation
Right Rotation
(3) Left/Right?
The head is on the right side
(1) Find its MOCAP data
(2) Get the head's location
MOCAP data
Left|right

Figure 4.2: Rotation accuracy measurement

4.2 IPCA

Experiments on PCA and IPCA were based on KNN as KNN is a basic and stable algorithm. To insure the accuracy, to IPCA, the distance accuracy with different number of features was compared first. In the experiment, the original size of training data in IPCA is 50 and the final size of the training data of both algorithms is 3000. It can be seen that the error difference between two algorithms are no more than 1 degree. For IPCA, the smallest error was 5.8 degree when the number of features is 20 and for PCA is 5.14 degree when the number of features is 50. As the error difference between IPCA and PCA was small, IPCA was used to reduce the dimension instead of PCA which will cause out of memory problem if the size of the training data was large.

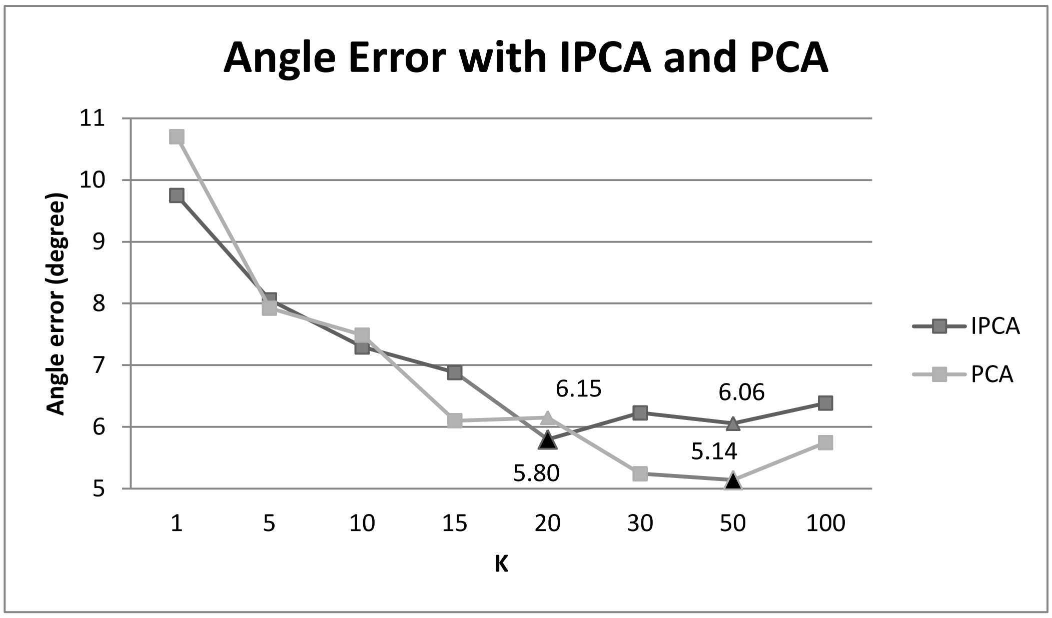


Figure 4.3: Angle error with IPCA and PCA

4.3 KNN

In terms of KNN, three sets of experiments were taken. The first one is to see whether dimension reduction will lose accuracy; secondly, results of KNN with and without IPCA were compared; to analysis the efficiency of the rotation algorithm, KNN with and without rotation were compared; the third one is to discover out the best parameter K, KNN with different K were analyzed. The image below shows the testing depth image and the closest frame found by KNN.

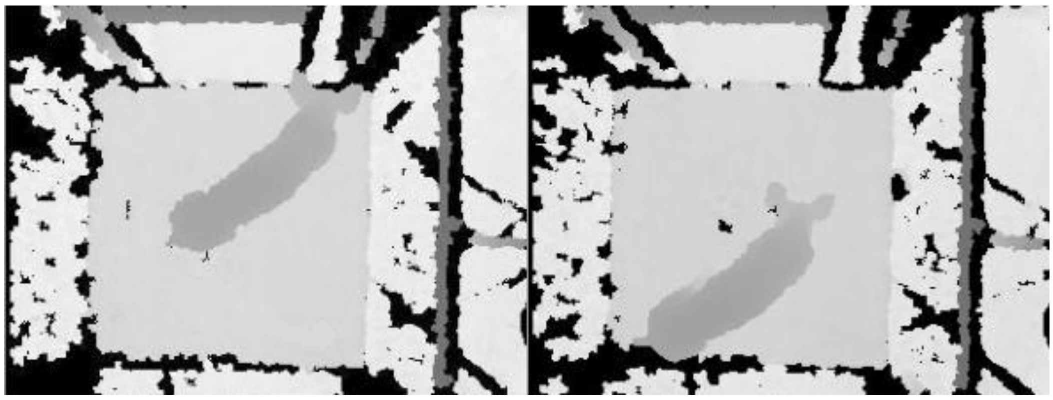


Figure 4.4: (a) Testing frame **(b) Closest frame found by KNN**

4.3.1 KNN and IPCA

When doing IPCA, the system will discard a part of information from data so that it is possible that the system will lose accuracy after doing so. In this case, it’s necessary to work out whether the algorithm is still efficient if IPCA is added. The figure below shows the distance error of two methods. Results of doing KNN directly will be better as K grows, while KNN with IPCA reaches its best result of 1.26 when K was 2, which is better than that without IPCA despite the fact that it normally takes eight hours to do KNN and half hour to do KNN with IPCA. Under this circumstance, it’s ideal to use IPCA before KNN.

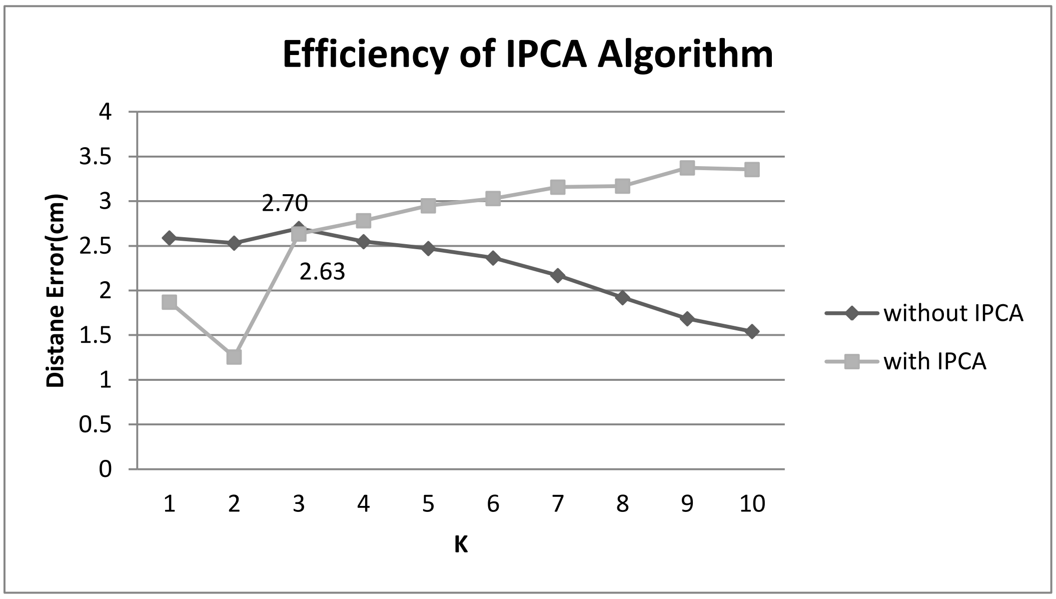


Figure 4.5: Efficiency of IPCA

4.3.2 KNN and Rotation

Visually, it can be seen that pig’s head is quite different from its bottom, which makes it possible to use classifiers directly on depth images without rotating pig’s head to the same direction. In this case, KNN on both original and rotated images were tested. The following figures illustrate the distance errors when the number of features was 20.

As can be seen from the figure below, as K grows, results with are always better than that without rotation. For KNN with rotation, when K is two, the smallest distance error is 1.26 cm while the best result without rotation is 3.38 when K is 8. In both methods, the trends appear gentle.

Whether big difference exists between the head and the bottom can be roughly demonstrated according to the results when K was one, as the algorithm will pick up the most similar image to the testing one from the database. Since the difference error of KNN was as large as 6.75 cm when K was one and the distance was computed according to the Euclidean distance of depth values, it can be concluded that only depth information of the pig was not strong enough to distinguish the head and the bottom. In this case, normalization with regard of both the shape and the depth value can help to improve the results. When the head and the bottom were wrongly segmented, the estimated motion capture data would be reverse to the real motion capture one, as shown in Figure 4.7.

