THE APPLICATION OF B-MODE ULTRASONOGRAPHY FOR ANALYSIS OF HUMAN SKELETAL MUSCLE

by

RYAN CUNNINGHAM

A thesis submitted to

The Manchester Metropolitan University

for the degree of

DOCTOR OF PHILOSOPHY

School of Computing, Mathematics and Digital Technology

& School of Healthcare Science

Manchester Metropolitan University

February 2015

Table of Contents

1 LIST OF TABLES

2 LIST OF FIGURES

3 DECLARATION

This thesis is submitted to Manchester Metropolitan University in support of my application for admission to the degree of Doctor of Philosophy. No part of this thesis has been submitted in support of an application for another degree or qualification of this or any other institution of learning. Work relating to this thesis has appeared in the following papers and abstracts:

3.1 PEER REVIEWED PUBLICATIONS

Automated measurement of human skeletal calf muscle contraction via b-mode ultrasound imaging. **Cunningham, Ryan, et al.** 2013. Medical Image Understanding and Analysis.

3.2 ABSTRACTS

Skeletal muscle states predicted from sequential b-mode ultrasound images. **Cunningham, Ryan, et al.** 2014. International Society of Electrophysiology and Kinesiology.

A b-mode ultrasound muscle model for hypothesis testing. **Cunningham, Ryan, et al.** 2014. International Society of Electrophysiology and Kinesiology.

4 ACKNOWLEDGEMENTS

I would like to sincerely thank my supervisor, Prof. Ian Loram, for his unfaltering and passionate help, support and guidance throughout this project. He has been an immense inspiration, a model researcher and an ambassador for research whose involvement has significantly contributed to my development as a student of research.

I would also like to sincerely thank my supervisor, Dr Peter Harding, for his uncompromising support and his hands-on help in the physiology and computing labs. He has been a good friend and a model for dedicated research.

I would also like to sincerely thank my supervisor, Dr Nicholas Costen, for his advice and guidance during my PhD, in my publication writing and in writing my thesis.

I would also like to sincerely thank Cornelis for his help and support around the posture labs. His influence has contributed significantly to my development as a research student by being a dedicated and passionate model researcher.

I would also like to sincerely thank my parents, Tracy and Martin, my brothers Jordan, Michael, Callum, and Dylan, my sister Colleen, and my girlfriend, Lise, who have given me love, support, and understanding throughout my dedication to this project and my life, and have constantly inspired me to aim far beyond my reach as a person.

I would also like to thank my very good friend, Mayur, for his support, encouragement and understanding throughout this dedicated period in my life.

I would also like to sincerely thank my work colleagues who have provided help, support and friendship at varying stages of my time at MMU, in no particular order: Alex, Kate, Maria, Diego, Milos, Kola, Joe, Steve, Jonathan Flavel, Jonathan Worms, Elizabetta, Emma, Neil, John Darby, Gemma, Adrian, Greg, Matt Crossley, and Daniel, Brian.

I would also like to sincerely thank Alison Loram, and Brian Bate for their help and support in my first year as a research student. They have also given me much encouragement and guidance throughout the entire research program and I am eternally grateful.

Without willing participants none of my work would be possible, so I am forever grateful to the following participants: Lise, Mikey, Jenna, Callum, Dylan, Jordan, Daniel Leightly, Daniel Davenport, Charlottte, James, Gemma, Lily, Katerina, Elizabetta, Diego, Maria, Kate, Pete, Ian, Cornelis, Fran, Brian Davies, Brian Bate, Nick, Matt Piaseki, Matt Crossley, Adrian, Jonathan Flavel, Jonathan Worms, Alex, Emma, Theo, Joe, and Steve.

I would also like to sincerely thank Mike Dempsey for his resolute support during times of difficulty at the final stages of the project. Without his support and the support of others like Martyn Amos and Keith Miller, completion of this project would have been extremely difficult.

Finally, I would like to thank the school of computing, mathematics and digital technology, and the School of Healthcare Science at MMU. I would also like to thank the Dalton Research Institute for funding this research.

5 ABSTRACT

Skeletal muscles control the joints of the skeletal system and they allow human movement and interaction with the environment. They are vital for stability in balance, walking and running, and many other skilled motor tasks. To understand how muscles operate in general and specific situations there are a variety of tools at the disposal of research scientists and clinicians for analysing muscle function. Strain gauges for example allow the quantification of forces exerted during joint rotation. However, skeletal muscles are multilayer systems and often different muscles are responsible for the overall force generated during joint rotation. Therefore, strain gauges do not reveal the extent of the contribution of individual muscles during muscle function. The most widely-used and accepted muscle analysis tool is electromyography (EMG), which can measure the activation level of individual muscles by measuring the electrical potential propagating through muscle resulting from local activations of motor units. However, EMG does not linearly relate to any real physical forces, meaning that without prior knowledge of the force exertion on the level of the muscle, force cannot be estimated. EMG can measure superficial layers of muscle non-invasively by attaching surface electrodes (surface EMG) to the skin over the belly of the muscle. To measure the activity of individual muscle beneath the superficial muscle, a needle or thin-wire electrode must be inserted through the skin and into the muscle volume (intramuscular EMG), which is invasive and not practical in many situations. Furthermore, intramuscular EMG can only provide measurement of a very small volume ($< 1mm^3$) which can have varying amounts of active motor units.

Ultrasonography is a powerful cost-effective non-invasive imaging technology which allows real-time observation of cross-sections of multiple layers of dynamic skeletal muscle. Recent advances in automated skeletal muscle ultrasound analysis techniques, and advances in image processing techniques make ultrasound a valuable line of investigation for analysis of dynamic skeletal muscle. This aim of this thesis is to study and develop advanced image analysis techniques applicable to the analysis of dynamic skeletal muscle. The broader aim is to understand the capacity/limits of ultrasound as a skeletal muscle analysis tool. The ideas presented within offer new approaches to modelling complex muscle architecture and function via ultrasound. Tools have also been developed here that will contribute to, and promote ultrasound skeletal muscle analysis as a new and emerging technology which may be used by clinicians and research scientists to develop our understanding of skeletal muscle function. The main findings of this thesis are that automated segmentation of architecturally simple and complex skeletal muscle groups is possible and accurate, and that information about joint angles and muscle activity/force can be automatically extracted directly from ultrasound images without the explicit knowledge of how to extract it. The techniques used offer new

possibilities for non-invasive information extraction from complex muscle groups such as the muscles in the human posterior neck.

6 PREFACE – OVERVIEW OF THESIS

My investigation is focused on the development of methodological approaches for exploiting the information content of ultrasound observations of skeletal muscle. The main starting point for this investigation comes from the work of Darby, and others [1]. In that study, the authors present an approach to fully automatic skeletal muscle analysis. Their analysis technique implements local regional motion tracking in the human gastrocnemius and soleus muscles in the calf. Their work showed that automatic segmentation of calf muscles can be achieved, and that regional intramuscular motion is related to external measurements of skeletal joint angles, and contractile force generated within the muscle. Their results revealed that the proposed segmentation technique was not fully developed and thus failed in certain situations, and the proposed motion tracking algorithm accumulated error over time (drift). The drift problem potentially renders the analysis useless for arbitrary long sequences/periods. This thesis is therefore generally about developing the segmentation further, and developing an alternative approach for extracting meaningful information from within segmented muscles.

My work began with the analysis of long sequences using the same method as Darby, and others [1]. The purpose of this initial investigation was to familiarise myself with the main issues and to assess whether tracking drift could be reduced by optimisation of the many parameters of the tracking algorithm. Firstly, I noticed that both the segmentation and tracking algorithms were impractically slow for a multivariate parametric optimisation of large volumes of data (20+ videos containing ≈ 4000 frames). Secondly, the sequences I was using were obtained from another study and the images were not optimal due to poor placement of the ultrasound probe. These reasons prompted me to analyse only a single sequence with manual region segmentation, and to reduce my investigation to only 2 tracking parameters, namely the height and the width of the tracking features. Results showed that drift can be reduced slightly for certain parameter choices, namely larger feature sizes. This work was repeated later with more suitable data, and the results revealed an optimal feature size (an upper limit). The results and interpretation of this work is presented in chapter 3.

The results in chapter 3 supported the development of an approach that was not based on tracking or propagation of information between frames. By this point in the study it was understood from the work of Yeung, and others [2, 3], that – because ultrasound is a 2D planar view of 3D dynamic structure – 2D feature tracking is an ill-posed problem which cannot be resolved without knowledge of the 3D movement (trans-planar motion) of features. Although the analysis presented in chapter 3 was not thorough or conclusive on the main issue, the results agreed with the general consensus that feature tracking in 2D ultrasound is ill-posed. The main conclusions of this study were that a new approach was needed which was not tracking-based, and that the segmentation needed further

development, particularly if it was to be applied to segmentation of more complex muscle groups with more individual segments and more components of variation.

In chapter 4 a generic segmentation algorithm was developed based on the work of Darby, and others [1]. Their approach makes use of muscle shape statistics derived from many annotations of ultrasound images of human calf muscle. That approach is powerful because it can be used to regulate a contour finding process by allowing some degree of missing boundary gradient information in the ultrasound, which is a common problem. The main problem with their technique was in the contour finding algorithm. Their algorithm did not use a proper search routine; instead they initialised a contour from their database of annotations (based on a boundary gradient cost), and then they use iterative gradient descent to optimise the fitting of the boundary. Gradient descent is problematic because, it converges on local minima, and can be slow depending on the number of required iterations. The other problem with their technique was that they fail to exploit the fact the boundaries of skeletal muscles in these images have a predictable uniform appearance right across the image, and they only sample gradients at discrete points along the boundaries.

The main purpose of chapter 4 was to optimise the speed of the fitting procedure. The secondary purpose was to make the segmentation more general and more robust. To address both problems, a stochastic generative search algorithm was proposed, which uses shape statistics to generate shapes, and a simple cost function that measures the gradient across the entire boundary to measure the fitness of a generated shape. The main advantage of this approach is that randomly generated shapes can be evaluated in parallel of one another, and later the best shape according to the fitness function can be taken. This potentially introduces a large speed improvement which scales with the number of parallel processing cores at the user's disposal. This approach also eliminates the possibility of getting stuck in local minima due to the independence of each shape evaluation. The resulting tool running in Matlab (therefore unoptimised) on a single core actually yielded a substantial speed increase on the previous technique, and a small improvement in accuracy. The speed of the techniques is an arbitrary issue without measuring the new approach on a parallel system and was therefore not quantitatively compared. However, the results of chapter 4 allowed quick and accurate segmentation of the human calf muscles which could then be used to process vast amounts of data within a practical time-frame.

At this point, an experiment was designed with the intention of discriminating between the motion of muscle fascicles, as computed by the feature tracking algorithm and the segmentation algorithm. The experiment involved participants standing on foot pedals and – while keeping their feet flat on the pedals – they would produce diminishing levels of contractile force by pushing their feet downwards on the pedals. The pedals were programmed to respond to the application of force by rotating. They were also programmed to respond to the release of force by returning to a neutral angle. The angle of

the pedal was recorded as a time series, and that data was used in subsequent trials to rotate the pedals without muscle activation. Ultrasound, gastrocnemius EMG, pedal force, and pedal angle were all recorded in synchronisation. The result was a dataset which contained examples of active and passive shortening of muscle over the same joint angle range. After tracking all of the muscle motions, they were normalised to remove the bias of velocity from the motion field, and the motion between pairs of frames was modelled and successfully delineated by an algorithm called Support Vector Machines (SVM). This work was subsequently published and can be found in Abstract 8.9, section 8.9.1.

The SVM work was found to be methodologically flawed for a number of reasons, and thus prompted a rethinking of both how the experiment should be executed, and how the decision space should be delineated thereafter. The main issue with the experiment arises from the fact that muscle motion is not linear; muscle motion resulting from contraction is different from that of passive movement, which essentially means that contraction movement looks different at different joint configurations, and likewise passive movement looks different at different levels of contraction. Another problem arises from the inherent property of an SVM, which is that an SVM can only make binary decisions about the classification which a certain data vector (motion field) belongs to. Therefore, by using SVMs I am accepting the fact that I will not be able to delineate contraction/force or joint rotation when the motion field contains both factors. Use of SVM also removes the possibility of predicting the magnitude of joint rotations and contractions in combination and in isolation[1].

In chapter 5 a new approach was proposed for extracting information from segmented ultrasound images of the human calf. The approach was based on a paradigm called 'feature learning', which is a powerful way of modelling intrinsic properties of data without knowledge of what information is contained within the data. The main question of this chapter was 'does the information content of ultrasound allow discrimination of muscle states caused by combinations of passive joint rotation and active muscle contraction?' To answer this question, first, an experiment was designed that would produce a dataset containing ankle joint rotations and active contractions, both in isolation, and in combination. In order to ensure a well-distributed range of joint rotations and contractions, two signals were designed. The first signal was used to actuate the motors on a foot pedal system which created rotations in the ankle joint. The second signal was displayed on a screen with feedback of measured muscle contraction, and participants were asked to match their contraction signal with the other signal on the screen. Both signals were designed such that they contained a range of frequencies and amplitudes, and that they were minimally correlated. The result of this experiment was a large dataset

[1] A multi-class SVM approach could be used however, this is not recommended for regression-type models; SVMs are generally very good at classification of 2 or more classes, and are very rarely used for continuous regression.

of long sequences containing ultrasound images of muscle which were created from known quantities of ankle rotations and active contractions.

The dataset described in chapter 5 was then segmented using the technique developed in chapter 4. Then, a feature learning technique called Restricted Boltzmann Machine (RBM) was used to model the appearance/states of the intramuscular regions of gastrocnemius and soleus. The resulting model reduced individual images of muscle to an 800 dimensional feature vector. That feature vector was compared over a sequence with the externally measured quantities of ankle joint angle and muscle activity. That comparison revealed that individual features were strongly correlated with isolated externally measured quantities, even when those quantities were uncorrelated throughout the sequence. Because the RBM was used to model static images, the only way that result can be correct is if internal muscle states caused by different quantities of external factors are unique. A linear optimisation showed that the feature vector can be weighted to delineate and predict the external quantities for every participant in the dataset. Initial results, not detailed in this thesis indicate that generalisation of the current feature set between participants is only partially successful. This implies there are further issues to be resolved and questions to be addressed before we know whether or not the limitations lie within the information content of muscle ultrasound, and the features intrinsically available or within the choices made within the processes of experimental design, normalisation, model building and analysis.

After the developments in chapters 4 and 5 I was asked to analyse of a number of ultrasound sequences of the human posterior neck muscles (10 muscles visible within a single image). These images are extremely complex and manual annotation of them was unrealistic and in some cases impossible. Since the segmentation method proposed in chapter 4 requires annotation of muscle segments to build statistical models of shape a different approach was required. Chapter 6 details the development of an approach to modelling muscle as observed via ultrasound, using MRI. Ultrasound and MRI images of the posterior neck were obtained from a group of participants. Image-plane markers (visible in the MRI) were used to identify the image-plane between the MRI and ultrasound machines. After annotating the MRI images, the annotations were scaled from MRI dimensions to ultrasound dimensions, whereupon they were manually registered to the visible boundaries within the ultrasound. A method was then developed which allowed simulation of muscle deformation upon probe placement by manually adjusting a pressure parameter until the misaligned muscle boundaries were aligned. This process resulted in anatomically correct annotations of a set of ultrasound images, and the data was then used to create a statistical model of posterior neck muscle shape.

The shape model derived from MRI annotations was used with the algorithm proposed in chapter 4 to automatically segment ultrasound images of posterior neck muscles. The results presented in chapter 6

showed that the technique was accurate to $3mm$ or $\approx 60\%$ in the majority of cases. Results also showed that due to the higher degrees of variation of this model compared to the calf muscle model, segmentation was slow when implemented on a single core. A suggestion was made that the technique should be implemented on a parallel architecture. This was not implemented due to time constraints on the project. Once the speed of this algorithm is optimised, it will then become possible to use the same approach detailed in chapter 5 for the analysis of all 10 muscles in the human posterior neck.

In the final chapter, chapter 7, each chapter is reviewed and the main conclusions are identified. Then suggestions for further work are presented before final overall conclusions are drawn.

Chapter 1 contains general introductions to skeletal muscle, electromyography (for measuring muscle contraction), ultrasonography, and ultrasound imaging of skeletal muscle. Chapter 1 also lists the specific aims and scope of the thesis. The following chapter, chapter 2 contains a broad review of the literature on ultrasound image segmentation and ultrasound analysis of skeletal muscle, followed by a review of the development of feature learning. The final part of chapter 2 is a critical review of the relevant studies leading to the development of the ideas within my thesis. A glossary of terms and a list of acronyms are presented immediately after this preface.

7 GLOSSARY OF TERMS

Active (movement) refers to skeletal muscle movement that has been initiated by the activation of motor unit(s), which results in contraction of the muscle.

Active (muscle) refers to skeletal muscle that has (above noise level) detectable electrical activity within.

Aponeurosis is a sheet-like tissue that binds skeletal muscle to other muscles to bone. ***Aponeuroses*** *is the pleural*.

Attenuation generally refers to the reduction in strength of a signal. Pertaining to ultrasound, this refers to the reduction in strength of signals resulting passing through muscle and other material.

Contraction is shortening of muscle caused by activation.

Contrast pertaining to ultrasound/MRI is the difference in pixel intensities between anatomically different regions. Low contrast makes it difficult to differentiate regions.

Cross-talk with respect to EMG is the unwanted detection of electrical activity from more than one muscle through a single EMG channel.

Deep muscle refers to any muscle not directly beneath the skin layer.

Dorsi-flexion is the action of decreasing the angle measured between shank and foot.

Electromyography is a device which is used to measure the electrical activity of muscle during contraction.

EMG Intramuscular is the use of fine-wire or needle electrodes to measure the electrical activity of muscle; typically used to measure the activity of deep muscles.

EMG Surface is the use of electrodes placed on the surface of the skin to measure the electrical activity of the muscle directly beneath the skin.

Fascicle is a bundle of muscle fibres. Fascicles bundle together to form the active components of muscle. Fascicles shorten in response to electrical activation, which results in muscle contraction.

Feature (GBRBM) is the collective term for a variable in an RBM/GBRBM model, the output of which is determined by a linear combination of an input vector and a set of coefficients/weights associated with the feature.

Feature (GBRBM) response the output of a feature determined by a linear combination of an input vector and a set of coefficients/weights associated with the feature.

Feature (KLT) a small patch of image, typically used as a template to track its movement in temporally adjacent image/frames.

Feature (KLT) tracking is the term for tracking the motion of a small patch of image (feature) between temporally adjacent images/frames.

Hypo-echoic (ultrasound images) are dark homogenous regions of an image resulting from objects which reflect very few sound waves such as water.

Innervation refers to the stimulation of nerves or body parts.

in vivo in the living organism.

Joint angle refers to the angle between connecting bones relative to their joint.

Joint rotation is a change in the angle between two connecting bones relative to their joint.

Lateral outward from the centre of the body.

Medial inward towards the centre of the body.

Passive (movement) refers to skeletal muscle movement that has not been initiated by the activation of motor unit(s).

Passive (muscle) refers to skeletal muscle that has no (or minimum) detectable electrical activity within.

Plantar-flexion is the action of increasing the angle measured between shank and foot.

Regularisation in general refers to restricting the influence of any individual free-parameter of a multivariate model; typically used to prevent the model from over-fitting for the purposes of improving generalisation.

Segmentation (usually refers to automatic) delineation of regions of interest within an image. E.g. delineating muscle compartments in an ultrasound image.

Signal dropout generally refers to the momentary loss of a signal. Pertaining to ultrasound, this refers to the loss of signals usually caused by passing through muscle and other material.

Signal to noise ratio refers to signal that is distinguishable from noise. A high ratio signal has a more distinguishable signal above the noise level, from a low ratio signal.

Skeletal Muscle is muscle that is responsible

for skeletal function.
Speckle refers to additive normally distributed noise to pixels in an image.
Superficial muscle refers to muscle that is directly beneath the skin layer.
Trans-planar motion (ultrasound) refers to the 3D motion of material that is not detectable in the 2D ultrasound plane.
Triceps surae refers to a group of muscle in the human calf: gastrocnemius, plantaris and soleus.

8 LIST OF ACRONYMS

AAM	Active Appearance Model
ANN	Artificial Neural Network
API	Application Programming Interface
ASM	Active Shape Model
BBN	Bayesian Belief Network
B-mode	Brightness Mode
CD	1. Contrastive Divergence 2. Centroid Distance (only in chapter 6)
CNN	Convolutional Neural Network
CRF	Conditional Random Field
EMG	Electromyography
sEMG	Surface Electromyography
iEMG	Intramuscular Electromyography
FIR	Finite Impulse Response
GA	Genetic Algorithm
GBRBM	Gaussian-Bernoulli Restricted Boltzmann Machine
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
GRF	Gaussian Random Field
GUI	Graphical User Interface
HOG	Histogram of Orientated Gradients
KLT	Kanade Lucas-Tomasi feature tracking
MCMC	Markov Chain Monte-Carlo
MG	Medial Gastrocnemius
MRF	Markov Random Field
MRI	Magnetic Resonance Imaging
fMRI	Functional Magnetic Resonance Imaging
MVC	Maximum Voluntary Contraction
MSE	Mean Square Error
PCA	Principal Components Analysis
RBF	Radial Basis Function
RBM	Restricted Boltzmann Machine
RMSE	Root Mean Square Error
RUSI	Rehabilitative Ultrasound
SBN	Sigmoid Belief Network
SIFT	Scale Invariant Feature Transform
SO	Soleus
SVM	Support Vector Machine

1 INTRODUCTION

This chapter is an overview of the scope/aims of this thesis and a general introduction to the core concepts and essential knowledge required to follow the content of the ensuing chapters. In principal this thesis is an investigation of the capacity of ultrasonography as a medium for *in vivo* analysis of dynamic skeletal muscle, and necessarily requires the investigation and development of (automated) computational techniques which would improve and expand the scope and consistency of such analyses. The specific aims are drawn up at the comclusion of this chapter. To communicate this investigation concisely general introductions are given on human skeletal muscle anatomy and function, the standard method of skeletal muscle activity measurement (electromyography), and b-mode ultrasonography in relation to skeletal muscle analysis. After the general introduction the presentation of background material should provide a basis for understanding the research questions addressed in the literature to date, which are reviewed in the next chapter. Other techniques for analysis of skeletal muscle are considered beyond the scope of this investigation and are therefore excluded. The final section of this chapter presents a description of the thesis and an explicit list of objectives and aims.

1.1 HUMAN SKELETAL MUSCLE ANATOMY AND PHYSIOLOGY

Skeletal muscles number greater than 600 individual muscles in the human body, and they allow balance, mobility and movement. Muscles are made up of thousands of cells called ‘fibres’, which bundle into ‘fascicles’ (see Figure 1). When a fibre receives a nerve impulse from the brain it shortens, which in turn causes shortening of fascicles and contraction in the muscle. Nerve impulses are transmitted to muscle fibres via motor neurons, which can connect to more than one fibre. The brain can recruit more fibres depending on the demand (e.g. lifting heavy objects). These nerve impulses can be measured directly via a technology called electromyography (explained in section 1.2). Fascicles and other intramuscular structures can be viewed as they shorten, via real-time ($25Hz$) imaging technology such as ultrasonography (explained in section 1.3).

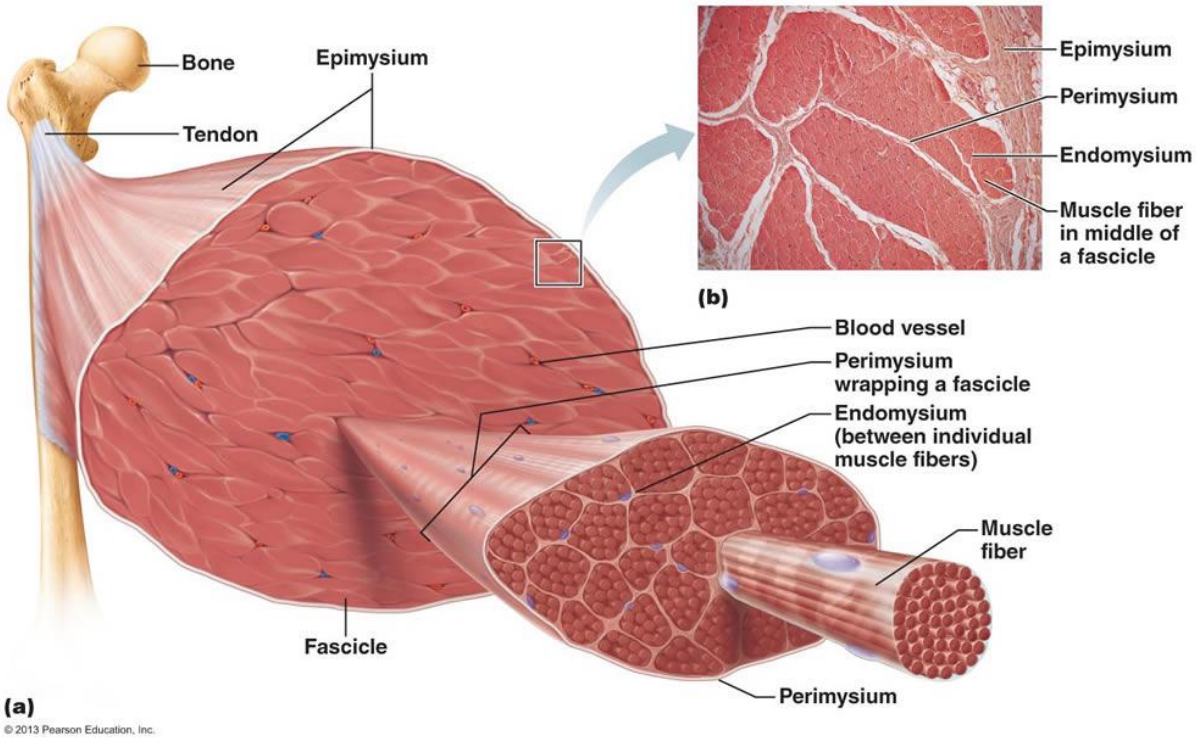


Figure 1. Skeletal muscle architecture (image obtained from [4]). This figure shows a skeletal muscle ensheathed by Epimysium. The muscle attaches to bones via its two tendon ends. Muscles are composed of fascicles ensheathed by Perimysium. Fascicles are bundles of fibres ensheathed by Endomysium. Muscle fibres shorten when they receive a nerve impulse from the brain, which causes contraction.

Muscles attach to bones via tendons, which are a continuation of the thin sheet-like membrane that encases the muscle, called aponeurosis (although not labelled, the epimysium in Figure 1 forms a thin sheet-like aponeurosis, which forms the tendon). Skeletal muscles attach to one or more bones about a joint and when they contract they cause the bone to move about the joint. If a muscle contracts and causes connecting bones to come together (flexion) they are termed protagonist muscles. If a muscle contracts and causes bones to move apart (extension) they are termed antagonist muscles. Importantly, fibres can only shorten when they receive nerve impulses, that is to say that an antagonist muscle cannot perform the job of the protagonist muscle with respect to the connecting bone, and *vice-versa*. Skeletal muscles are also layered and each muscle performs a specific function during movement; although many muscles appear to exhibit redundant functionality (they have approximately the same function as some other muscle that attaches to the same joint). As the content of this thesis is focused on human calf muscles and human posterior neck muscles, the following two sections will introduce their anatomy and physiology.

1.1.1 Triceps Surae Anatomy and Physiology

The human triceps surae is made up of 3 separate muscles in 2-3 layers (see Figure 2) – in order of superficial to deep – gastrocnemius, plantaris and soleus. Approximately 25% of humans do not have

a plantaris muscle, and it is relatively very small, therefore it is not the subject of any investigation in this thesis. The soleus muscle is often thought of as the second layer of muscle due to the insignificance and lack of presence of the plantaris muscle. Collectively, these muscles are known as the plantar-flexors because they produce the plantar-flexion action in the ankle joint. Plantar-flexion is the term for increasing the joint angle between the shank and the foot. The opposite action is call dorsi-flexion, which decreases the joint angle between shank and foot.

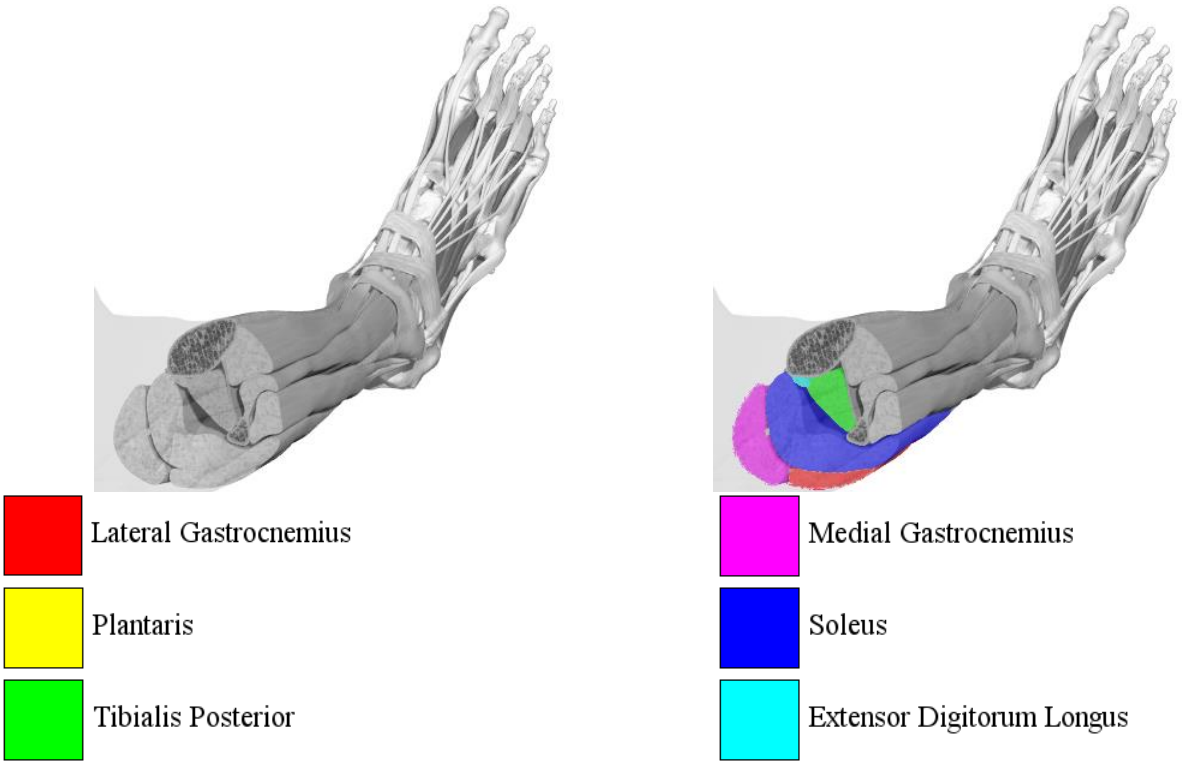


Figure 2. Calf muscle anatomy. A transverse cross-section of the human calf (left), highlighting the gastrocnemius (medial and lateral) muscles which forms the superficial muscle layer, then the very small plantaris which lies between the medial gastrocnemius head and the soleus muscle. The soleus muscle forms the second layer of muscle. Other plantar flexor muscles such as the extensor digitorum longus and the tibialis posterior can be seen, although those muscles are not part of the triceps surae which forms the bulk of the calf and is the subject of investigation in this thesis.

The calf muscles are recruited for balance, walking, and jumping to name just a few functions. Figure 2 is a graphical representation of a transverse cross-section of the human calf. Functionally they exhibit a large degree of redundancy due to the fact that all 3 muscles combine to form the Achilles tendon, which has the purpose of flexing the ankle joint. However, the gastrocnemius muscles originate (attach at the opposite end of the Achilles tendon) on the femur bone and – as a consequence – crosses both the knee joint and the ankle joint. Whereas the soleus and plantaris both originate on the tibia and – as a consequence – only cross the ankle joint. Therefore, these muscles have a degree of functional independence and play distinct roles in sensory feedback (proprioception).

1.1.2 Posterior Neck Anatomy and Physiology

The muscles in the posterior human neck have a complex multilayer architecture, and are made up of over 12 muscles, in over 5 symmetrical depth-wise muscle layers (see Figure 3). Muscles in the neck can cross 2 joints and attach to multiple different bones [5]. The size and shape of muscles across a population exhibit wide variability, and cross-sectional areas do not scale proportionally with body height and/or weight [5], furthermore there are significant differences in muscle shape between genders [6]. The human neck muscles are also functionally complex [7], while also exhibiting functional redundancy across muscles. These muscles allow controlled head rotation about all 3 axes (6 degrees of freedom), while also allowing lateral and frontal head extension/retraction.

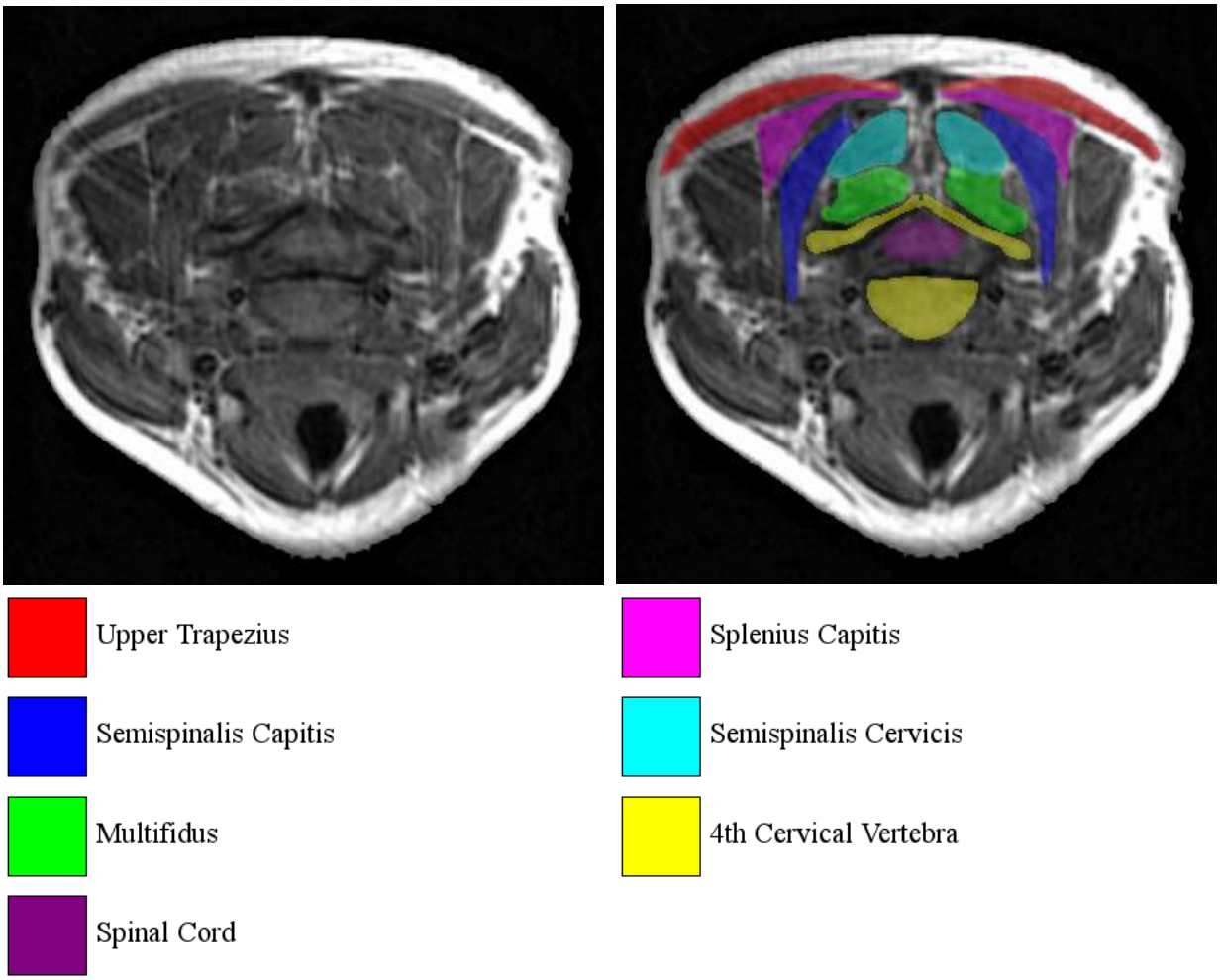


Figure 3. Posterior neck muscle anatomy. A transverse MRI cross-section of the human neck is shown on the left, while a colour-coded version of the same image is shown on the right. There are many identifiable muscles representing 5 symmetric layers of muscle (triceps, splenius, semispinalis capitis, semispinalis cervicis, and multifidus). Other muscles make up the rest of the posterior neck however, these muscle are not identifiable in these images due to their small size and/or vertical location in the neck.

1.2 ELECTROMYOGRAPHY

As previously stated in section 1.1, within skeletal muscle, bundles of fibres called fascicles contract in response to neural innervation from the brain. This causes skeletal muscle to contract with the ultimate aim of changing or sustaining the angles between joints in the skeletal system. Neural innervation causes electrical activation of motor units (motor neurons) which results in a low-voltage (millivolts) residual electrical current propagating through the muscle and that voltage can be measured and recorded by electromyography (EMG). Generally, with increasing demand (larger contractions) more motor units activate and an increased electrical current is registered. EMG signals can be processed and used to estimate the level of muscle activity and/or the amount of internal force generation during contraction. Importantly, and perhaps intuitively EMG does not register passive changes in muscle length/architecture. There are two main types of EMG which have different strengths, weaknesses, making them suitable for different applications. The next two sections give further details on the application and advantages/disadvantages of both types.

1.2.1 SURFACE ELECTROMYOGRAPHY

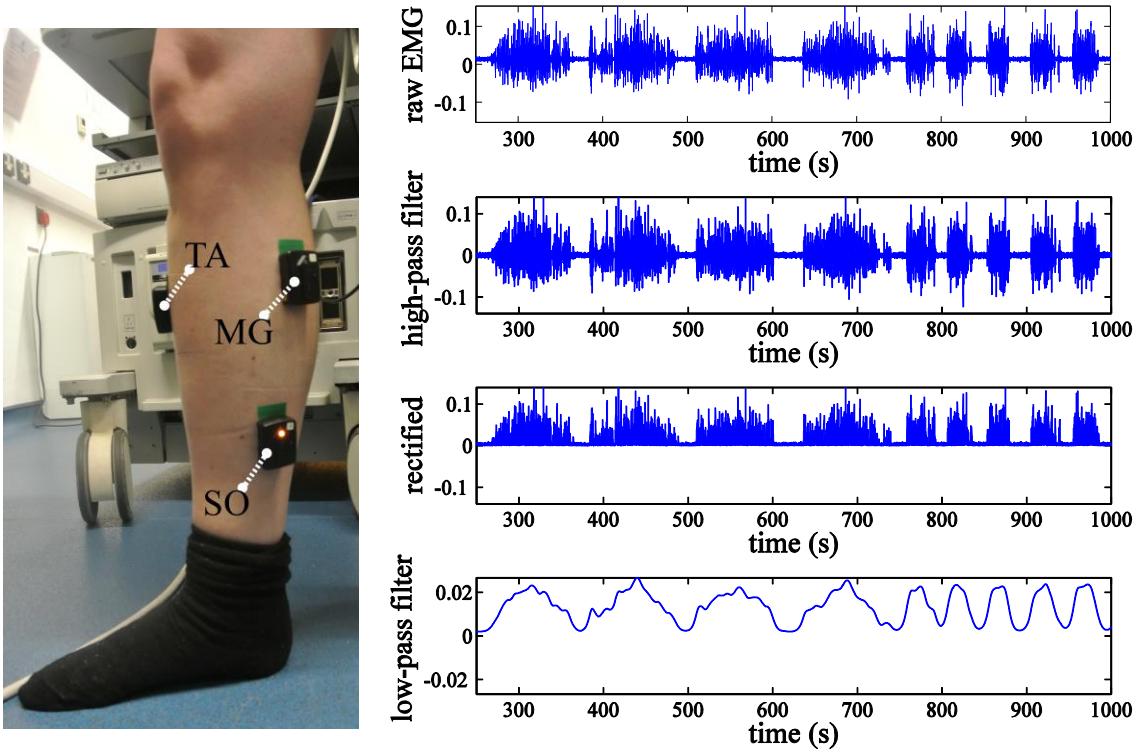


Figure 4. A typical EMG filtering process. Left: 3 wireless surface EMG electrodes can be seen taped to tibialis anterior (TA), medial gastrocnemius (MG), and soleus (SO) on the right leg of a human participant. These electrodes record a raw voltage signal (right, top) during muscle contraction. Right: from top to bottom: The raw EMG signal is recorded, where the voltage indicates muscle activation. Then a high-pass filter is used to remove low frequency noise (has the effect of removing positive signal bias). Then the filtered signal is rectified (taking the absolute value). Finally, a low-pass filter is used to extract smooth activity peaks often referred to as the signal envelope. This is just one of many methods, and it is parametric so it can have varying results. Usually some consideration is required before filtering takes place.

Surface EMG (sEMG) is a non-invasive technology used to measure the electrical activity of skeletal muscle subcutaneously (through the skin). To perform sEMG measurements, electrodes are fixed to the skin over the muscle of interest (typically over the belly of the muscle) and the electrical activity is detected through the skin (see Figure 4). sEMG is relatively cheap, non-invasive, easy to deploy and can give valuable information about the contractile properties of muscle during movement, particularly when combined with body kinematics. Sometimes multiple different sensors (or an array of sensors) can provide an activation map, which can help differentiate the discrete functional properties of muscle. However, sEMG has the following limitations (in no particular order):

1. ***Low dimensionality***

   Muscles are physiologically complex and information rich therefore, reducing the information content of muscle down to one or two signals representing local motor unit activity is extremely limiting. Surface electrodes are also sensitive to different parts of muscle and therefore if the whole muscle is not measured, then regional information is lost.

2. ***Subjectivity & limited information***

   sEMG recordings can be sensitive to physiological mechanisms within muscle, such as fatigue. Fatigue in a muscle influences sEMG measurements but are not necessarily detectable. Furthermore, sEMG measurements do not necessarily explain force production, strength or effort within a muscle, and certainly do not convey information about muscle length or joint angles. The amplitudes of sEMG peaks in different muscles can be equivalent but the force production, strength or effort of the two muscles can be different within a person. Likewise, sEMG peaks over a group of people can be similar but almost certainly will convey entirely different meanings for each person.

3. ***Low signal-to-noise ratio***

   sEMG is characteristically sensitive to nearby low frequency electromagnetic interference from motors, electrical mains. sEMG is also susceptible to high frequency noise as the signal propagates through different materials before being recorded. Therefore the sEMG signal is usually processed with one or more filters before it is interpreted, although the raw sEMG can still be used. Filtering the sEMG signal inherently results in a signal that is sensitive to the parameters of the filter(s), which are usually tuned to exploit specific types of information present in the raw signal. See Figure 4 for an illustration of a typical EMG signal filtering process.

4. ***Cross-talk***

   Cross-talk is where the electrical activity from nearby muscles (usually adjacent) propagates over into the recording field of another muscle. Cross-talk is problematic because it inherently means that single muscles (and particularly specific regions within a muscle) cannot be measured in isolation.

5. ***Only measures superficial muscle***

   Perhaps the main drawback of sEMG is that it is only capable of measurement and interpretation of superficial muscle activity. In terms of the human calf, this intuitively means that electrodes placed over the belly of the gastrocnemius muscle cannot measure activation in the soleus or plantaris muscle, although the soleus is accessible through the surface lower down the leg where it contacts the skin. However, considering the deep muscles in the neck (see section 1.1.2), there is no way to measure activity in any muscle other than the upper trapezius in the posterior neck, which does not even contribute significantly towards head function. To access the deeper muscle, intramuscular electrodes are required (see next section).

1.2.2 INTRAMUSCULAR ELECTROMYOGRAPHY

Intramuscular EMG (iEMG) is an invasive technology used to measure the electrical activity of skeletal muscle by directly inserting electrodes through the skin into the muscle volume. This is usually done with a needle, or fine-wire electrode. Whereas sEMG can only measure the activity of superficial muscle layers, iEMG can measure the activity of both superficial and deep muscle layers. The main disadvantage of iEMG is that it is an invasive technology often inducing some level of discomfort and restriction of movement. iEMG is also susceptible to noise, with the further disadvantage that the measurement volume can be very small ($\leq 1mm^3$ compared sEMG which typically has a measurement volume of $\leq 2cm^3$). In the previous section it was noted that sEMG was capable of measuring the activity of the main muscles of interest in the triceps surae, but not capable of measuring the activity of the main muscles of interest in the posterior neck. iEMG can measure deep muscles in the posterior neck and it has successfully been used previously [8] however, in that case the authors suggest that iEMG can only realistically be used to simultaneously measure activity in 2 deep muscles. Furthermore, their work highlights the difficulty in locating deep muscles with the electrode, stating that there can be no certainty about the location of the electrode in the muscle.

1.2.3 SUMMARY

In summary, neither sEMG nor iEMG can provide the full picture of muscle activity. Whether there are regional differences, or an inability to acquire measurements from all muscles under observation

simultaneously, EMG can only provide partial descriptive information. The next section will introduce a completely non-invasive technology – b-mode ultrasound – that is capable of observing the architectural and physiological properties of many layers of dynamic skeletal muscle simultaneously, in both passive and active situations.

1.3 B-Mode Ultrasonography

Ultrasound is a powerful imaging technology, which is widely used in medicine and research. It is cost-effective, time-effective, portable, and free of ionizing radiation. Ultrasound is typically used for imaging anatomical structures in the human body. The images produced by ultrasound are often referred to as tomographic, which translates as 'cross-sectional'. 'B-mode', or brightness mode, refers to the way that the images are represented; each pixel in an ultrasound image is as a grey-scale intensity value. Ultrasound is a valuable technology for analysis of *in vivo* dynamic skeletal muscle function due to its relatively high temporal resolution. Temporal resolution refers to the frequency that images are created at, which for standard ultrasound is usually between $20 - 100Hz$. In comparison, Magnetic Resonance Imaging (MRI) typically has a higher spatial resolution (and better image quality) and a much lower temporal resolution of $\approx 0.0033Hz$, and functional MRI (fMRI) typically has a lower spatial resolution and a lower temporal resolution of $\approx 0.5Hz$. Therefore, ultrasound is the preferred technology for analysis of dynamic skeletal muscle.

Unlike EMG, ultrasound offers a high-dimensional information stream that is currently under-utilised in research. Ultrasound also offers a view of dynamic passive internal motion of skeletal muscle features, whereas EMG can only offer a low-dimensional measure of muscle activity, and that measure does not relate to force production, or strength. The following section presents a basic explanation of how b-mode ultrasound images are formed. The section after presents a cross-sectional ultrasound view of human calf muscle, with an explanation of some of the information content within that view.

1.3.1 Image Formation

Ultrasound images are formed from very high frequency (above the normal audible range) sound pulses which are transmitted and received by arrays of transistors in the head of a probe. Ultrasound probes come in a variety of sizes and shapes, and operate at a variety of different frequencies ($0.5 - 30MHz$). The size and shape of an ultrasound probe is typically designed for imaging specific parts of the body. The head of the probe is where the transducers are aligned, and is therefore the part of the probe that contacts the skin when imaging the body. Within the probe head is a linear array of transducers, which convert electricity into sound and vice versa. The sound waves from the transducers travel through material and when they encounter boundaries between materials some of the sound waves are reflected back (backscatter) while some of them carry on through the boundary. Some of the sound waves that carry on through boundaries can be reflected back when they encounter

a further boundary, and this intrinsic property is applicable to all sound waves that pass through boundaries. The reflected sound waves arrive back at the transducers, which convert the vibrations back into electricity. Because the sound waves arrive at different times, the ultrasound machine can compute the delays into a depth map, and because the sound waves arrive back at positions along the linear array, the machine can further compute a full 2D spatial map of the sound echoes.

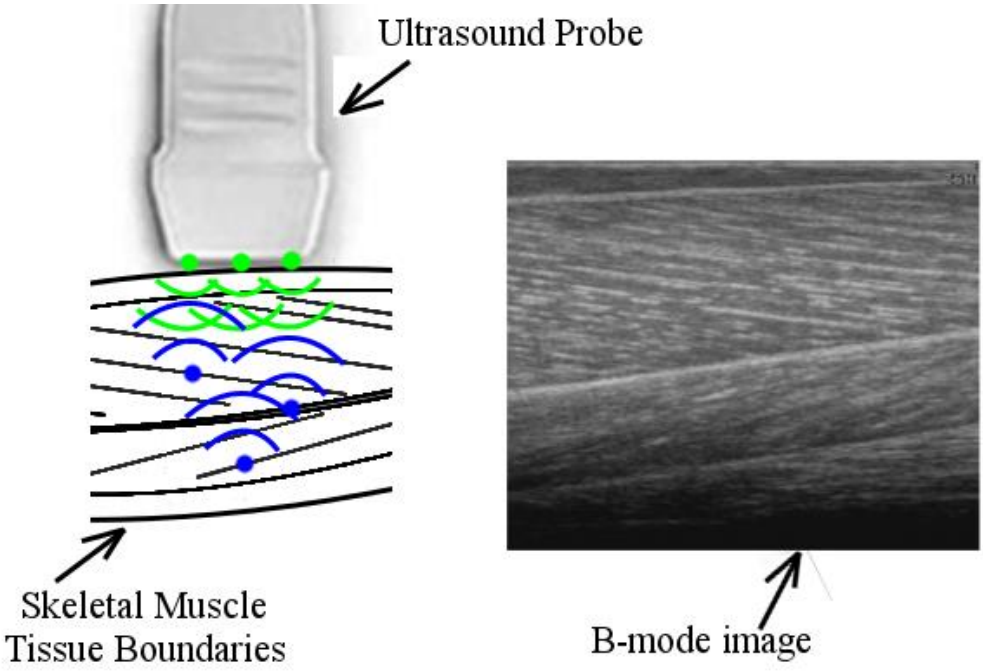


Figure 5. B-mode ultrasound image formation. The green dots represent transducers that are emitting waves, while the blue dots represent sound wave reflections off boundaries between tissues. The image on the right is formed from the backscatter where the brightness of any pixel represents the amplitude of the reflected sound waves at that location. The boundaries between materials in the b-mode image can be clearly differentiated from other materials in the image which appear much darker in comparison. Note that the b-mode image represents only a planar slice of the material being imaged.

The amplitudes of the reflected sound waves represent the level of acoustic impedance of the materials they have reflected back off. Materials with high acoustic impedance reflect many more sound waves than those with low acoustic impedance and therefore have higher amplitudes. In b-mode (brightness mode) ultrasonography, the amplitudes (or acoustic impedances) are represented as a grey-scale pixel intensity, where the brighter the pixel, the higher the acoustic impedance. This intrinsic property of ultrasound can be used to differentiate between materials observed in an ultrasound images, and can help identify the boundaries between anatomical structures (see Figure 5). A table of acoustic impedance values for anatomical structures can be found in Table 1 below.

Table 1. Acoustic properties of tissue (Source: [9]). Note that muscle, tendon, bone, soft tissue, connective tissue, and blood all have different impedance values and would therefore appear in an image at slightly different brightness's, which may make them distinguishable from one another

Material	Velocity (m/s)	Density (kg/m^3)	Attenuation ($dB/cm\ MHz$)	Acoustic Impedance (MRayl)
Air	330	1.2	-	0.0004
Blood	1584	1060	0.2	1.68
Bone, Cortical	3476	1975	6.9	7.38
Bone, Trabecular	1886	1055	9.94	1.45
Brain	1560	1040	0.6	1.62
Breast	1510	1020	0.75	1.54
Cardiac	1576	1060	0.52	1.67
Connective Tissue	1613	1120	1.57	1.81
Cornea	1586	1076	-	1.71
Dentin	3800	2900	80	8.0
Enamel	5700	2100	120	16.5
Fat	1478	950	0.48	1.40
Liver	1595	1060	0.5	1.69
Marrow	1435	-	0.5	-
Muscle	1547	1050	1.09	1.62
Tendon	1670	1100	4.7	1.84
Soft Tissue (Average)	1561	1043	0.54	1.63
Water	1480	1000	0.0022	1.48

1.3.2 *The Appearance of Skeletal Muscle via B-mode Ultrasonography*

The anatomy and physiology of skeletal muscle has been described briefly in section 1.1. This short section presents an observation via ultrasonography of some of the features of skeletal muscle that were referenced in that section; specifically fascicles and aponeuroses of the human triceps surae. This section also presents some of the types of measurement that can be obtained directly from these images without focusing on automatic extraction of those measurements at this point in the thesis.

As previously mentioned, skeletal muscle is composed of bundles of fascicles, which are bundles of fibres. Fascicles are visible via ultrasound, and measurements such as their curvature, orientation, or pennation angle can be extracted. The image below shows a typical acquisition of a single frame of ultrasound of the human triceps surae, specifically the medial gastrocnemius, soleus and tibialis posterior muscles.

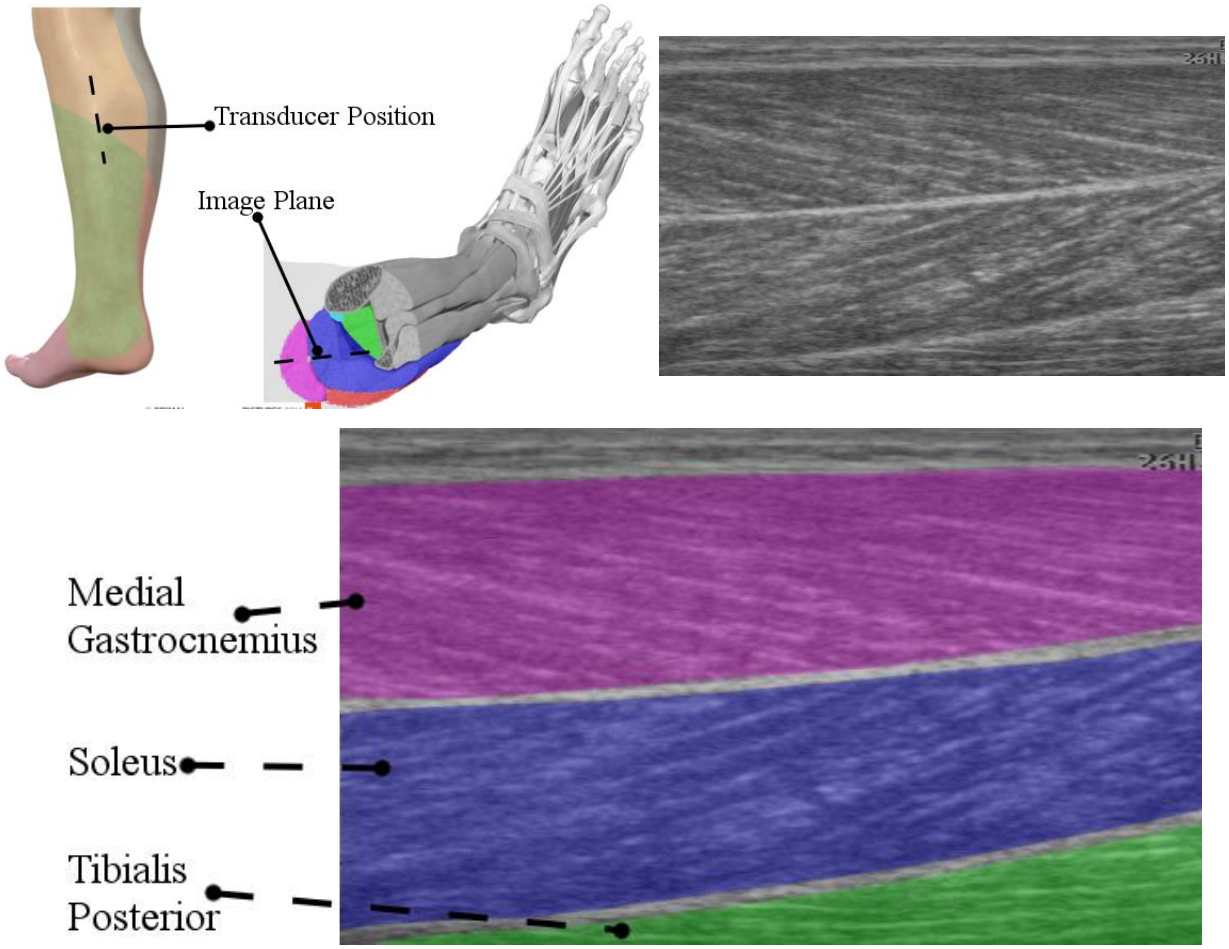


Figure 6. Top, left: illustrations of a human leg showing the position of the linear head of the probe over the medial gastrocnemius, and the direction of the image plane through the muscle layers. Top, right: a b-mode ultrasound image showing the 2D planar slice of the 3 muscles in the calf. Bottom: colour-coded muscles in the human calf, referring to the top left leg illustrations. The bright boundaries between muscles are structures known as the aponeuroses, which eventually form the tendon of each muscle. The bright diagonal striations within the muscle region are the boundaries of individual fascicles.

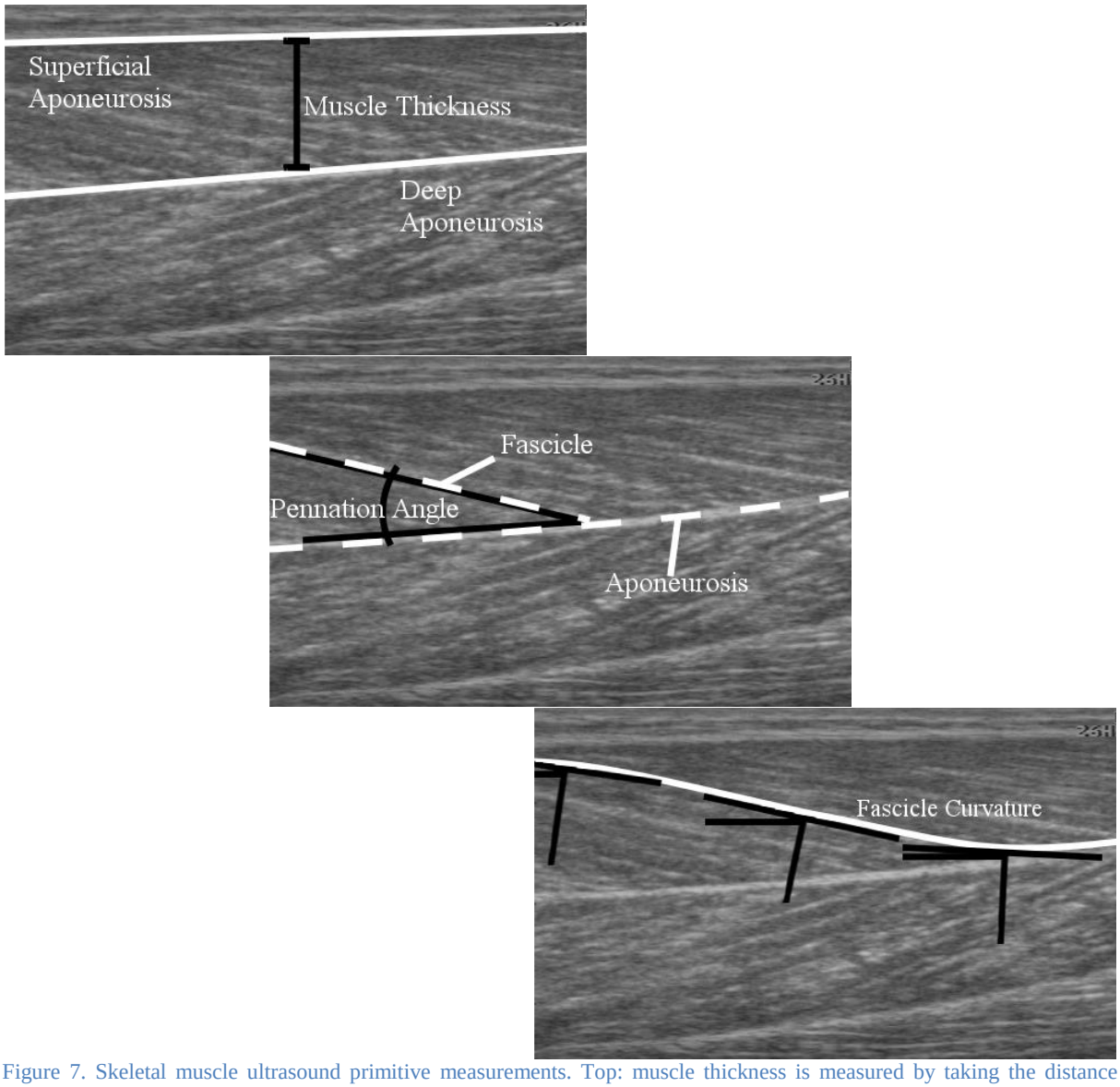


Figure 7. Skeletal muscle ultrasound primitive measurements. Top: muscle thickness is measured by taking the distance between superficial and deep aponeuroses. This measure is simple to calculate compared to taking the cross-sectional area, which is the area of the muscle region between superficial and deep aponeuroses. Middle: the pennation angle of fascicles as measured by computing the angle at which fascicles connect to the deep aponeurosis. Bottom: fascicle curvature is more difficult to measure as it involves computing the change in fascicle orientation between aponeuroses. All of these quantities change during contraction, and also during passive joint rotation.

Ultrasound allows real-time observation of the dynamic properties of skeletal muscle such as those illustrated in Figure 7. These properties change during active and passive muscle activity which makes ultrasound invaluable for understanding the mechanical properties of skeletal muscle. It shall therefore be the goal of this thesis to investigate the use of advanced image analysis techniques for automated analysis of skeletal muscle ultrasound image sequences, as well as to develop new techniques and approaches in order to exploit this information-rich medium. The following section will concretely lay out the specific aims of this thesis, expanding upon this point.

1.4 THESIS AIMS

The following objectives provide the terms of reference and define the scope of this thesis. The aims are as follows (in order of descending importance/priority).

1. The general aim of this thesis is to investigate the use of advanced image analysis techniques for automatic analysis of skeletal muscle via b-mode ultrasonography. Many existing segmentation and analysis methods have been developed, yet there is still no general solution or standard method to analyse ultrasonography sequences of skeletal muscle. The types of issues that will need to be addressed are tracking drift, automatic segmentation, algorithm speed optimisation, acquisition of truth data for validation and training of methods, and information extraction.
2. The more specific aim is to investigate the intrinsic spatial information content of the pennate muscle group in the triceps surae. Going beyond asking specific questions about measurable parameters in relation to muscle function, this requires an experimental design and novel analysis to determine the content and limits of information contained within the ultrasound images of functional muscle.
3. The third aim is to investigate the feasibility of ultrasonography for analysis of architecturally and functionally complex muscles. To date, analysis of skeletal muscle via ultrasonography has generally been limited to simple muscle architectures (architecturally and/or functionally non-complex muscle). Application of ultrasound has most relevance when applied to muscles of high importance for motor control and for which information is less easily available via electromyography. Hence specifically this aim focuses on the multilayer muscle group in the human posterior neck.
4. The final aim is to develop a set of advanced tools and image processing algorithms using MALTAB, Simulink, C++, and CUDA for use in answering the previous 3 objectives, and for use in future projects and investigations into ultrasonography analysis.

These aims set out clear objectives for the project, and provide the research questions that must be considered while reviewing the literature. A broad review of the background literature and a detailed review of the most relevant literature immediately follow this chapter.

2 LITERATURE REVIEW

This chapter is split into three main sections: a broad overview of ultrasound analysis, a review on feature learning from images, and a critical review of selected seminal works. The broad overview is divided into three main sections. Section 2.1.1 explores some of the attempts to segment ultrasound images with mathematical models of visible landmarks, as well as techniques for tissue classification in the absence of well-defined landmarks and shapes. Section 2.1.2 explores some of the literature on the use of ultrasound to analyse skeletal muscle, as well as some of the more contemporary computational methods of that have been developed for the analysis of skeletal muscle via ultrasonography. Finally, section 2.2 focuses on feature learning, exploring the background to some recent developments in machine learning in application to image understanding and analysis.

Before reviewing any published works, it is of note that a preliminary investigation of publication rates reveals an upward trend in the area of skeletal muscle ultrasound imaging and computational analysis of skeletal muscle ultrasound images. Figure 8 illustrates the growing interest in the use of ultrasonography for analysis of skeletal muscle, and a more recent proportional growth in computational techniques in application to ultrasound images. Note that the acceleration of interest in ultrasound as a skeletal muscle imaging technology increases with the onset of development in computational methods of analysis. With modern computational approaches and a wealth of background literature, the potential to exploit the hidden information within visible contractile tissue via *in vivo* ultrasonography has never been greater.

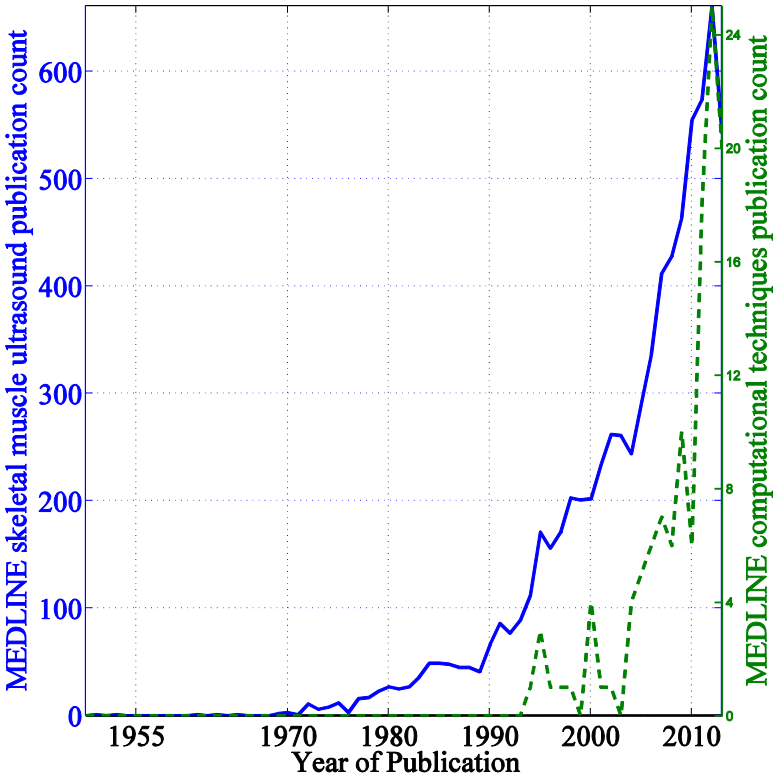


Figure 8. MEDLINE data (retrieved from [10]) on publication trends from 1950 to date (2014). Search terms for the solid blue line: “skeletal” + “muscle” + “ultrasound”. Search terms for the green dashed line: “skeletal” + “muscle” + “ultrasound” + “automatic” + “automated” + “segmentation”

2.1 Background: Techniques in Ultrasound Image Analysis

2.1.1 Ultrasound Image Segmentation

There are many issues surrounding the segmentation of ultrasound images: signal attenuation, speckle noise, inter-participant variability, sensitivity to view-plane orientation, signal dropout, lack of labelled data, electromagnetic background noise, and poor regional contrast [11]. For an expert there is typically some visible landmark that can be identified, and because the ultrasound probe always contacts the skin, imaging a depth-wise plane of tissue impedance, that landmark will present in a consistent orientation although often with some variability in location within the image. A key problem from a computational perspective is defining landmarks (obtaining ground truth). Without a sound definition of shape and/or texture it is difficult to measure the performance of any technique. A typical response is to create synthetic data and evaluate the performance of a proposed segmentation technique on that data, which in most cases does not truly reflect real-world performance. Good segmentation techniques typically use priors of shape and/or texture in order to segment well-structured regions such as organs and muscles in the human body. In most cases human experts annotate regions within the ultrasound images, which can be used to build models of shape and appearance. Because of the quality issues pertaining to ultrasound images, in some cases higher

contrast and/or higher spatial resolution imaging mediums (MRI, X-ray, etc…) can be used to create accurate models of shape as regions are often more readily identifiable for annotation by experts. Models of shape and texture can be used to regularise heuristic searches within the image, where segment boundaries are incomplete and/or poorly contrasted with the surrounding region.

One of the earliest approaches to segmentation of ultrasound images is one in which the authors attempt to extract features (patches of texture) directly from the image using a multilevel, iterative algorithm [12]. The authors first divide the image into four quarters, which form the top level of a quadtree structure. Each of the four top level nodes are examined for homogeneity of pixels, and where any node is inhomogeneous (high energy; based on some cost function) the node is split into 4 child nodes, where the process is then repeated. Each leaf in the quadtree represents a feature which is then used to segment the image into unique regions. The authors evaluate the performance on small amounts of both synthetic and real anatomical data. Both their synthetic and real images contain very well defined bright and dark regions. While they address the issue of image bias (creating images fit for their approach) by transforming the synthetic images, they do not attempt to show generalisation to a variety of other *in vivo* images. The paper also does not demonstrate the segmentation of any well-defined structures, and further lacks the solid validation of comparison with real image annotations; instead the authors evaluate real data segmentation qualitatively. Such an approach is missing an obvious application to with any relevance to other problems. Further, its lack of real-world validation, coupled with its complicated nature places this approach in poor standing for further investigation.

A similar approach makes use of Gaussian Random Fields (GRF) and K-means clustering to implement a probabilistic segmentation routine [13]. Their goal is to group large contiguous regions within general speckle-laden ultrasound images; validated on synthetic data. Their algorithm uses a Finite Impulse Response (FIR) filter on the original image, followed by multiple stages of K-means clustering to prime a Gaussian Random Field for segmentation. They evaluate their algorithm quantitatively on synthetic images, reporting success and an improvement on their previous method [12] – except for segments with low contrasting boundaries. They also qualitatively assess the performance on two different *in vivo* ultrasound images, reporting moderate success in each case. Without significant amounts of labelled data it is difficult to ascertain the viability of this technique on real-world data. The technique at best can group tissues and roughly contiguous regions, but cannot reliably pick out structure or shape from a real ultrasound image. Such a technique might have an application in anomaly detection (tumour detection, etc…), rather than specific region extraction however, the lack of external validation is worrying and creates ambiguity in its usefulness.

The first notable step towards detecting structured regions in ultrasound images can be found in a semi-automatic intravascular border detection algorithm [14]. In contrast to the two previous

approaches, the authors use a priori knowledge of coronary artery anatomy to guide a heuristic search. In order for their technique to work, a manual region of interest is defined by an operator prior to segmentation. The authors use a Prewitt compass with a quasi-Sobel edge detector to find oriented edges near the manually defined region of interest. Together with the knowledge of the expected double-echo texture pattern present in coronary artery ultrasound images, a heuristic search extracts the boundary of interest. Their approach is less convoluted than previous methods, and yet by including a priori knowledge in their algorithm they successfully validate their algorithm on over 35 real ultrasound images. This technique is already a positive step towards fully automated segmentation, demonstrating the power of prior knowledge for contour finding in real ultrasound images.

The development of semi-automated ultrasound segmentation has continued with boundary recognition paradigms such as the Active Contour [15]. One example evaluates first and second order grey-level pixel intensity lines at fixed intervals perpendicular to a basis spline [16]. This method deals with segmentation of 3D *in vivo* ultrasound images of the gall bladder. The authors state that after an initial operator defined region of interest (an ellipse roughly defining the shape boundary) in a single planar slice of the gall bladder, the algorithm maximizes the gradient edge along the B-spline by iteratively moving each point towards high gradients. Once segmented, the algorithm becomes fully automated, using shape and texture statistics with knowledge of the region of interest defined by initial segmentation to segment subsequent planar slices of the gall bladder. The authors claim that an operator typically needs to correct points in the contour during the initial frame segmentation when high gradients are not consistent across the shape boundary. The authors note the apparent lack of temporally/spatially persistent high gradient edges in real image but, they demonstrate that with a statistical model of contour shape this deficiency can be overcome.

More recently, powerful methods of shape modelling have become prevalent throughout the literature. The Active Shape Model (ASM; [17]) combines statistical machine learning with active contour optimization, providing a robust solution to ultrasound image segmentation. A good example of segmentation with an ASM can be found in [18]. Here, the authors actively construct a model of prostate boundary shape and appearance (pixel profiles measured across the shape boundary at fixed intervals), by annotating a large number of real-world example ultrasound images of the prostate. The authors demonstrate the power of the ASM, as their real-world images contain poor shape boundaries which present as very flat edge gradients. They use the ASM statistics to regularise shape deformation while maximising the edge gradient across the shape boundary. While they claim full automatic segmentation, they do clearly state that some user intervention is required, and rather importantly they do not demonstrate generalisation of the shape model across different participants.

In conclusion, ultrasound image segmentation is difficult due to the inherent features of ultrasound images (speckle, attenuation, low contrast between regions, signal drop-out, discontinuation of segment boundaries). The literature is incomplete in terms of tried methods and specific application to well-defined skeletal muscle. The best methods make use of shape and/or texture priors to regularise and help avoid fitting to erroneous boundaries. This concludes the broad review of ultrasound-specific segmentation. The following section is a review of manual and automated skeletal muscle ultrasound analysis techniques.

2.1.2 *Skeletal Muscle Ultrasonography: Information Extraction*

Protocols for measuring attributes of skeletal muscle can be found in the literature as early as 1985 [19]. Here, the authors take manual measurements of muscle thickness and focus on their interpretation of recorded measurements, rather than proposing an automatic method. Soon after in 1989, computational methods of skeletal muscle ultrasonography analysis appeared in the literature [20]. The authors proposed an ultrasonic strain gauge based on speckle tracking. They validated their results by applying varying degrees of strain to a phantom and correlating the tracking results with the recorded strain. They further validated their approach by recording ultrasonography of the forearm during isolated finger movements, and qualitatively comparing the speckle tracking strain map with the expected muscle activation map.

The standard now set for estimating strain in muscle via ultrasonography, improved tracking algorithms were proposed, with an aim to improve either speed and/or accuracy of results. A multi-level speckle tracking algorithm was proposed to increase accuracy by combining the effects of tracking small (11×11) and large (31×31) regions of interest [3]. The authors note that the combined effect of improved spatial resolution of smaller regions over larger regions, and the improved region tracking accuracy of large regions over small regions, can be used to improve tissue strain estimation. In another attempt to improve tracking accuracy, an adaptive *a priori* tracking algorithm was proposed [2]. Their approach makes use of image-wide speckle tracking information along with a priori knowledge obtained from ultrasound speckle motion models, to regularise the motion field. The general consensus at this point is that the accuracy of local motion tracking in ultrasonography sequences is bounded by out-of-plane motion of non-rigid material. These studies represent some of the first comprehensive analyses of the properties of feature tracking, and its limitation in ultrasonography. Tracking local features in a 3D bounded volume, with only a single planar slice of grey-level information is an ill-posed problem, i.e. the distribution of the 3D structure of the data from the 2D image-plane is not deducible, only approximations can be made, and therefore drift over time is an inevitable artefact of this technique.

Work on the analysis of skeletal muscle via ultrasonography can be found in the literature, with manual efforts to estimate muscle strain [21], moment arm length [22], tendon parameters [23], and sensitivity comparisons between electromyograms and ultrasonography [24]. Manual methods look at specific human-measurable parameters, such as muscle thickness, fascicle orientation/curvature, and fascicle length. While these methods are manual, they make the case for the wide breadth and access to *in vivo* information about dynamic muscle accessible via ultrasonography. Furthermore, studies have been conducted on the sensitivity of measureable parameters (information content of ultrasonography) which revealed that ultrasonography captures more detail about isometric contraction than EMG (the current gold standard technique for estimating aggregated activation of motor units

during contraction) under certain conditions [24]. The authors empirically showed that in certain muscles (tibialis anterior, biceps brachii, brachialis, transversus abdominis, obliquus internus abdominis, and obliquus externus abdominis) under isometric contraction, changes in motor unit activation as recorded by EMG related non-linearly to changes in measureable parameters, as recorded from ultrasonography. They demonstrated that in principal, for low force contractions EMG was less sensitive than ultrasonography, and for high force contractions ultrasound was less sensitive than EMG. This result means that, without any way to quantify motor unit activity via ultrasonography output, there exists a gap in the knowledge about muscle contraction at low force output. However, they claim that the same results imply that ultrasound cannot be used to discriminate between large contractions.

Early computational techniques were geared towards the analysis of motion fields generated from ultrasonography sequences during muscle contraction [25, 26, 27, 28]. After the establishment of manual analysis protocols the development of alternative computational approaches followed; these make absolute estimates of muscle state, rather than tracking changes in state over time. There are many notable works which aim to estimate muscle fascicle orientation/curvature and/or length from single images [29, 30, 31, 32, 33]. The most promising work on the quantification of fascicle orientation used a vessel enhancement filter to enhance the anisotropic (tube-like) textures (which correspond with, and are assumed to be fascicles) in images of human calf muscle with two different techniques. The Radon Transform was used to quantify global fascicle curvature and discrete oriented wavelets were used to quantify local fascicle orientation from ultrasound images of the vastus lateralis and gastrocnemius muscles [30].

This work was later extended and applied to 3D ultrasound where 3D fascicle curvature is quantified locally over the whole muscle [31]. The authors report accurate measurements of fascicle curvature to within 0.06^o which can be used to estimate total fascicle length between connecting aponeuroses. A number of issues are raised on repeatability; the authors point out that homogenous regions of muscle and regions with visible blood vessels contribute to erroneous results, and are therefore manually excluded. For static muscle the technique is very reliable. However, for sequences of contracting/lengthening muscle the analysis is vulnerable to the appearance of homogenous regions and structures that are not fascicles such as blood vessels. Their method also requires that an operator defines a region of interest (usually selected to include well defined fascicles), as opposed to automatic, unbiased segmentation of muscle regions[2], although segmentation is not the research question they pose. An extension of this led to the development of a technique for the quantification of regional fascicle curvature [32]. While there are other methods for quantifying regional fascicle

[2] It should be noted that automatic segmentation may require the presence of well-defined aponeuroses.

curvature [34], their approach revealed that fascicle curvature varies in different regions of the muscle, which is important for understanding muscle mechanics.

In conclusion, many robust manual and automated techniques have been developed for extraction of information from skeletal muscle via ultrasonography. This information is rich, meaningful, and readily accessible non-invasively at relatively low cost and without requiring excessive operator skill. The variety of information/estimates that have been extracted (muscle length, fascicle curvature/orientation, muscle thickness) has been shown to be related to real externally measured physiological signals (joint angles, EMG, force). There is now a question of what the limit of the information that can be extracted is, and under what conditions it can be measured. The following section is a review of the evolution of a class of information extraction techniques called feature learning.

2.2 Machine Learning: The Evolution of Feature Learning

Machine learning is a field of computer science in which the aim is to develop algorithms that can automatically learn to extract knowledge from data. Some of the most successful machine learning algorithms are inspired by the human brain; Artificial Neural Networks (ANN; a.k.a. neural nets). This review will focus on ANNs and their natural progression from simple single-layer models, to complex multilayer models, and unsupervised feature learning.

2.2.1 Linear Models

A binary threshold neuron is the simplest model of a neuron in the ANN paradigm [35]. This model is a governed by a simple thresholded linear function,

$$J(x,w) = b + x^T w = \begin{cases} 1 \ \ if \ \ \geq 0.5 \\ 0 \ otherwise \end{cases}$$

Equation 1

where x is an input vector, w is a weight vector of equal length, and b is a bias term. The model is represented graphically below.

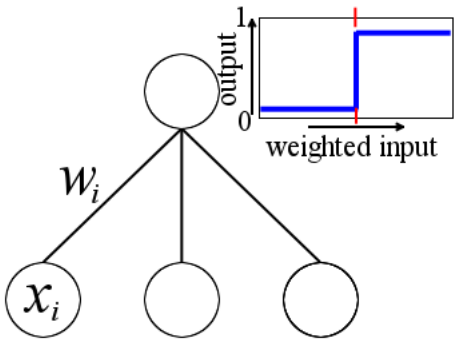


Figure 9. A binary threshold neuron is shown with 3 inputs $\boldsymbol{x}$, and 3 associated weights $\boldsymbol{w}$ (the first weight can be thought of as a bias, and the first input can be thought of as a constant with a value of 1) connecting the inputs to an output node (the neuron). In the graphic i represents the index of the input/weight. The output of this system according to Equation 1 is shown as a function of its weighted input (the red line markings indicate the threshold; the blue line shows the output).