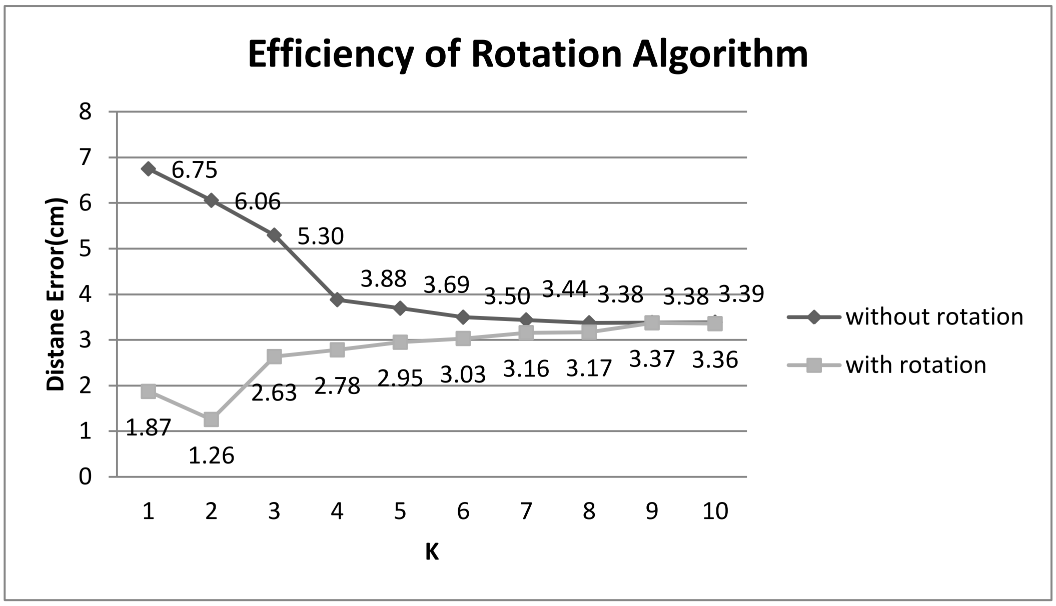


Figure 4.6: Efficiency of rotation

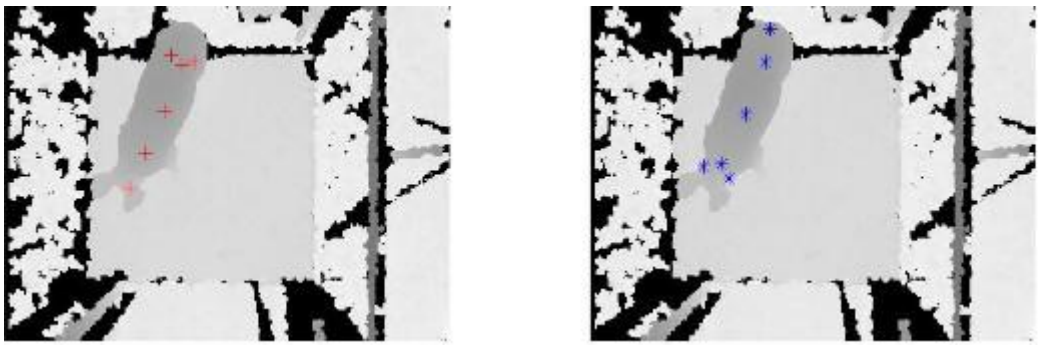


Figure 4.7: (a) Real MOCAP data (b) Estimated MOCAP data with wrongly segmented head

4.3.3 The choice of K

Figure 4.8 shows the accuracy of measurement tested on Set II. When K is 2, the highest accuracy of both distance and angle measurement, which are 95 % and 85 %, can be achieved. After that, both of the accuracies fall gradually. In terms of KNN method, the distance accuracy is overall higher than the angle accuracy. When K is growing bigger, the accuracy falls down. The trends of two measurements are similar to each other. The figure also indicates that the angles were more sensitive to the choice of K.

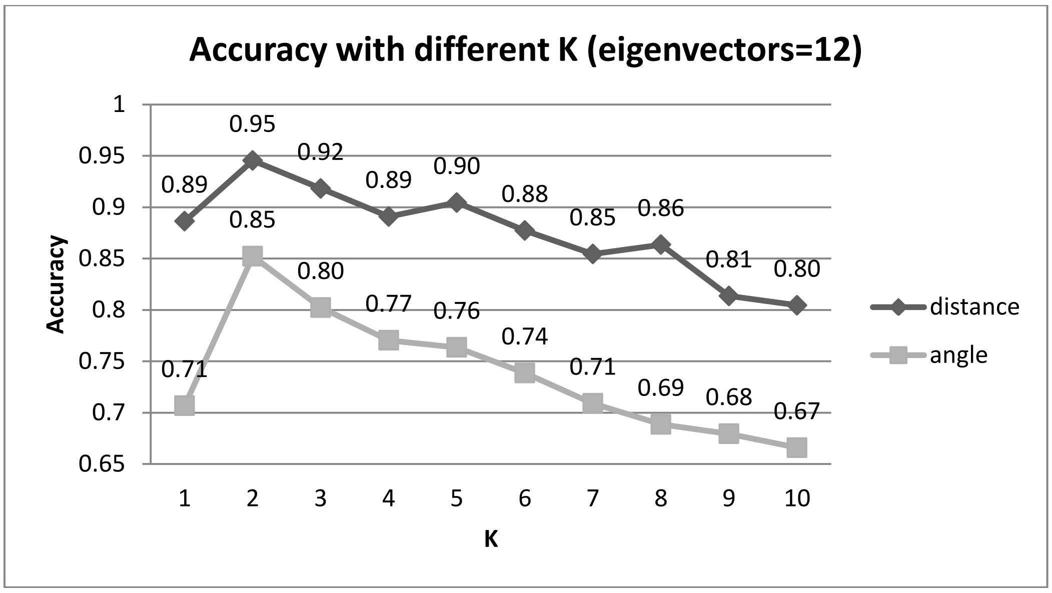


Figure 4.8: Accuracy with different K

4.4 Random forest

Set II was used in the experiment of RF. In this project RF algorithm was used as a classification. And the table below shows the results when the number of trees was 500 and the number of predictors was 4. According to the table, the angle accuracy can be as high as 92.50 %.

Table 3: RF classification

	Accuracy	Error
Distance	81.82%	3.4956 cm
Angle	92.50%	5.4236 degree

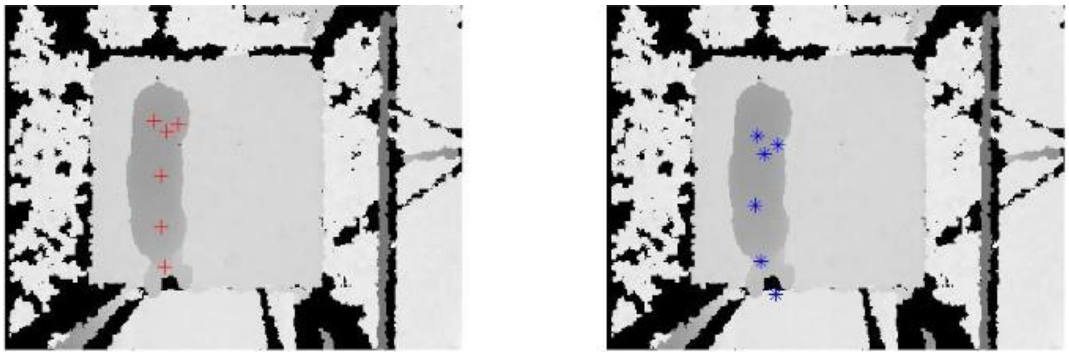


Figure 4.9: (a) Real MOCAP **(b) MOCAP by RF method**

4.4.1 Number of trees

The following graph shows accuracy changes with different number of trees. Not surprisingly, the angle accuracy grew up gradually as the number of trees was larger. The distance accuracy, on the other hand, fluctuated slightly. Besides, when applying RF, the angle accuracy was always higher than the distance accuracy which was different from results of KNN and NN. Results above indicate that the weak classifiers were more sensitive to angle changes than drifts when doing classification. Here the number of predictors was four.

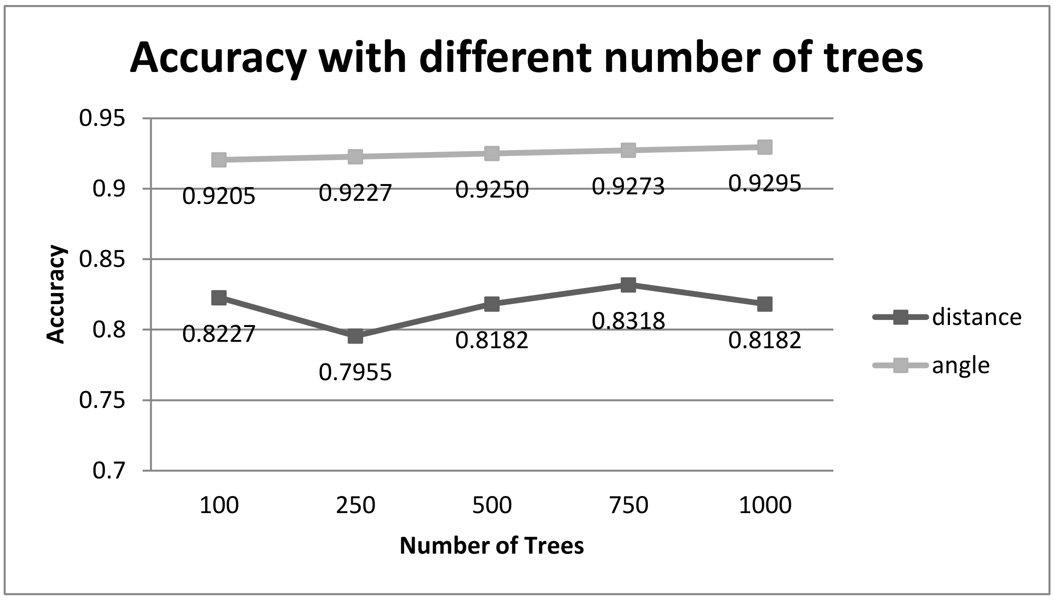


Figure 4.10: Accuracy with different number of trees

4.4.2 Number of predictors

Different number of predictors will also affect the accuracy. As Figure 4.11 illustrates, fluctuations of angle and distance accuracy were similar to each other while the accuracy of angle was still higher than that of the distance. Besides, the number of predictors did little contribution to improve the algorithm, as the accuracy changed little as the number of predictors grew. It can be assumed that the number of predictors has little influence on the accuracy.