The binary threshold neuron was proposed as a mathematical model of how biological neurons might make decisions on whether to fire an axon potential to propagate via its axon to other neurons. The weights represent synapses (the connecting junction between neurons), and output function represents the decision function of a structure known as the axon hillock. Finally the input represents the dendrites, which connect via synapses to the axons of other neurons. This model, when presented with an input vector takes the weighted sum of its inputs and then makes a decision about whether or not the input belongs to a particular class of inputs or not (i.e. is the weighted sum of inputs greater than the threshold or not?).

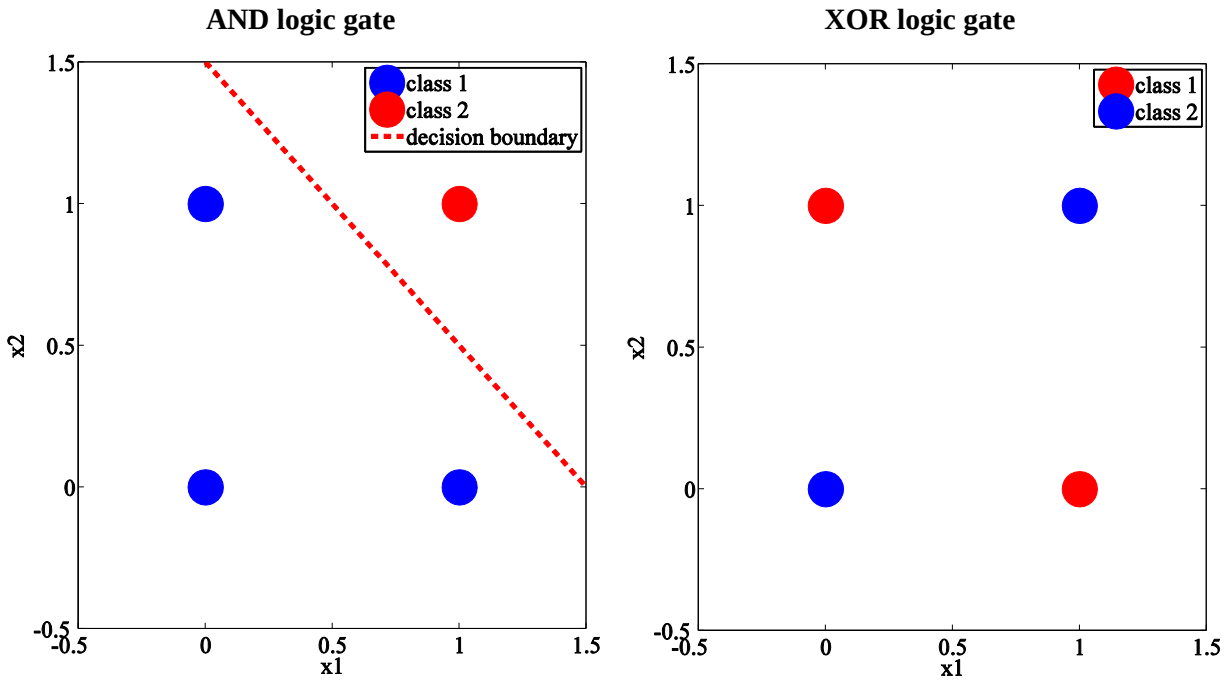


Figure 10. Examples of linearly separable and not linearly separable problems. Left: the AND logic gate. Right: XOR logic gate. Two inputs, x1 and x2, have binary states (0 or 1). The states can be separated in data-space by a straight line in the case of the AND gate, but not the case of the XOR gate.

Later, a learning rule was proposed that suggested how the weights might be changed automatically based on previous examples of input-output combinations [36]; i.e. learning. This led to the development of the first generation of neural networks, the perceptron, and the perceptron convergence procedure for automatically learning a set of weights [37]. The perceptron was fundamentally proven to be limited in its capabilities, in a machine learning sense [38]. The authors proved that the perceptron was only capable of classifying linearly separable classes of input vectors; data that can be separated by a straight line, plane, or hyper-plane (see Figure 10). The authors also noted that a multilayer network of perceptrons could be used to solve non-linear problems (networks with more than one layer are called deep neural networks). However, the perceptron convergence procedure learnt by adjusting weights based on an input vector and a corresponding target output, which meant that only the top layer perceptrons had a learning signal with which to change their weights; there was no efficient algorithm[3] that would allow optimization of the weights of the other layers of perceptrons.

[3] Numerous inefficient algorithms were subsequently developed that essentially created random perturbations of weights or activities of neurons, thereby discovering how to change the weights to improve the performance of the network.

2.2.2 *NONLINEAR MODELS*

After many attempts to learn multiple layers of neurons automatically, the error backpropagation algorithm was proposed with a mathematical proof of its convergence [39]. The error backpropagation algorithm could (theoretically) efficiently learn any number of layers of connected neurons. The technique was widely accepted as the standard method for learning multiple layers of neurons for solving nonlinear problems. However, while the algorithm was proven to converge, its convergence was heuristic and for arbitrary complex problems it often converged on locally optimum solutions (a solution that is not as good as some globally optimum solution). The technique can also suffer from slow convergence when the transfer function experiences 'flat spots' in terms of its derivative [40].

After the apparent practical failure of the backpropagation algorithm, the Support Vector Machine (SVM) algorithm was proposed [41]. The SVM algorithm provides a single layer binary classification model, with an algorithm that guarantees convergence for linear problems, using a constrained optimization method called Lagrange multipliers. The objective of the SVM is to find a set of weights w that linearly separate the input space x into binary classes t, subject to the constraint that the separating line is maximally distant from all input vectors of both classes; this is called margin maximisation. The authors also proposed an extension of the margin maximisation algorithm to solve nonlinear problems. SVMs use one additional layer of nodes and a technique known as the 'kernel trick' to compute similarity measures (such as the Gaussian Radial Basis Function; RBF) between input vectors and algorithmically selected vectors (these are called 'support vectors') up to the number of vectors in the data set. The similarity measures provide a projection/transformation of that data-space such that it may be linearly separated. SVMs (a.k.a. 'kernel machines') are a class of machine learning algorithms that are said to have 'shallow architectures'. Research has identified the importance of 'deep architectures' (many layers) in the context of circuit complexity theory (the premise that certain circuits require multiple layers to be efficiently implemented), for learning efficient and concise models [42], which imposes an upper limit on the usefulness of SVMs for sufficiently complex problems.

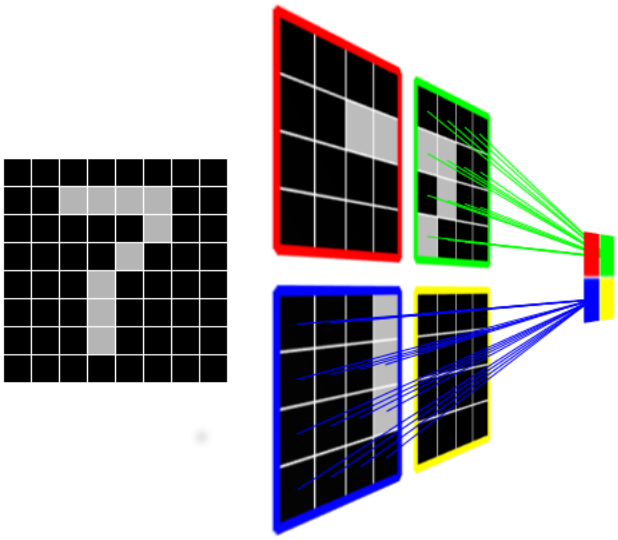


Figure 11. Example of a simple 4-neuron convolutional neural net architecture. An $\mathbf{8 \times 8}$ binary image of a digit is split into 4 sections (in real systems these sections can overlap), and each of the sections is colour coded, representing the inputs to 4 neurons (typically there are many more neurons in a real example). Each neuron is connected via weighted connections to every pixel in its local (colour-coded) region. The weights of each of the 4 neurons are shared meaning that they are represented by a single matrix of coefficients. This allows replication of local feature detectors; i.e. if one neuron learns to detect and oriented edge, the other neurons will have the same capability. The 4 neurons form a spatially organised layer of features ($\mathbf{2 \times 2}$ representation of the original image). The layer of spatially organised features can form the input to a second convolutional neural net, and this process can be repeated indefinitely (this is called deep learning).

Convolutional Neural Networks (CNN) were proposed as a method for learning robust deep architectures via the backpropagation algorithm [43]. CNNs overcome the weakness of shallow architectures (SVMs), and this is evident in the literature on handwritten digit recognition/classification [44], and more substantially for object recognition [45]. CNNs use replicated local neurons over divisions of the input space, and the incoming weights of each neuron are shared (see Figure 11). Each neuron is known as a convolutional filter, and each filter is spatially organised and can be thought of as a filtered image. Additional layers of filters can be leant on top of each layer, indefinitely. Each layer of a CNN is highly regularised because of the weight sharing, and each layer builds in additional shift/translation invariance. The original CNN was a 'supervised' learning algorithm which required that for every image of a handwritten digit there was an associated class label. The main problems of learning from labelled data is that it is difficult to acquire in large volumes (often large volumes are required), and there is not a great deal of information contained in a label (low bandwidth) that would lead to learning a good/generalised model. The solution to learning robust deep architectures without labels is a class of learning techniques known as unsupervised feature learning.

2.2.3 *Unsupervised Feature Learning*

Feature, is the collective term for a variable input to a model; the features of an image are its pixels and the output(s) of a model can be considered the features of another model. Unsupervised feature learning of deep architectures originated with the Cognitron/Neocognitron [46, 47], which was the predecessor of the more contemporary CNN. Modern CNNs typically favour a technique called Sparse Coding [48] with weight sharing to learn sparse representations of local partitions of a data vector all unsupervised (without labels), a layer at a time [49]. At each layer, the responses of neurons are aggregated (typically by k-means or PCA) to reduce the dimensions of the layer-wise representation, and to build in some shift invariance. The aggregation of feature responses of each sparse layer is known as 'max pool sampling'. Max pool sampling of layer-wise feature responses inherently means that the spatial information about features is lost increasingly with every additional layer.

The Bayesian Belief Network (BBN) is a powerful technique for learning multiple layers of features without labels, while retaining all the spatial information relating to features [50]. The Sigmoid Belief Network (SBN) was proposed as an expansion of this idea [51]. The objective of a belief network is to create a generative model of the features/inputs at every layer. Creating generative models means modelling the data such that from the model alone, realistic example data vectors can be generated; i.e. the structure and rules/correlations about the input vector have been modelled accurately in a concise model. The general idea is to learn multiple layers of features without labels, then fine-tune the final model with a small amount of labelled data, and this was a powerful new concept. The main problem with SBNs is that the learning time does not scale well with additional layers, and deep models are prone to local minima due to difficulty in obtaining true samples for intermediate-layer neurons during learning (i.e. this is the same problem as for the multilayer perceptron before backpropagation).

The Boltzmann Machine is another type of generative model, capable of learning a single layer of nonlinear features from input data [52, 53] however, its connective architecture is complex which slows the learning considerably. Restricting the connectivity of Boltzmann Machines (Restricted Boltzmann Machine; [54]) led to a much more efficient model however; the learning was still too slow. It was later demonstrated that RBMs could learn a layer of nonlinear features and by transforming data (via the RBM features) over the dataset, additional layers of nonlinear features could be learnt on the transformed data. This process could be repeated with each new set of transformed data indefinitely; this was shown to be exactly equivalent to learning a deep SBN [55]. Recently, a faster learning algorithm for RBMs was proposed which substantially improved the learning time for an RBM [56]. Pre-training deep networks with RBMs overcomes the limitations of backpropagation, and simplifies the process of learning very deep networks of nonlinear features, and this was shown to outperform some of the state-of-the art techniques like SVMs [57]. This technique is now used to pre-train layers of features in CNNs [58]. CNNs are the current state of the art in

machine learning, although experts believe that the max pooling protocol for learning a CNN is the incorrect thing to do, because valuable spatial feature information is lost as the complexity (deepness) of the network increases.

In conclusion, feature learning is a powerful technique for extracting important, descriptive features from large volumes of data, without labels or prior knowledge about what the descriptive information is. Feature learning could be a powerful alternative method for modelling the states of skeletal muscle via ultrasonography, and relating those states to externally measured physiological signals. The following is a review of the current state of the art in skeletal muscle analysis, and image analysis and inference with deep learning of features.

2.3 Review of Selected Publications

This section of the review presents an expanded critique/appraisal of five selected publications on ultrasound segmentation, calf muscle segmentation and fascicle tracking, quantification of curvature/orientation of fascicles, and feature learning in images. These works represent the state of the art, and the relevant publications lead the field in automated analysis of human calf muscle via b-mode ultrasonography. By demonstrating a multitude of techniques, these works show the potential for ultrasound analysis, and its intellectual value. On the question of how to exploit the potential of ultrasonography for analysis of skeletal muscle, the last two papers in this critical review section explore two potential methods for automatic information encoding from raw images for segmentation, and information extraction.

2.3.1 Ultrasound Image Segmentation: A Survey [11]

This paper has been selected because it is the first comprehensive review of techniques for segmentation of medical b-mode ultrasonography. The authors review many different segmentation techniques with respect to their clinical application, since new techniques are typically proposed to solve a given clinical problem. The authors open with the statement that segmentation of images is strongly influenced by the quality of data. They further state that due to orientation of the ultrasound probe, artefacts of ultrasonography such as speckle, attenuation, and signal dropout can result in missing/incomplete boundaries of objects in the image. They also note that contrast between regions in the image can be low, although they go on to say that image quality has improved, making these issues less of a problem. With this, it can be said that their first conclusion is that ultrasound image segmentation is very challenging.

The most popular and successful approaches make use of shape and/or texture priors to regularise and guide heuristic searches, based on Bayesian statistics, Genetic Algorithms (GA), Markov Random Fields (MRF), Artificial Neural Networks (ANN), and Active Shape Models (ASM). The authors note that statistical techniques such as those mentioned often require expert annotation, stating that it is difficult to do due to image quality. The authors further state that this also makes evaluation of the accuracy of new techniques difficult. A popular approach for evaluating performance was to use multiple expert annotations, which reveals that the variability between experts is often higher that the variability of the segmentation techniques for identifying borders. Some techniques used initial global contour detection using global shape priors, followed by temporal segmentation using the initial segmentation as a location prior. This allowed relatively robust temporal sequence segmentation for whole-sequence processing.

The relative success of techniques is dependent on the problem at hand. The authors attempt to score the techniques based on attributes such as image resolution, degree of automation, and problem

classification (e.g. cardiac segmentation, anomaly detection/classification, prostate segmentation, etc…). For anomaly detection the superior technique was the active contour paradigm [15] often in conjunction with various other methods for filtering images and modelling shape priors. For prostate and cardiac segmentation the best approaches used were often based on shape or texture models with the Active Shape Model paradigm [17], or the Active Appearance Model (AAM) paradigm [59]. The authors then make the final conclusion that "a good ultrasound image segmentation method needs to make use of all the task-specific constraints or priors". The authors wrap up the review with a list of 10 carefully selected ultrasound image segmentation techniques and an accompanying summary. The techniques were selected not for their status as a successful technique, but for their approach/paradigm and/or for their evaluation on reasonable volumes of real clinical datasets. This paper to date represents a seminal work in the field of ultrasound image analysis.

2.3.2 *AUTOMATED REGIONAL ANALYSIS OF B-MODE ULTRASOUND IMAGES OF SKELETAL MUSCLE MOVEMENT [1]*

This paper has been selected because it represents the first automated regional analysis of dynamic human skeletal muscle via ultrasonography. The paper presents a method of fully automatic calf muscle segmentation, and within-muscle regional tracking of fascicles. The authors highlight that the purpose of their work is to provide an automated analysis of skeletal muscle to understand its functional significance in anatomy, during dynamic tasks. Their work is based upon previous work in which tracking of local features about (on, and nearby) the aponeuroses of the calf muscle was used to show that changes in muscle length could be approximated various passive and active ankle and knee rotations [60]. In the original method, the features to be tracked were manually selected on and about the aponeuroses and they were tracked using a cross-correlation method [61]; no automatic segmentation was proposed.

Use of a more robust and widely accepted tracking method was proposed in this paper, namely Kanade Lucas-Tomasi (KLT) feature tracking [62]. This method made use of a multiple resolution tracking technique which helps quantify large feature displacements, and therefore was shown to be more robust when tracking larger contractions and joint rotations. The KLT also uses a quasi-Newton-Raphson iterative approach to localising feature displacement, which led to faster feature tracking. Furthermore the KLT has methods for replacing features that have drifted too far away from their original state (features that look very different from when they were first selected; i.e. lost features).

The segmentation method used was the Active Shape Model (ASM), which required manual marking of over 600 images of calf muscle boundaries in order to capture enough statistics about relative muscle size/shape/orientation. The ASM required initialisation before a gradient descent fitting procedure was executed. Manual and automated initialisation approaches were implemented, and the subsequent segmentations were compared. This comparison revealed that on average the automatic approach was more robust. However, out of 8 total participants, two cases showed poor segmentation, with less than half of the sequence being segmented to less than 1 millimetre error. The authors attributed the poor results to loss of muscle boundary definition, which may imply that the fascicle plane orientation is important. One could further suggest the small training set of only 7 participants may have also contributed to any failure.

For regional analysis, the KLT algorithm automatically selects many (≈ 200) image-wide non-overlapping scattered ‘good’ features based on an automatic measure of tracking reliability [63]. Then these features are tracked throughout each video sequence and their coordinates over time are logged. The ASM was then initialised and subsequently segmented the entirety of each sequence into two muscle regions (and aponeuroses). Because features are not persistent over time, and because the KLT

algorithm can replace ‘lost’ features, a set of ghost markers were automatically initialised on and about the aponeuroses (as in [60]) within each muscle via the segmentation. These ghost markers (termed probes) interpolated the underlying feature movement via a Delaunay mesh constructed from those features. The ghost markers were then stable over time and could be used to analyse local regional propagation of fascicles.

The authors conclude that this is the first time a fully automated segmentation and analysis technique have been applied to skeletal muscle. Their first significant conclusion is that the automated segmentation is accurate up to $\approx 0.3mm$. The second conclusion is that their proposed regional tracking approach is significantly more accurate than the previous cross-correlation method for large movements resulting from large changes in joint angle ($\approx 20°$), or large force contractions ($\approx 50Nm$). The results also show that for smaller movements, the previous method is more accurate, although this is expected due to the iterative nature of the KLT. Upon analysis of the feature loss rate over sequences, one very interesting result is that the loss rate increases with activity, which is a strong indicator of trans-planar motion of features, since features cannot possibly retain identity if they leave the image plane. They also point out that both the tracking technique and the segmentation technique is parametric, and they have not fully explored parametric optimisation.

This paper to date represents the state of the art in fully automatic regional analysis of dynamic skeletal muscle – specifically in the human calf muscle.

2.3.3 *Computational Methods for Quantifying in vivo Fascicle Curvature from Ultrasound Images [32]*

This paper is selected because it represents the first time that local regional fascicle curvature has been automatically determined in the human calf muscle via ultrasound. The authors present a robust and accurate solution to direct automatic estimation of regional fascicle curvature, in application to investigating the mechanical properties of *in vivo* pennate muscle architecture. Their work is an expansion of their existing techniques for quantifying regional local fascicle curvature, and overall fascicle orientation [30].

The original method was capable of computing local orientations of fascicles by convolving regions of the manually segmented muscle region with wavelets at discrete angular intervals (in 1 degree increments); the maximum convolution value revealed the orientation of the fascicle at that location. The result of the wavelet convolution process over the entire muscle region resulted in a fascicle orientation field. The proposed method then used a fibre tracking method (originally developed for diffusion tensor imaging) to track individual fascicles between aponeuroses [64]. That method essentially divides the orientation field into a discrete lattice structure, and then computes the median orientation within each element of the lattice. Then points are initialised in the image where the fractional anisotropy (has high variance in one direction) was greater than some threshold. Then the fascicles are tracked by following the trajectory (direction of variance) of the lattice until either the region boundary is met, or until the fractional anisotropy is lower than the threshold.

When fascicles have been tracked, their curvature is quantified by measuring the angle along each fascicle at 15 pixel intervals (Euclidean). This resulted in a curvature grid which can track changes in local regional fascicle curvature over a sequence of contracting calf muscle. The authors only consider superficial muscle layers (e.g. gastrocnemius), but they have demonstrated a powerful concept for analysing pennate muscle architecture *in vivo*. Their approach is parametric, and they note that there is a trade-off between speed and accuracy. Their technique is validated against synthetic images with known orientation of tube-like structures, reporting a typical standard error of less than $0.026m^{-1}$. Their original wavelet technique was also validated against multiple-expert annotated muscle ultrasound sequences, reporting a standard error of $1.35°$, and synthetic images with known curvature, reporting typical errors of less than $0.06°$.

The authors highlight that their technique is based on the assumption that all features within the ultrasound image are fascicles. Therefore, poor probe placement or appearance of spurious features during contraction can result in erroneous quantification of regional curvatures. The authors conclude that their methods allow the quantification of local regional curvatures, and that this may lead to better understanding of muscle mechanics under different functional conditions.

This paper represents the state of the art in quantification of local regional fascicle curvature – specifically in the human superficial calf muscle.

2.3.4 LEARNING HIERARCHICAL FEATURES FOR SCENE LABELING [65]

This paper is selected because it represents the state of the art in one of the most challenging areas of computer vision, namely automatic scene parsing of natural colour images. In the near future this work may be adapted to perform full scene parsing of ultrasonography sequences of skeletal muscle, and is therefore worthy of review. Scene parsing is the collective term for labelling every pixel in an image with the class it belongs to (e.g. road, pedestrian, car, building, etc…). The proposed approach uses a convolutional neural network (CNN) to model the features of natural images. That model was then used with a variety of segmentation methods to label every pixel in natural scenes. The hierarchical deep learning (CNN) approach is based on the premise that "good internal representations are hierarchical. In vision, pixels are assembled into edglets, edglets into motifs, motifs into parts, parts into objects, objects into scenes."

The CNN was trained in supervised mode (with labels) on the raw pixels of the natural images. The authors point out that a big problem of CNNs is the max pool sampling process, which takes a local neighbourhood of learned feature activations and aggregates their response with some arbitrary pooling function. This means that features lose some of their spatial information with every additional layer of the CNN. Spatial information is important for pixel-level scene parsing, so the authors propose a multi-resolution image approach. First, a Laplacian pyramid image is constructed at full-scale, half-scale, and quarter-scale resolution. That image pyramid is the input to a CNN, where the classical approach of weight sharing across a lattice is used in conjunction with weight sharing across the resolutions of the pyramid. This multi-resolution approach goes some way to solving the loss of spatial feature information problem, since the spatial information is somewhat encoded in the different pyramid image levels.

With the resulting parsed scene, the authors use a superpixel technique to classify pixels in local regions with the same label [66]. This information is then combined with the class predictions from the CNN, and a linear classifier predicts the final class labels. The use of superpixel information with the CNN class predictions gives good priors for additional linear classification. The authors also present two other solutions based on previously successful techniques, namely, Conditional Random Field (CRF) and tree hierarchies of segmentations with optimal cover. These additional techniques are beyond the scope of this review due to their high complexity however, their function is said to 'clean up' initial poor segmentations predicted by the CNN, to give state of the art final segmentations.

Figure 12. Example of scene parsing via multi-resolution Convolutional Neural Networks (CNN). Two different images are shown first as raw pixels (top), then as colour-coded automatic CNN segmentations with text-coded class names (bottom).

The authors conclude with results that represent the state of the art in scene parsing on 3 large datasets, reporting better than state of the art results on 2 datasets and near-best results on the other dataset. In all cases the authors report a segmentation time approximately 1 order of magnitude faster than the next best techniques. Results are compiled for the raw CNN with a linear classifier on top, the combined superpixel/CNN, the combined CRF/CNN, and the combined tree/CNN approach. This paper represents the current state of the art in scene parsing of natural colour images, and is a strong candidate for future investigation into ultrasonography segmentation.

2.3.5 *IMAGENET CLASSIFICATION WITH DEEP CONVOLUTIONAL NEURAL NETWORKS [67]*

This paper was chosen because it won a competition (ImageNet) in 2010 in which the aim was to automatically predict the probability of the subject of 1.2 million high-resolution (variable resolution) images with 1000 possible classifications. This is one of the largest datasets of labelled images to date and offered an unrivalled opportunity to test the capability of the state of the art machine learning techniques. The proposed technique was a colossal (by standards to date) Convolutional Neural Network (CNN). The eventual CNN had over 650,000 neurons in 5 fully connected layers which resulted in over 60 million free parameters. Such a model is only feasible to train due to the efficiency of learning a deep CNN, and for the introduction of GPUs to perform parallel convolution operations.

This work approaches the current limits on what expert thought on image understanding and computer vision. The authors state that this is one of the largest convolutional neural networks ever trained, and as such required "highly-optimised" GPU implementations of 2D convolution operations and other primitive parallel operations. The authors also state that any network of that size (60 million parameters) was in serious danger of over-fitting, and conventional techniques such as cross-validation of regularisation (a term in the learning function that prevents neurons becoming too specialised) parameters would be infeasible due to the sheer volume of data (1.2 million samples) and the size of the network. To address over-fitting the authors used a powerful recently developed technique, namely Dropout [68].

The premise of Dropout is surprisingly simple and is empirically shown to be extraordinarily effective at regularising large deep neural networks [68]. Dropout is the neural network response to powerful model averaging techniques like Random Forests [69]. Random Forests train lots of simple models (decision trees) – often 1000's – each with a different subset of the features in the dataset, and the average (mode) response for any given input vector is taken as the final model. This method is shown to result in a model that outperforms any of the individual models and is well generalised to out-of-sample data. Decision trees are easy to learn and literally tens of thousands can be learned if one does not worry about runtime performance after learning. In contrast it is extremely infeasible and time consuming to do the equivalent with large neural networks. However, the Dropout technique was shown to achieve the equivalent effect of model averaging by randomly dropping neurons and their connections from the neural network during training, which is said to prevent too much co-adaptation between neurons. The effect of randomly omitting neurons during learning is that lots of models are trained separately within the same architecture, since every neuron that remains after dropout is an entirely unique configuration of the network.

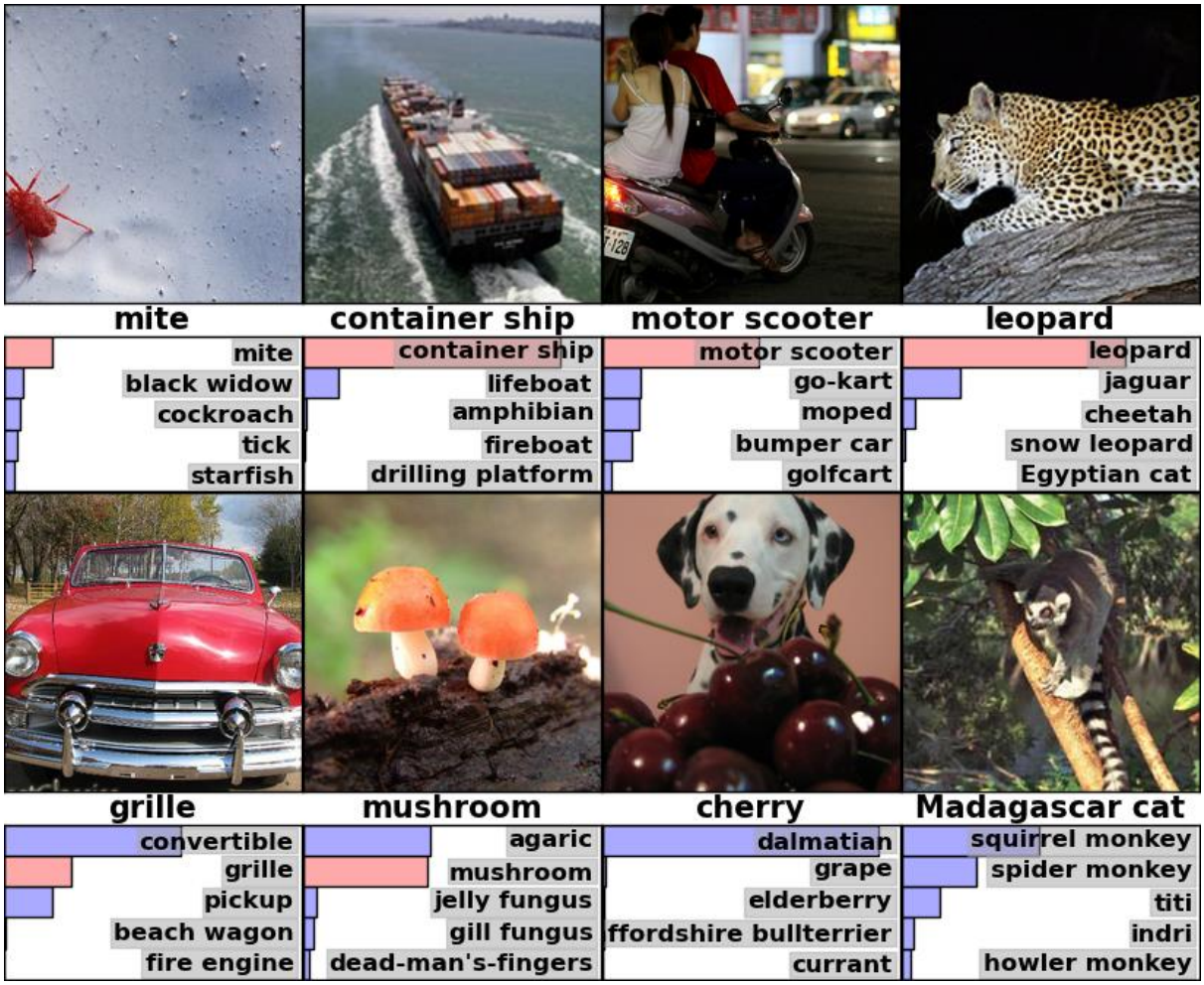


Figure 13. Images from ImageNet with class labels (directly below each image in bold text), and top 5 probabilistic CNN predictions of what the subject of each image is (just below the class labels in order of likelihood). The top row of images show successful classifications in the CNNs first guess, and even in the subsequent 4 guesses the answers are plausible (e.g. the leopard image influences guesses on a number of other big cats like the jaguar). What is perhaps more interesting is the bottom row of images, which represent incorrect classifications in the in the CNNs first guess, because the CNN appears to be predicting entirely rational guesses about the subject (e.g. the picture of the Dalmatian, or the convertible appear to be labelled somewhat debatably incorrectly).

The results of the trained system far outperform the best Computer Vision techniques like Sparse Coding [49] and Scale Invariant Feature Transform (SIFT; [70]) + Fisher Vector (Fisher kernel framework; [71]). One of the authors' most important statements was that of the necessity of the deepness of their model; the authors report that the performance was 'inferior' if any one of the layers was removed from the model (which they note only consisted of only 1% of the total parameters). To date, this result is the strongest empirical evidence for the superiority of hierarchical feature learning methods. This paper represents the current state of the art in image understanding, and is a strong candidate for investigation into ultrasound image understanding, in application to skeletal muscle analysis.

3 THE FAILURE OF FEATURE TRACKING FOR ANALYSIS OF SEQUENTIAL ULTRASOUND IMAGES OF HUMAN SKELETAL MUSCLE

3.1 ABSTRACT

This chapter is an introductory investigation into feature tracking as a paradigm and general approach to the analysis of dynamic skeletal muscle. Feature tracking is a technique which involves the selection of one or more local features (patches of texture) in an image, and approximating the local displacement of that feature in a subsequent frame. Typically features then update their locations to the new approximated positions so that subsequent tracking from that location can take place. Features can drift away from the object they are tracking as small tracking errors accumulate over a sequence. This can result in poor estimates of absolute feature positions in the long term. Tracking errors occur in the ultrasound domain due to presence of speckle noise, and when objects traverse the image plane (trans-planar motion) to name the two main causes. While drift is shown to be unavoidable in ultrasound image sequence analysis [2, 3], there have been few studies relating this to skeletal muscle. The primary aim of this chapter is to provide a parametric analysis pertaining to drift of a widely used adaptive feature tracking algorithm, namely KLT. The secondary aim is to identify the stability of feature tracking over a lengthy sequence of dynamic skeletal muscle. A single 3 minute sequence of dynamic human calf muscle was tracked via a grid of manually initialised local features using the KLT algorithm with varying feature widths and heights. Then, a single frame of the same sequence was replicated over a 3 minute sequence, and varying amounts of speckle noise was introduced to each replicated frame. Those static sequences were then tracked via the same process (grid of KLT features), and the analyses of both methods (dynamic and static) were compared. Results indicate that the paradigm of motion tracking is inappropriate as a method for the analysis of skeletal muscle over lengthy sequences. Both speckle noise and trans-planar motion contribute greatly towards drift; therefore a different approach is required for robust sequential analysis of dynamic skeletal muscle.

3.2 INTRODUCTION

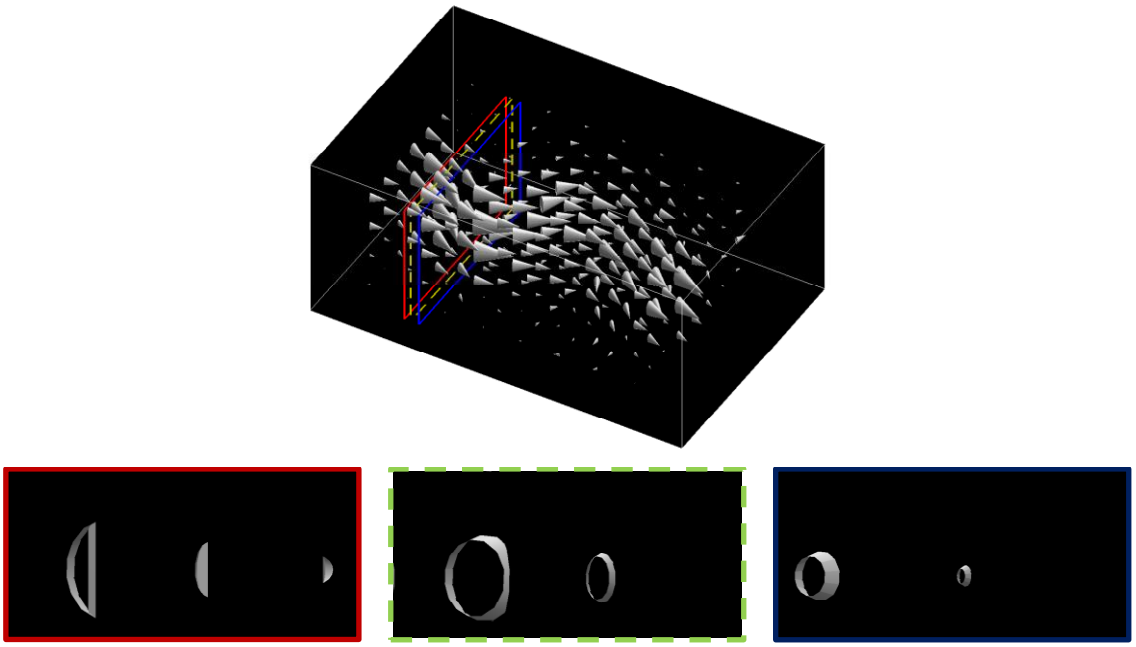
Feature tracking drift is a well-known problem and it usually only becomes significant for arbitrarily long sequences, where some other system needs to interpret the tracking results. Some contributing factors include partial object occlusion, nonlinear feature transformations (affine transformation), various types of image noise, and local changes in lighting (pixel intensity). Unfortunately ultrasonography is susceptible to all items on this list with the addition of a few other problems such as signal dropout. However, the two main issues concerning feature tracking in ultrasonography are:

1. Trans-planar motion of features (affine transformations)
2. High frequency speckle noise

A brief description is given in the following two sections.

3.2.1 TRANS-PLANAR MOTION OF FEATURES

Trans-planar motion of material is arguably the main reason for tracking failure regarding ultrasonography. It is impossible to track the motion of a feature as it moves along the non-visible third axis of the image plane because all of the local information about that feature has been removed from the image. The remaining feature information is likely to be correlated because of its connectivity however; these correlations are coincidental and can lead to tracking drift.



(a) (b) (c)

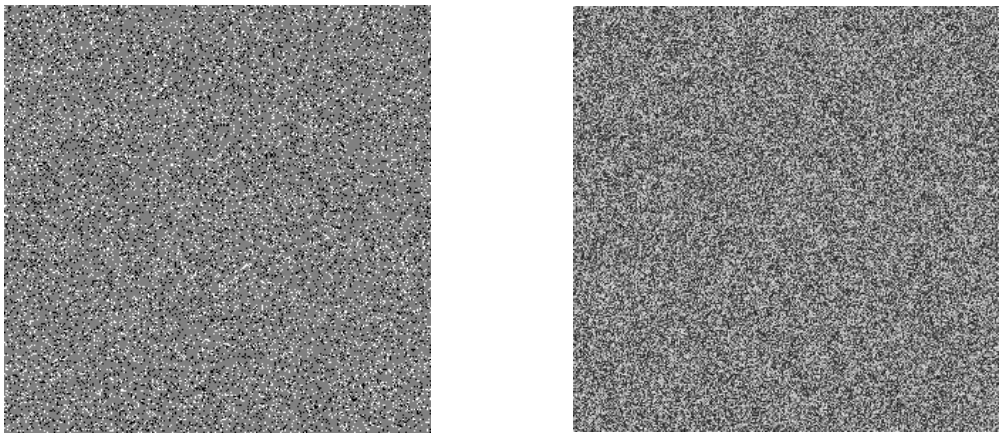
Figure 14. 3D oriented cones (Matlab wind dataset [72]) (top) as viewed from 3 planar perspectives (a) red solid plane, (b) green dashed plane, (c) blue solid plane.

Figure 14 illustrates the problem of trans-planar motion by showing a 2 dimensional slice of a 3 dimensional data space in 3 consecutive nearby locations. This can inversely be thought of as a single

fixed plane, with 3 dimensional objects passing through it, with each 2D view representing a snapshot in time. The visible objects in the 2D plane appear to move along the x/y (horizontal/vertical) axes in 2 dimensions, when in fact they are moving only in the third axis (normal to the 2D image plane). The visible objects also appear to change shape and/or size, when in fact they never do. Finally, objects can appear/disappear between frames. This is clearly problematic in the case of ultrasonography of a tightly packed dynamic tissue, since features can leave the plane, and similar (in 2D appearance) features can enter the plane nearby, resulting in tracking error.

3.2.2 HIGH FREQUENCY SPECKLE NOISE

Due to the image formation process, ultrasound images are characteristically noisy. The type of noise may be referred to as salt and pepper noise. However; speckle noise is more appropriate model of the type of noise inherent to ultrasound images. Salt and pepper noise is the presence of uniform distributed random binary (black/white) pixels over the image. Speckle noise is the presence of normal distributed continuous random pixels, and is a more accurate description of the type of noise exhibited by ultrasonography.



(a) (b)

Figure 15. Examples of salt and pepper noise (a) and speckle noise (b) when applied to a homogenous grey-level image.

For arbitrarily small features, one would expect additional random bits of information to prevent the tracking error term from reaching its absolute minimum, even for static images. Therefore one should expect the contribution of speckle noise to lead to tracking drift over arbitrarily long sequences.

3.3 METHODS

To investigate the usefulness of feature tracking as an approach for analysis of dynamic skeletal muscle via ultrasonography, a popular algorithm called the KLT was chosen to analyse a long sequence of dynamic human calf muscle. Specifically the KLT algorithm was chosen because it has been used successfully to analyse human calf muscle in previous work [1]. The KLT algorithm works by minimising the difference between two features ($I(x,y)_t$ and $I(x+\Theta_x, y+\Theta_y)_{t+1}$, where Θ is the displacement term, and t is the frame). The width and height of a feature can vary however, generally the larger the feature the more texture information there is above the noise threshold to locate it in a successive frame. Furthermore, the KLT algorithm only attempts to track local linear image transformations. Affine (non-linear) transformations cause local mismatches between frames, and the KLT drops those features that exhibit 'too much' (parametric error threshold) affine transformation during tracking.

Research has shown that regional movement of local features is nonlinear [32]. This has also been confirmed in relation to the KLT, where a study showed that with increasing dynamic contractions, greater feature losses were experienced [1]. Given those reasons, it can be assumed that by making features too small the KLT algorithm has too little information per feature above the noise threshold for reliable (drift-less) tracking. Conversely, making features too large inherently includes surrounding nonlinear transformations of image patches, leading to mismatched features between frames. Therefore one would expect an intermediate optimal feature size for tracking local image transformations over a sequence of dynamic human calf muscle. The approach taken here was to explore the possibility of stable tracking over a lengthy sequence of dynamic human calf muscle by optimizing the feature size. It is not expected that feature tracking drift can be resolved however, there may be the possibility of reducing drift to make regional motion tracking a feasible approach for analysis of sequences of dynamic skeletal muscle.

Two separate methods of analysis were used to address the two main issues surrounding feature tracking of ultrasonography sequences; trans-planar motion, and speckle noise. For both methods, a grid of KLT feature points was initialised within a manually segmented muscle region (see Figure 16). After manual muscle region segmentation, an 8×8 grid of KLT features was initialised within the muscle region. All default KLT settings were used (other than window sizes) as per the Matlab (2015a) KLT implementation (vision.PointTracker). The fascicle region was segmented manually due to lack of a robust segmentation algorithm at this point. The features were then tracked into every successive frame, always assuming continuity (i.e. no dropped features, or re-initialisation). At this point the two separate methods differ very slightly in their approach. The following two sections expand on the details.

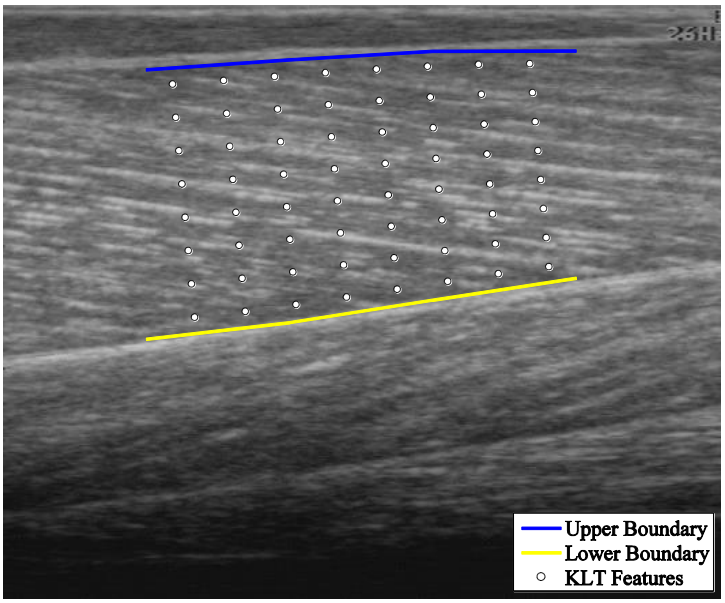


Figure 16. Initial feature positions placed between the boundaries of the medial gastrocnemius muscle. The feature grid is focused on the central region of the muscle to avoid features tracking off the edge of the image during the characteristic left-right motion of fascicles.

3.3.1 METHOD #1 – TRACKING DYNAMIC CALF MUSCLE

This method is designed to analyse the effect of dynamic skeletal muscle, where movement in the ultrasound image plane is assumed to contain trans-planar motion. A long video sequence (data taken from the experiment detailed in chapter 5, section 5.3.2) of a human medial gastrocnemius was analysed using the KLT algorithm with varying feature sizes (widths and heights between 15 and 43 in steps of 4), in order to measure tracking stability as a function of feature size. The video sequence records a human participant performing isometric calf muscle contractions in pulses dictated by instructions in the form of an on-screen feedback system. Concurrently, the electrical activity of the muscle was recorded via sEMG and processed with a low pas filter (details in chapter 5, section 5.3.2), meaning that for every frame of ultrasound, there is an associated measure of contractile activity. Because the participant is performing isometric contractions and their ankle joint is fixed, the amplitude of the sEMG signal represents a quantitative distance from the muscle position; where the participant's muscle is at rest (i.e. no sEMG amplitude) the muscle should be approximately in its original configuration, since the joint angles (knee and ankle) were fixed for the duration of the trial. The same statement can then be said about any local features that are tracked for the duration of the sequence. Accepting arbitrarily small errors, all local features should return approximately to their original position at the end of the sequence assuming the absence of drift. Therefore, this fact can be used to measure drift from the initial feature grid at the end of, and over the sequence.

3.3.2 *Method #2 – Tracking Static Calf Muscle with Speckle Noise*

The second method is designed to analyse the effect of speckle noise on feature tracking. The initial frame of the long video sequence used in method #1 (see previous section) was replicated to make a sequence of the same length, which only contained a single repeated frame. This is in order to discount the possibility of drift caused by trans-planar motion. The optimal feature size (35×35) – discovered from the analysis of method #1 – was used to give a fair comparison. For each frame in the new sequence, a degree of speckle noise was added according to the equation,

$$I = I + N(0, v) \odot I,$$

where I is the image matrix, $N(0, v)$ is a matrix of normally distributed random numbers with 0 mean and v variance. For this analysis varying degrees of noise are added by increasing v from 0.001 to 0.01 in steps of 0.001 (i.e. [0.001:0.001:0.1] in Matlab notation). It can be seen in Figure 17 that towards the upper end of values ($v \geq 0.007$) the image becomes quite distorted and grainy in appearance, which is not necessarily a good representation of the typical amount of speckle noise expected from an ultrasound image.

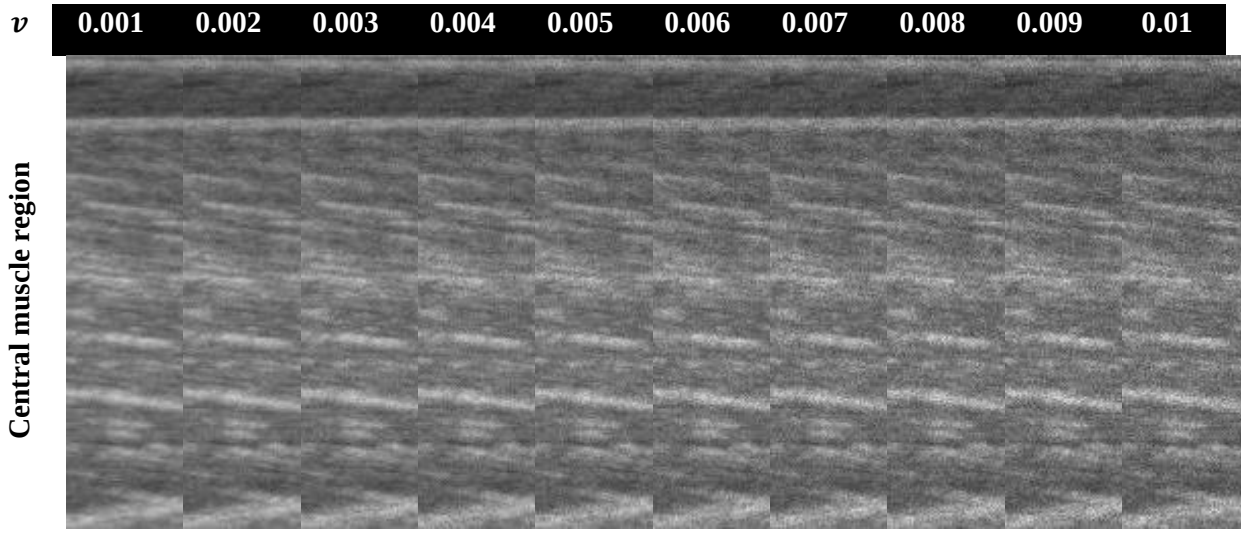


Figure 17. Illustration showing the effect of adding increasing amounts of speckle noise to a central portion of the gastrocnemius in the ultrasound image. Notice the loss of feature definition (especially for relatively small features) as v increases (left to right).

3.4 RESULTS

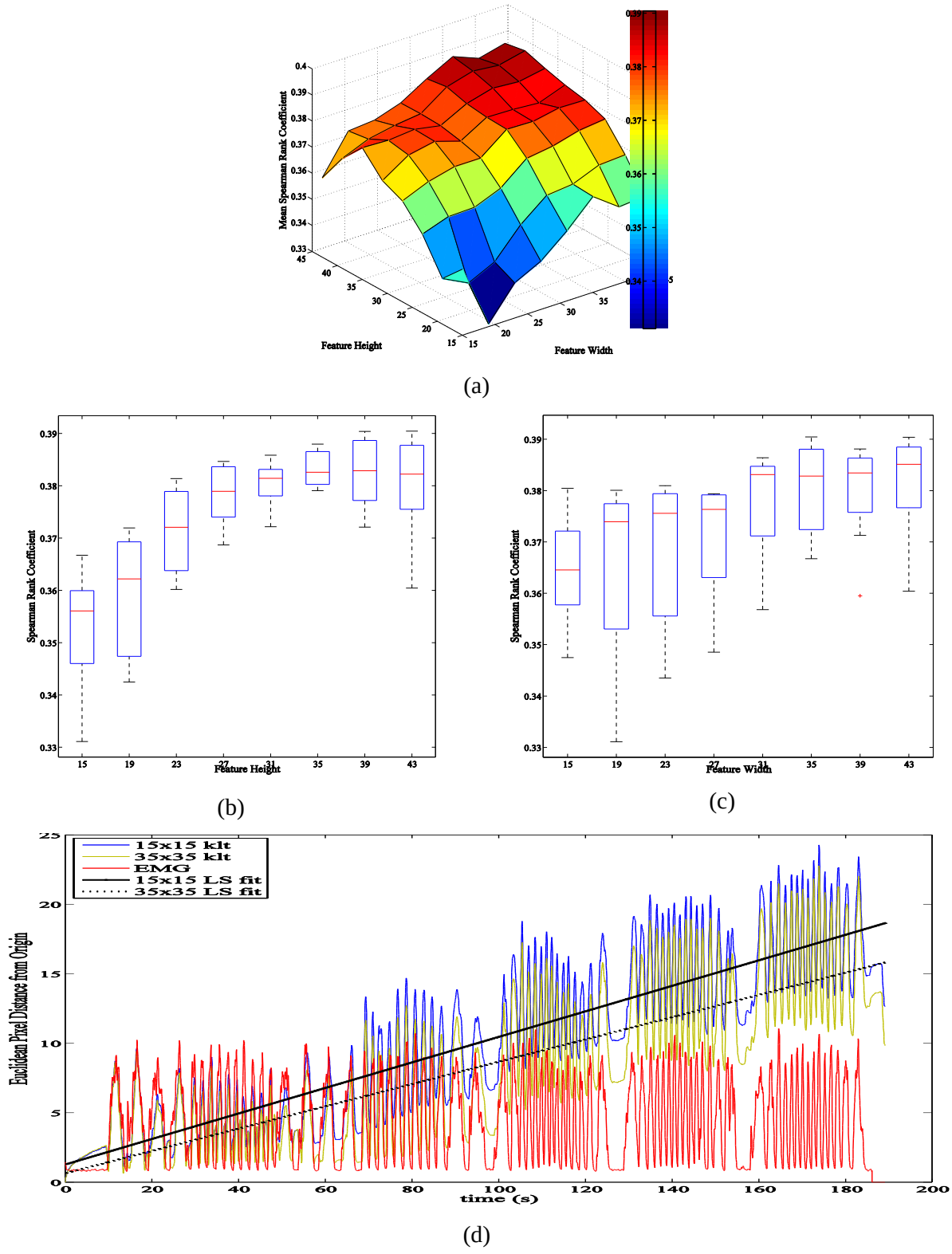


Figure 18. (a) Mean spearman rank correlation between Euclidean feature distances from their initial positions, and muscle activity (sEMG), as a function of feature size. (b) and (c) Box and whisker plots of Spearman rank correlations between Euclidean feature distances from their initial position, and muscle activity (sEMG), as a function of feature height/width respectively. (d) Time series of the mean upper median (Spearman) of Euclidean feature distances from their initial position as a function of time, plotted with muscle activity (sEMG).

In Figure 19 sEMG and feature displacement exhibit a monotonic relationship therefore the Spearman correlation method was used to quantify tracking reliability. The results in Figure 18 reveal that there is an optimal feature size that falls between ‘large’ and ‘small’. A feature size of 35×35 was shown to have the least variance and one of the highest median correlation values of all feature sizes. Smaller features exhibit a much lower average correlation. Larger features exhibit greater average reliability yet greater variance in correlations. This result can be explained with the results of previous studies, where the authors report that different regions of the image require different feature sizes for reliable tracking [2, 3]. Importantly, the smallest feature size (used in previous work [1]), gives the poorest correlation and largest variance.

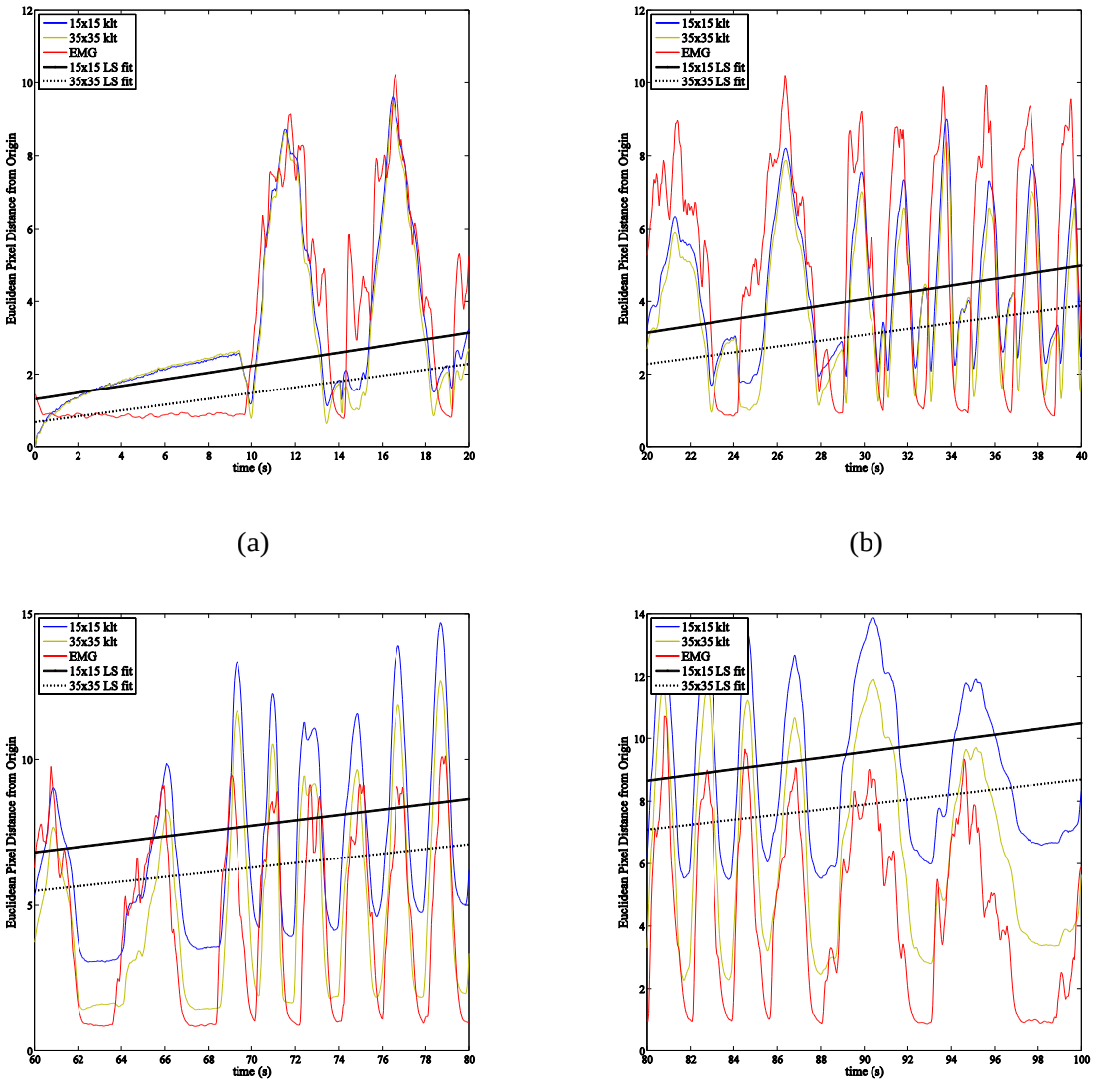


Figure 19. 4 individual 20 second segments of the time series (d) in Figure 18 highlighting the de-correlation between KLT features and sEMG over time as a result of tracking drift.

While feature sizes have a clear effect on tracking reliability, if one examines Figure 18 (d) it is clear that this effect is minimal. What is of more importance is the apparent similarity between the resulting signals of 15×15 and 35×35 features; the signals seem to encode the same information, with only a slight difference in least squares fit lines over the entire sequence. Further evidence of the failure of adaptive tracking can be seen in Figure 19, which shows 20 second segments of time from the beginning of the sequence where tracking appears to be robust, to $t \equiv 100s$ where both the maximum and minimum correlating feature sizes have drifted from their origin enough to cause a de-correlation with sEMG.

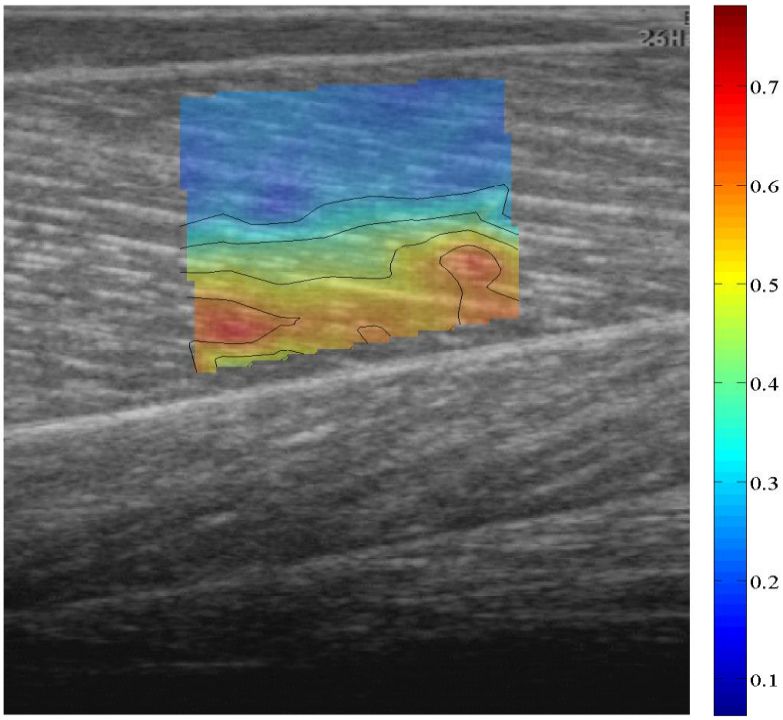


Figure 20. Heat map of mean per-feature Spearman rank coefficients over all feature sizes. Note over half the muscle region fails to provide enough detailed information such that it can be reliably extracted by a feature tracking technique. The contour lines represent steps of $\frac{1}{5}$ of the spearman coefficient range.

One can examine the distribution of these correlations by looking at a heat map of average correlations over the muscle region. One reason for poor feature tracking is use of 'bad features'. Good features to track are defined in a seminal paper [63] as corner features; features that have a high variance over both axes of orientation, so that the feature has only one cost minimum (theoretically) in terms of template matching. The authors of this work also describe a method of automatically selecting good features from the distribution. Figure 20 shows that at least superficially this statement appears to be true; the best correlations are found in the deeper part of the muscle where the definition and presence of corner features appears to be greater. However, the two strongest areas of correlation appear to be in areas with very little fascicle definition – albeit they are deep in the muscle. This seems to suggest

that well defined features appeared in those areas at some point in the sequence which made those regions more stable, but ultimately, there is no indication that starting with ‘good features’ would appear to significantly help.

More generally, the reliability of the deeper part of the muscle might be due to less non-linear and affine transformation than the superficial part, and/or the signal to noise ratio being lower in the superficial part than the deep part. As is apparent in Figure 20 and Figure 21, features track more reliably in deeper muscle regions. One might hypothesise that this is most probably caused by the deformation and compression of superficial fascicle ends due to probe placement (see chapter 6, section 6.3.3.1 for details).

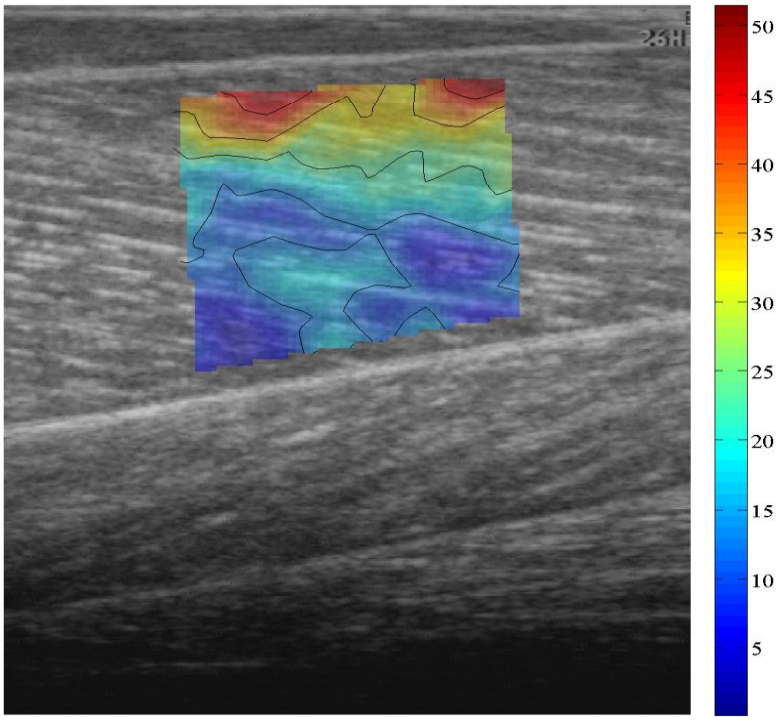


Figure 21. Heat map of mean Euclidean feature distances (in pixels) per feature over all feature sizes.

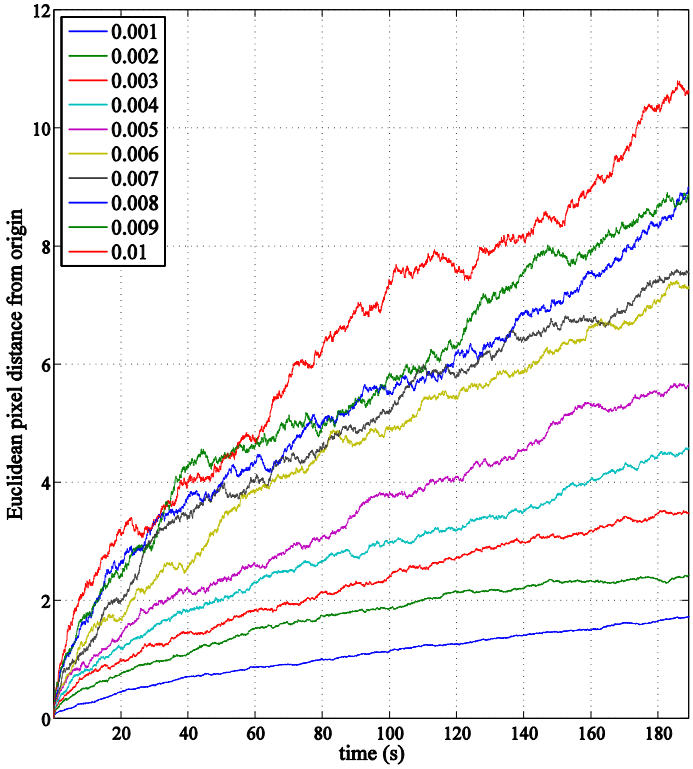


Figure 22. Shows how features distance themselves (on average over the feature grid) from their initial location's as they track increasing amounts of speckle noise over a **3′** sequence. Note: for a static image, there should be no variation for the origin if speckle noise had no effect.

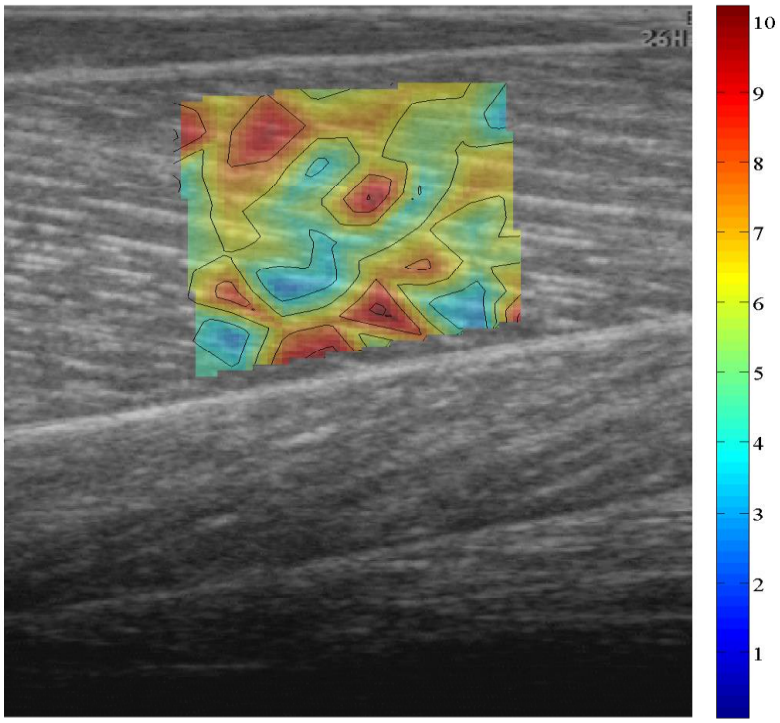


Figure 23. Heat map of mean Euclidean feature distances (in pixels) per feature over all feature sizes.

There can be little doubt that speckle noise inherent to all ultrasound images adds to the problem of tracking drift (see Figure 22). However, one must consider the degree of effect that speckle noise has over motion tracking in ultrasound terms. A comparison of Figure 21 with Figure 23 reveals that the effect of trans-planar motion and/or affine feature transformation inherent to dynamic skeletal muscle causes a much greater problem than speckle noise, with a five times greater effect on feature drift from origin (in the worst cases). Figure 23 also shows a much more uniformly distributed error, more representative of 'good feature' selection as per [63]; i.e. regions within the image that contain poor fascicle definition record poor tracking (red patches in the heat map). Furthermore, the apparent stable nature of the superficial features in Figure 23 seems to suggest that there is in fact enough definition and detail for reliable tracking in that region, and that the features in Figure 21 represent features that cannot track contraction accurately due to trans-planar motion, affine transformations, and possibly deformation of the feature field resulting from probe placement.

3.5 Discussion & Conclusions

The purpose of this chapter was to provide an introductory investigation to the use of features tracking for analysis of dynamic human skeletal muscle. While there was already a strong intuition derived from the literature that feature tracking is an inappropriate paradigm for analysis of dynamic skeletal muscle [3, 2, 1, 60], this study aimed – in the first instance – to ascertain which of the two artefacts (trans-planar motion or speckle noise) of ultrasound contributed more towards the failure of tracking algorithms. Secondly, this study aimed to analyse the effect of optimising the parameters – namely the feature sizes – to ascertain the feasibility of using the algorithm for analysis of short sequences of dynamic skeletal muscle. The results have indicated that traditional feature tracking is not an ideal methodology for analysis of dynamic skeletal muscle, and that linear optimisation has little effect on tracking reliability.

The results of tracking a dynamic sequence containing muscle contractions show that feature tracking has limited applications here. Firstly, the results in Figure 21 show that features can drift anywhere up to 50 pixels ($\approx 5mm$) over a 3 minute sequence. Figure 21 and Figure 23 provide strong evidence for trans-planar motion of fascicles, since much of the feature drift seen in Figure 21 is in the superficial part of the muscle. An argument can be made that this is due to poor definition of fascicles in that region however; Figure 23 shows that, even with large amounts of additional speckle noise, features in that region appear to be more stable when there is no motion resulting from contraction. That fact points towards the conclusion that the features in that region either exhibit strong affine transformations like curvatures, and/or those features are moving in and out of the image-plane and can therefore not be tracked. Previous studies have suggested that the fascicles in that region do curve more than the fascicles in any other region in the muscle [32] however; this question has not been resolved further. Whether fascicle curve/warp or leave the image-plane, tracking local linear texture displacements is problematic and inappropriate in the domain of ultrasound.

Analysis of static frames with additive speckle noise reveals another weakness of adaptive tracking as a method of analysis of ultrasound image sequences. Adaptive tracking uses only the current appearance of a feature to track its movement into the next frame, and then the appearance of the feature in its new location is used to track into the following frame, etc… This property of adaptive tracking means that even for static images, the addition of noise causes drift over a lengthy sequence. Figure 22 shows that increasing the noise level causes increased tracking drift. Other feature tracking algorithms (even variations of the KLT) use the appearance of the feature when it is first selected (in the first frame), which would stabilise features if the signal to noise ratio was not too low. While speckle noise would cause features to jitter because the templates would never perfectly match, the algorithm would always use the information in the first frame, making it more likely to stick to well-

defined details above the noise threshold. However, that would be in the ideal case and because these sequences inherently contain nonlinear and affine texture transformations, in order that features can be tracked even when they warp somewhat, an adaptive tracker would be the appropriate choice. This is due to the fact that adaptive tracking allows a feature to change appearance slowly over a sequence.

The main conclusion is that trans-planar motion and/or affine transformations of local image patches make feature tracking a poor proposition for analysis of dynamic skeletal muscle. The results have shown that some regions allow for more accurate tracking of local feature movement than others, which can be attributed to regional variations in fascicle curvature [32], and trans-planar motion (features leaving/entering the image plane). The average feature drift while tracking dynamic muscle is twice that of tracking static muscle with additive noise in the worst case. While a possible approach exists that would allow tracking above the noise threshold (non-adaptive tracking), there is currently no known way to track trans-planar motion, or affine transformations without drift.

Ultimately the aim of tracking skeletal muscle is to interpret the motion field, and typically researchers would like to know how fascicles change during contraction. There are promising studies that aim to directly measure fascicle curvature over the muscle using *a priori* knowledge of the quasi-anisotropic appearance of fascicles [32, 31, 30]. A small wavelet template (15×15) is convolved with the image at discrete angular intervals over every pixel in the image, which reveals the local orientation of fascicles in the 2D plane. This approach is rich because it requires no knowledge of the local orientation of fascicles in previous or successive frames, and is therefore an absolute objective measure of local fascicle orientation. Due to the inherent properties of ultrasound images, errors can and do occur still, these errors are not propagated to successive frames since every measure is objective. More substantial errors can result from this method when erroneous boundaries enter the image plane, which are mistaken for fascicles. The authors present an alternative method which convolves the whole image (muscle region) with an oriented template of striated lines (representing fascicles), at discrete angular intervals. The dominant orientation (maximum convolution) reveals the general/global or dominant fascicle orientation. While this second technique is less susceptible to erroneous local boundaries, important information about local fascicle curvature is lost, which the authors report is critical for understanding the mechanics of muscle.

To conclude, feature tracking is inappropriate for the analysis of dynamic skeletal muscle, and a new approach is required. There are methods for making objective, time-invariant measures of muscle state [32, 31, 30], however these approaches make the assumption that fascicles are the only structure in the muscle region, and that fascicle orientation is the most important factor in understanding the mechanics of muscle. A new technique is required that makes no prior assumptions about the information content of the ultrasound images of muscle. Such a technique should also be robust to the

artefacts of ultrasound and should model the time-invariant states of dynamic skeletal muscle. In the last decade there has been a resurgence of interest in machine learning methods for modelling data without prior knowledge of the structure of the information in the data. For example, a technique called Restricted Boltzmann Machines (RBM) has been used to automatically model the components of handwritten digits and natural images, such that inference can be made about the information content as represented by the model [73, 67]. This technique could be used to model skeletal muscle states including the local and global orientation of fascicles, without any prior assumptions about what information is important and what information the ultrasound is conveying.

By building a model directly from the data, trivial issues like the size of the wavelet convolution kernel for finding fascicle orientations can be optimized to be representative of real data, and not some human interpretation of the data. Convolutional neural networks (CNN) are the state of the art for modelling local features from images however; CNNs throw away important spatial information about features (where the local features are in the image) and spatial information is important for understanding muscle mechanics [32]. Therefore an RBM would be the recommended method of the two. RBMs require a lot of data to build representative models. Acquisition of many images of human calf muscle is somewhat trivial compared to automatically extracting the region of interest. Manual region delineation is often performed in previous studies, which is subjective and time-consuming [30]. In the absence of a good automatic segmentation technique it would not be possible to build a representative model of muscle states due to the large amount of bias that would be included in the model regarding relative muscle orientations and thicknesses over a population. Including an entire image would also mean that the input dimensions would make model building an impractically slow process due to dimensionality of the input vector (an image). Therefore a very accurate segmentation technique is required for the extraction of normalised muscle regions (segmented and fixed image dimension representation of the muscle region). The following chapter presents a proposed ultrasound calf segmentation method and a practical tool for segmentation of entire sequences of dynamic calf muscle.

4 A FULLY AUTOMATIC ULTRASOUND IMAGE SEQUENCE SEGMENTATION TOOL FOR BOUNDARY DETECTION OF THE MEDIAL GASTROCNEMIUS AND SOLEUS OF THE HUMAN TRICEPS SURAE

4.1 ABSTRACT

Ultrasonography has been used to non-invasively analyse the architectural and functional properties of multiple layers of human skeletal muscle. However, many of these techniques require manual definition of intramuscular regions of interest. This process is time consuming and subjective. Automatic identification of regions of interest is challenging due to artefacts pertaining to ultrasound image formation such as shadows, signal dropout, speckle noise, poor contrast, and incomplete or missing boundary gradient information. There has been a recent resurgence of interest in ultrasound image analysis due to improvements in equipment and processing of data resulting in a reduction of many of these artefacts. Recent advances in automatic segmentation techniques, coupled with improved imaging technology makes automatic segmentation of relatively simple (well defined, semi-rigid) dynamic structure from ultrasonography sequences, a realistic and feasible possibility. This chapter presents a Matlab tool for automatic segmentation of the medial gastrocnemius (MG) and the soleus (SO) in the human triceps surae (calf) via b-mode ultrasonography. The tool makes use of shape statistics based on expert annotations of the boundaries of dynamic calf muscle. A global heuristic search is performed in the first frame of a new video sequence, based on a principal component model of spatial muscle configuration. Thereafter, a less exhaustive heuristic search is performed to segment every successive frame, using the segmentation from the previous frame. The initial segmentation is relatively time consuming ($\approx$ 1 minute) however, every successive frame took $< 0.5s$. Segmentation was evaluated on sequential images of dynamic calf muscle obtained from 20 participants, against expert annotations at discrete intervals over lengthy sequences ($>$ 3 minutes). The technique was shown to be accurate to $< 1mm^2$.

4.2 INTRODUCTION

A recent review on ultrasound image segmentation concludes that ultrasound segmentation is a challenging and complex issue [11]; speckle noise, shadows, signal attenuation, signal dropout, and poor contrast are all artefacts of ultrasound images that make manual and automatic segmentation challenging. The authors show that advanced computer vision techniques have led to many previously successful examples of segmentation by region boundary detection, using statistical models of shape and/or grey-level statistics [74, 75, 76]. Shape/texture models are typically constructed from many annotations of many example images. The standard computer vision technique for modelling shape and texture gradients across the shape boundary is the Active Shape Model (ASM; [17]). The main

problem concerning the derivation of a shape model from annotated ultrasound images is the difficulty of the expert annotation process. Segments are usually identifiable by the presence of high contrast/gradient boundaries around the perimeters of individual segments. However, the presence of such well-defined regions in ultrasound images is sensitive to, and dependant on probe placement relative to the region of interest; this includes movement of the segment relative to the probe [11].

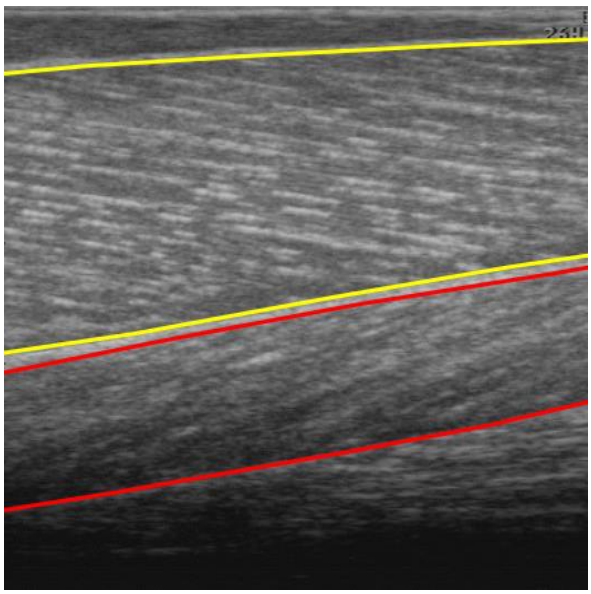


Figure 24. Longitudinal view of the human medial gastrocnemius (MG) and soleus (SO) muscles via b-mode ultrasound (the view represents approximately a $\mathbf{5.94 \times 5}\boldsymbol{cm}$ cross-section). The yellow lines delineate the MG at the boundary of its superficial and deep aponeuroses (top and bottom line respectively). The red lines delineate the SO at the boundary of its superficial and deep aponeuroses (top and bottom line respectively). Within each segment the visible striations represent individual muscle fibres called fascicles, and experts use their orientation or curvature to understand muscle function.

The triceps surae is a functionally complex, multi-layered muscle group [77, 78]. Its architecture with respect to ultrasonography is relatively simple (see Figure 24), and this fact has previously been exploited in an attempt to automate measurements of muscle thickness by searching images of calf for relatively (to the rest of the image) large step changes in texture gradients horizontally across the image [79]. However, the technique used there does not accurately trace the boundary, rather measuring the distance between two straight edges positioned about the aponeuroses of the MG. Their technique also gives no solution to segmentation of the deeper muscle, SO. There is relatively little literature on the segmentation of pennate skeletal muscle via ultrasonography. However, the current benchmark is a technique that uses the ASM paradigm to automatically segment MG and SO [1]. Their technique is validated on a small sample of 8 participants but, the authors report errors within $1mm^2$ for most cases, and they report failure on one of their participant's data. The failure of their technique is attributed to poor definition of SO boundaries. Their approach is to construct grey-level statistics that are machine-specific, therefore one must assume that their method would fail to segment

images recorded on different machines, and even to some extent the same machine with different configuration setting. Furthermore, their technique fails to take advantage of the continuous and consistent appearance of the aponeuroses intensity gradient; instead they sample gradients at discrete intervals along the contour, perpendicular to the segment boundaries.

A better approach should take into consideration the apparently simple structural appearance of MG and SO. The technique presented here uses expert annotations of MG and SO to construct a shape model via Principal Component Analysis (PCA), and a stochastic generative search about the component axes, while aiming to minimise a simple gradient cost function that maximises the positive gradient over the entire segment boundary. This technique could in theory be applied to images generated on different machines, or on the same machine with different configuration settings. The proposed technique is evaluated on sequences of ultrasonography of human dynamic skeletal calf muscle obtained from 20 participants.

4.3 METHODS

This section describes the methods used to create a model of calf muscle shape, the development of a stochastic search algorithm, and a simple cost function for automatic segmentation of MG and SO from ultrasound images. First, the data collection is briefly described, leaving out much of the details about the experiment since they are covered in a later chapter (chapter 5, section 5.3). Then the image annotation protocol is described, which was used to supply the statistics about calf muscle boundary shape. Then, the procedure for modelling calf muscle boundary shape via PCA is explained in detail. Finally, the stochastic segmentation algorithm is described in detail, along with the definition of a suggested gradient cost function for evaluation of segmentation fitness during the stochastic search.

4.3.1 DATA COLLECTION

In order to build a comprehensive model of the shape boundaries of MG and SO that generalises across a population and in different functional conditions, data collection of representative ultrasonography sequences was required. Ultrasound image sequences of the calf were acquired from 20 consenting human participants (details in chapter 5, section 5.3), which contained controlled dynamic contractions with combined controlled joint rotations over a 3 minute period. These sequences encapsulated a wide functional range of changes in calf muscle thickness and curvature in the aponeuroses. This dataset also contained a wide inter-participant variation in relative MG and SO muscle depth and thickness, all of which provides good data for modelling the shape and appearance of these muscles. The following subsection describes how this data were annotated by an expert operator in order to provide shape statistics for the model.