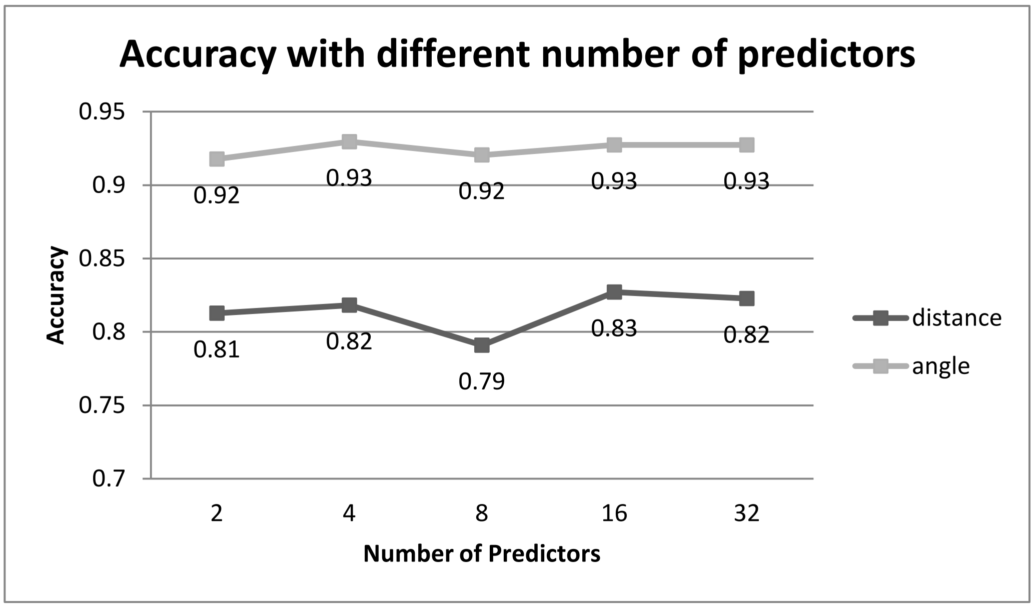


Figure 4.11: Accuracy with different number of predictors

4.5 NN

Data tested in this experiment was based on Set II. Regression Neural network function *newgrnn (P, T)* provided by Matlab was used in this experiment. The inputs are 12 features computed by IPCA and the outputs are 15 normalized coordinates (x, y, z) in a 3D space. Matlab decide that the number of neurons in the hidden layer is the same as that in the input layer, which is 12.

To avoid overfitting problem mentioned in Section 2.5.3 while keeping the accuracy, the radial basis should be adjusted. Figure 4.12 shows the accuracy and the proportion of images without overfitting problem. When the radial basis is less than 70, despite the fact that the angle accuracy was high, no more than half of the frames can be segmented, which means the system cannot detect the pose of most of the frames. The system was able to segment all the images only when the parameter was set to be 130, when the angle accuracy fell down to 86%. In this experiment, it can be seen from Figure 4.13 that the overfitting problem took place when the radial basis was less than 130. As radial basis grows, the accuracy fell down but the system was able to recognize most of the frames.

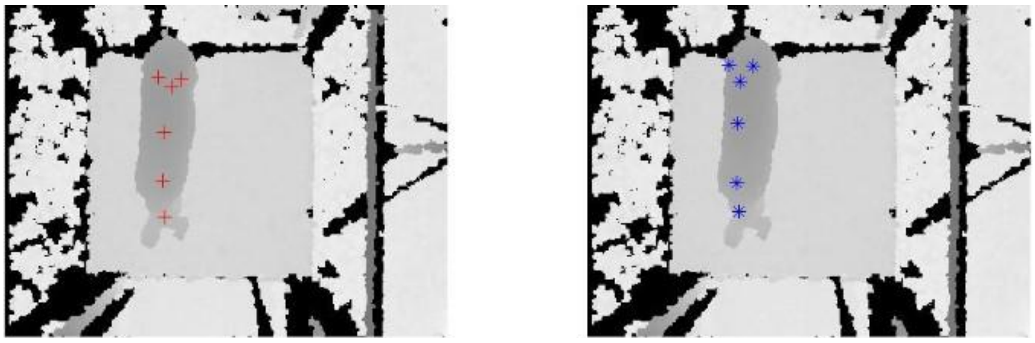


Figure 4.12: (a) Real MOCAP (b) MOCAP data by NN Regression

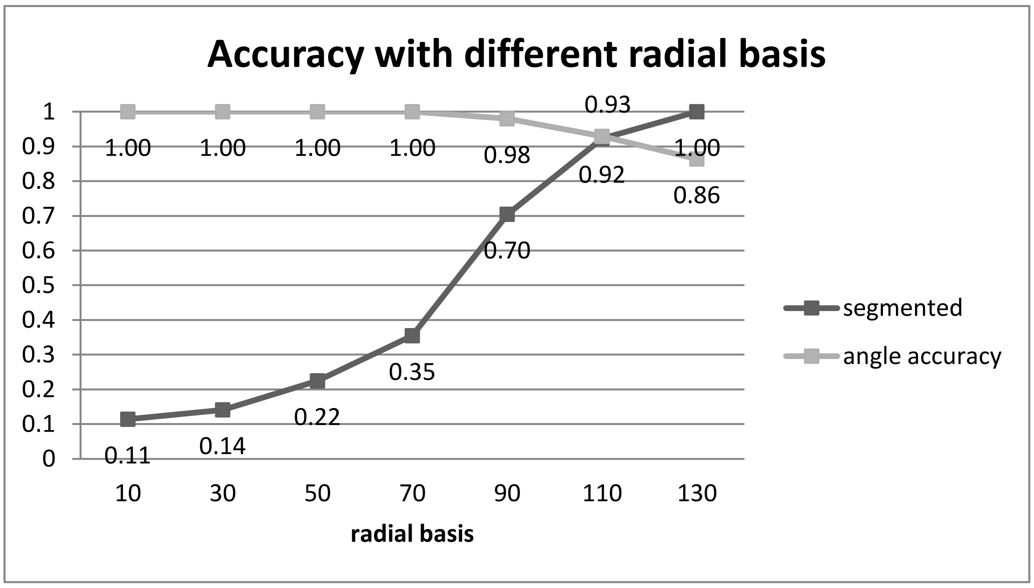


Figure 4.13: Accuracy with different radial basis

4.6 Comparison

4.6.1 Time Cost

Table 4 indicates the time cost of tracking the pose of the pig in one clip. In this section, all the comparisons were based on data of Set I.

Table 4: Testing time (s)

Stage / Classifier	Modeling	Testing	Average Cost (per 1 frame)
KNN	**0**	209.51392	0.952336
RF	55.693192	**38.41508**	**0.427765**
NN	72.265039	41.03374	0.514994

Both RF and NN can do a real-time testing while it takes KNN almost 1 second to do so. While KNN do not need time to get a training model, RF takes least time to test data. Overall, RF is the quickest algorithm. Compared with RF and NN which will take some time to get a model without any outputs, KNN is more user-friendly as it is able to get the pose immediately.

4.6.2 Head Accuracy

The position of the head plays an important role in motion capture. While the bottom, the back and the neck usually kept in one line in the video, the pig's head rotated frequently. In another word, it is comparatively more difficult to track the head correctly than other body parts. However, the head plays an important role in behaviour analysis. Under this circumstance, an experiment measuring the accuracy head detection is introduced here. Despite the fact that RF beat NN when calculating the overall angle accuracy, NN had the highest accuracy in head detection. The reason why KNN performed the best in distance error is that the algorithm classified images according to the distance. On the other hand, NN can detect the head in the highest accuracy, with an error of 2.7621 degree.

Table 5: Head detection

	Distance Error	Distance Accuracy	Angle Error	Angle Accuracy
KNN	**3.4500**	**0.8207**	5.7983	0.7432
RF	5.1618	0.7681	3.0436	0.8304
NN	5.4365	0.8056	**2.7621**	**0.8726**

4.6.3 Distance Accuracy

As mentioned above, distance accuracy may contribute little in behaviour analysis because location is not important. But it is a normal way to measure the method by distance accuracy. Meanwhile, according to experiments above, distance and angle accuracy usually have the same trend as parameters of the algorithms vary.