4.3.1.1 Ethical Approval

These experiments were approved by the Research Ethics Committee of the Faculty of Science and Engineering, Manchester Metropolitan University (MMU). Participants gave (written) informed consent to these experiments, which conformed to the standards set by the latest revision of the Declaration of Helsinki. Experiments were performed at the Cognitive Motor Function laboratory, in the School of Healthcare Science at MMU.

4.3.2 IMAGE ANNOTATION

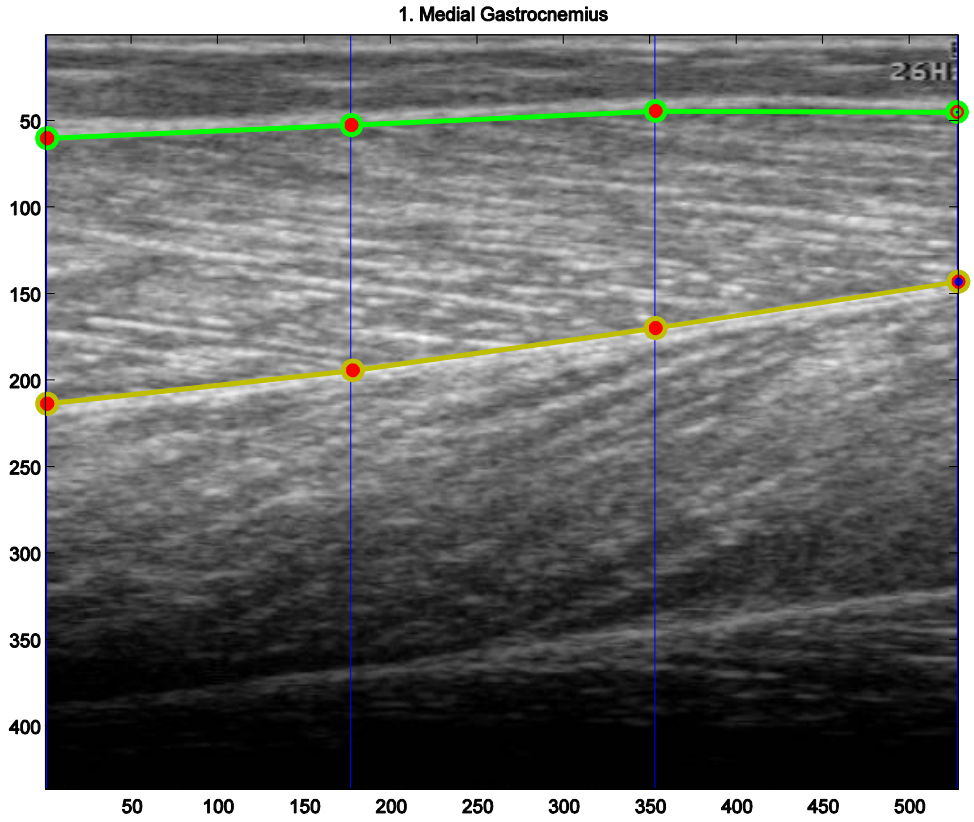


Figure 25. Image annotation GUI developed for use with Matlab. The application allows a user to mark 4 points at fixed horizontal intervals for the upper and lower boundaries of each muscle. The image shows an annotation for the boundaries of the medial gastrocnemius. The red dots show the points marked by the expert, and the green lines show their connectivity. The blue lines show where the horizontal position of each point is fixed.

A human expert annotated ultrasound images, identifying the contours of the following 2 regions in every participant (ordered *superficial* to *deep*):

1. Medial Gastrocnemius
2. Soleus

To regulate the annotation process, a simple GUI was implemented (see Figure 25) using Matlab's keyboard and mouse function window callback interface [80]. Functions were developed that allowed a 2 segment mark-up of 10 fixed, equidistant temporal intervals in each video sequence. The expert

was instructed to mark from left to right, just inside the aponeuroses of each muscle segment. To remove horizontal variance, the expert was restricted to marking 4 points per aponeuroses at fixed, equidistant horizontal intervals across the image. The expert was allowed to cycle through each segment, selecting erroneous markings for removal and subsequent replacement. The ultrasound images and their annotations were subsequently saved. A total of $20\ participants \times 10\ images = 200$ frames were marked up by the expert. All annotations were recorded in millimetres to form an accurate anatomical model that will generalise easily to images from other machines if the measurement area is known on that image.

4.3.3 Modelling Calf Muscle Shape with Principal Component Analysis

The Active Shape Model (ASM) is a powerful paradigm for capturing the average shape and variance of an arbitrary segment [17]. A variation of the ASM is used here to create shape statistics for use with the proposed stochastic segmentation algorithm (explained in the next section). The standard approach for building an ASM is to model the variance of points in a set of shapes using PCA. Before this can be done, shapes in the annotated set must be spatially aligned using a procedure called Procrustes analysis; minimising the Euclidean distance between contours of shapes in a set by linear transformations. Procrustes analysis requires that the annotated points in each example shape correspond to the same points in every other shape (these points re called 'landmarks'), so that – during alignment – the distance between those points can be minimised. However, in this case the expert to generate zero variance in the *x*-plane, meaning that the shapes are already aligned in the *x*-plane. This substantially simplified the Procrustes procedure by reducing the alignment problem to a linear translation in the vertical plane. Vertical muscle boundary alignment means positioning the boundaries of each muscle such that all points across boundaries of different muscles are minimally distant from each other without scaling (i.e. removing the difference between mean vertical point positions).

4.3.3.1 Shape Alignment

In order to model the variance of shape contours from a set of shapes, all shapes in the set must be aligned. Before shape alignment a subset of shapes belonging to 10 carefully selected[4] participants was extracted from the set of all annotated shapes, and the mean shape of that subset was computed. That subset – termed the 'training set'– consisted of shapes, carefully selected to encapsulate a wide range of motions and relative muscle appearances. The vertical average of the boundary points of the mean shape, and all shapes in the training set were computed. Then, the differences between the averages of the mean shape and all shapes in the training set were computed. Finally, the respective differences

[4] For clarification, the term "carefully selected" refers to the selection of patrticipants which would make a good training set for a chape model, which involved the selection of participants with thick muscles, participants with thin muscles, and participants with a range of orientations and curvatures of deep and superficial aponeuroses. Participants were *not* selected based on segmentation results on the testing set.

were subtracted from every shape in the training set, which resulted in vertical alignment of muscle contours. Those differences were recorded for later use. See the following figure (Figure 26) for an illustration of this process.

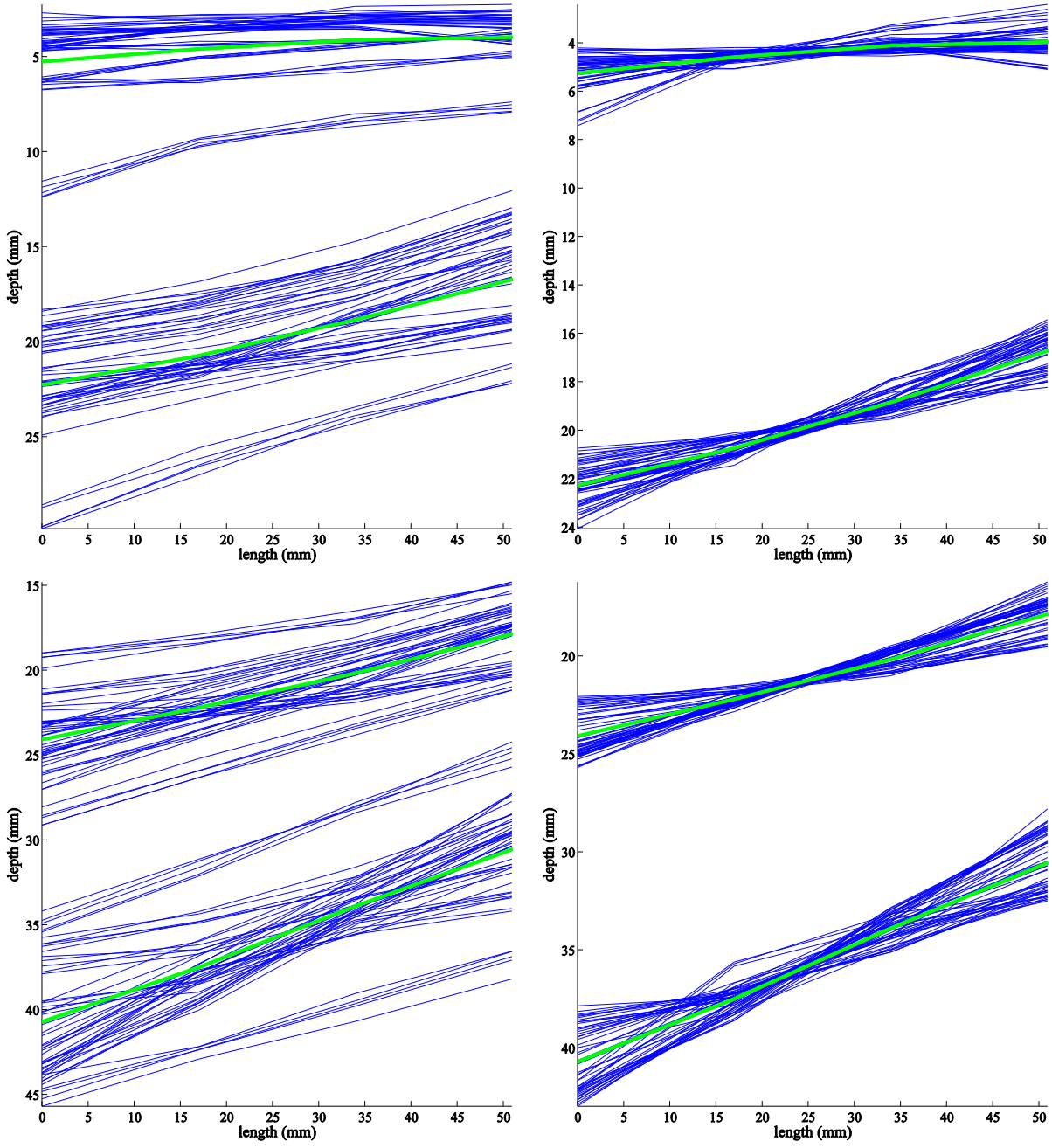


Figure 26. Shape alignment for MG (top) and SO (bottom) muscles. The blue lines represent 5 annotations of superficial and deep aponeuroses per 10 participants. The green lines show the mean shape of each muscle. The figures on the left show all shapes before vertical alignment to the mean shape. The figures on the right show all shapes after vertical alignment to the mean shape. It is clear from the images on the right that there is actually very little variation in the orientation of aponeuroses however, there appears to be a fair amount of variation in the thickness of both muscles.

4.3.3.2 Shape Modelling with PCA

After shape contour alignment, a model of contour variance can be constructed using PCA. With PCA the main vectors (components) of contour variance can be captured as a set of linear components. The principal components of a set of shapes are equivalent to the eigenvectors and eigenvalues of the covariance matrix of that set. In Matlab the covariance matrix is given with,

$$shapeCov = cov(shapesY),$$

where $shapeCov$ is the 8×8 covariance matrix of all vertical coordinates of every contour point of every shape in the training set. $shapesY$ is a matrix of all vertical boundary points of all shapes, where each row is a set of points corresponding to one shape. Also in Matlab, the eigenvectors and corresponding eigenvalues are given with,

$$[\lambda, v] = eig(shapeCov),$$

where λ is a set of eigenvectors (components), v is a set of associated eigenvalues (variances), of the covariance matrix $shapeCov$. The principal components, together with the mean shape constitute the shape model. A second component model can be constructed from the stored shape alignments (see section 4.3.3.1) using exactly the same method. First the covariance matrix of upper and lower boundary displacements can be constructed with,

$$alignCov = cov(alignY),$$

where $alignCov$ is the 2×2 covariance matrix of all vertical displacement coordinates for upper and lower boundaries of every shape in the training set. $alignY$ is a matrix of all vertical boundary displacements of all shapes, where each row is a set of 2 points corresponding to the upper and lower vertical boundary displacements for a single shape, respectively. The eigenvectors and corresponding eigenvalues are given with,

$$[\lambda, v] = eig(alignCov),$$

A separate shape model was constructed for each muscle such that the variance of each shape was captured independently, resulting in extra modes of variance in the training set. The following subsection shows how these models can be used to generate hypothetical shapes that can be evaluated for fitness against an arbitrary calf muscle image.

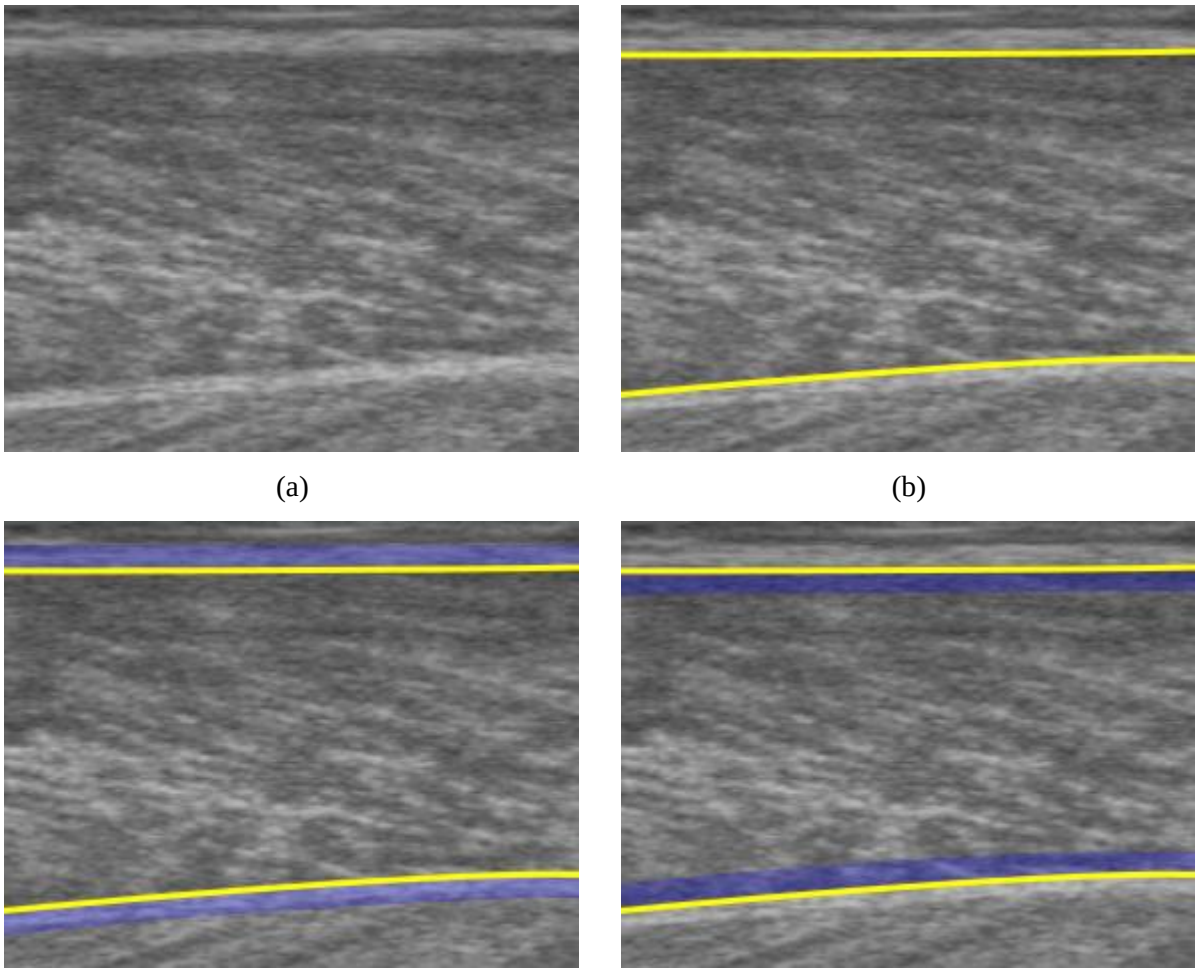
4.3.4 SEGMENTATION ALGORITHM

According to Cootes, and others [17] a good statistical model of shape is capable of generating realistic examples by adding weighted linear combinations of eigenvalues to the mean shape with,

$$S = \mu + \sigma \sum_{i=1}^{i=n} \lambda_i v_i N(0,1),$$

Equation 2

where S is a randomly generated shape (vector of 8 boundary points), μ is the mean shape (vector of 8 boundary points), λ_i is the eigenvector of the component i, v_i is the corresponding eigenvalue (variance) and $N(0,1)$ is some normally distributed random noise with 0 mean and unit variance. n can be an integer from 1 to the number of components in the model, which in this case is 8. σ is a variance control parameter, which can be used to increase or decrease the proportion of variation in possible shapes/locations that can be generated. Increasing σ potentially produces shapes that are further away from the statistical mean, and decreasing it (towards zero) produces shapes that are closer to the statistical mean. Equation 2 can also be used to generate a random location for the upper and lower shape boundaries within the image. The advantage of separately generating a shape and a location for that shape is that the operator can control the amount of variance separately. This means that if unusually large/small muscles are encountered that the algorithm is not able to segment initially, the σ parameter of Equation 2 can be increased/decreased for the location model, independently of the contour variance. This allows a wider search for the boundaries of unusually large/small muscles while prevents fitting of erroneous boundaries (boundaries outside the statistics of the shape model).



(a) (b)

Figure 27. Shows a typical medial gastrocnmeius grayscale segment (a) with a generated contour overlay (b), an outer area (c), and an inner area (b). The difference between outer and inner mean pixel intensities (within the blue highlighted areas) is measured.

The process of generating a hypothetical shape and a hypothetical location for that shape can be used to search for muscle boundaries within the grey-scale of the image. It is known *a priori* that the boundaries of skeletal muscle (aponeuroses) appear as bright bands of pixel intensity (approximately 15 pixels in thickness), relative to the internal appearance of the muscle, and this is due to the acoustic impedance properties of muscle and aponeuroses (see chapter 1, subsection 1.3.1). To measure the goodness of fit for some boundary hypothesis, a cost function can be created that will minimise with the largest intensity difference between the inner and outer boundary of the aponeuroses (see Figure 27). The cost function is as follows,

$$C(S,I) = \sum_{i=1}^{i=n} (\mu_1 - \mu_2),$$

Equation 3

where $C(S, I)$ is the cost of the fit of shape S, on the image I, n is the number of segments, and μ_1, and μ_2 are the (mean normalised) average (mean) inner and outer pixel intensities, respectively. The minimum of this function therefore locates the shape S with respect to the image I, with the greatest relative positive difference between the outer pixel region and the inner pixel region, within the constraints of the shape and location models. See Figure 28 for a flow diagram of the segmentation process.

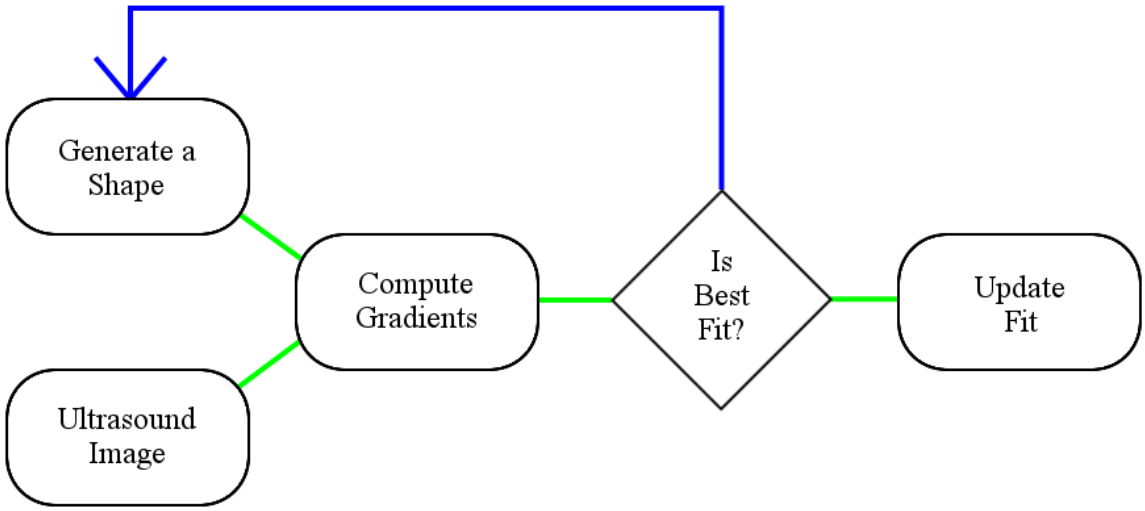


Figure 28. Flow diagram from left to right: Shapes are continually generated from the model and evaluated against the ultrasound image gradients. If by some arbitrary measure of the gradients the fit is considered the best fit in a list of previous fits, then that fit is taken as the current best result. This process can run up to some maximum number of iterations in serial, or in parallel, since no random shape evaluation depends on any other shape evaluation. For parallel execution the best fit is determined from a list of costs and associated shapes after execution.

It is possible to continuously generate shapes and locations from the respective component models, evaluating each shape for fitness according to Equation 3 (see Figure 28). That algorithm can be used to segment an initial frame with an exhaustive stochastic search (5000 maximum iterations), generating many shapes and taking the best fit according to Equation 3. Thereafter the resulting segmentation of each successive frame replaces the mean shape in Equation 2 for shape/location generations in each additional frame. Additionally, the σ coefficient can be reduced to perform a less exhaustive local search (25 maximum iterations). This has the effect of allowing a generalised image-wide search in the first frame, and then a specialised local search in each successive frame, which is important for obvious practical reasons.

4.3.5 VALIDATION

The segmentation algorithm was used to segment MG and SO in every frame of the ultrasonography sequences of all 20 participants. The following algorithm and model parameters[5] were used for every participant:

Instance	Parameter	Value
First Frame	Maximum Iterations	5000
	Shape Model σ	3
	Location Model σ	3
Frame #2 onwards	Maximum Iterations	50
	Shape Model σ	0.5
	Location Model σ	0.5

The resulting automatic segmentations were compared only in the frames that were annotated by the expert (see section 0), which spanned 5 equidistant intervals from the first frame to the last. Results are evaluated on the following terms:

1. **Percentage accuracy**

$$P = \frac{S \cap A}{\mu(A)} \times 100,$$

Equation 4

where P is the proportion of the area of the expert annotation A that intersects with the area of the automatic segmentation S, expressed as a percentage, and $\mu(A)$ is the area of the expert annotation.

2. **Area Classification Error (mm^2)**

$$E = \mu(A) - \mu(S \cap A),$$

Equation 5

where E is the area of misclassified regions in square millimetres, computed as the difference between the total area of the expert annotated segment $\mu(A)$, and the total area of the intersection between the automatic segmentation segment and the expert annotated segment $\mu(S \cap A)$. $\mu(x)$ computes the area of an arbitrary polygon x.

[5] These parameter values were selected initially to permit a global search for shapes within 3 standard deviations of the mean shape, and a local search for shapes within 0.5 standard deviations of the mean shape. Since they showed empirical success, no further investigation of these parameters was required.

4.4 RESULTS

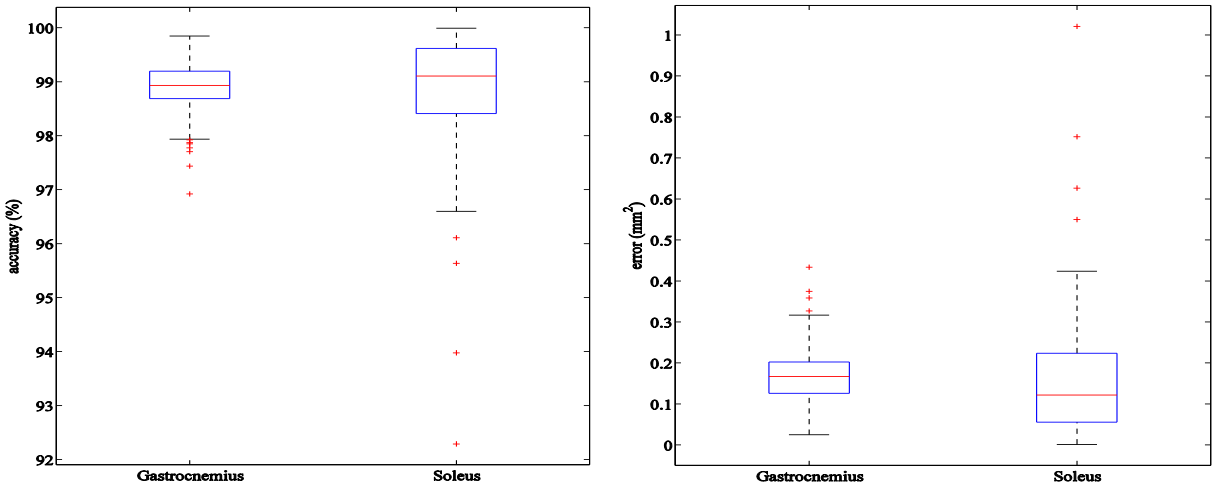


Figure 29. Segmentation results for MG and SO for all annotations. Left shows the percentage accuracy given by Equation 4. Right shows total area of misclassified regions in millimetres, given by Equation 5.

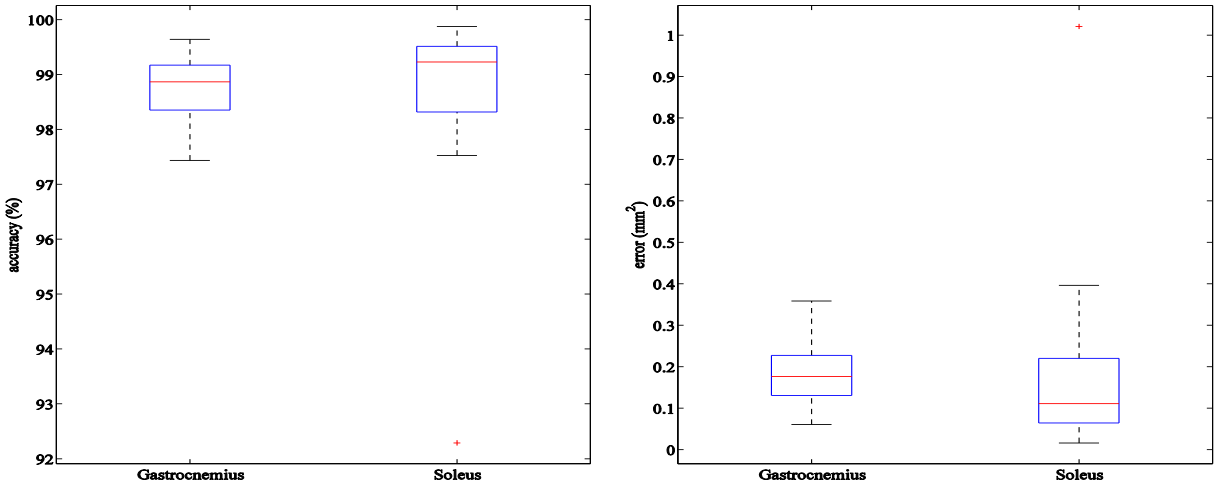


Figure 30. Segmentation results for MG and SO for only first frame annotations. Left shows the percentage accuracy given by Equation 4. Right shows total area of misclassified regions in millimetres, given by Equation 5.

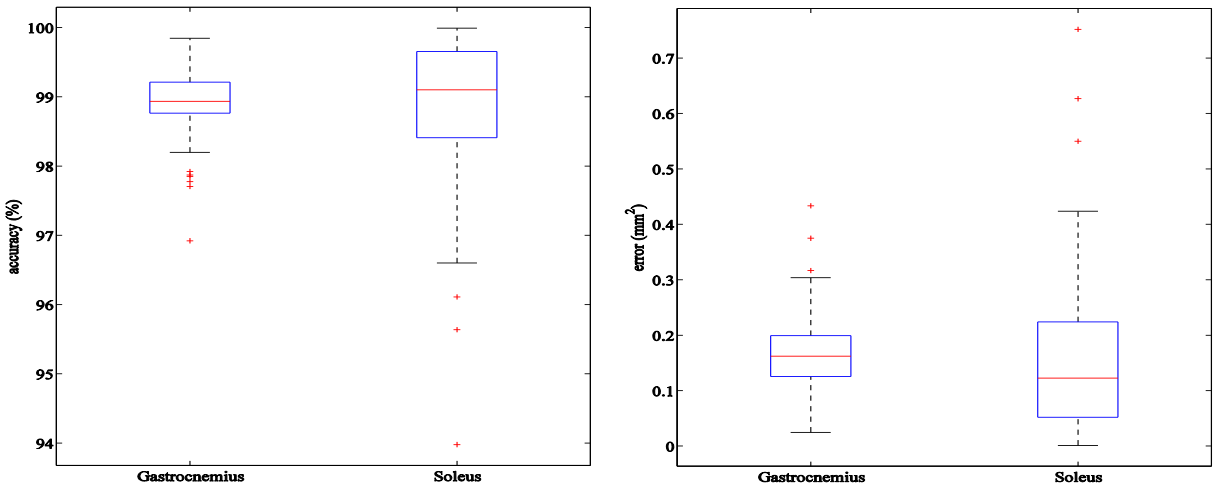


Figure 31. Segmentation results for MG and SO all annotations after the first frame. Left shows the percentage accuracy given by Equation 4. Right shows total area of misclassified regions in millimetres, given by Equation 5.

The results given in the 3 figures (Figure 29, Figure 30, and Figure 31) show a marked improvement on the previous technique [1]. The authors report an average accuracy of approximately 0.3 millimeteres, with errors up to 6 millimetres in some cases. They also report a number of incorrect initialisations via their automated initial frame segmentation. Furthermore, they only evaluate their results on a pool of 8 people, all of whom are included in the model building process. In contrast, the method presented here is evaluated on 20 people (10 held out) and shows good overall segmentation (see Figure 29) with average (median) errors of $\approx 0.16mm^2$ for MG and $\approx 0.12mm^2$ for SO, good initial frame segmentation (see Figure 30) with average (median) errors of $\approx 0.17mm^2$ for MG and $\approx 0.11mm^2$ for SO, and stable sequential segmentation (see Figure 31) with average (median) errors of $\approx 0.16mm^2$ for MG and $\approx 0.12mm^2$ for SO. The average (mean) segmentation time for each frame at 50 iterations was ≈ 0.9 seconds. The figure below shows further evidence that the tracking of muscle segments appears to be smooth and accurate over time.

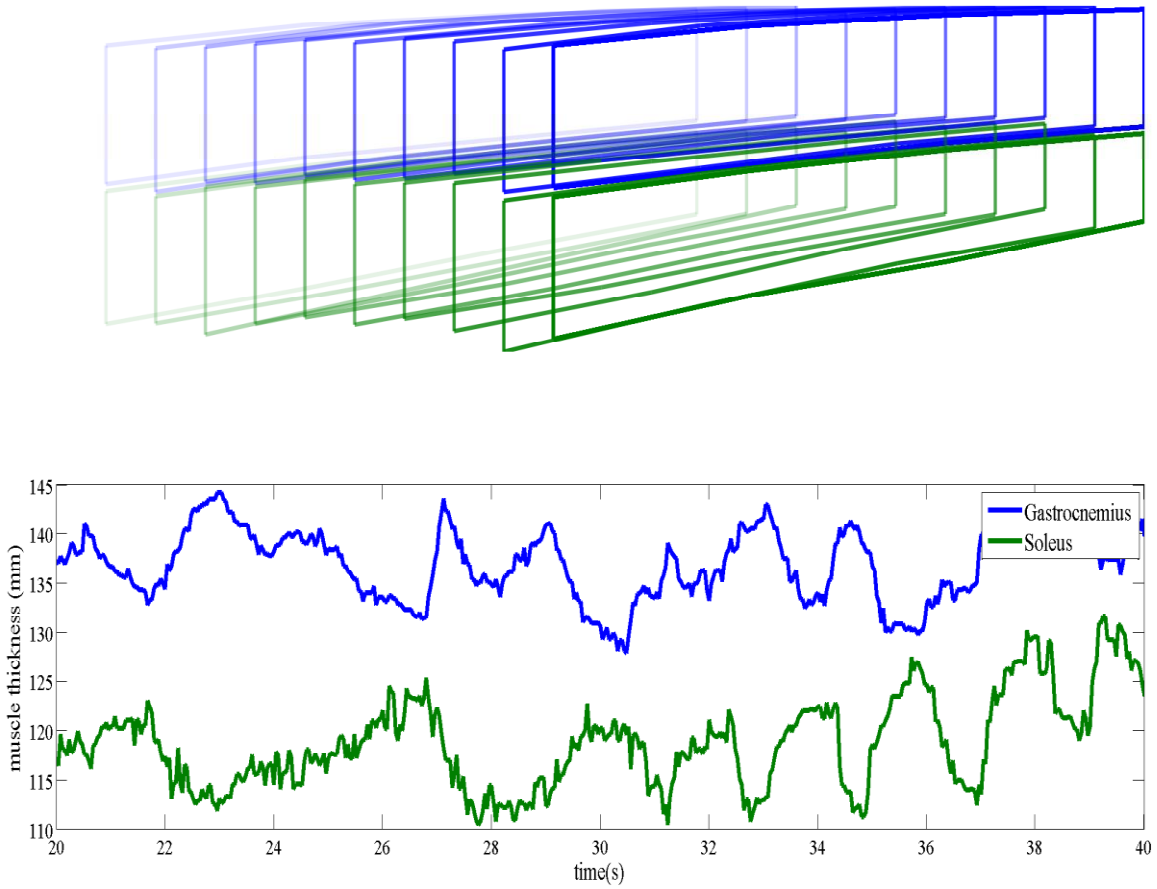


Figure 32. Time series of automatic calculation of muscle thickness for a representative case. Top: each shape from left to right represents the shape of MG and SO at 2-second intervals between 20 seconds and 40 seconds corresponding to the time series in the figure below. Bottom: muscle thickness (mean distance between aponeuroses) for each muscle over time. This figure shows how MG and SO can change thickness independently during dynamic muscle movements.

4.5 DISCUSSION & CONCLUSIONS

This study has presented an automatic human calf muscle segmentation tool, capable of detecting the boundaries of the gastrocnemius and soleus muscles as defined by the aponeuroses of those muscles. Results have shown that the tool operates at a very practical temporal frame rate of less than 1 second. Comparisons with expert annotated images show that the segmentations are more accurate than the previous benchmark (average errors of $0.3mm^2$ [1]), showing average errors of less than $0.2mm^2$, and this result was shown to be stable of a lengthy sequence (see page 94). This study has presented a possible modelling protocol and segmentation solution to automatically segmenting any such muscle group similar in appearance to the triceps surae via ultrasonography. Such a tool would allow full automation of powerful techniques for quantification of the internal muscle states [31, 30, 32, 1, 34]. This in turn would allow extensive research and testing over thousands of images and/or sequences without the need for expert annotation.

The tool is presented as a serial solution however, the algorithm for segmenting images is atomic in nature, and therefore could be parallelised. Parallelisation means the possibility of real-time segmentation, which would provide operator feedback, or biofeedback of skeletal muscle thickness to patients undergoing physiotherapy for example. This optimisation is a trivial matter of implementing the stochastic routing on a Graphics Processing Unit (GPU), which has many parallel compute cores for executing thousands of routines synchronously [81]. The tool is also presented as a collection of Matlab scripts and functions, which are by no means efficient. A valid suggestion would be to implement the code in a more efficient language such as C++, and further optimisation could be achieved via multi-core processing (which Matlab already does to some extent) however, this is less trivial.

In summary, this is the first generic solution for segmentation of the human triceps surae. The technique is robust and accurate and may have an application to more complex muscle groups that have many more segments and much more relative muscle variability such as the muscle in the human posterior neck. The grey-scale contour cost function has the inherent property that the tool may be applicable to ultrasound images of the triceps surae that were collected and processed on a completely different machine to which the statistical model was created from, as well as the same machine with different configuration settings. The statistical model has anatomical dimensions (mm), and the cost function aims to maximise the intensity gradient about the aponeuroses, therefore, if the dimensions of the muscles are known for any given ultrasound image, then theoretically the same cost function would ideally be minimised. This makes this tool a very practical segmentation solution, since not all researchers and clinicians use the same imaging equipment. Full code and brief usage instructions for this tool are given in Appendix 8.6. This tool can now be used to accurately segment thousands of images of human calf muscle, which can then be used to build a model of muscle states.

5 Modelling The Time-Invariant States of Human Calf Muscles from Ultrasonography Sequences via Restricted Boltzmann Machines

5.1 Abstract

Ultrasonography has been used to non-invasively measure human-intelligible parameters from layers of skeletal muscle. Previous research has been able to establish connections between these parameters and non-ultrasound measured parameters (force exertion, joint angle, and aggregated motor unit activity). However, these relationships have only been established in functionally isolated conditions; isometric contractions, or passive joint modulation. Intuitively, linear changes in parameters such as fascicle pennation angle correlate reliably with force exertion in the isometric case, and joint angle in the passive case. A non-linearity (or de-correlation) presents in the combined case of modulating joint angle and modulating contraction. On the question of information content of ultrasound, this chapter presents preliminary work on the use of a powerful feature extraction method, namely Restricted Boltzmann Machine (RBM), for modelling the time-invariant states of human calf muscle in combined and isolated cases, without making prior assumptions about the information content. Linear regression models were then used to establish relationships between the RBM model and the externally measured ankle joint angle, force, and surface electromyography (sEMG). Results indicate that RBMs can learn powerful feature representations of muscle states by demonstrating the presence of automatically extracted features that represent independent joint angle, sEMG, and force over a sequence. Results also indicate that it is possible to separate out combined activities directly from the ultrasound by optimising weighted linear combinations of the RBM feature representation, in a comprehensive model of isolated and combined functional conditions.

5.2 Introduction

The human calf (specifically the triceps surae) is a 2 layer skeletal muscle system made up of 3 muscles; medial/lateral gastrocnemius (superficial), and soleus (deep). These muscles allow coordinated control of the ankle joint during musculoskeletal functions such as balance, jumping and gait [82, 83, 84]. The internal structure of each muscle is architecturally and functionally complex [77, 78], made up of bundles of muscle fibres called fascicles. Each muscle layer attaches to different bones and each muscle has a different neural innervation and a different internal architecture (variations in density and organisation of fascicles), and therefore is functionally independent; this independence has been confirmed by measured changes in muscle parameters such as fascicle length/orientation during passive ankle joint rotation, and isometric contractile activity [85, 86, 87].

Muscle states are a function of multiple different factors such as neural excitation and passive joint rotation (see Figure 33). Neural excitation causes active internal generation of force by excitation/contraction mechanisms, and passive joint rotation causes stretch and force (tension) within muscle. Other lesser factors including contact forces (e.g. drag, pressure) from adjacent muscles also contribute to muscle states. These factors yield measurable change in muscle states such as muscle activity (EMG), muscle length (joint angle), muscle force (tension). Therefore, these measurable quantities are fundamentally related to muscle states. Muscle states also include unmeasured quantities such as local strain, pressure and tension and relate to the changes occurring at the tissue and molecular level.

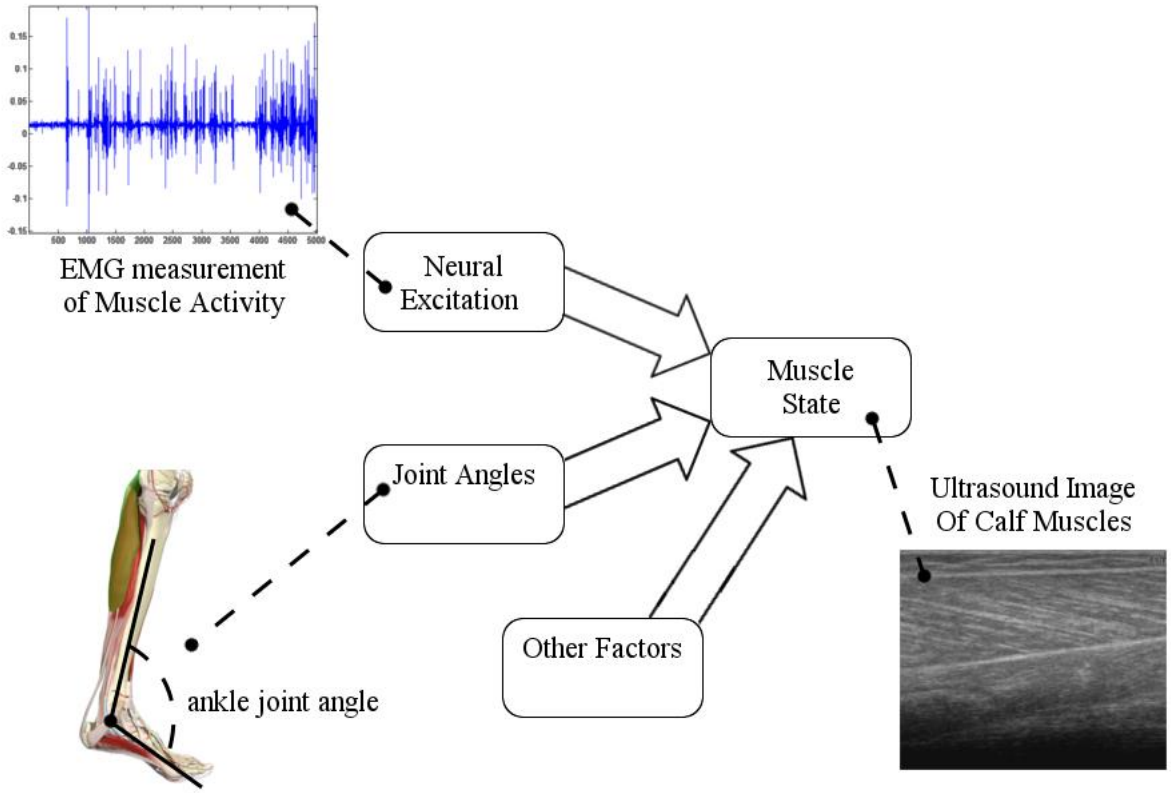


Figure 33. Inputs to muscle states. Neural excitation, joint angles, and other less influential factors are inputs which determine the internal state of muscle. Neural excitation causes electrical activity in the muscle which is measurable by EMG. Neural excitation leads to contraction of fibres/fascicles which can lead to changes in muscle length. Joint angles are measurable by fixing joints to devices with known or measurable joint angles, or by using body markers and a motion capture system. Changes in joint angle can lead to changes in muscle length. All of these inputs can vary independently of one another. Ultrasound is an imaging technology which allows direct observation of the internal state of the muscle. Do the visible internal muscle states uniquely determine the individual contribution of the inputs?

One way of observing muscle states is ultrasound, which provides a planar image of muscle shape and texture. An ultrasound image of skeletal muscle can reveal the shape of the internal architecture of muscle, determined by fascicles, which produce contractile force and change shape (curve) and length

resulting from active internal force generation (via neural innervation) and/or joint rotations [88]. Measurements of fascicle curvature via ultrasonography have been shown to be reliable and accurate [89], although their reproducibility from images acquired from the same subject on different days has been shown to be lower [90]. Numerous studies have shown that fascicle lengths and curvatures can be automatically quantified *in vivo* via b-mode ultrasonography [30, 32, 34, 31, 91]. However, these methods primarily focus on the superficial muscles, possibly in part because of the difficulty in imaging the deeper muscles, and the lack of well-defined fascicles. Furthermore, these methods assume the appearance of well-defined fascicles and their persistent definition throughout an ultrasound sequence of active muscle contraction and/or passive joint rotation.

While fascicle curvature has been linked to internal muscle force [88], research has demonstrated that the relationship between changes in fascicle length/curvature, joint angle, and plantar flexion (increase in ankle joint angle) force is nonlinear for contractions at different joint angles [92]. Ultrasound allows the possibility for measurement of muscle shape and motion parameters like muscle thickness, fascicle shearing, pennation angle, fascicle curvature/length. However, these primitive measurements are low-dimensional and therefore have a limited information bandwidth with respect to inverting the nonlinear information pathway that presents in functional conditions containing a mixture of inputs (neural innervation and joint angle rotation). While local fascicle motion [1] and local fascicle curvature [30] can provide a high-bandwidth information stream with which to invert the pathway, these methods of analysis either exhibit time-varying signal drift in the case of tracking, or spurious curvature measures in the presence of erroneous edges gradients mistaken for fascicles. Feature extraction methods such as the Restricted Boltzmann Machine (RBM; [54]) may provide a workable solution. RBMs build density models of the natural descriptive features of raw data without human intervention (i.e. unsupervised learning paradigm).

The primary objective of this work is to ascertain whether function-specific information (joint angles, force, and sEMG) can be extracted from ultrasound of skeletal muscle in functional conditions which contain a mixture of inputs (neural innervation and joint angle rotation). This study explores the use of a generic feature learning technique, namely RBM, for modelling the functional states of medial gastrocnemius and soleus, as represented by sequential ultrasound images. RBMs have been used to model the descriptive features of handwritten digits (without human intervention). The features of that model were then used to successfully classify handwritten digits at over 98% accuracy [73] using only a small amount of labelled data to optimise (weight) the features towards predicting the labels. The use of an RBM to automatically extract descriptive features of muscle states removes the human-biased assumption that fascicle curvature is the only determining factor of joint angle and muscle force. The use of an RBM would also allow modelling of soleus muscle states in absence of well-defined

fascicles, and provides a time-invariant representation of muscle states, which avoids any possibility of measurement drift.

In order to use an RBM to encode the complex variety of combined and isolated skeletal muscle states, a comprehensive dataset is required. That dataset should contain many examples of changes in muscle state determined by, isometric contraction, passive ankle rotation, and combined ankle rotation with active contraction. For each combination of inputs there should be an associated, synchronised ultrasound image of gastrocnemius and soleus. Referring to Figure 33, this dataset would attempt to build a data space where all possible inputs (neural innervation, ankle joint rotation) that determine muscle states are induced and observed via ultrasound. Any combination of inputs in that data-space will be associated with an ultrasound image of muscle state that was induced by those inputs. Then if an RBM could build a good density model of those states, linear models might provide a way to reproduce the inputs that caused those states in such a way that is time-invariant, and no assumptions are made about the descriptive content of the ultrasound.

The following section details the design of an experiment in which the goal is to construct a comprehensive dataset of images and associated ankle joint angle, sEMG, and force, during isolated and combined muscle function. This is done by designing two command signals; one signal will be used to rotate the ankle joint within a defined range at multiple frequencies and amplitudes, while the other signal will provide a target for participants to match against integrated feedback of EMG from their calf muscles. Then an expanded explanation of the RBM architecture and learning algorithm is given for the benefit of the reader. The remainder of the section gives an analysis of the features of the trained RBM in relation to externally measured muscle states. This is followed by an empirical evaluation of linear models for predicting skeletal muscle states from sequences of RBM features containing combined and isolated changes in functional states.

5.3 METHODS

In order to model the appearance of skeletal muscle via ultrasound under active and passive shape changing conditions, a comprehensive dataset was required. That dataset should contain examples of multiple frequency/amplitude joint rotations and active contractions, both in combined functional cases and in isolation. This required the design of a pair of signals; one of which was used to modulate the ankle joint angle via foot pedals, and the other was used along with feedback of muscle activity via sEMG to guide participants to produce a variety of contractions. During the design of the signals, the main considerations were to include different frequencies and amplitudes of contraction and joint rotation in order to maximise the distribution of combined joint rotation and active contraction. These considerations were important for ensuring the acquisition of a comprehensive model of muscle states. These signals provided the input for 3 trials designed to exploit active and passive components of shape change in isolated and combined conditions.

5.3.1 SIGNAL DESIGN

Two types of signal were designed in order to cover a range of frequencies and amplitudes over a 3 minute period. The first signal was designed to provide oscillations and gradual transitions between high and low frequency sine waves. The second signal was designed to provide gradual transitions between amplitudes via the combination of different frequency sine waves. For both signals, a second version was created such that there were temporary correlations, de-correlations, and anti-correlations between both versions over the 3 minute period. A version of each signal was used to control the angle of a foot pedal system on which the participants were standing. This allowed controlled passive manipulation of the ankle joint angle. The second version was displayed on a screen adjacent to integrated feedback of the participants' sEMG measurements. This was used to guide the participants in producing controlled contractions by matching the amplitude of their sEMG with the amplitude of the displayed signal.

The amplitudes of all signals used to control the joint angle were optimised in order to produce angular ankle rotations within the range of $-10°$ to $20°$ (where $0°$ is the neutral standing position, and positive and negative angles represent plantar flexion and dorsiflexion respectively). The amplitude of each participant's sEMG feedback signal was optimised such that the maximum amplitude of sEMG at 50% Maximum Voluntary Contraction (MVC) was roughly aligned with the maximum amplitude of the target signal, and the minimum amplitude of sEMG at 0% MVC was roughly aligned with the minimum amplitude of the target signal. MVC tests took place before data collection once for each participant. A detailed description of the design of these signals is given in the following sections.

5.3.1.1 Modulating Frequency Signal

A modulating frequency signal was constructed from a high frequency (0.4Hz) sine wave and a low frequency ($0.125Hz$) sine wave (see gain blocks 1 and 2 in Figure 34). In 20 second pulses (see pulse block A in Figure 34), occurring at regular intervals for predetermined periods of time, the signal switches between sine waves. Also in pulses (see pulse block B in Figure 34) but less frequently, the signal will introduce a gradual transition between $0.125Hz$ and $0.4Hz$ and back to $0.125Hz$ over a 30 second period. This signal was used to actuate the motors on a foot pedal device that modulated the ankle joint angle.

To create an accompanying signal for muscle activity guidance, the same frequencies were used but, the pulse blocks were modified such that different frequencies would overlap to create temporary correlations, de-correlations, and anticorrelations. The second signal was presented as a dot on an oscilloscope, adjacent to another dot which represented integrated sEMG feedback of muscle activity. This signal was then used to provide participants with a target with which to match their sEMG. A Simulink (Matlab 2013a, Mathworks) model was designed to implement both signals, such that the model could be used for real-time data acquisition and signal output.

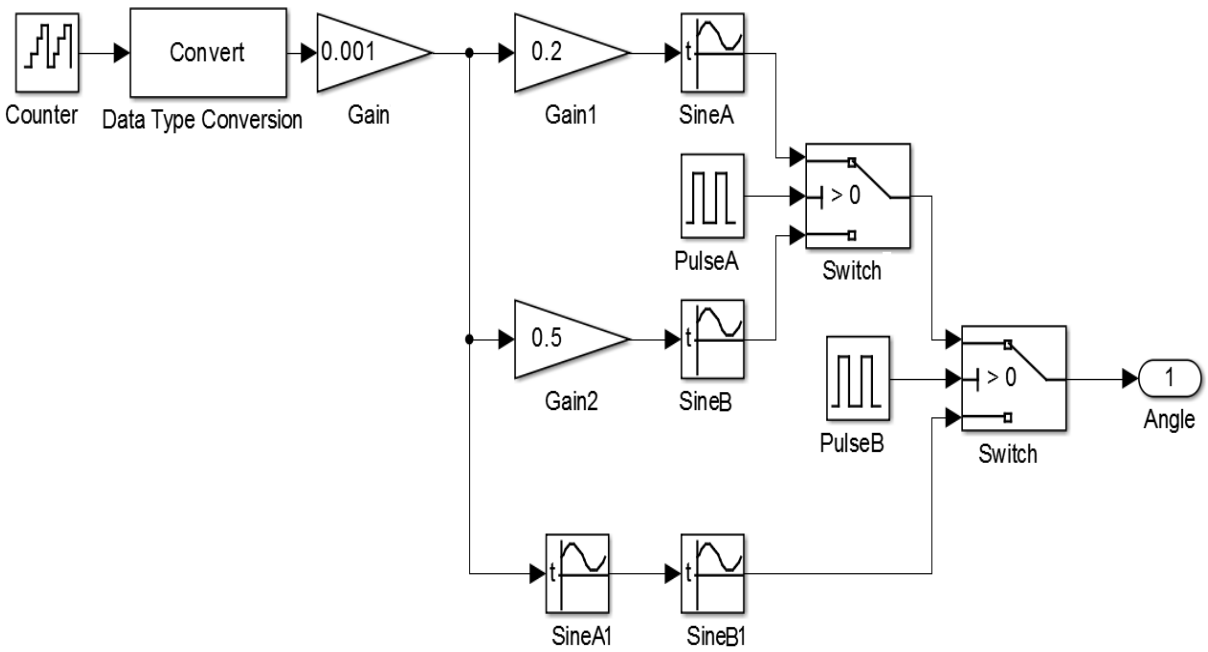


Figure 34. Modulating frequency signal: Simulink diagram. This illustrates the signal design for modulating the angle of the foot pedals at varying frequencies swapped out for one another by pulse activated switches.

Table 2. Sine Block Parameters (Pedal Angle). This table lists the parameters of the four sine blocks from Figure 34.

	Amplitude	Bias	Frequency (rad/s)	Phase (rad)
SineA	$\frac{1}{2}$	$\frac{1}{2}$	2π	$\frac{-\pi}{2}$
SineB	$\frac{1}{2}$	$\frac{1}{2}$	2π	$\frac{-\pi}{2}$
SineA1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{\pi}{30}$	$\frac{-\pi}{2}$
SineB1	$\frac{1}{2}$	$\frac{1}{2}$	30π	$\frac{-\pi}{2}$

Table 3. Pulse Block Parameters (Pedal Angle). This table lists the parameters of the two pulse blocks from Figure 34. These parameters are used to drive the pedal angle.

	Amplitude	Period (s)	Pulse Width (% period)	Phase delay
PulseA	1	20	50	0
PulseB	1	60	50	0

Table 4. Sine Block Parameters (sEMG Command). This table lists the parameters of the four sine blocks from Figure 34. These parameters are used to guide participants' sEMG amplitude.

	Amplitude	Bias	Frequency (rad/s)	Phase (rad)
SineA	$\frac{1}{2}$	$\frac{1}{2}$	2π	$\frac{-\pi}{2}$
SineB	$\frac{1}{2}$	$\frac{1}{2}$	2π	$\frac{-\pi}{2}$
SineA1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{\pi}{30}$	$\frac{-\pi}{2}$
SineB1	$\frac{1}{2}$	$\frac{1}{2}$	30π	$\frac{-\pi}{2}$

Table 5. Pulse Block Parameters (sEMG Command). This table lists the parameters of the two pulse blocks from Figure 34.

	Amplitude	Period (s)	Pulse Width (% period)	Phase delay
PulseA	1	40	50	0
PulseB	1	180	50	0

5.3.1.2 Modulating Amplitude Signal

A modulating amplitude signal was constructed from a high and low frequency sine waves. The sum of a $1Hz$ sine wave and a $\frac{2}{3}Hz$ sine wave composed the initial 90 seconds of the signal. The sum of a $\frac{2}{3}Hz$ sine wave and a $\frac{2}{5}Hz$ sine wave composed the final 90 seconds of the signal. This signal was used to actuate the motors on a foot pedal device that modulated the ankle joint angle. The first 90 seconds of this signal was switched with the last 90 seconds in order to compose a second signal. The second signal was presented as a dot on an oscilloscope, adjacent to another dot which represented integrated sEMG feedback of muscle activity. This signal was then used to provide participants with a target with which to match their sEMG. A Simulink model was designed to implement both signals, such that the model could be used for real-time data acquisition and signal output.

5.3.1.3 Signal Distributions

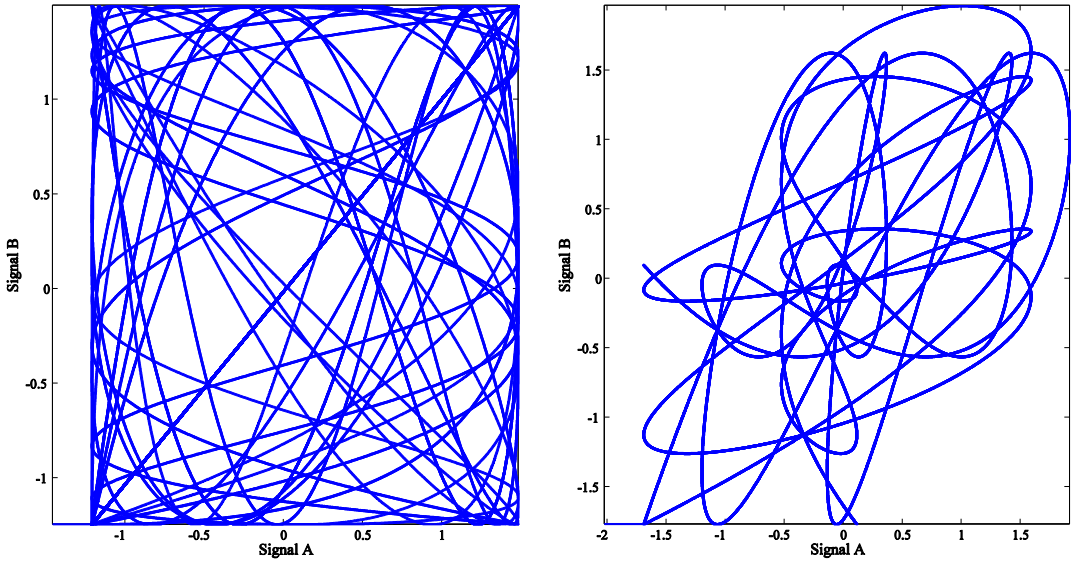


Figure 35. Signal distributions. This figure shows the distribution of data points for both modulating frequency (left) and modulating amplitude (right) signal types. Signal A represents the joint angle control signal, and Signal B represents the sEMG target signal. The graphics clearly show that in the combined case these signals are independent of one another.

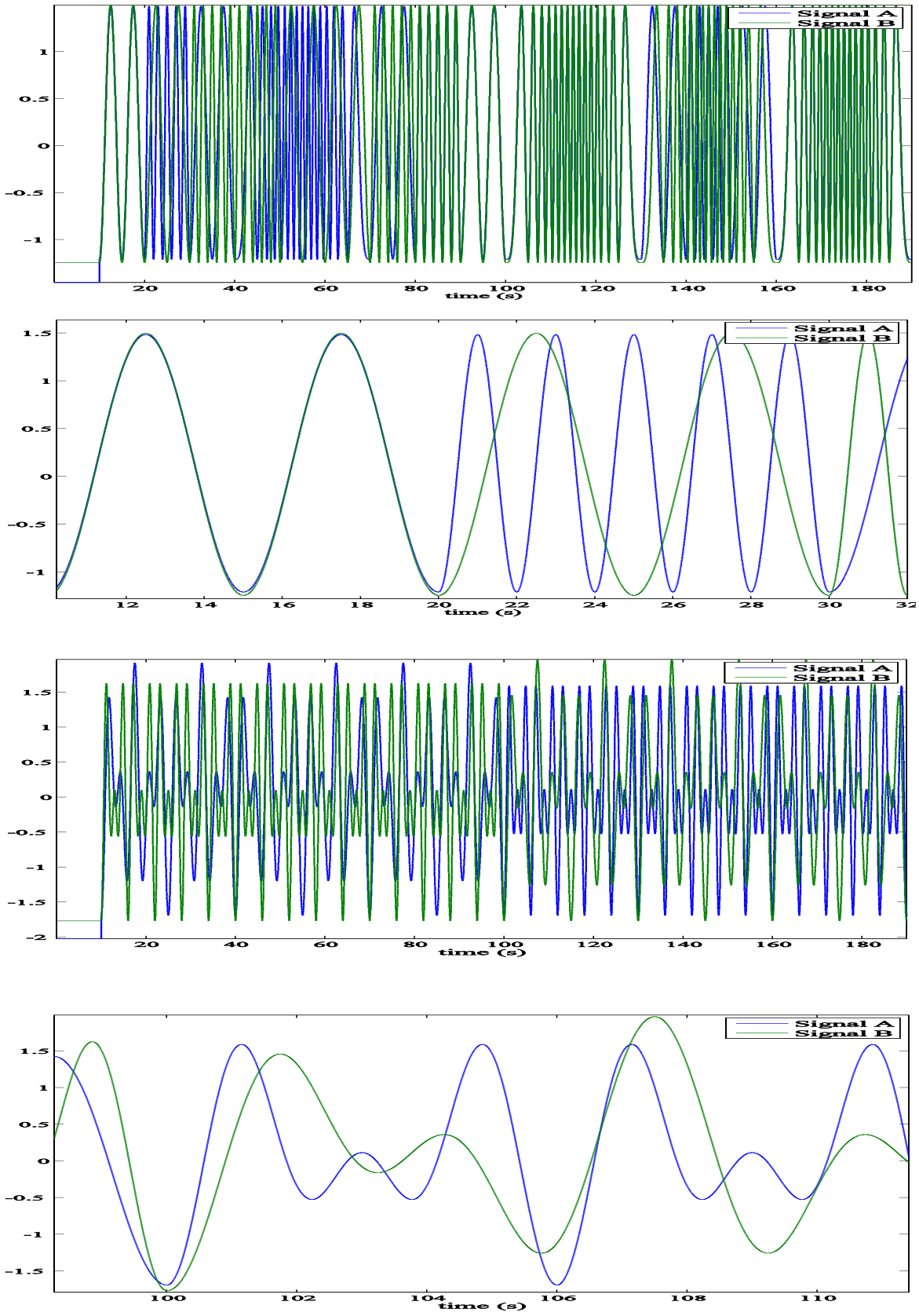


Figure 36. Signal time series. This figure shows the combined frequency modulation and a zoom view (top 2) and amplitude modulation and a zoom view (bottom 2).

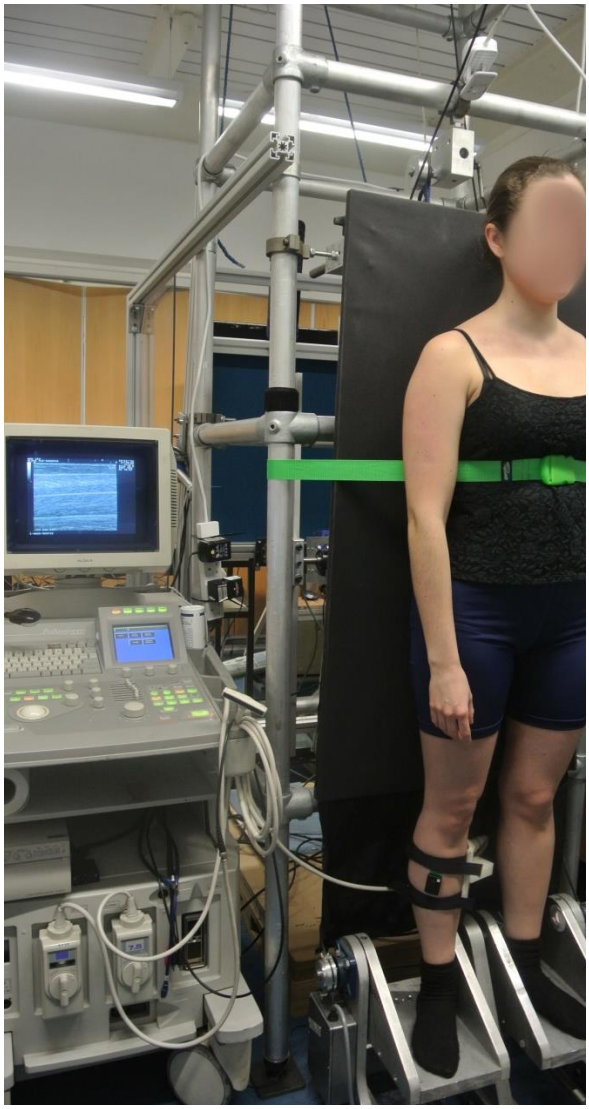
5.3.1.4 Trial Design and Order

Three trials were designed to exploit the active and passive components to shape change under functionally isolated and combined conditions. A total of six trials were designed and presented to each participant in a specific order to help each participant familiarise with the process and the equipment. The duration of each trial was $00:03:10$ $(hh:mm:ss)$ (3 minutes of signals with 10 seconds of static standing before each trial begins). For a comprehensive dataset 3 distinct trials were required:

1. ***Isometric***: participants' ankles remained fixed at a neutral angle while one of the pre-designed signals was displayed on an oscilloscope. Participants were given feedback of their integrated sEMG from the medial gastrocnemius muscle. Participants were asked to match the two signals as best they could by contracting their calf muscles and keeping their joints as fixed as possible. This provided a model of isolated active shape change.
2. ***Passive***: participants were asked to relax while remaining standing on foot pedals which were actuated via one of the pre-designed signals; this caused a change in the ankle joint angle. Again, participants were asked to keep their feet flat on the pedals, while avoiding knee bends without hyperextension. Feedback was provided of their integrated sEMG from the medial gastrocnemius muscle via oscilloscope. Participants were asked to monitor their sEMG activity levels and keep it to a minimum. This provided a model of isolated passive shape change.
3. ***Combined***: participants were asked to do the same signal-matching as in the isometric trials, while allowing their ankle joint to rotate freely as the foot-pedals are controlled. One of the pre-designed signals was used to modulate their joint angle, while the other of the pre-designed signals was given as a target sEMG signal. This provided a model of combined active and passive shape change.

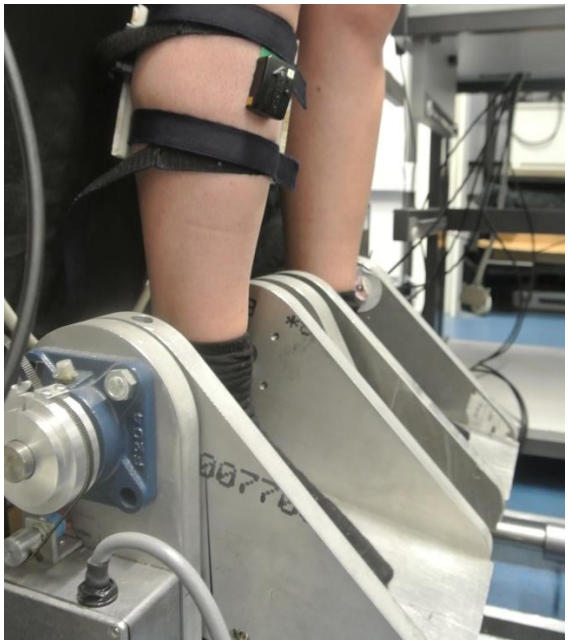
All trials lasted 3 minutes and in the first instance were completed using modulating frequency signals as the control. The reason for beginning with modulating frequency is that the signals are more easily anticipated by participants (i.e. regular peak intervals, or slow transition between frequencies), requiring less control to follow and match activity peaks, and they therefore provide a good opportunity to introduce the concept of the experiment to participants. Likewise, the trials were completed in the order given above so that participants had the opportunity to experience the signals in isolation before experiencing the combined situation. Once all 3 trials were completed, participants then repeated the process with the modulating amplitude signals, having familiarised themselves with the equipment and the process. Participants found it more difficult to anticipate peaks and troughs, which made it difficult to match their sEMG signal with the control signal; therefore this set of trials was completed last, also in the same order as given above.

5.3.2 *Data Collection*



(a)

(b)



(c)

Figure 37. Experiment set-up. The participant stood upright with their back against a board (strapped in above the waist; this prevented unwanted joint articulation and/or muscle contraction as the pedal angle changes during the experiment). The ultrasound machine (on the left of image (a)) imaged the medial gastrocnemius and soleus muscles in the right leg. Feedback of gastrocnemius sEMG and target sEMG amplitudes were delivered respectively via two adjacent points on the oscilloscope in (b). The participant aligned their ankle with the axis of rotation of the pedals in (c).

In order to collect a comprehensive (mixed and isolated function) dataset of images of calf muscle associated with pre-designed muscle state input signals (sEMG, joint angle, force exerted), data were collected in a setting which allowed signal-controlled manipulation of joint angles and signal-assisted contractions (see Figure 37). Ultrasonography of the calf muscles was recorded from the right leg only. Likewise, electromyography data were sampled from 4 muscles in the right leg; medial gastrocnemius (MG), soleus (SO), lateral gastrocnemius, and tibialis anterior. Both MG and SO were

visible in the ultrasound which was acquired at $25Hz$ (frame per second) via an image acquisition card. The angles of the pedals were collected from a potentiometer fixed to the axis of rotation of the two pedals on which the participants stood. A load cell attached to the same pedals recorded force applied to the pedals. Force, pedal angle, and all sEMG signals were recorded at $1000Hz$. Data were synchronised at the beginning of each trial by a device voltage trigger on the image acquisition card, which was fired at the beginning of the Simulink model.

After placing the 4 sEMG electrodes and testing their raw and filtered signals via Simulink live feedback scopes, the ultrasound was fixed near to the MG sEMG with a custom probe holder and two straps (see Figure 37). During probe placement, care was taken to ensure full visibility of both MG and SO. The probe was placed to ensure visibility of superficial and deep muscle fasciae in both muscles, with pennate appearance of fascicles between both muscles. It was important to reduce inter-participant bias by ensuring a consistent quality and appearance between participants' muscles. Data were collected from a total of 20 participants (11 male, 9 female aged between 18 and 50).

5.3.2.1 Ethical Approval

These experiments were approved by the Research Ethics Committee of the Faculty of Science and Engineering, Manchester Metropolitan University (MMU). Participants gave (written) informed consent to these experiments, which conformed to the standards set by the latest revision of the Declaration of Helsinki. Experiments were performed at the Cognitive Motor Function laboratory, in the School of Healthcare Science at MMU.

5.3.3 *Methods of Analysis: Restricted Boltzmann Machines*

Restricted Boltzmann Machines (RBMs) are a powerful machine learning method for encoding multi-level feature representations of images from raw image data [93, 73, 54] without labels/annotations, akin to Principal Components Analysis (PCA). The use of RBMs for encoding skeletal muscle states from ultrasound would enable the RBM learning algorithm to decide what features are descriptively important and how best to represent data vectors with a sparse latent vector of descriptive features.

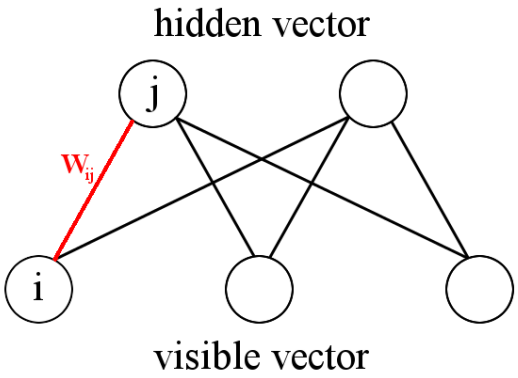


Figure 38. Architecture of a Restricted Boltzmann Machine. A vector of visible nodes has bipartite connectivity (with associated weights) to a vector of hidden nodes. Visible nodes are presented with an input vector then the RBM computes a vector of probabilities – termed the hidden (or latent) vector – by taking weighted linear combinations of the input variables. For a trained system, if the states of the hidden vector are known, the visible vector can be approximately reconstructed by taking weighted linear combinations of the hidden node states.

The architecture of a RBM is a two layer bipartite graph with weights on its connections (see Figure 38). There are no lateral interactions between nodes (hence the name, 'Restricted') which helps to speed up the learning. RBMs learn a feature representation (hidden vector) such that: if the states of the features are known, then the probable input vector that caused those states can be inferred (or reconstructed). The probabilities of hidden states are computed with,

$$P(h_j = 1|v) = \frac{1}{1+e^{-\sum v_i w_{ij} + b_j}},$$

Equation 6

where h_j is a hidden node, v is the visible vector, w_{ij} is the weight connecting $v_i$and h_j, and b_j is the bias associated with h_j. Inversely, the probabilities of the visible vector are computed with,

$$P(v_i = 1|h) = \frac{1}{1+e^{-\sum h_j w_{ij} + b_i}}.$$

Equation 7

where v_i is a visible node, h is the hidden vector, w_{ij} is the weight connecting $v_i$and h_j, and b_i is the bias associated with v_i.

The objective function of an RBM is to maximise the probability of generating realistic examples of input vectors with a process known as alternating Gibbs sampling. Starting with either random states or a real input vector clamped to the visible vector (setting *v* equal to *I*, where *I* is the input vector), the RBM infers all the hidden variables (Equation 6), then infers all the visible variables (Equation 7), and back and forth until 'thermal equilibrium' (when the states are in a low energy configuration, i.e. the network has settled). After a very long time of Gibbs sampling, the visible vector will be far away from where it started and will hop in and out of deep energy ravines (clusters), revealing probable visible vectors. This can also be stated as: an RBM learns to maximise the probability of randomly generating any visible vector *v* from a data set when sampling at equilibrium. A good model should also be able to generate unseen plausible examples (i.e. those not in a training set) with equivalent frequency.

5.3.3.1 Markov Chain Monte Carlo

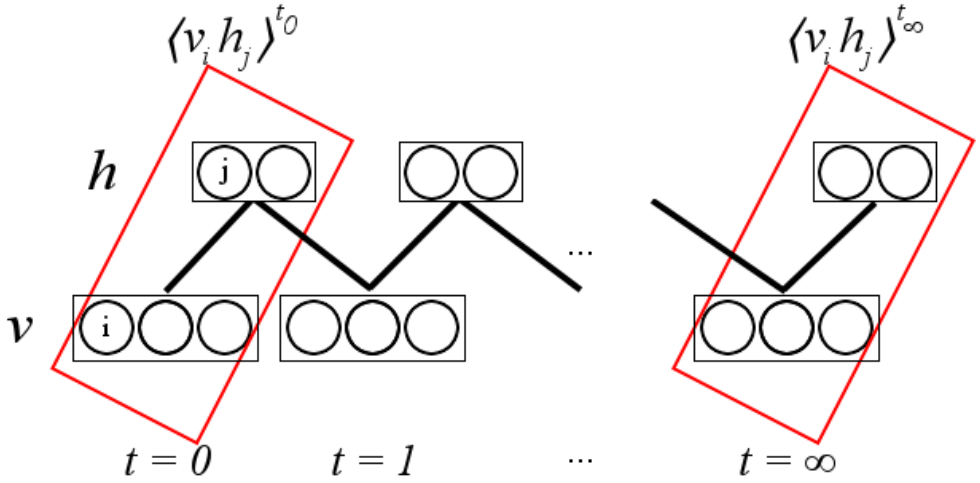


Figure 39. Markov Chain Monte Carlo (MCMC). The graphic shows the stages of MCMC; on the left at time $\boldsymbol{t = 0}$ a data vector is clamped to the visible units and then a set of hidden variables is constructed (Equation 6), followed by alternating updates of the visible states (Equation 7) and corresponding hidden states (Equation 6). The positive statistics are recorded (red square on the left), then after running the Markov chain for a long time ($\boldsymbol{t = \infty}$) the negative statistics are recorded (red square on the right).

The maximum likelihood learning algorithm for a RBM (i.e. as per previous paragraph) is Markov Chain Monte Carlo (see Figure 39) and it is notoriously slow. First, an RBM is created with some large number of hidden variables (typically an order of magnitude less than the number of training cases in a set [94]), and a visible vector with a number of variables equal to the dimensions of a single input vector. Small random (normal distribution) weights are initialised between hidden and visible

variables to help break the initial symmetry of the model and speed up learning (although the RBM has stochastic units so the initial values are not as important as in other machine learning algorithms like backprop nets, where model symmetry is bad for convergence). Then, a data vector is 'clamped' onto the visible vector (this means making all the *v*'s equal to the data points of some input vector). Following, the hidden vector probabilities are computed and both the hidden and visible states are recorded. This initial phase of clamping a data vector, inferring a hidden vector, and recording the statistics is known as the positive phase, occurring at time $t = 0$ (time denotes steps in a Markov chain; the positive phase being the initial state). After the positive phase, a negative phase is initiated which involves alternating Gibbs sampling from the initial state until the network reaches equilibrium (when the states have settled into a low energy ravine); for the purpose of argument the number of steps until equilibrium can be thought of as being reached at time $t = \infty$. To complete the negative phase, the states of the hidden and visible vectors are recorded. The weight adaptation rule is then the difference between positive and negative phase correlations,

$$\Delta w_{ij} = \mu\left(\langle v_i h_j \rangle^{t_0} - \langle v_i h_j \rangle^{t\infty}\right),$$

where $\langle v_i h_j \rangle^{t_0}$ is the correlation between v_i and h_j after the positive phase and the angle brackets denote that $v_i h_j$ is averaged over some batch (subset) of the data, $\langle v_i h_j \rangle^{t\infty}$ is the correlation of v_i and h_j after the negative phase, and μ is the learning constant. This learning rule is trying to minimise the difference between the states of the hidden and visible vectors when there is data clamped (t_0) and when the model is allowed to run for a long time generating 'fantasy' states (t_∞). If the algorithm is run for long enough, eventually the two correlations will be the same and the learning will stop.

5.3.3.2 Minimising Contrastive Divergence

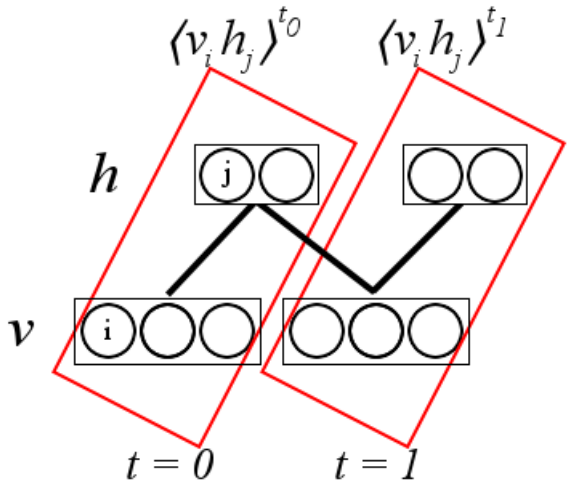


Figure 40. Contrastive Divergence ($\mathbf{CD^1}$). The graphic shows the stages of $\mathbf{CD^1}$; on the left at time $\boldsymbol{t = 0}$ a data vector is clamped to the visible units and then a set of hidden variables is constructed (Equation 6), followed by a single update of the visible states (Equation 7) and corresponding hidden states (Equation 6). The positive statistics are recorded (red square on the left), the negative statistics are recorded after only a single step in the Markov chain ($\boldsymbol{t = 1}$; red square on the right).

For proper maximum likelihood learning, the algorithm must reach equilibrium before collecting negative statistics, which is extremely slow and impractical in many cases. However, relatively recent developments have brought about a revival of interest in RBMs with the introduction of a short-cut called Contrastive Divergence [56]. The concept essentially exploits the fact that when learning starts with random initialisations, it is a waste of time running to equilibrium (i.e. it can be assumed with confidence that the random initial weights are a bad guess at an optimal configuration); after only a few Gibbs steps the fantasy particles (vectors) reveal what direction they are wandering in, whereupon the weights can be moved closer to optimum with,

$$\Delta w_{ij} = \mu\left(\langle v_i h_j \rangle^{t_0} - \langle v_i h_j \rangle^{t_1}\right).$$

This learning rule is denoted CD^1 because only a single Gibbs step is taken before learning occurs (see Figure 40). The technique works well in practice except for particles (vectors) that are far away from the global minimum. This is due to the fact that a single Gibbs step may not be enough to construct a large enough energy ravine at the data for the particle to fall into the global minimum [94]. A good solution is to run the learning for a while (until the mixing rate of the Markov Chain starts to slow down) with CD^1 and then run the learning for a while with CD^3 (3 steps), then CD^5 (5 steps), and so on until eventually all stray particles spill into the global minimum.

5.3.3.3 Gaussian-Bernoulli Restricted Boltzmann Machines

One of the main practical issues with the binary stochastic state RBM is that it is inappropriate for modelling continuous visible states like those found in natural images. By modelling images as a matrix of probabilities of pixels and their neighbours being active, RBMs fail to exploit the natural correlations among neighbouring pixels (i.e. the intensity value of an arbitrary pixel in a natural image is nearly always the average of its neighbours). One solution is to treat the raw probabilities as intensity values without assigning a discrete binary state (so called 'mean-field' logistic units), but in practice this does not work well for real images. One widely accepted solution is to use stochastic binary hidden units with real valued visible units plus some Gaussian noise. This solution is referred to as the Gaussian-Bernoulli Restricted Boltzmann Machine (GBRBM; [95]).