The best results of three segment methods are as shown below. In terms of distance, KNN had the best result of approximately 95% of accuracy, while RF had the lowest of 83%. It is reasonable that KNN performed the best because it worked out the pose by computing the distance between training and testing images. In other words, the algorithm is based on distance. On the other hand, as trees in RF use weak classifiers to segment images so that they are not sensitive to features which are less important. After normalization, the pose of the body is similar to each other, while the pose of the head is still quite different. In this case,

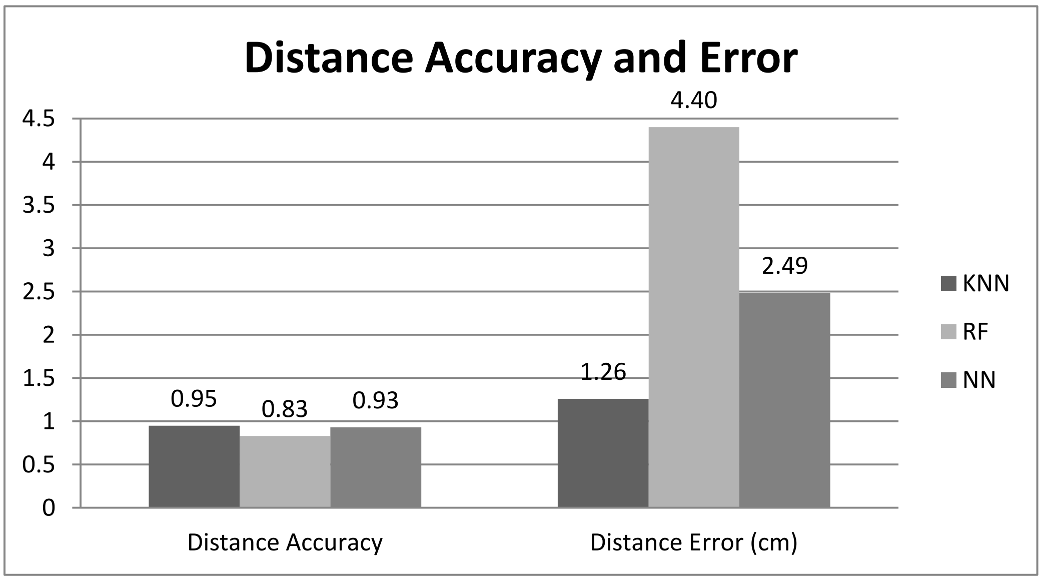


Figure 4.14: Distance accuracy and error

4.6.4 Angle Accuracy

By analyzing angle changes between frames, behaviours such as raise the head and turn around can be detected, which will be helpful in behaviour analysis. As a result, the accuracy of measuring angles is vital. The figure below shows the results of angle measurement. Both RF and NN work well in measuring angles. As they were more sensitive to more distinguishable features, while KNN laid emphasis on the distance of each image and paid little attention to the different weights of each feature.

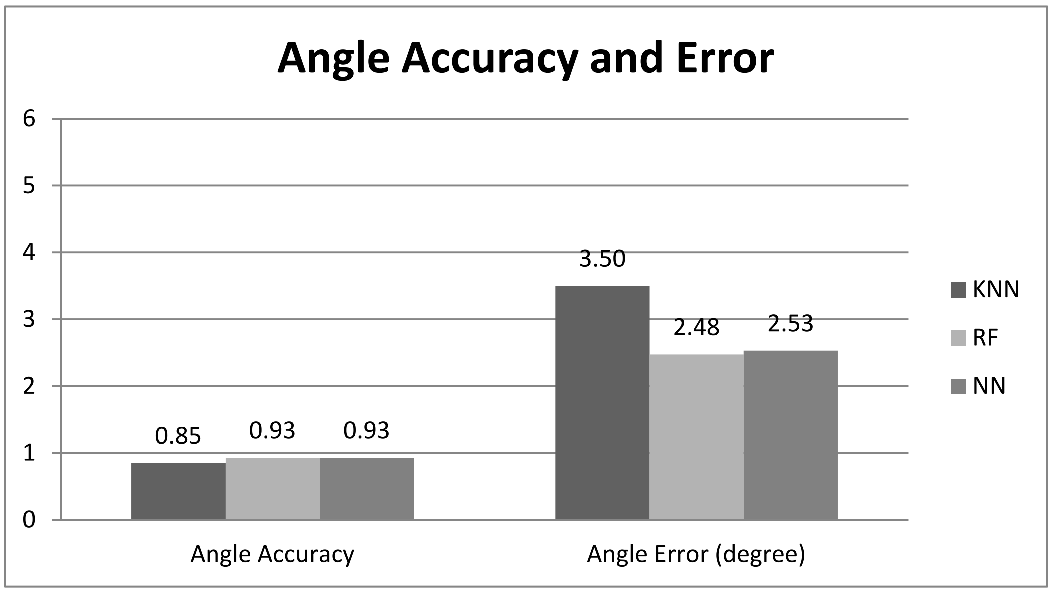


Figure 4.15: Angle accuracy and error

As Figure 4.16 demonstrates, MOCAP worked out by KNN (Figure 4.16 (c)) has the closest distance to the real MOCAP data. However, for MOCAP worked out by KNN, angles between the neck and the back and the one between the back and bottom were straighter than shown in real MOCAP data. On the other hand, while angle difference of the head between the real MOCAP data and data generated by NN was the smallest one among three methods, RF provided the data whose angle is most similar to the real MOCAP data.

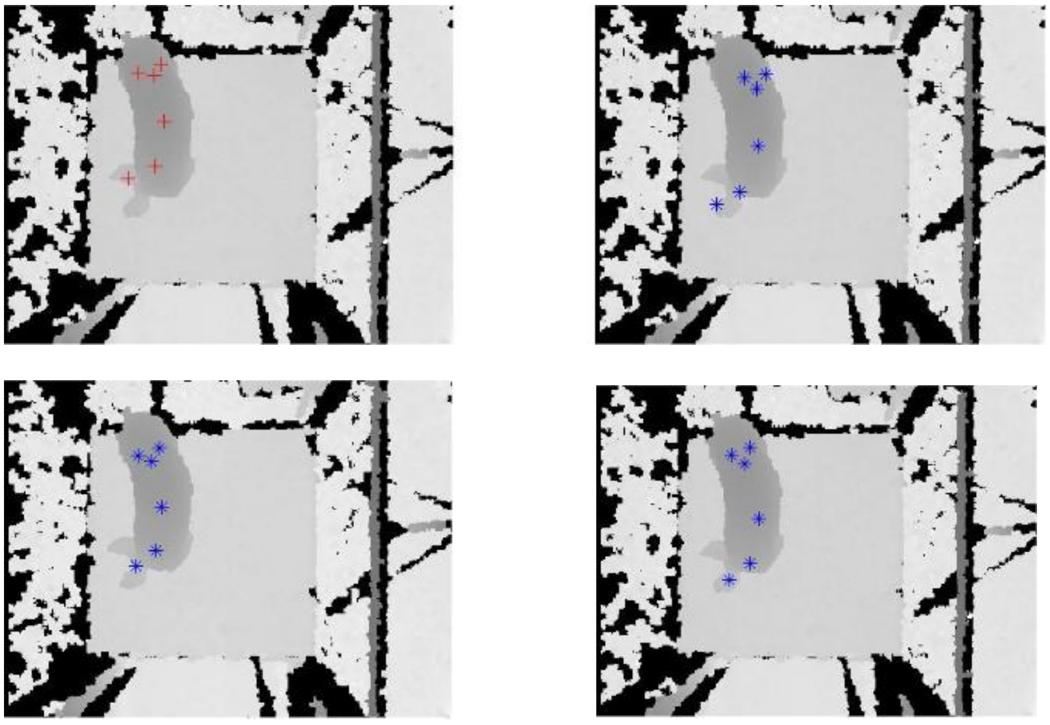


Figure 4.16: (a) Real MOCAP data (b) MOCAP data by NN (c) MOCAP data by KNN (d) MOCAP data by RF

5. Conclusion

Inspired by the Kinect motion capture technology for human and for the welfare of animals, the pig behaviour analysis project was launched to capture the pose of the pig, as automatic motion capture will simplify the process and improve the efficiency of animal behaviour analysis.

In this project the pose of the pig in depth videos was successfully tracked by using different classifiers with a high accuracy. It strongly proved that except tracking human beings, the Kinect technique is also suitable to track the pose of the pig. This approach can make a contribution to automatically behaviour analysis in the field of biology.

5.1 Work has been done

The pig behaviour analysis project has made some achievements such as proving that the Kinect technologies, which had been used only on human beings, can also be used on animal motion capture and working out a piece of software based on the KNN regression method to extract the pose of the pig automatically. The system can be used for pig behaviour analysis in the future.

In addition, it also made three improvements based on the original task: an algorithm has been created to normalize the pig and capturing the head according to the depth information and the shape of the body in an incoutinuous environment; IPCA has been applied save calculation time by reducing the number of features; classification methods (KNN, RF, NN) have been tested on depth images and got high accuracy in distance with the angle measurement whose accuracy were 95% and 93% by analyzing the results.

5.2 Results

Overall, as different classification methods worked well on the depth. To lower the dimensions of the feature vectors, IPCA (Incremental Principal Components Analysis) was applied on depth data before classification. IPCA can largely reduce the time of classification while keeping a relatively high accuracy. In the process of normalization, the results were better after normalizing the input images by rotating the pig to the horizontal axis with the head facing right. To make segmentation, KNN (K-nearest Neighbours), RF (Random Forest), NN (Neural Network) were tested on the depth images. While KNN got the best result in distance measurement, RF and NN worked well on angle measurement.

The most important results were shown below:

(1) All the algorithms can make classification in less than 1 second. For each frame, RF is the quickest algorithm of 0.43 second while KNN is able to make classification in 0.95 second and NN 0.51 second.
(2) IPCA can reduce the time on a large scale and get the best angle accuracy of 5.8 degree when selecting 20 features.
(3) Images with normalization can get more accurate segmentation.
(4) In KNN classification, the highest distance accuracy of 95% occurred when K was 2.
(5) In RF classification and NN regression, the angle accuracy was 93%, which were higher than that of KNN
(6) When using NN as a regression method, the angle accuracy between the neck and the head was the highest, the accuracy of which reached 87%.

5.3 Results analysis

5.3.1 Normalization

In terms of normalization method, the experiments showed that when applying KNN as a classification method, motion capture results can be better with rotation than that without rotation. It turned out that the accuracy of the rotation algorithm was 98%.

Since the experiments used KNN as a classifier which made classification according to the difference of depth values between training and testing data, the large degree of error indicates that the depth values only could not segment the bottom and the head accurately.

5.3.2 Feature reduction

Accuracy resulting from PCA and IPCA were close and the difference of angle error of two algorithms was less than one degree. However, while creating a new eigen-space, PCA did an orthogonal algorithm on all the data, the matrix of which can be as large as 2500 by 5200, Matlab system was very likely to face an out of memory problem. Instead of building the eigenspace in a single calculation, IPCA built the eigen-space with regard of a small number of samples and adjusted the space gradually. As a result, the approach of IPCA solved the out of memory problem.

In terms of accuracy, KNN was applied as the classifier. As K grew, results without IPCA became better while results with IPCA became worse. The reason for this phenomenon is that the KNN normally has a peak with the best result when K reached a certain value. When doing IPCA, less features need to be compared so that the peak came earlier than the algorithm without IPCA, when there were too many features to calculate leading the estimated MOCAP data adjusted in a slow rate.

5.3.3 Classification and regression

All three classifiers tested in the project had their advantages and disadvantages. KNN has the highest accuracy of 95% in distance measurement, since the algorithm itself makes classification according to the distances. Angle accuracy of RF and NN were close around 93%. The system being able to get high accuracy when applying RF as a classifier rather than a regression method indicates that the size of the training database was sufficient enough so that the system can often make the right classification even if there was only one sample in each class. The regression method NN was able to get the highest accuracy of 87% when measuring angle error of the head. This is possibly because compared with the other two algorithms, NN treated features more 'fairly' by giving weight to all the features. Meanwhile, the regression method is more flexible than classification as it is able to work out more concrete values by measuring the similarity between the testing data and each class.

5.4 Challenges

In this project, there are two main challenges: to find the head of the pig and to accelerate algorithms for classification.

5.4.1 Find pigs' head

As training data was randomly chosen from videos, in order to rotate the pig, the head of the pig needs to be recognized first. In this step, the biggest difficulty is to segment the head between inconsistent frames so that common algorithms depending on the consistency between frames. To find the head of the pig, a new algorithm based on both shape and depth information was introduced. The algorithm is able to find the body with an accuracy of 98%.

5.4.2 Accelerate classification algorithms

The project used IPCA to accelerate classification methods. IPCA is able to reduce the training time from seven and eight hours to half an hour.

5.5 Problems and difficulties

Except the limitations of the algorithms, estimation errors may also be caused by other factors such as measurement error and environments change.

5.5.1 Reflection markers

Sometimes the estimated MOCAP data seemed to be more reasonable than the real data. Since when the pig moves quickly, the reflection markers cannot respond in real time, the delay in which causes drift errors in the MOCAP data. Even worse, some of the depth images were wrongly labeled so that noises were added in classification. As shown in Figure 5.1, the location of estimated head (right image) is better than the real location the marker has labeled. In the experiment section, frames with wrong labels on pigs' body were deleted manually.

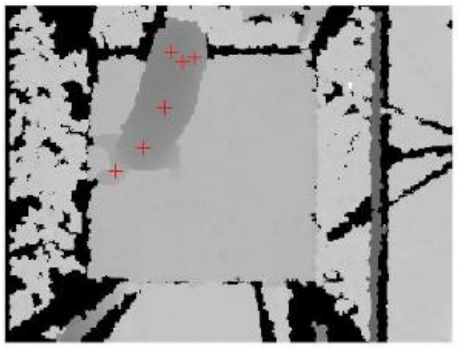
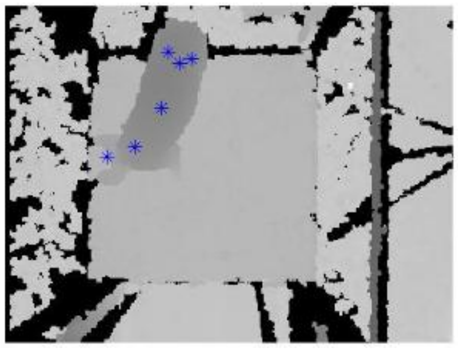



Figure 5.1: (a) Real MOCAP data **(b) Estimated MOCAP data**

5.5.2 Size of the pig

Videos were recorded on different pigs at different times so that the size of the pig varies in different videos. To solve the problem, the images were reshaped into a fixed image. However the problem still occurred when testing data. As the image below shows, when the angles between markers of estimated MOCAP data (Figure 5.2 (b)) was similar to the real MOCAP data (Figure 5.2 (a)), the area covered by the estimated regions was smaller than the real one, which indicates that the pig in the best match image is smaller than that in the testing image.

One possible solution to this problem is to record the size of the pig in the video before normalization and regard the pig's size as a feature when doing classification.

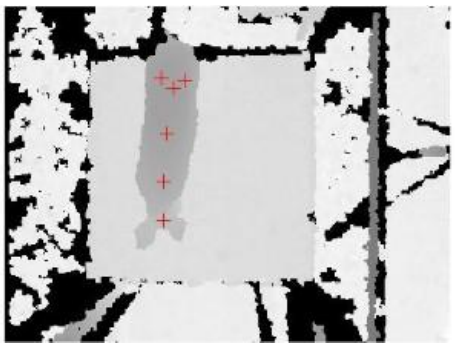
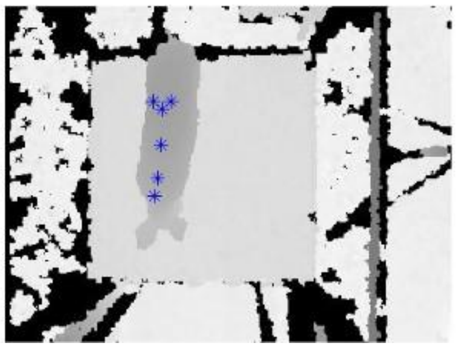



Figure 5.2: (a) Real MOCAP **(b) MOCAP with size problem**

5.6 Improvements and future work

The result from analysis indicates that there are still some limitations which need to be improved and some future work can be done based on this system.

Firstly, the problem caused by different sizes of the pig, which will cause big distance errors, need to be solved.

Furthermore, there is still no method which can be work as a real-time motion capture system. Despite the fact that making classification by RF and NN cost no more than half a second per each frame, both of the algorithms need time for initialization. In this case, there is still space for improvement.

After getting the pig's pose accurately, the next step might be work out the pig's behaviour according to the pig's postures like eating or playing. MOCAP data generated by the system needs to be used and tested to analyze the pig's behaviour to see how much it can contribute to behaviour analysis.

Apart from this, for the welfare of the pig, one more marker can be added on the pig's tail. Getting the motion of the tail can be helpful to analyzing the health condition of the pig [50].

Nevertheless, to see whether the Kinect method can be applied on other creatures, the same experiments can be done on animals such as mice and dogs.

Bibliography

[1] Tirakoat, S.; , "Optimized Motion Capture System for Full Body Human Motion Capturing Case Study of Educational Institution and Small Animation Production," *Digital Media and Digital Content Management (DMDCM), 2011 Workshop on* , vol., no., pp.117-120, 15-16 May 2011

[2] Hamano, Seiya; Takano, Wataru; Nakamura, Yoshihiko; , "Motion data retrieval based on statistic correlation between motion symbol space and language," *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on* , vol., no., pp.3401-3406, 25-30 Sept. 2011

[3] Manoj Perera; Takaaki Shiratori; Shunsuke Kudoh; Atsushi Nakazawa; Katsushi Ikeuchi; , "Task Recognition and Style Analysis in Dance Sequences," *Multisensor Fusion and Integration for Intelligent Systems, 2006 IEEE International Conference on* , vol., no., pp.329-334, Sept. 2006

[4] Wang Manjun; Wang Wei; Li Yan; Qi Baojun; Lv Zhiguo; , "Motion capture data segmentation using kernel dynamic texture," *Audio Language and Image Processing (ICALIP), 2010 International Conference on* , vol., no., pp.592-596, 23-25 Nov. 2010

[5] Zhe Zhang; Qiang Fang; Ferry, F.; , "Upper limb motion capturing and classification for unsupervised stroke rehabilitation," *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society* , vol., no., pp.3832-3836, 7-10 Nov. 2011

[6] Shotton, J.; Fitzgibbon, A.; Cook, M.; Sharp, T.; Finocchio, M.; Moore, R.; Kipman, A.; Blake, A.; , "Real-time human pose recognition in parts from single depth images," *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* , vol., no., pp.1297-1304, 20-25 June 2011