The activation of hidden units then becomes,

$$p(h_j = 1|v) = \varphi\left(b_j + \sum \frac{v_i}{\sigma_i^2} w_{ij}\right),$$

Equation 8

where σ_i is the standard deviation of the visible unit, and φ is the sigmoid transfer function $\varphi(x) = \frac{1}{1+e^{-x}}$. The activation of the visible units is,

$$p(v_i = v|h) = N\left(b_i + \sum h_j w_{ij}, \sigma_i^2\right),$$

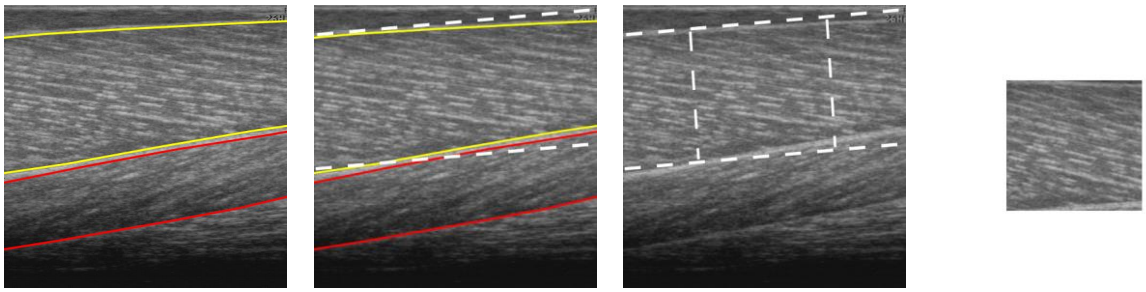
where $N(\mu, \sigma)$ is a sample from a normal distribution with mean μ and standard deviation σ. The learning rule is a slight modification on the original rule, which still minimises contrastive divergence,

$$\Delta w_{ij} = \mu\left(\langle \frac{1}{\sigma_i^2} v_i h_j \rangle^{t_0} - \langle \frac{1}{\sigma_i^2} v_i h_j \rangle^{t_1}\right).$$

With linear units plus Gaussian noise, and stochastic hidden variables, real-valued raw image pixels can be modelled much more efficiently and accurately. An original C++/CUDA implementation of a GBRBM and the contrastive divergence learning algorithm is given in Appendix 8.7.

5.3.4 *Muscle Segmentation & Normalisation*

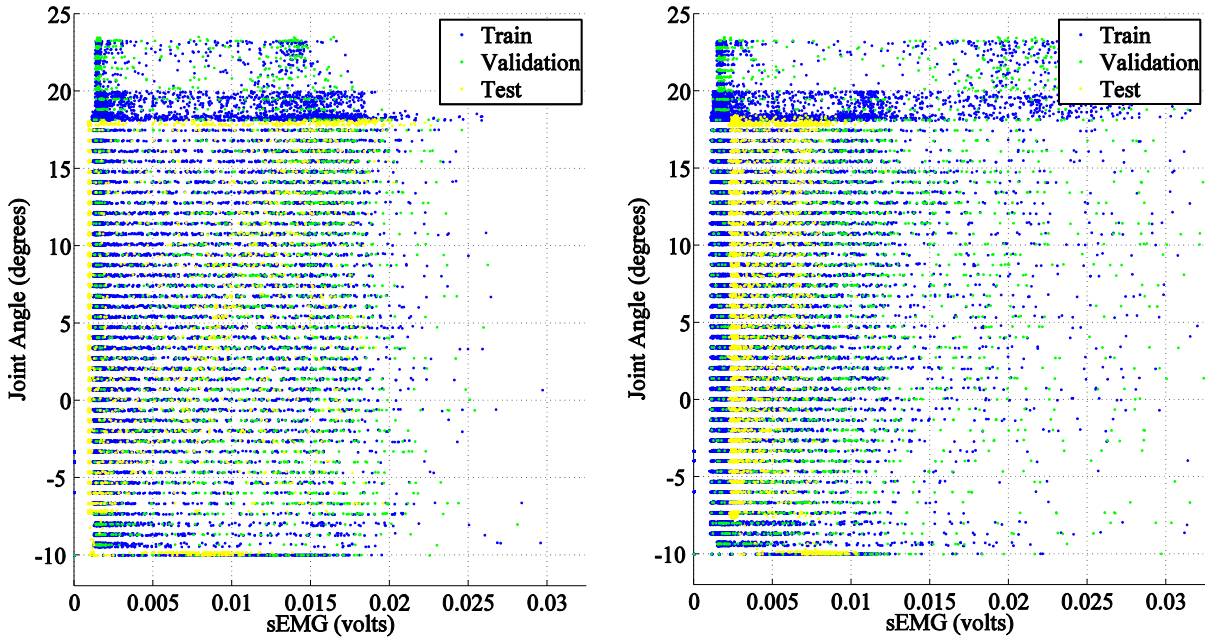
Parametric models such as GBRBMs require fixed input dimensions. Furthermore, to reduce the influence of inter-participant bias (relative muscle thickness, relative orientation of facia/aponeuroses and fascicles [96]) some standard representation of each muscle is required prior to building a GBRBM model. For these reasons, very accurate muscle region segmentation is required in order that each muscle could be extracted and resampled at fixed dimensions. The MG and SO muscles were automatically segmented using the method described in chapter 4 for every ultrasound image of each participant's set of trials, followed by sampling pixels from within a central bounding box from each muscle (see Figure 41). This process is henceforth referred to as image standardisation.



(a) (b) (c) (d)

Figure 41. MG image standardisation. In order from left to right this figure shows: a) Initial segmentation of MG (yellow contours) and SO (red contours). b) Least squares fit of two parallel lines (dashed white lines) to the superficial and deep contours of MG. c) A bounding box is created by the addition of two perpendicular parallel lines (dashed vertical lines) a distance apart 50% of the total image width, about the centre of the image. d) The pixels within the box are cropped and rotated, later to be resized to a standard $\mathbf{80 \times 80}$ pixel matrix. This process is repeated for SO to form a separate pixel matrix.

5.3.5 *Sampling from the Data Space*



(a) (b)

Figure 42. Model sample selection by nearest Euclidean distance to a 2D grid of equidistant points within the data range for (a) medial gastrocnemius, and (b) soleus. The training set is 40 nearest points to a $\mathbf{24 \times 24}$ grid, the validation set is 5 nearest points to the same $\mathbf{24 \times 24}$ grid and set of participants, while the test set is 5 nearset points to a $\mathbf{24 \times 24}$ grid on a held out participant.

A total of 570,000 images were segmented ($25Hz \times 190\ seconds \times 6\ trials \times 20\ participants$), standardised and put into a database with associated truth signals (since images in each sequence were synchronised with acquisition of sEMG of MG and SO, force, and joint angle). Then, for any given combination of joint angle and sEMG one can find the nearest (normalised Euclidean distance) pair of signals recorded in the database and extract the associated image and labels. Due to the sheer volume of images and the high dimensionality (in data vector terms) of each image (80×80) – and given that the space contains a huge bias towards particular signal combinations (e.g. static muscle) – a subset of the space was sampled for model building.

Before sampling, the signals were normalised by subtracting the mean and dividing by their standard deviations (mean normalisation). Then, a grid of 24×24 data points was generated, evenly spaced within the range of the normalised signals. Each data point represented a desired sEMG coupled with a desired angle. The space was then subsequently sampled by taking standardised images of MG and SO, and their associated un-normalised signals for the nearest n (by Euclidean distance) pairs of signals to each point in the grid (see Figure 42). The images in the selected samples were used to build two separate GBRBM models, without using the labels at this stage.

In order to assess the performance of the GBRBM on out-of-sample data (not involved in model selection), good practice was followed by dividing the data into training, validation, and testing sets; where the training and validation sets were constructed from 19 of the 20 participants, and the remaining participant made up the testing set. The sampling method described in the previous paragraph was used to select the 5 nearest points to each point in the grid to make up the validation set; subsequently the selected points were removed from the following selection of the 40 nearest points in the set, which made up the training set. The same method was used on the held out participant to select the nearest 5 points to each point in the grid.

The training set consisted of 23,040 images, and the validation and testing set both consisted of 2,880 images. Because of the selection methods, these images are representative examples of evenly-distributed combined functional muscle state across the pool of 20 participants. The justifications for using such a small subset of the 570,000 available images is (other than the already stated reasons of data redundancy/bias) were to speed up the GBRBM training process[6], and high confidence in the image segmentation and standardisation process. The normalisation problem is one of the big challenges in machine learning and since this data contains only the well-normalised muscle region, it should take fewer examples to train a good GBRBM model (although future work may make use of the entire set and/or even more participants).

5.3.6 GBRBM Architecture Selection and Model Training

A GBRBM model architecture was chosen based on the size of the data vectors (images), the size of the training set, and in order to maximise computational efficiency. With images of resolution 80×80 the number of visible vector units required was 6,400. The number of required hidden vector units is more speculative. However, following widely accepted recommendations [94] (which states that the chosen number of hidden units is typically an order of magnitude less than the number of examples in the training set), and an attempt to minimise the number of free parameters in the system (due to training times and over-fitting problems), the chosen number of hidden units was 768. This specific number was chosen to equal the number of cores on the Graphics Processing Unit (GPU) (device that was used to speed up training) for efficient parallel computation. A graphical representation of the GBRBM architecture follows:

[6] To train a single model took on the order of 2-3 weeks for each model (MG, SO). To train a larger model would have taken considerably longer.

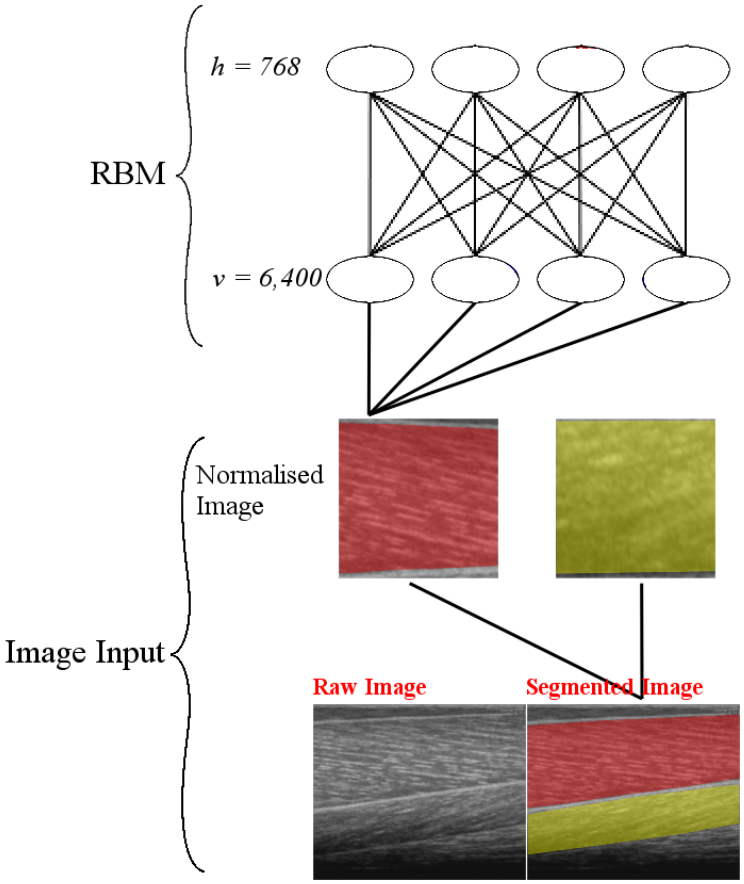


Figure 43. GBRBM model architecture. 768 hidden units *h* will learn to form a probabilistic state representation of the normalised images, such that the visible (input) vector *v* can be accurately reconstructed. Each muscle MG and SO had its own GBRBM model.

Training separate models of MG and SO required a training cycle of 2,000 iterations; where a single iteration consisted of a full pass over the entire training set, then updating the weights (this is known as batch training). After each iteration, the reconstruction mean squared error (MSE between all input vectors in a set and all reconstructed input vectors in the set, after a single step of Gibbs sampling) of the training and validation sets was evaluated. The training error typically descends rapidly – initially – and then begins to descend more slowly with each additional iteration. If there are enough free parameters in the model (large *h*) the reconstruction error on the validation set is expected to start increasing as the model over-fits the training data. Good practice dictates that this is a good time to stop training. Once training is complete, the testing set reconstruction error can be measured to give a good idea of true out-of-sample performance. In this instance there were not enough free parameters in the model for it to over-fit the training set so the reconstruction errors effectively plateaued. It should be noted that the model did not fully converge as the error was still descending very slowly. A decision was made terminate training because the training error was showing no signs of further improvement leading upto 2,000 training iterations.

5.3.7 GBRBM Receptive Fields (Features)

The result of training was a model of muscle states in the form 768 feature detectors. Each feature detector has learned to respond to highly correlated (with the weights connecting a feature h_i with the input vector v) input vectors. For example, a good feature detector might have a mostly zero weight vector with high weights in the location of the superficial aponeurosis. Such a feature detector would output a high probability when there were light pixels in the region of the superficial aponeuroses, and a low probability when there were dark pixels in that region. It is possible to visually inspect the weight vectors (termed the receptive field, or feature) as follows:

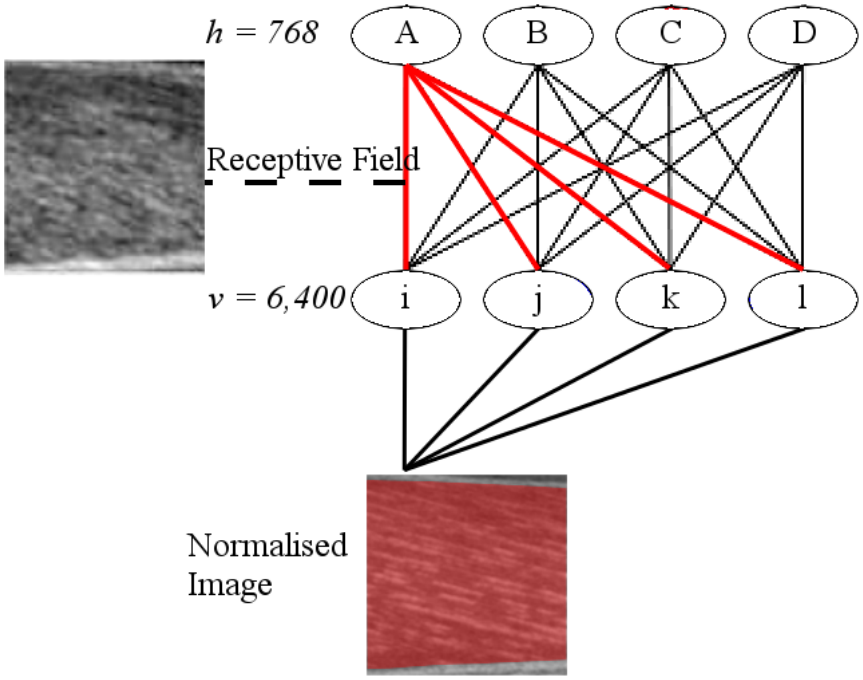


Figure 44. Receptive field of a feature detector of the MG GBRBM model. Figure shows the receptive field (black & white image on the left corresponding to red connecting lines) of the hidden unit A. Since each weight (red lines) is associated with a pixel (I, j, k, l …) in the original image, the weights can be visualised as an image of the same dimensions as the original input vector. This gives experts the opportunity to try to understand what it is about these images that a given feature detector is responding to; in this case the feature seems to be interested in the presence/orientation of deep aponeuroses, and a few relative gradients across the MG muscle.

One can visually analyse the entire ensemble of features in both networks to check that the learning has worked sensibly. One can also examine what features the GBRBM considers important for reconstructing hypothetical images of skeletal muscle. The following page shows a randomly selected subset of features from both GBRBMs. Features can be discovered that appear to be modelling fascicle curvature and presence/position of aponeuroses.

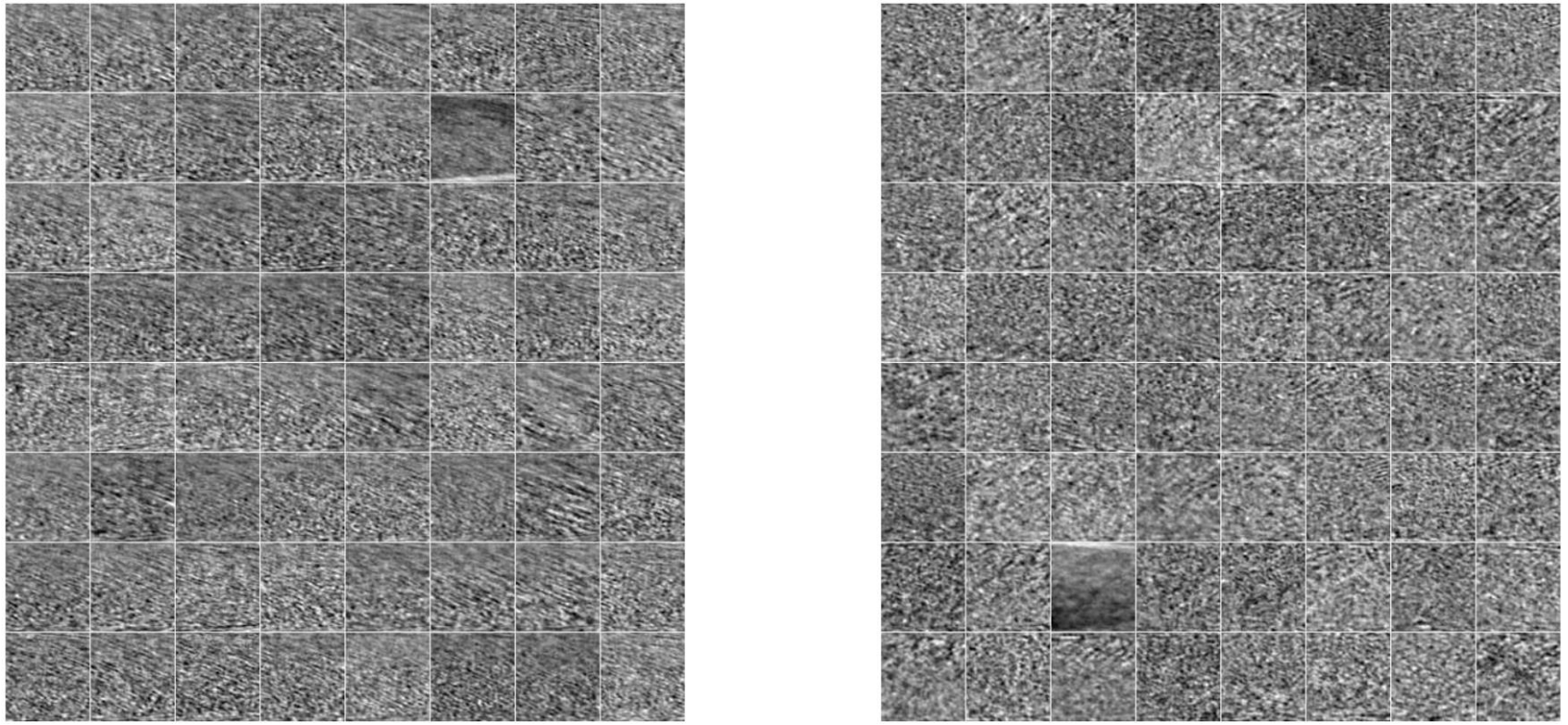


Figure 45. Each square in the two **8 × 8** grids shows the receptive fields of 64 (out of 768) randomly selected hidden nodes (feature detectors) for MG GBRBM (left) and SO GBRBM (right). The majority of MG GBRBM feature detectors appear to be detecting configurations of fascicles, while the SO GBRBM feature detectors appear to be modelling something more complex and less intuitive although curved and orientated fascicles (this is likely because images of SO do not typically present fascicles as well-defined as those in MG, and in some cases no fascicles are visible at all).

5.3.8 *Linear Estimation of Muscle States from MG/SO GBRBM Features*

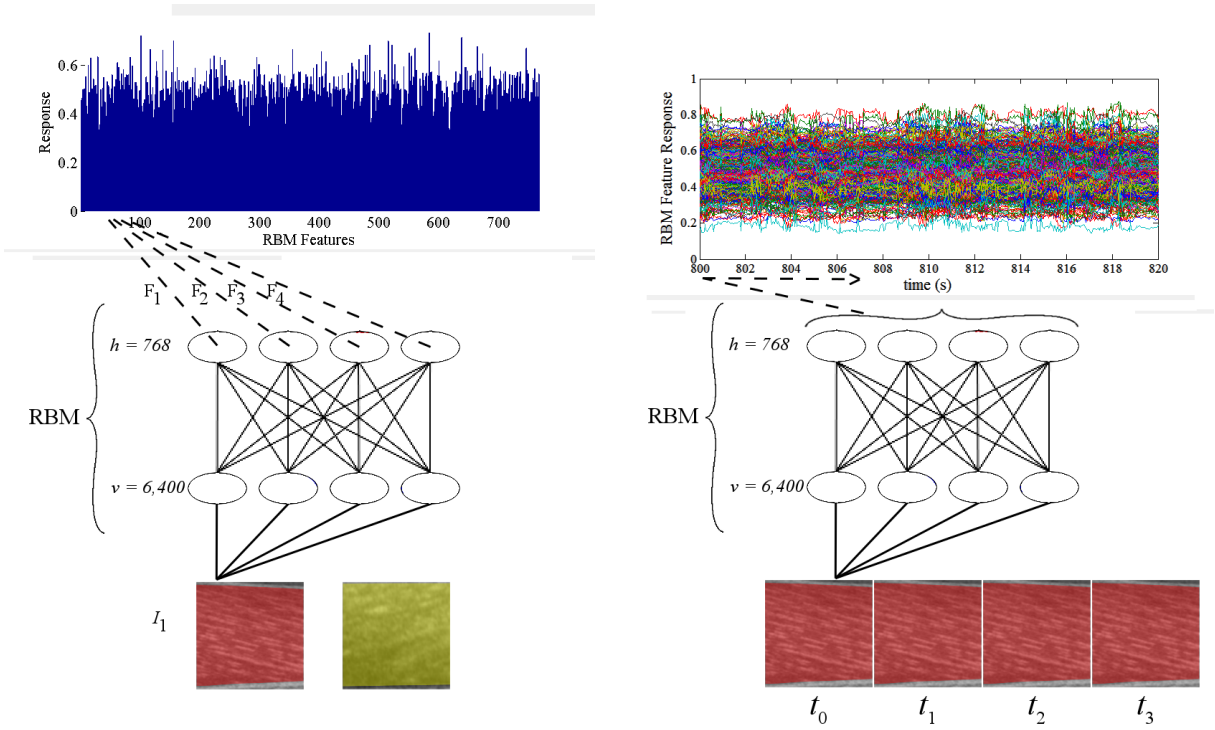


Figure 46. Transforming standardised image sequences to a time series of feature responses. The illustration on the left shows how an image (standardised MG; red highlighted muscle at the bottom) can be transformed into a 768 static feature vector F. The bar plot at the top shows the responses of each feature (the hidden variables) after a linear combination of pixels via the connecting weights of h to the vector v. The illustration on the right shows how feature responses can be constructed for every frame of a sequence $\boldsymbol{t_n}$ of standardised images, resulting in a time series of 768 feature responses.

The two trained GBRBM models (MG/SO) were used to transform all 570,000 images into their respective GBRBM feature representations (see Figure 46). A linear combination of these feature responses over time can be correlated by optimisation of a linear set of coefficients connecting the associated external signals (force, sEMG, joint angle). A linear neural network (with no time series history) [97] was used to build individualised models for each participant:

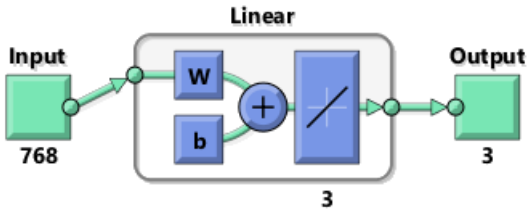


Figure 47. Linear layer neural network. Diagram shows the connectivity of the linear regression model used to predict the 3 output signals (force, sEMG, and joint angle) from the 768 features for any given frame in a given sequence. The linear layer shows linear $\boldsymbol{768 \times 3}$ coefficients w and 3 bias variables b. The diagonal line after the + sign denotes a linear transfer function was used.

K-fold cross validation was used to ensure good intra-participant model generalisation. A K value of 20 was used in the following manner:

```
1. for i = 1 to K
2.         idx1= [(i-1)*(num_frames/K)+1]
3.         idx2 = [i*(num_frames/K)]
4.         cross_val_set = get_frames(idx1:idx2)
5.         cross_val_labels = get_labels(idx1:idx2)
6.         training_set = get_frames([1:idx1, [idx2+1]:num_frames)
7.         training_set_labels = get_labels(([1:idx1, [idx2+1]:num_frames)
8.         model = linear_layer(training_set, training_set_labels)
9.         prediction = model(cross_val_set, cross_val_labels)
10.        save_results(i, model, cross_val_labels, prediction)
11. end
```

This method of leaving out a section of data, building a model on the remaining data, and evaluating performance on the left-out data is widely accepted by experts as the standard method for evaluating model generalisation. By leaving out every image and associated labels in succession allows model evaluation of the entire sequence. This method was repeated for every participant in succession and the results are measured with variance accounted for and mean square error. Ideally one would use a larger K ($K = num\_frames$) however, this is an unrealistic measure due to time constraints[7].

[7] To complete a round of cross validation with every participant at $K = 20$ took on the order of one day.

5.4 RESULTS

5.4.1 EVALUATING THE PERFORMANCE OF MG & SO GBRBMS

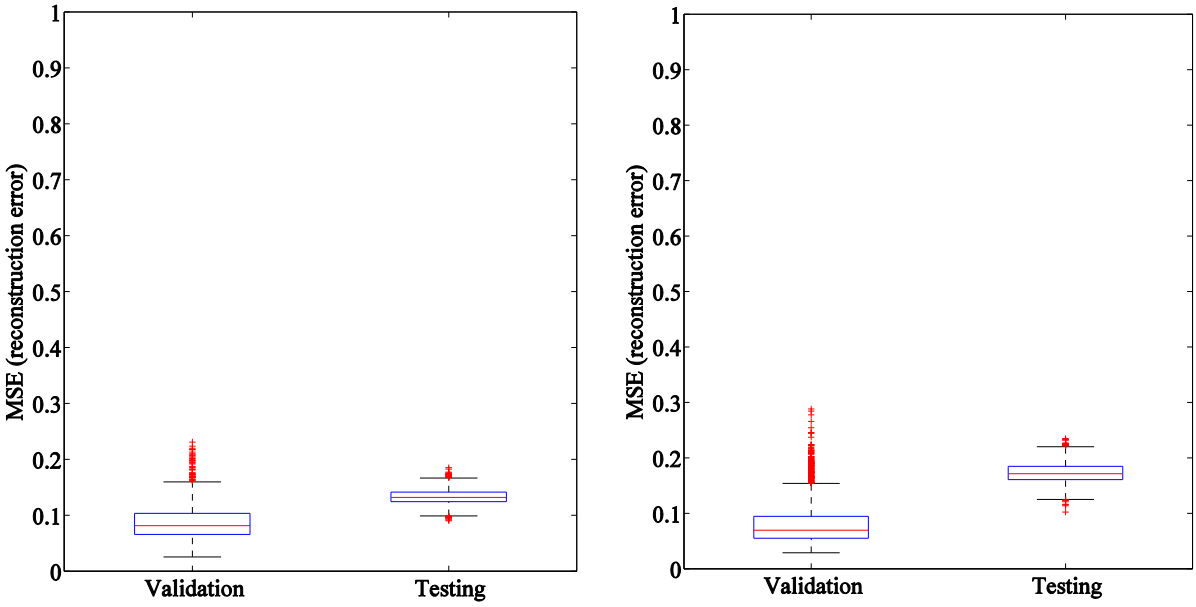


Figure 48. Box and whisker plot of GBRBM validation and testing image reconstruction errors per image for the MG (left) and SO (right) GBRBM models. Each data point is the average square reconstruction error (averaged over all pixels) per image.

After training the GBRBMs for 2,000 training iterations, the performance of each GBRBM was evaluated on their ability to reconstruct standardised images of MG and SO (respectively) from the training, validation and testing sets. The results (see Figure 48) showed that images in the testing set were reconstructed less accurately than those in the training and validation sets. This is expected and is a good sign that the learning has worked correctly. Results also show that on average SO images were reconstructed more accurately than MG images according to MSE of reconstructions.

MSE is a good way to measure model performance in numeric terms for setting a baseline/standard for future work. However, in order to understand the performance of the models in real terms, it is recommended to visually inspect the reconstructed images against the original images. Because there are too many images to evaluate each one, the top and bottom 4 reconstructions from both models were selected for visual inspection (see Table 6 and Table 7). Visual inspection of the top/bottom 4 image reconstructions hints at the suggestions that many of the errors could arise from the inability of the network to reconstruct noise present in the images. The reconstructed images appear to have been filtered and a lot of the high-frequency contrast has been attenuated, while all of the 'important' descriptive features such as fascicles and aponeuroses have been retained. This statement can be said of all images that have been visually inspected. In particular, notice the bottom 4 reconstructions from

each GBRBM as the grainy original image appears to have been filtered, yet the definition and presence of fascicles appears to have been retained in all cases.

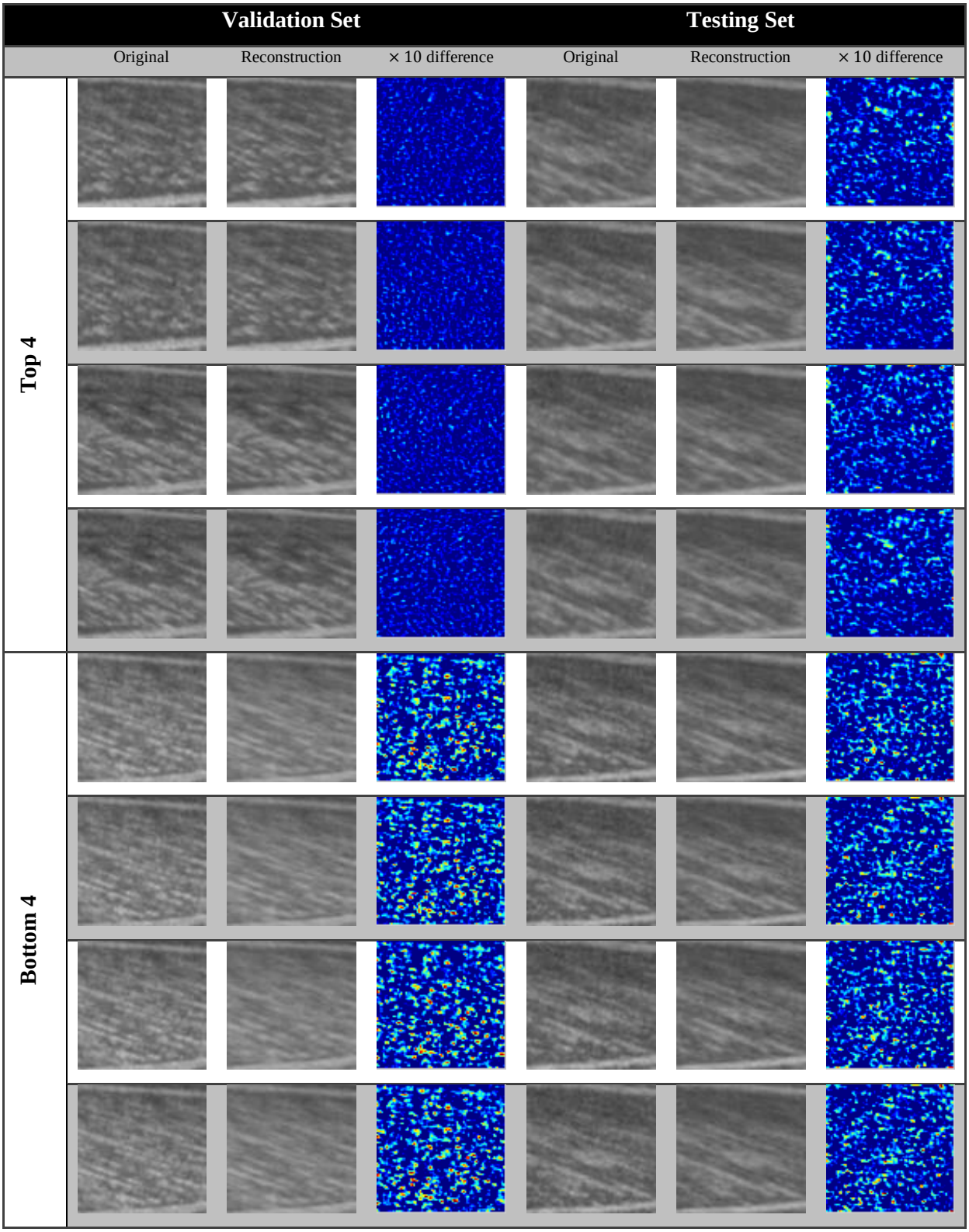


Table 6. Top/Bottom (MSE in descending order) medial gastrocnemius image reconstructions from the GBRBM.

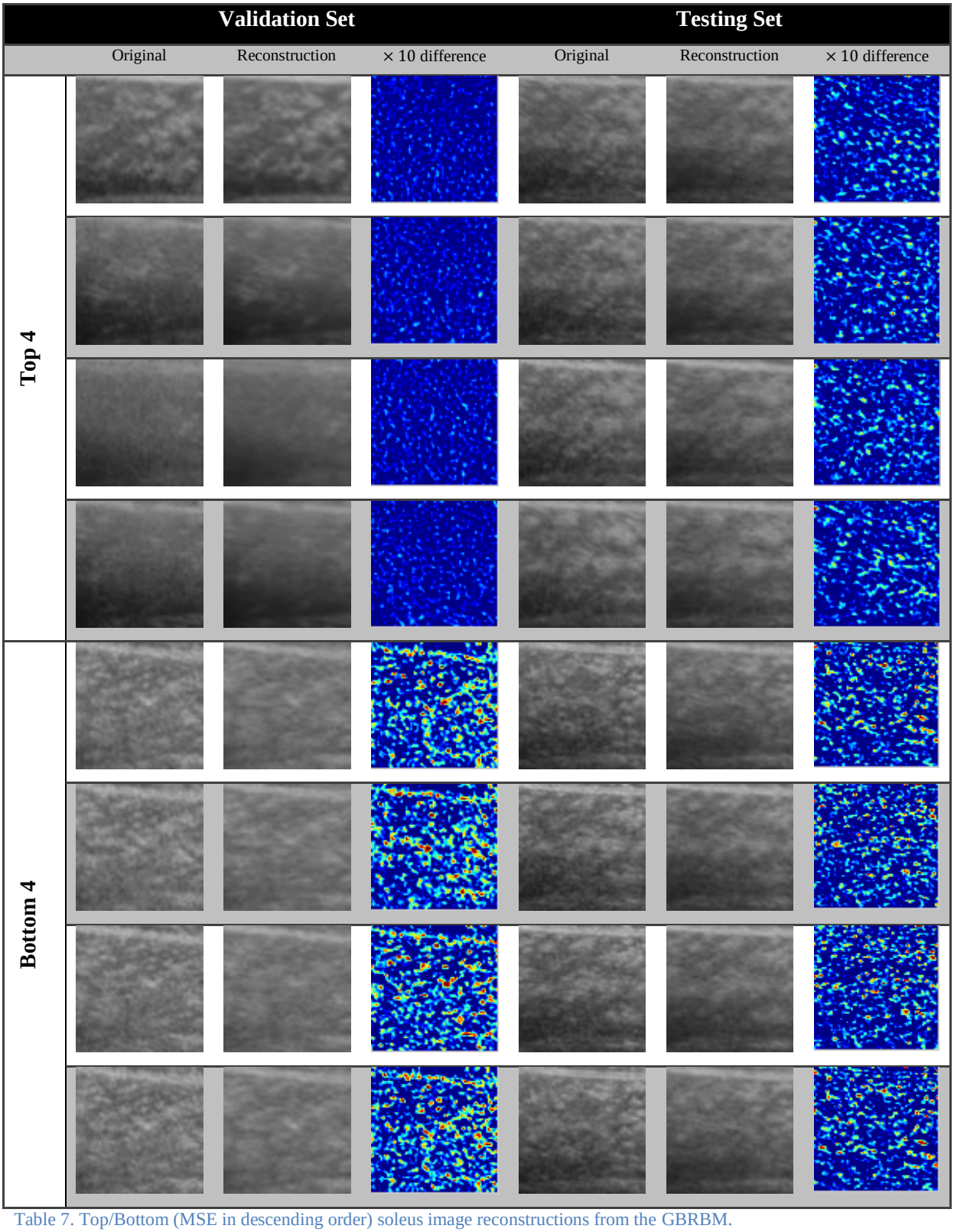


Table 7. Top/Bottom (MSE in descending order) soleus image reconstructions from the GBRBM.

5.4.2 GBRBM RECEPTIVE FIELD (FEATURE) ANALYSIS

Further to analysing the reconstruction of images within held out (out-of-sample) sets of images, it is possible to analyse the receptive fields of hidden nodes in relation to external signals (force, sEMG, ankle angle). Images presented to the GBRBM can be transformed into their probabilistic hidden vector representations (hereafter referred to as feature responses) via Equation 8 by retaining only the raw probabilities (without producing a stochastic binary value). Feature responses were generated for every image in every trial for a representative participant. The feature responses of individual elements (features) of the hidden vector were compared with each of the external signals. Correlations between the responses of all features of the hidden vector and all external signals were carried out over all trials to ensure that highly correlating feature responses held up across trials in all functional (combined, isometric, passive) cases. Before the comparison, all signals and features were normalised by subtracting the mean and dividing by their standard deviations. After the correlations (Spearman) were computed the features were sorted in order of average (across all signals) correlation rank over all trials. The results of this are presented in Figure 49.

This analysis revealed that natural correlations exist between GBRBM features and the external signals in isolation. Figure 49 shows that there are many features that exhibit relatively high correlations with respect to force and sEMG, and some of those features exhibit relatively high (absolute) correlations with respect to the joint angle, and some do not. The interpretation of this is that there are a lot of features would appear to respond only to changes in discrete muscle states in combined and isolated functional cases. For example, the implication of a feature correlating significantly with sEMG and not joint angle over combined, isometric, and passive functional cases, is that there exists a natural feature in the GBRBM that responds only when there is contractile (sEMG) activity present within the muscle, regardless of combined or isolated joint rotation. This result is demonstrates the power of the GBRBM approach; the GBRBM was never given access to any labelled data during training and this information was extracted solely from the raw data.

Further to analysis of the distribution of correlations among features, one can take the average responses of the top *n* features over all trials and make qualitative comparisons against the external signals. Figure 50 and Figure 51 show the results of averaging the top three (Spearman rank) feature responses for MG and SO respectively, with respect to each signal (sEMG, joint angle, force) over all trials for a representative case. These results clearly show that there are features that were automatically learned from the data by each GBRBM which independently respond to changes in muscle state that were caused by completely separate external/internal factors in isolation and even when the factors are proportionally mixed in the combined case.