[7] Kalgaonkar, K.; Raj, B.; , "Recognizing talking faces from acoustic Doppler reflections," *Automatic Face & Gesture Recognition, 2008. FG '08. 8th IEEE International Conference on* , vol., no., pp.1-6, 17-19 Sept. 2008

[8] Glardon, P.; Boulic, R.; Thalmann, D.; , "PCA-based walking engine using motion capture data," *Computer Graphics International, 2004. Proceedings* , vol., no., pp.292-298, 19-19 June 2004

[9] Akazawa, Y.; Okada, Y.; Niijima, K.; , "Real-time video based motion capture system based on color and edge distributions," *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on* , vol.2, no., pp. 333- 336 vol.2, 2002

[10] Seferidis, V.; Ghanbari, M.; , "Hierarchical motion estimation using texture analysis," *Image Processing and its Applications, 1992., International Conference on* , vol., no., pp.61-64, 7-9 Apr 1992

[11] Stone, E.E.; Skubic, M.; , "Passive in-home measurement of stride-to-stride gait variability comparing vision and Kinect sensing," *Engineering in Medicine and Biology Society,EMBC, 2011 Annual International Conference of the IEEE* , vol., no., pp.6491-6494, Aug. 30 2011-Sept. 3 2011

[12] Chan-Soo Park; Sung-Wan Kim; Doik Kim; Sang-Rok Oh; , "Comparison of plane extraction performance using laser scanner and Kinect," *Ubiquitous Robots and Ambient Intelligence (URAI), 2011 8th International Conference on* , vol., no., pp.153-155, 23-26 Nov. 2011

[13] Lu Xia; Chia-Chih Chen; Aggarwal, J.K.; , "Human detection using depth information by Kinect," *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on* , vol., no., pp.15-22, 20-25 June 2011

[14] Abramov, A.; Pauwels, K.; Papon, J.; Worgotter, F.; Dellen, B.; , "Depth-supported real-time video segmentation with the Kinect," *Applications of Computer Vision (WACV), 2012 IEEE Workshop on* , vol., no., pp.457-464, 9-11 Jan. 2012

[15] Rakprayoon, P.; Ruchanurucks, M.; Coundoul, A.; , "Kinect-based obstacle detection for manipulator," *System Integration (SII), 2011 IEEE/SICE International Symposium on* , vol., no., pp.68-73, 20-22 Dec. 2011

[16] Boyd, J.E.; Godbout, A.; Thornton, C.; , "In Situ Motion Capture of Speed Skating: Escaping the Treadmill," *Computer and Robot Vision (CRV), 2012 Ninth Conference on* , vol., no., pp.460-467, 28-30 May 2012

[17] Pearson, K. "On Lines and Planes of Closest Fit to Systems of Points in Space" . *Philosophical Magazine* 2 (6): 559–572 1901.

[18] Glardon, P.; Boulic, R.; Thalmann, D.; , "PCA-based walking engine using motion capture data," *Computer Graphics International, 2004. Proceedings* , vol., no., pp.292-298, 19-19 June 2004

[19] Tsuruta, S.; Kawauchi, Y.; Woong Choi; Hachimura, K.; , "Real-Time Recognition of Body Motion for Virtual Dance Collaboration System," *Artificial Reality and Telexistence, 17th International Conference on* , vol., no., pp.23-30, 28-30 Nov. 2007

[20] Razali, N.S.; Manaf, A.A.; , "Gait recognition using motion capture data," *Informatics and Systems (INFOS), 2012 8th International Conference on* , vol., no., pp.MM-67-MM-71, 14-16 May 2012

[21] Danijel Skočaj, Aleš Leonardis, Incremental and robust learning of subspace representations, *Image and Vision Computing*, Volume 26, Issue 1, 1 January 2008, Pages 27-38, ISSN 0262-8856, 10.1016/j.imavis.2005.07.028.

[22] Juyang Weng; Yilu Zhang; Wey-Shiuan Hwang; , "Candid covariance-free incremental principal component analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* , vol.25, no.8, pp. 1034- 1040, Aug. 2003

[23] D. 0. Loftsgaarden and C. P. Quesenhery, “A nonparametric estimate of a multivariate density functions,” *Ann. Murh. Srut.,* vol. 36, pp. 104Y-1051, June lYh5.

[24] Zhe Zhang; Qiang Fang; Ferry, F.; , "Upper limb motion capturing and classification for unsupervised stroke rehabilitation,"*IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society* , vol., no., pp.3832-3836, 7-10 Nov. 2011

[25] Xueyi Wang; , "A fast exact k-nearest neighbours algorithm for high dimensional search using k-means clustering and triangle inequality," *Neural Networks (IJCNN), The 2011 International Joint Conference on* , vol., no., pp.1293-1299, July 31 2011-Aug. 5 2011

[26] Lijuan Zhou; Linshuang Wang; Xuebin Ge; Qian Shi; , "A clustering-Based KNN improved algorithm CLKNN for text classification,"*Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on* , vol.3, no., pp.212-215, 6-7 March 2010

[27] Breiman, Leo. "Random Forests". *Machine Learning* 45 (1): 5–32. 2001

[28] Khan, R.; Hanbury, A.; Stoettinger, J.; , "Skin detection: A random forest approach," *Image Processing (ICIP), 2010 17th IEEE International Conference on* , vol., no., pp.4613-4616, 26-29 Sept. 2010

[29] Hong Bo Li; Wei Wang; Hong Wei Ding; Jin Dong; , "Trees Weighting Random Forest Method for Classifying High-Dimensional Noisy Data," *e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on* , vol., no., pp.160-163, 10-12 Nov. 2010

[30] Singh, R.K.; Naik, S.K.; Gupta, L.; Balakrishnan, S.; Santhosh, C.; Pai, K.M.; , "Hybrid SVM - Random Forest classication system for oral cancer screening using LIF spectra," *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on* , vol., no., pp.1-4, 8-11 Dec. 2008

[31] Cortes, Corinna; and Vapnik, Vladimir N.; "Support-Vector Networks", Machine Learning, 20, 1995.

[32] Das, S.; Trutoiu, L.; Murai, A.; Alcindor, D.; Oh, M.; De la Torre, F.; Hodgins, J.; , "Quantitative measurement of motor symptoms in Parkinson's disease: A study with full-body motion capture data," *Engineering in Medicine and Biology Society,EMBC, 2011 Annual International Conference of the IEEE* , vol., no., pp.6789-6792, Aug. 30 2011-Sept. 3 2011

[33] Wang Jingfang; , "Non-linear pattern recognition based on SVM and genetic algorithm," *Image Analysis and Signal Processing (IASP), 2011 International Conference on* , vol., no., pp.694-698, 21-23 Oct. 2011

[34] Jarrett, K.; Kavukcuoglu, K.; Ranzato, M.A.; LeCun, Y.; , "What is the best multi-stage architecture for object recognition?," *Computer Vision, 2009 IEEE 12th International Conference on* , vol., no., pp.2146-2153, Sept. 29 2009-Oct. 2 2009

[35] Bain. *Mind and Body: The Theories of Their Relation*. New York: D. Appleton and Company. 1873

[36] David Marr, “A theory of cerebellar cortex,” *Journal of Physioloyg, 1969* vol., no.,pp.437-470, 1969

[37] Yeqin Wang; Hui Wang; Lihong Mo; , "Research on recognition of wood texture based on integrated neural network classifier," *Intelligent Control and Information Processing (ICICIP), 2010 International Conference on* , vol., no., pp.512-515, 13-15 Aug. 2010

[38] Burke, H.B.; Rosen, D.B.; Goodman, P.H.; , "Comparing artificial neural networks to other statistical methods for medical outcome prediction," *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on* , vol.4, no., pp.2213-2216 vol.4, 27 Jun-2 Jul 1994

[39] Matyunin, S.; Vatolin, D.; Berdnikov, Y.; Smirnov, M.; , "Temporal filtering for depth maps generated by Kinect depth camera," *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2011* , vol., no., pp.1-4, 16-18 May 2011

[40] Baak, A.; Muller, M.; Bharaj, G.; Seidel, H.-P.; Theobalt, C.; , "A data-driven approach for real-time full body pose reconstruction from a depth camera," *Computer Vision (ICCV), 2011 IEEE International Conference on* , vol., no., pp.1092-1099, 6-13 Nov. 2011

[41] Bo Wu; Nevatia, R.; , "Detection of multiple, partially occluded humans in a single image by Bayesian combination of edgelet part detectors," *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on* , vol.1, no., pp. 90- 97 Vol. 1, 17-21 Oct. 2005

[42] Lopes, N.V.; Mogadouro do Couto, P.A.; Bustince, H.; Melo-Pinto, P.; , "Automatic Histogram Threshold Using Fuzzy Measures," *Image Processing, IEEE Transactions on* , vol.19, no.1, pp.199-204, Jan. 2010

[43] Jhuang, Hueihan et al. “Automated Home-cage Behavioural Phenotyping of Mice,” *Computer Science and Artificial Intelligence LaboratoryTechnical Report*, MIT-CSAIL-TR-2009-052, CBCL-283, October 26, 2009

[44] Joy M. Greer, Mario R. Capecchi, “Hoxb8 Is Required for Normal Grooming Behaviour in Mice,” *Neuron*, Vol. 33, no.1, pp. 23-34, ISSN 0896-6273, 10.1016/S0896-6273(01)00564-5, 3 Jan 2002

[45] Weixing Zhu; Xuefeng Pu; Xincheng Li; Xiaofang Zhu; , "Automated detection of sick pigs based on machine vision," *Intelligent Computing and Intelligent Systems,*

2009. ICIS 2009. IEEE International Conference on , vol.2, no., pp.790-794, 20-22 Nov. 2009

[46] D. M. Weary, J. M. Huzzey, and M. A. G. von Keyserlingk, “BOARD-INVITED REVIEW: Using behaviour to predict and identify ill health in animals,” *Journal of Animal Science*, vol. 87, no. 2, pp. 770-777, Feb, 2009.

[47] J. Hu, and H. Xin, “Image-processing algorithms for behaviour analysis of group-housed pigs,” *Behaviour Research Methods Instruments & Computers*, vol. 32, no. 1, pp. 72-85, Feb, 2000.

[48] C. G. Van Reenen et al., “Behavioural reactivity of heifer calves in potentially alarming test situations: a multivariate and correlational analysis,” *Applied Animal Behaviour Science*, vol. 85, no. 1-2, Jan 12, 2004.

[49] J. J. Vatine et al., “A novel computerized system for analyzing motor and social behaviour in groups of animals,” *Journal of Neuroscience Methods*, vol. 85, no. 1, Nov 1, 1998.

[50] J. J. Zonderland et al., “Tail posture predicts tail damage among weaned piglets,” Applied Animal Behaviour Science, vol. 121, no. 3-4, pp. 165-170, Dec, 2009.

[51] Asano, M.; Takano, H.; Nakamura, K.; , "Eye detection method robust to facial pose changes for eye input device," *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on* , vol., no., pp.602-607, 9-12 Oct. 2011

[52] Rahman, M.M.; Ishikawa, S.; , "Robust appearance-based human action recognition," *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on* , vol.3, no., pp. 165- 168 Vol.3, 23-26 Aug. 2004

[53] Canny, John; , "A Computational Approach to Edge Detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* , vol.PAMI-8, no.6, pp.679-698, Nov. 1986

[54] Lu Xia; Chia-Chih Chen; Aggarwal, J.K.; , "Human detection using depth information by Kinect," *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on* , vol., no., pp.15-22, 20-25 June 2011

[55] Chakraborty, I.; Paul, T.K.; , "Identification and analysis of human pose in video," *Applied Machine Intelligence and Informatics (SAMI), 2011 IEEE 9th International Symposium on* , vol., no., pp.297-301, 27-29 Jan. 2011

[56] Sharma, A.M.; Venkatesh, K.; Mukerjee, A.; , "Human pose estimation in surveillance videos using temporal continuity on static pose," *Image Information Processing (ICIIP), 2011 International Conference on* , vol., no., pp.1-6, 3-5 Nov. 2011

[57] Balan, A.O.; Black, M.J.; Haussecker, H.; Sigal, L.; , "Shining a Light on Human Pose: On Shadows, Shading and the Estimation of Pose and Shape," *Computer*

Vision, 2007. ICCV 2007. IEEE 11th International Conference on , vol., no., pp.1-8, 14-21 Oct. 2007

[58] Siriteerakul, T.; Sato, Y.; Boonjing, V.; , "Estimating change in head pose from low resolution video using LBP-based tracking," *Intelligent Signal Processing and Communications Systems (ISPACS), 2011 International Symposium on* , vol., no., pp.1-6, 7-9 Dec. 2011

Appendix

1. getPig.m

```
%function: elimate noise
function [pig,rm,cm]=getPig(a)
    resize=201;
    halfsize=(resize-1)/2;
    largestArea=0;
    largestSize=0;
    [L,num] = bwlabel(a);
    for j=1:num
        tempSize=bwarea(L==j);
        if tempSize>=largestSize&&tempSize>100
            largestSize=tempSize;
            largestArea=j;
        end
    end
    if largestArea~=0;
        pigTemp=L==largestArea;
    else
        pigTemp=zeros(240,320);
    end
pigTemp=[zeros(halfsize,319+resize);zeros(240,halfsize),pigTemp,z
eros(240,halfsize);zeros(halfsize,319+resize)];
    [r,c]=find(pigTemp==1);
    rm=round(mean(r(:,1)));
    cm=round(mean(c(:,1)));
    if isempty(r(:,1))==1
        pig=zeros(resize,resize);
    else

pig=pigTemp(rm-halfsize:rm+halfsize,cm-halfsize:cm+halfsize);
    end
end
```

2. segment_v1.m

```
%Normalization
function [hp,a2,rm,cm,angR]=segment_v1(a)
    resize=201;
    halfsize=(resize-1)/2;

    %% delete background rigions
    r2=[3,57,68,77,77,68,1];
    c2=[212,212,197,197,213,240,240];
    r3=[209,224,234,234,217,209];
    c3=[197,197,218,240,240,203];
    r4=[240,320,320,240];
    c4=[28,28,0,0];
    r5=[251,251,320,320];
    c5=[0,240,240,0];
    r6=[220,320,320,220];
    c6=[200,200,240,240];
    r7=[55,55,0,0];
    c7=[204,240,240,204];
    %% head positions
    up=0;down=1;left=2;right=3;
    hp=0;


    background=zeros(240,320);

noise=roipoly(background,r2,c2)+roipoly(background,r3,c3)+roipoly
(background,r4,c4)+roipoly(background,r5,c5)+roipoly(background,r
6,c6)+roipoly(background,r7,c7);
    %% get the basic rigion of the pig
    a0=flipud(reshape(a, [320, 240])');
    a2=a0>=1560&a0<=2150&~noise;
    %%eliminate noise
    [a2,rm,cm]=getPig(a2);
    %% cut the image
    if isnan(rm)||isnan(cm)
        a0=zeros(resize,resize);
    else

a0=[zeros(halfsize,319+resize);zeros(240,halfsize),a0,zeros(240,h
alfsize);zeros(halfsize,319+resize)];
        a0=a0(rm-halfsize:rm+halfsize,cm-halfsize:cm+halfsize);
        for i=1:resize
            for j=1:resize
                if a2(i,j)==0
                    a0(i,j)=0;
                end
            end
       end
    end
    %% region of the head
    [rl,cl]=find(a2~=0);
    rD=max(rl)-min(rl);
```

```
cD=max(cl)-min(cl);
if rD>cD
    tempHead=a0(1:halfsize,:);
    topM=sum(tempHead(1:end))/length(find(tempHead~=0));
    tempHead=a0(halfsize+2:resize,:);
    buttomM=sum(tempHead(1:end))/length(find(tempHead~=0));
    if topM>=buttomM
        hp=up;
    else
        hp=down;
    end
else
    tempHead=a0(:,1:halfsize);
    leftM=sum(tempHead)/length(find(tempHead~=0));
    tempHead=a0(:,halfsize+2:resize);
    rightM=sum(tempHead)/length(find(tempHead~=0));
    if leftM>=rightM
        hp=left;
    else
        hp=right;
    end
end
a2=a0;
%% rotate the pig into horizontal axis
[rl,cl]=find(a2~=0);
rD=max(rl)-min(rl);
cD=max(cl)-min(cl);
[y x] = find(a2 ~= 0);
p=polyfit(x,y,1);
p=p(1);
if rD>cD
    if p>=-0.9&&p<=0.9
        angR=90;
    else
        angR=atan(p(1))*180/pi;
    end
else
    if p>=-0.9&&p<=0.9
        angR=atan(p(1))*180/pi;
    elseif p>0.9
       angR=atan(p(1))*180/pi-90;
    else
       angR=atan(p(1))*180/pi+90;
    end
```

```
end
a2=imrotate(a2,angR);
[y x]=find(a2~=0);
pig_length=round((max(x)-min(x))/3);
b1=a2(min(y):max(y),min(x):min(x)+pig_length);
b2=a2(min(y):max(y),max(x)-pig_length:max(x));
color_b1=mean(b1(find(b1~=0)));
color_b2=mean(b2(find(b2~=0)));
[y x]=find(b1~=0);
b1_center=[round(mean(x)),round(mean(y))];
[y x]=find(b2~=0);
b2_center=[round(mean(x)),round(mean(y))];
a3=edge(a2~=0,'canny');
b1=a3(min(y):max(y),min(x):min(x)+pig_length);
b2=a3(min(y):max(y),max(x)-pig_length:max(x));
[y x]=find(b1~=0);
getD1=[];
for i=1:length(y)
    getD=pdist([x(i),y(i);b1_center],'euclidean');
    if getD<=pig_length;
        getD1=[getD1;getD];
    end
end
getD1=std(getD1);
[y x]=find(b2~=0);
getD2=[];
for i=1:length(y)
    getD=pdist([x(i),y(i);b2_center],'euclidean');
    if getD<=pig_length;
        getD2=[getD2;getD];
    end
end
getD2=std(getD2);
if getD1>getD2
    if abs(getD1-getD2<=2)
        if color_b1>color_b2
            angR=angR+180;
            a2=imrotate(a2,180);
        end
    else
        angR=angR+180;
        a2=imrotate(a2,180);
    end
end
```

```
    [y x]=find(a2~=0);
    min_size=min([min(x),min(y)]);
    max_size=max([max(x),max(y)]);
    a2=a2(min_size:max_size,min_size:max_size);
end
```

3. train_PCA_v1.m

```
%======IPCA======%
clear
%root_folder:root of the trained data
%sampleVideo:video in the last folder
root_folder = 'D:\works\bristol cs
ms52\DEPTH_MAPS\BOB_MOCAP_SUBSET\';
d = dir(root_folder);
d(1:2) = [];
d_sampleVideo = dir([root_folder, d(2).name, '\KINECT\*.depth']);
close all
diff=zeros(length(d_sampleVideo),1);
frameNum=220;
newsize=50;
elNum=12;%number of eigenvector
%% IPCA:intialize test data
PCAT=[];
reShapeT=[];
for i=2:40
    d_labelled = dir([root_folder, d(i).name, '\KINECT\*.depth']);
    reShapeB=zeros(100,6);
    reShapeB=[];
    PCAB=[];
    load([root_folder, d(i).name, '\mocap_data.mat']);
    xs = mocap_data.coords(:,1:3:18) * -1; %% Rotate 180
    %randomly choose 130 frames from each video
    if length(xs)<=length(d_labelled)
        framegetAll=randperm(length(xs));
    else
        framegetAll=randperm(length(d_labelled));
    end
    tk=1;
    for k = 1:length(framegetAll)
        tf=framegetAll(1,k);
        error=length(find(xs(tf,:)>-0.1&xs(tf,:)<0.1));
        if error==0
            dfp =
fopen([root_folder,d(i).name,'\KINECT\',d_labelled(tf).name],'rb'
);
            b = fread(dfp, 320*240, 'float');
            fclose(dfp);
            reShapeB(tk,4)=i;
            reShapeB(tk,5)=tf;

[reShapeB(tk,1),b,reShapeB(tk,2),reShapeB(tk,3),reShapeB(tk,6)]=s
egment_v1(b);
            b=imresize(b,[newsize,newsize]);
            b=b.*(b>0.1);
            PCAB=[PCAB,reshape(b,[],1)];
            tk=tk+1;
        end
        if tk==101;
            break
        end
    end
    PCAT=[PCAT PCAB];
    reShapeT=[reShapeT;reShapeB];
end
save('poseNormal.mat','reShapeT');
save('poseNormal.mat','PCAT','-append');
wrongN=0;
for i=1:length(reShapeT)
    load([root_folder, d(reShapeT(i,4)).name, '\mocap_data.mat']);
    xs = mocap_data.coords(:,1:3:18) * -1; %% Rotate 180
    ys = mocap_data.coords(:,2:3:18) * -1; %% Rotate 180
    xs=abs(xs(reShapeT(i,5),:)./7.7+140);
    ys=240-abs(ys(reShapeT(i,5),:)./7.7+162);
    coord=[xs' ys'];
    center=[reShapeT(i,3)-100 reShapeT(i,2)-100];
    coord=rotation(coord,center,-reShapeT(i,6));
    if coord(1,1)<coord(4,1);
        reShapeT(i,6)=reShapeT(i,6)+180;
        b=reshape(PCAT(:,i),newsize,newsize);
        b=imrotate(b,180);
        b=imresize(b,[newsize,newsize]);
        b=b.*(b>0.1);
        PCAT(:,i)=reshape(b,[],1);
        wrongN=wrongN+1;
    end
end
wrongN/length(reShapeT)
clearvars PCAB;
clearvars k;
clearvars b;
clearvars d;
clearvars x;
clearvars x1;
clearvars x2;
clearvars y;
clearvars y1;
clearvars y2;
clearvars mocap_data;
clearvars mocap_image;
clearvars d_labelled;
clearvars reShapeB;
PCAT=[PCAT;reShapeT'];
PCAT=PCAT(:,randperm(size(PCAT,2)));
reShapeT=PCAT(end-5:end,:)';
save('trainningDATA.mat','reShapeT');
clearvars reShapeT;
PCAT=PCAT(1:end-6,:);
[XmTrain,EVTrain]=createESinc(PCAT,elNum,50);
save('trainningDATA.mat','XmTrain','-append');
save('trainningDATA.mat','EVTrain','-append');
b=zeros(elNum,length(PCAT));
for i=1:length(PCAT')
    b(:,i)=EVTrain(:,1:elNum)'* (PCAT(:,i) - XmTrain);
end
save('trainningDATA.mat','b','-append');
```

4. KNN_v1.m

```
%======test all data======%
clear;
root_folder = 'D:\works\bristol cs
ms52\DEPTH_MAPS\BOB_MOCAP_SUBSET\';
d = dir(root_folder);
d(1:2) = [];
close all;
newsize=50;
elNum=12;
K=2;
load('trainningDATA.mat');
error_mean=zeros(220,2);
testNum=2;
%% test every depth file
    angle_error=zeros(220,2);% mean angle error of the head;mean angle
```

```
error of the bottom
   d_test = dir([root_folder, d(testNum).name, '\KINECT\*.depth']);
   error_frame=0; % how many error frames in the testing video
   wrong_pose=0;
   for j=1:220
      %% open depth file
       dfp =
fopen([root_folder,d(testNum).name,'\KINECT\',d_test(j).name],'rb
');
       a = fread(dfp, 320*240, 'float');
       fclose(dfp);
      %% plot the image
       a2 = flipud(reshape(a, [320, 240])');
       subplot(1,2,1);
       imshow(a2./max(a2(:)));
       hold on;
      %% preprocess the image
       [hp,a2,rm,cm,angR]=segment_v1(a); %% cut the image into a
201x201 segmented one
       a2=imresize(a2,[newsize,newsize]); %% resize the image into
a 50x50 one
      %% do IPCA on testing file
       a2=reshape(a2,[],1);
       a2 = EVTrain(:,1:elNum)' * (a2 - XmTrain);
      %% KNN
       dis=pdist([a2'; b'],'euclidean');
       dis=dis(1:length(b'));
       dis=[reShapeT,dis'];%1:hp 2:rm 3:cm 4:video 5:frame 6:angle
7:distance
       dis=sortrows(dis,7);
       tempK=1;
       trueK=zeros(K,6);
       for p=1:length(dis)
          load([root_folder, d(dis(p,4)).name,
'\mocap_data.mat']);
          xs = mocap_data.coords(:,1:3:18) * -1; %% Rotate 180
          if dis(p,5)<=length(xs)

errTrack=length(find(xs(dis(p,5),:)>-0.1&xs(dis(p,5),:)<0.1));
             if errTrack==0
                trueK(tempK,:)=dis(p,1:6);
                tempK=tempK+1;
             end
          end
```

```
            if tempK>K
                break
            end
        end
        dis=trueK;
       %% get test mocap data
        load([root_folder, d(testNum).name, '\mocap_data.mat']);
        xs = mocap_data.coords(:,1:3:18) * -1; %% Rotate 180
        ys = mocap_data.coords(:,2:3:18) * -1; %% Rotate 180
        mocap_Test=zeros(6,2);
        erro_estimate=0; %% detect wrongly labelled joints in the
testing frame
        for p=1:size(xs,2)
            if j>length(xs)
                erro_estimate=1;
            elseif(xs(j,p)>-0.1&&xs(j,p)<0.1)
                erro_estimate=1;
            else mocap_Test(p,:)=[abs((xs(j,p)./7.7) +
140),240-abs((ys(j,p)./7.7) + 162)];
            end
           %% plot test mocap data
            plot(mocap_Test(p,1),mocap_Test(p,2),'r:+');
        end
       %% open depth file
        dfp =
fopen([root_folder,d(testNum).name,'\KINECT\',d_test(j).name],'rb
');
        a = fread(dfp, 320*240, 'float');
        fclose(dfp);
       %% plot the image
        a2 = flipud(reshape(a, [320, 240])');
        subplot(1,2,2);
        imshow(a2./max(a2(:)));
        hold on;
       %% get mocap data
        tempK=sum(dis(:,4)~=0);
        mocap_Best=zeros(6,2);
        for p=1:tempK
            load([root_folder, d(dis(p,4)).name,
'\mocap_data.mat']);
            xs = mocap_data.coords(:,1:3:18) * -1; %% Rotate 180
            ys = mocap_data.coords(:,2:3:18) * -1; %% Rotate 180
            xs = abs(xs(dis(p,5),:)./7.7+140)+cm-dis(p,3);
            ys = 240-abs(ys(dis(p,5),:)./7.7+162)+rm-dis(p,2);
            coord=[xs',ys'];
            coord=rotation(coord,[cm-100,rm-100],angR-dis(p,6));
            if pdist([mocap_Test(1,:); coord(1,:)],'euclidean')>80
               coord=rotation(coord,[cm-100,rm-100],180);
            end
            mocap_Best=mocap_Best+coord;
        end
       %% plot test mocap data
        mocap_Best=mocap_Best./tempK;
        for p=1:length(mocap_Best)
            plot(mocap_Best(p,1),mocap_Best(p,2),'b:*');
        end
        error=0;
        error_s=0;
        if erro_estimate==1
            error_frame=error_frame+1;
        else
            for p=1:length(mocap_Best)

error=error+sqrt(sum((mocap_Best(p,:)-mocap_Test(p,:)).^2))/10*7.
7;
            end
            error=error/length(mocap_Best);

error_s=pdist([mocap_Best(1,:);mocap_Test(1,:)])/10*7.7;
            if error>=10;
               wrong_pose=wrong_pose+1;
               error=0;
               error_s=0;
            end
        end
        error_mean(j,1)=error;
        error_mean(j,2)=error_s;
    pause(0.005);
end
```