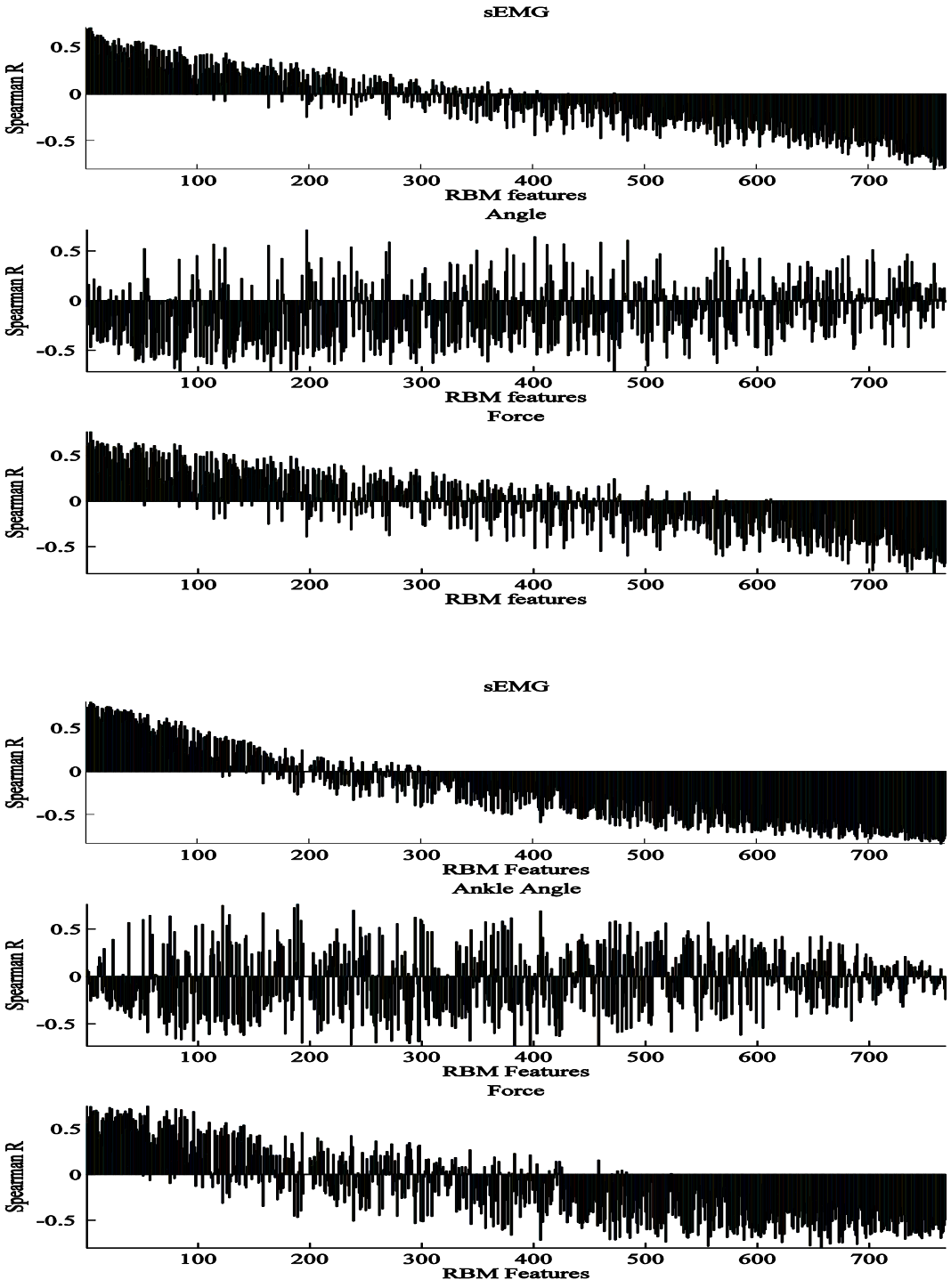


Figure 49. Natural correlations (Spearman) between features and external signals. This figure shows correlations between GBRBM features and the 3 external signals for MG (top) and SO (bottom) over all 6 trials for a typical participant. The correlations are sorted (descending) by the average absolute magnitude of the correlation with truth signals per feature. Notice that the features that correlate highly with sEMG correlate well with force in isolation from joint angle, while many features correlate well with joint angle in isolation of force/sEMG. This result empirically demonstrates that there are features representing discrete skeletal muscle state that are stable over time.

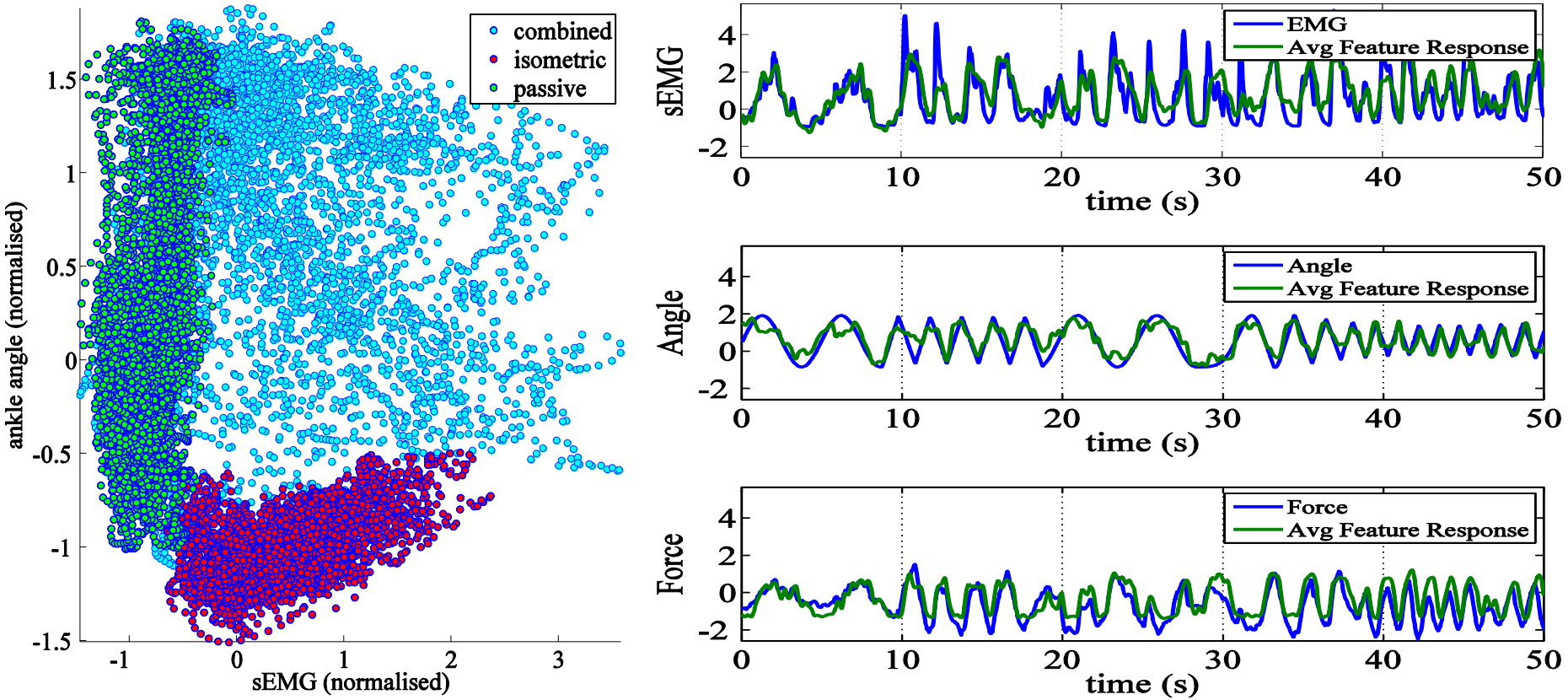


Figure 50. Averaged MG GBRBM feature response over all trials (for a the representative case in Figure 49) for the top 3 features in terms of correlation with sEMG (top), ankle angle (middle), and force (bottom). The averages of the top 3 raw features contain enough information to predict the respective signals in isolation of one another, without any combinatorial design (weighting the combinations of features). Left: shows a scatter plot of the averaged top 3 features in terms of correlation with sEMG and ankle angle over all 3 trials. Right: shows a selected subsection of the averaged top 3 features in terms of correlation with sEMG, ankle angle in the combined-function trial. Notice the remarkable level of detail to which joint angle can be separated from sEMG/force in the combined case.

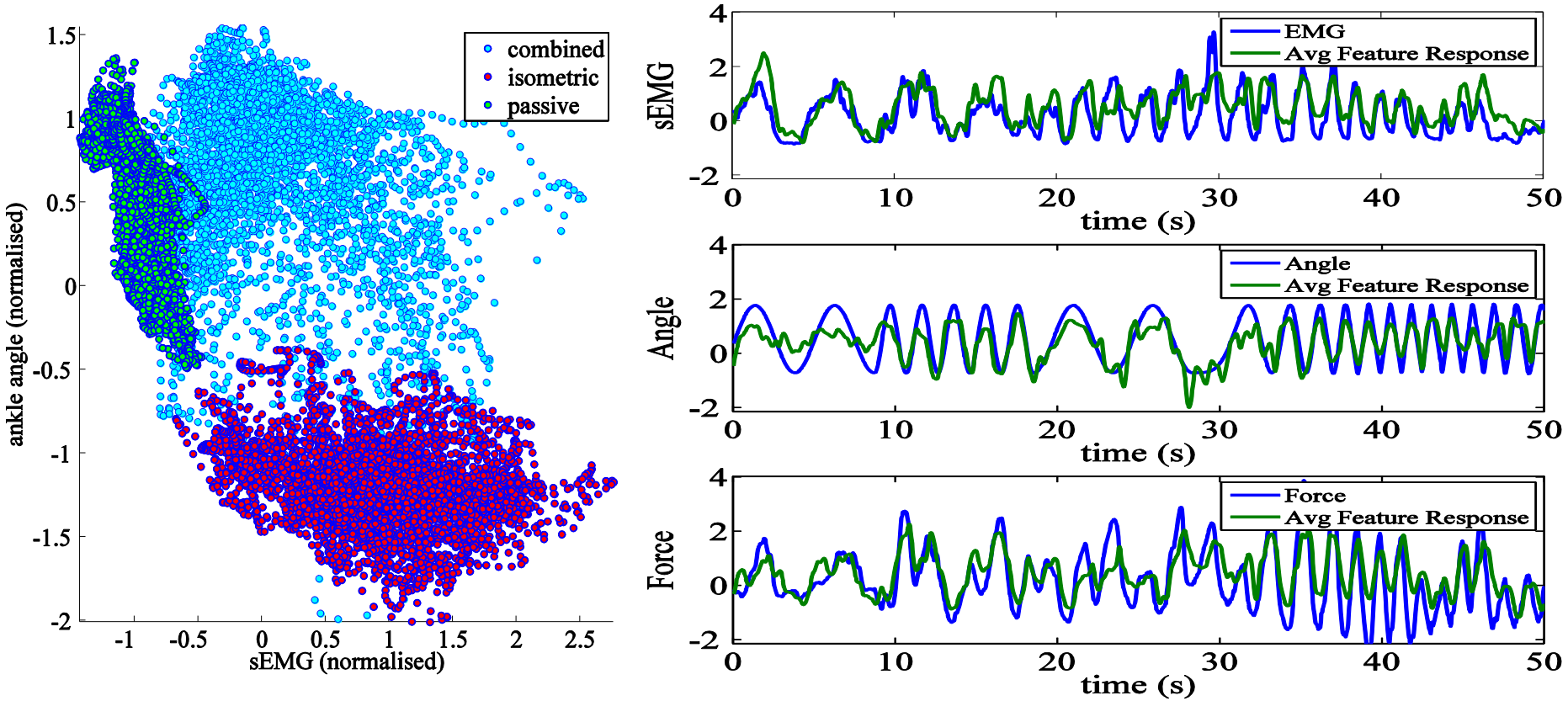


Figure 51. Averaged SO GBRBM feature response over all trials (for a the representative case in Figure 49) for the top 3 features in terms of correlation with sEMG (top), ankle angle (middle), and force (bottom). The averages of the top 3 raw features contain enough information to predict the respective signals in isolation of one another, without any combinatorial design (weighting the combinations of features). Left: shows a scatter plot of the averaged top 3 features in terms of correlation with sEMG and ankle angle over all 3 trials. Right: shows a selected subsection of the averaged top 3 features in terms of correlation with sEMG, ankle angle in the combined-function trial. Notice the remarkable level of detail to which joint angle can be separated from sEMG/force in the combined case.

5.4.3 Linear Estimation of Muscle States from GBRBM Features

The top 5 (Spearman rank) average GBRBM feature responses of images of MG and SO have revealed that there are features which appear to respond to changes in muscle state that were caused by completely different internal/external factors. These correlations exist naturally within the model without optimisation (weighting of the responses to improve the prediction). The results presented here show the effect of linear optimisation of those feature responses. Linear regression models were used to weight every feature in order to optimise that prediction on a per-person basis, with 20-fold cross validation. The results (see Figure 52, Figure 53, Figure 54, and Figure 55) on the following 3 pages show that weighted linear combinations of GBRBM feature responses of raw images of muscle can accurately predict the internal state of the muscle, in terms of active contraction (sEMG), ankle joint angle, and force production. The results show that prediction of muscle states was less accurate in the combined case, which is expected. Results also show that the passive joint angle states are predicted with the greatest accuracy over the isometric active contraction states.

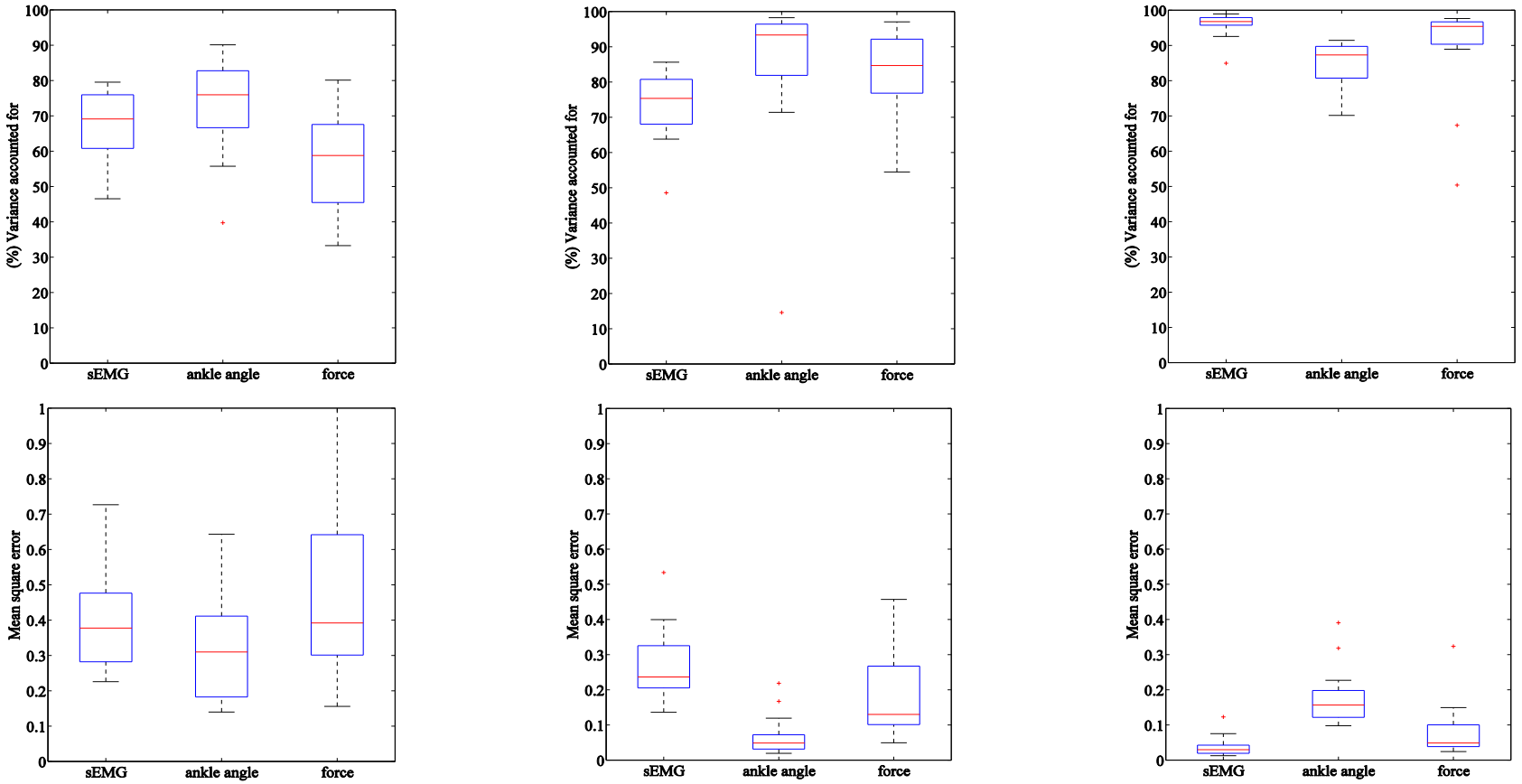


Figure 52. 20-fold cross validation results for linear prediction of muscle states for the MG GBRBM model over all 20 participants. Left: shows results for combined trials, Middle: shows results for isometric trials, and Right: shows results for passive trails. Results are presented in two forms Top: percentage of variance accounted for (%VAF), and Bottom: mean square error (MSE). The combined trials demonstrate the poorest relative performance, and in general it can be said that when there is activity, the prediction error of that activity increases (e.g. see sEMG in (b)). See Table 8 for results in real values.

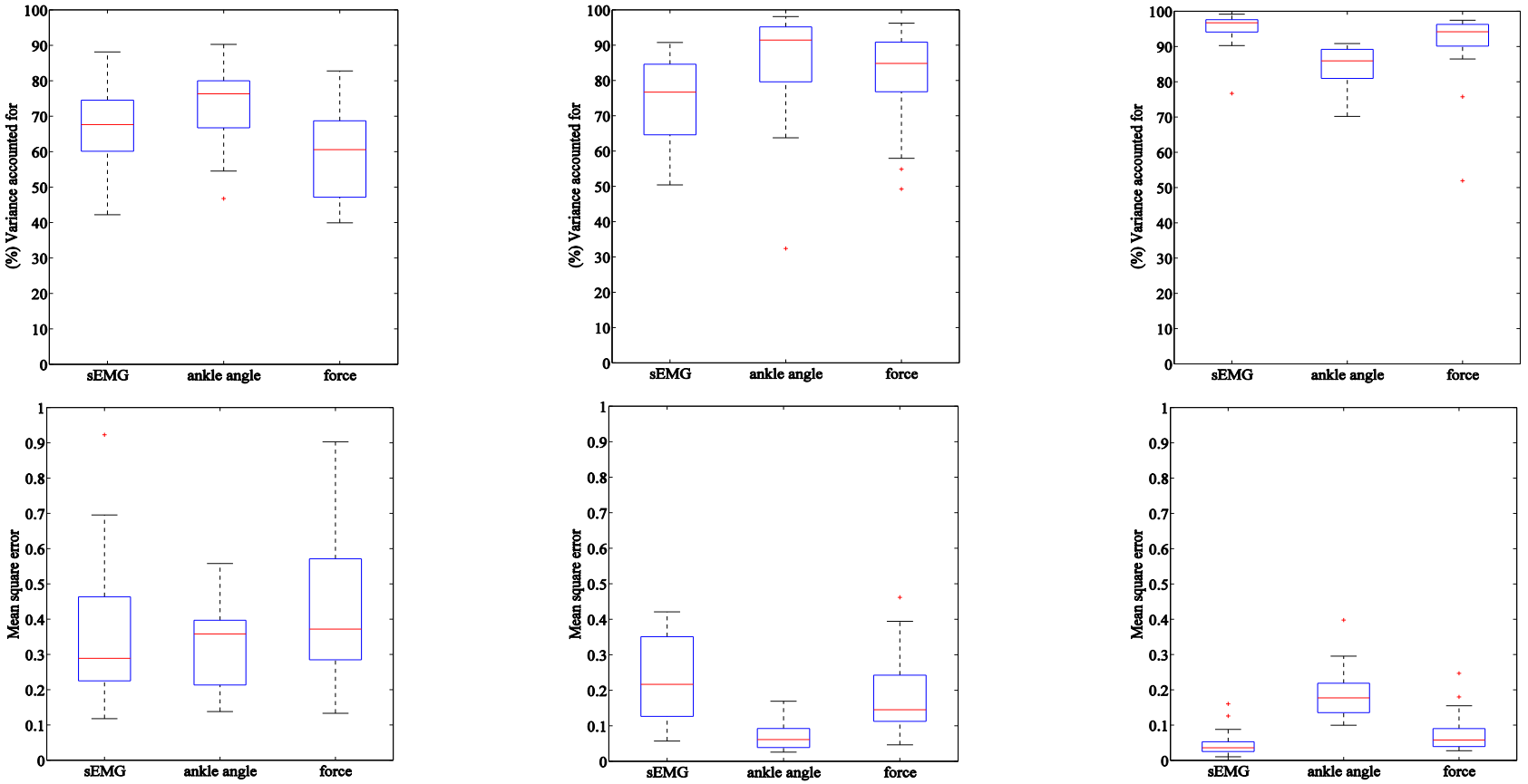


Figure 53. 20-fold cross validation results for linear prediction of muscle states for the SO GBRBM model over all 20 participants. Left: shows results for combined trials, Middle: shows results for isometric trials, and Right: shows results for passive trails. Results are presented in two forms Top: percentage of variance accounted for (%VAF), and Bottom: mean square error (MSE). The combined trials demonstrate the poorest relative performance, and in general it can be said that when there is activity, the prediction error of that activity increases (e.g. see sEMG in (b)). See Table 8 for results in real values.

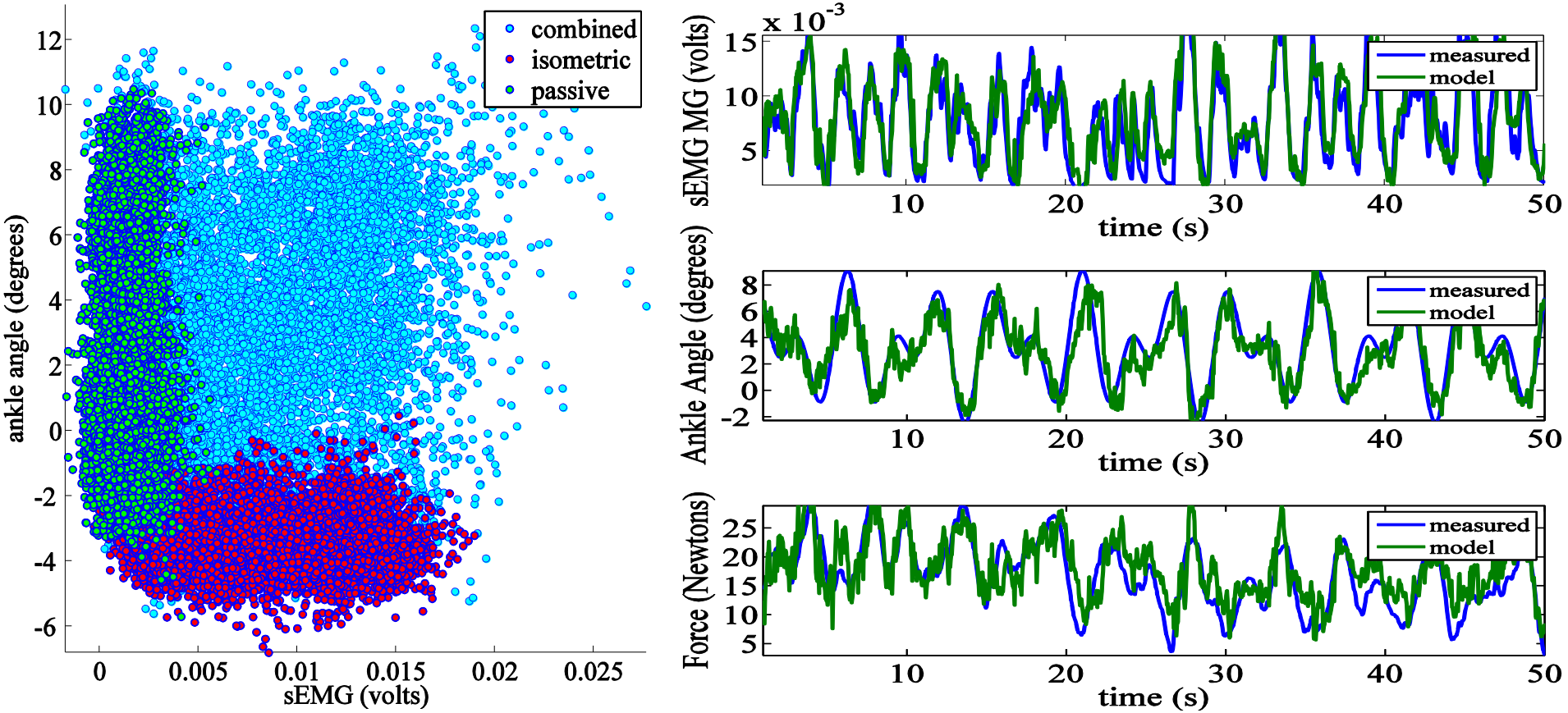


Figure 54. 20-fold cross validation results for linear prediction of muscle states (sEMG, joint angle, force) from MG GBRBM feature responses over all 3 trial types for a typical participant (same participant in Figure 50). Left: shows a scatter plot of the prediction of ankle angle and sEMG over all 3 trial types. Right: shows a subsection of predictions in the combined case. These results support the evidence (given by raw correlations) that there are stable relationships over time between GBRBM feature responses and externally measured muscle states.

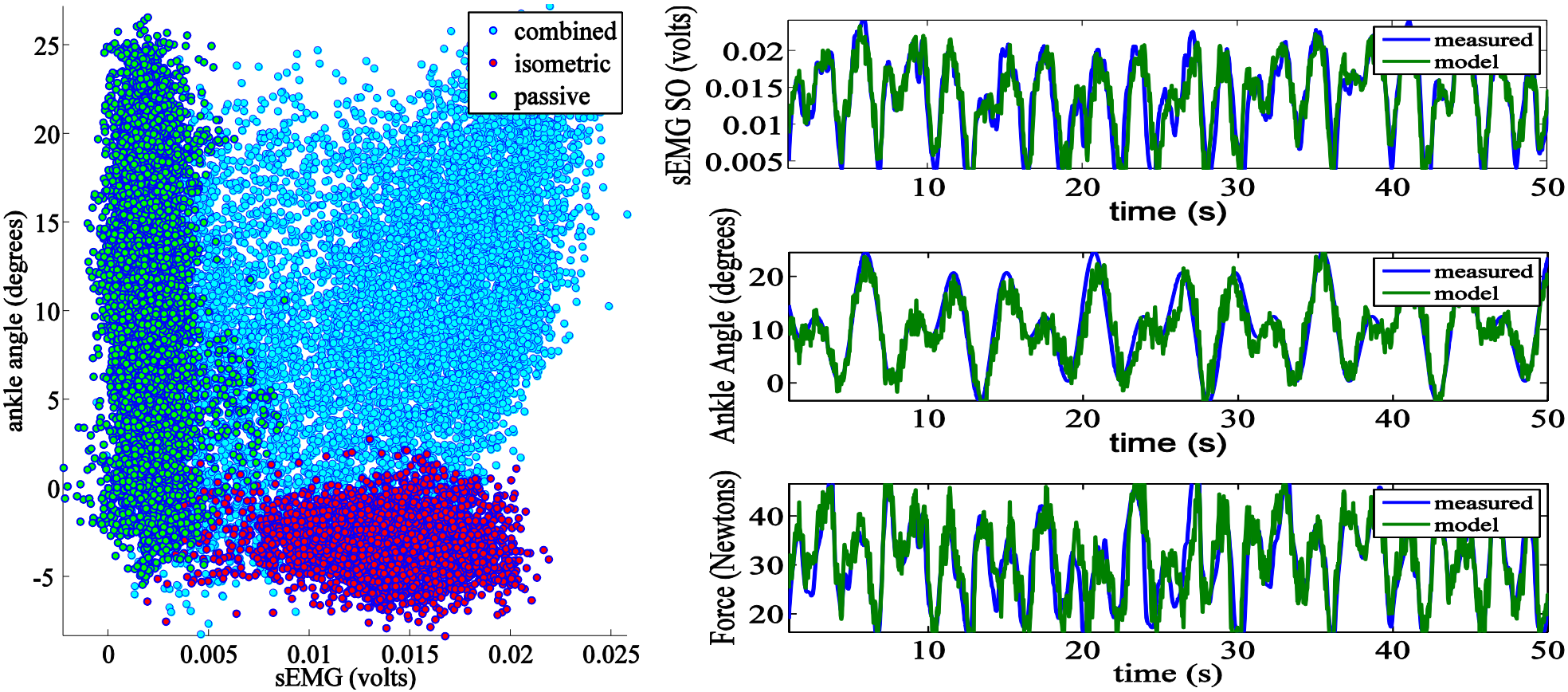


Figure 55. 20-fold cross validation results for linear prediction of muscle states (sEMG, joint angle, force) from SO GBRBM feature responses over all 3 trials types for a typical participant (same participant in Figure 50). Left: shows a scatter plot of the GBRBM prediction of ankle angle and sEMG over all 3 trial types. Right: shows a subsection of predictions in the combined case. These results support the evidence (given by raw correlations) that there are stable relationships over time between GBRBM feature responses and externally measured muscle states.

Table 8. Cross-validation MSE real value results. This table shows the results of linear optimization of the feature responses in real values (mean ± standard deviation).

Muscle	Condition	EMG (Volts)	Angle (Degrees)	Force (Newtons)
MG	**Combined**	$1.78e\text{-}10V \pm 9.31e\text{-}11V$	$6.07° \pm 4.36°$	$11.49N \pm 14.96N$
	Isometric	$1.28e\text{-}10V \pm 6.72e\text{-}11V$	$0.99° \pm 0.93°$	$4.93N \pm 7.79N$
	Passsive	$1.15e\text{-}11V \pm 5.77e\text{-}12V$	$4.53° \pm 3.42°$	$1.48N \pm 0.99N$
SO	**Combined**	$1.07e\text{-}10V \pm 1.63e\text{-}10V$	$6.14° \pm 3.79°$	$14.28N \pm 19.04N$
	Isometric	$6.37e\text{-}11V \pm 1.08e\text{-}10V$	$1.21° \pm 1.21°$	$6.06N \pm 9.04N$
	Passsive	$9.61e\text{-}12V \pm 1.12e\text{-}11V$	$4.16° \pm 1.88°$	$1.87N \pm 2.01N$

5.5 Discussion & Conclusions

This chapter has presented a general investigation into the information content of ultrasound with respect to dynamic human skeletal calf muscle. Specifically, this chapter has tested the hypothesis that muscle states are unique and independently determined by muscle activation and ankle rotation. The results presented in this chapter have shown that GBRBMs can be used to create robust, time-invariant feature representations of skeletal muscle states via ultrasonography, which has been used to exploit the information content of ultrasound with respect to dynamic human skeletal muscle. Results have also observed that the GBRBM features produce responses from images in a sequence, which proportionally predict the individual inputs that determined those states (muscle activity, joint angle). Because the GBRBM features were static, that result must mean that ultrasound-observed muscle states determined by muscle activity and joint angle are unique.

The use of GBRBMs for modelling skeletal muscle states is novel and represents a wide departure from the standard approach, feature tracking [1, 60]. GBRBM features provide a time-invariant representation of the appearance of observed muscle via ultrasound. Those states are related to the states observed in previous and successive frames; GBRBMs output a distance metric between states since features that respond to a particular state change proportionally with the (probabilistic) distance from that state. This property of a GBRBM means that over a time series of feature responses features can be used to track propagation between states without the possibility of drift.

The correlations between the changes in the responses of features over a series of combined/isolated function trials and the externally measured signals (sEMG, ankle angle, and force) was shown to be stable over time (see Figure 49). Furthermore, a representative example of this demonstrated that linear combinations of features were able to respond (proportionally) to independent state changes without optimising/weighting the combinations (see Figure 50 and Figure 51). Linear regression models were then used to demonstrate that feature responses could be weighted in order to accurately predict the externally measured states of the muscle for every frame in a sequence, independent of the history of states in the sequence (see Figure 54 and Figure 55). The fact that this can be done in

combined cases without priors (history) implies that the determining factors of muscle shape are encoded in unique internal muscle configurations.

Both GBRBM models predicted externally measured states with a high degree of accuracy (see Figure 52). It could be said that the SO GBRBM model gave slightly less accurate predictions of muscle state. However, an argument can be made that the experiment was not designed around the SO muscle; MG sEMG alone was used to drive participant feedback. One possible reason for better MG model performance could be that the fascicles are more easily visible in the MG muscle than in SO. One could even go as far as to say that many SO muscles had no identifiable fascicle curvature or pennation angle due to attenuation and/or difficulty in optimising a viewpoint with the ultrasound probe. These reasons only make the SO results more profound, as they suggest that fascicles orientation/curvature is not the only descriptor of muscle state that encodes the information about muscle activity and joint angle. The receptive field of the SO GBRBM supports this assertion (Figure 45), showing that there are very few features that appear to model fascicle orientation and curvature like those of the MG GBRBM.

Future work should examine the inter-participant generalisation of feature responses, asking whether features that predict the input to the states of a particular participant's muscle, also describe/predict the inputs to the states of another participant's muscle. Further to this investigation, one could also ask 'what other information is encoded in the state of muscle?' (e.g. fatigue, pain, etc…). While this work used linear models for estimation of muscle states, the use of non-linear models can help to resolve the expected inter-participant biases by modelling the nonlinearities between participants [96].

In conclusion, this is the first time that externally measured parameters (force, sEMG, and joint angle) have been estimated directly from ultrasound images of human calf muscle, when there are variations in the joint angle combined with variations in active contraction. This has only previously been achieved in isolated conditions; i.e. passive joint rotation, or isometric active contraction [1, 34, 30, 60, 32]. It is also therefore the first time that this has been achieved in a deep muscle (SO). A generic approach to modelling muscle states has been proposed, which allows experimentally designed models to be created without prior knowledge/assumptions of what is the important/descriptive information in the ultrasound. This approach also allows expert interpretation of the model by examination of the feature responses over time, and examination of the receptive fields of the features. Such an approach might provide a way to model the internal muscle states of more complex muscle groups such as those in the human posterior neck. There are over 12 muscles in the human posterior neck which are simultaneously observable via ultrasound. These muscles have much fewer visible measureable parameters (such as fascicles) than the muscles in the calf and therefore are a good candidate for this

approach, provided a similar segmentation technique can be developed to allow for the muscle region normalisation process prior to GBRBM modelling.

6 Modelling and Segmentation of Multiple Deep Muscles in the Human Posterior Neck via Transverse Ultrasonography

6.1 Abstract

The human posterior neck has over 5 depth-wise muscle layers. Ultrasonography can be used to non-invasively image at least 5 layers of muscle and the spine simultaneously. The morphometry of skeletal muscles in the human neck is architecturally complex, and defining a generalised model of shape/structure of neck muscle is challenging. This study presents a comparison of two methods for identifying 10 muscle regions and the spine directly from transverse ultrasonography. Method one defines the muscles within the MRI domain. Method two defines the muscles within the ultrasound domain. Ultrasound images of 5 muscle layers in the posterior neck were acquired from 23 participants. Image-plane markers were taped either side of the probe, to the neck. An MRI was then obtained at 19 transverse cross-sections of the neck, orthogonal to the spine. For the first method an expert operator annotated 10 muscles and the spine in the MRI image that presented the image-plane markers. Those annotations were then registered to the associated ultrasound image by manual linear transformations and a parametric ultrasound probe deformation function. For the second method a statistical shape model was created using all of the expert registered MRI annotations. A stochastic generative search algorithm was developed using the shape statistics along with an edge-gradient cost function, which were subsequently used to automatically segment all images on a leave one out basis. These two methods were analysed for agreement and consistency in the ultrasound domain. Results show that the two techniques demonstrated over 90% classification agreement, demonstrating that the information content of ultrasonography allows segmentation of 5 layers of muscle in the human posterior neck.

6.2 Introduction

The morphometry of skeletal muscles in the human neck is architecturally complex [5]; the posterior neck contains over 5 symmetrical depth-wise muscle layers (see Figure 56). Muscles in the neck can cross 2 joints and attach to multiple different bones [5]. The size and shape of muscles across a population exhibit wide variability, and cross-sectional areas do not scale proportionally with body height and/or weight [5], furthermore there are significant differences in muscle shape between genders [6]. This presents a challenge in creating generalised models of skeletal neck muscle shape/architecture. The human neck muscles are also functionally complex [7], while also exhibiting functional redundancy across muscles. Kinematic analysis of head orientation can reveal force loads in

the neck [98, 99] however, due to muscle groupings (functional redundancy), complex muscle structure [5], and the counterintuitive nature of muscle activation [7], information extraction at the level of the muscle is not currently possible. Surface electromyography (sEMG) can measure motor unit activation during contraction in superficial and partially superficial muscles in the neck [100], although it has been argued that sEMG is not appropriate for measurement of splenius capitis [101]. Intramuscular EMG (iEMG) is required to measure contraction in deep muscles. However, it is impractically invasive, time-consuming, has a relatively small measurement volume ($1mm^3$), and realistically only allows simultaneous acquisition of no more than 2 deep muscles in the posterior neck [8]. Furthermore, there can be no certainty about the clarity of the positioning of intramuscular electrodes within the desired muscle [8]. Research has documented the numerous methodological issues with sEMG and iEMG, such as cross-talk, electrical interference, mechanical artefacts, and activity of other muscles [102].

Imaging technology such as Magnetic Resonance Imaging (MRI) and ultrasonography allow non-invasive imaging of cross-sectional areas of every layer of muscle in the neck [6, 103, 104, 105]. MRI is infeasible for detailed analysis of skeletal muscle function due to imaging at very low temporal frequency ($\approx 0.0033Hz$). Studies have shown that functional MRI (fMRI) has sufficient temporal resolution ($0.5Hz$) to measure contractions in deep cervical flexors [105]. However, image quality is severely impaired. In comparison to MRI, ultrasonography has higher temporal frequency ($\approx 25Hz$) and has lower image quality. In comparison to fMRI, ultrasonography has higher temporal frequency and higher image quality. Furthermore, studies have shown that ultrasonography can be just as accurate as MRI and computerized tomography (CT) for measuring thickness of deep muscles in the human posterior neck [106]; the authors note that consistency between comparisons of the different modalities was improved by the use of image-plane markers.

Previous work has considered shape parameter measurement consistency in the deep multifidii via ultrasonography, concluding that it was a reliable method for measuring muscle dimensions, both while at rest, and under contraction [104, 107, 103]. A survey on ultrasonography of the cervical muscle [108] concludes that there is insufficient literature on assessment of the cervical muscles via rehabilitative ultrasound imaging (RUSI), and that there is a need for proper identification of muscle boundaries, using landmarks and knowledge of functional anatomy. The authors further state that standardised positions of subjects and ultrasound transducers are important for statistical analysis of shape parameter measurements.

Automatic segmentation of ultrasonography of the human posterior neck would provide a clinical tool for assisted probe placement, leading to standardised and consistent data acquisition. Such a tool would also provide a platform for further within-muscle, and whole architecture information

extraction [1]. Segmentation of structure from ultrasonography is challenging due to discontinuation of object boundaries, signal attenuation, image artefacts, speckle noise, shadows, and signal dropout [11]. There are examples of automatic carotid artery segmentation via ultrasonography from 2 different view-planes [109, 110] however, those segmentation problems are much more trivial; high contrast, single segment problems. The most promising attempts at segmentation of semi-rigid (well-defined) regions in ultrasonography involve the use of shape and/or texture statistics to inform semi-automatic segmentation of single segment regions [14, 18]. Active Shape Models (ASM; [17]) provide a powerful method of combining shape and texture statistics to regularise heuristic segmentation algorithms, although they require shape statistics built up from many annotations of shape boundaries over a population. Previous work has used ASMs to achieve fully automatic segmentation of a 2-layer skeletal muscle system in the human triceps surae [1]. However, annotating images of the human neck – directly from ultrasound – is extremely difficult due to the complexity of the architecture of the neck [5], and the challenging image quality [11].

The primary goal of this work is to determine the extent to which the information content of neck ultrasound images allows consistent, anatomically valid segmentation of individual neck muscle. In order to achieve this, two different methodological approaches are used and compared for agreement. The first method defines the specific, individual segmentation within the MRI domain which favours spatial, anatomical accuracy. The second method defines the specific individual segmentation within the ultrasound domain which uses the information content of the ultrasound images.

The first method uses expert annotation of the high resolution MRI images. The annotation specific to that individual is registered to the ultrasound domain with only minimal single adjustment of shape required to accommodate progressive, unidirectional deformation of muscle structures due to pressure of the probe. The second method uses a statistical shape model derived from registered MRI annotations from 23 participants. The shape model defines the correct relational structure of the ten neck muscle segments and the degree of compression resulting from ultrasound probe placement. This anatomically correct, general shape model has multiple degrees of freedom for adjusting shape to individuals while preserving those correct anatomical relationships between muscle segments. Within the ultrasound domain, the general model is adjusted, automatically to the grey-level gradients which are hypothesised to identify muscle boundaries. The open question is whether grey-level gradients are sufficient in quality and contain the right information to correctly identify anatomical muscle segments.

If successful, the MRI-informed ultrasound annotations will provide a protocol for annotating complex ultrasound images. The automatic segmentation algorithm will provide a potential tool for regularising

data collection, and for extracting within-muscle information about shape and motion from 5 layers of muscle and the spine in the human posterior neck.

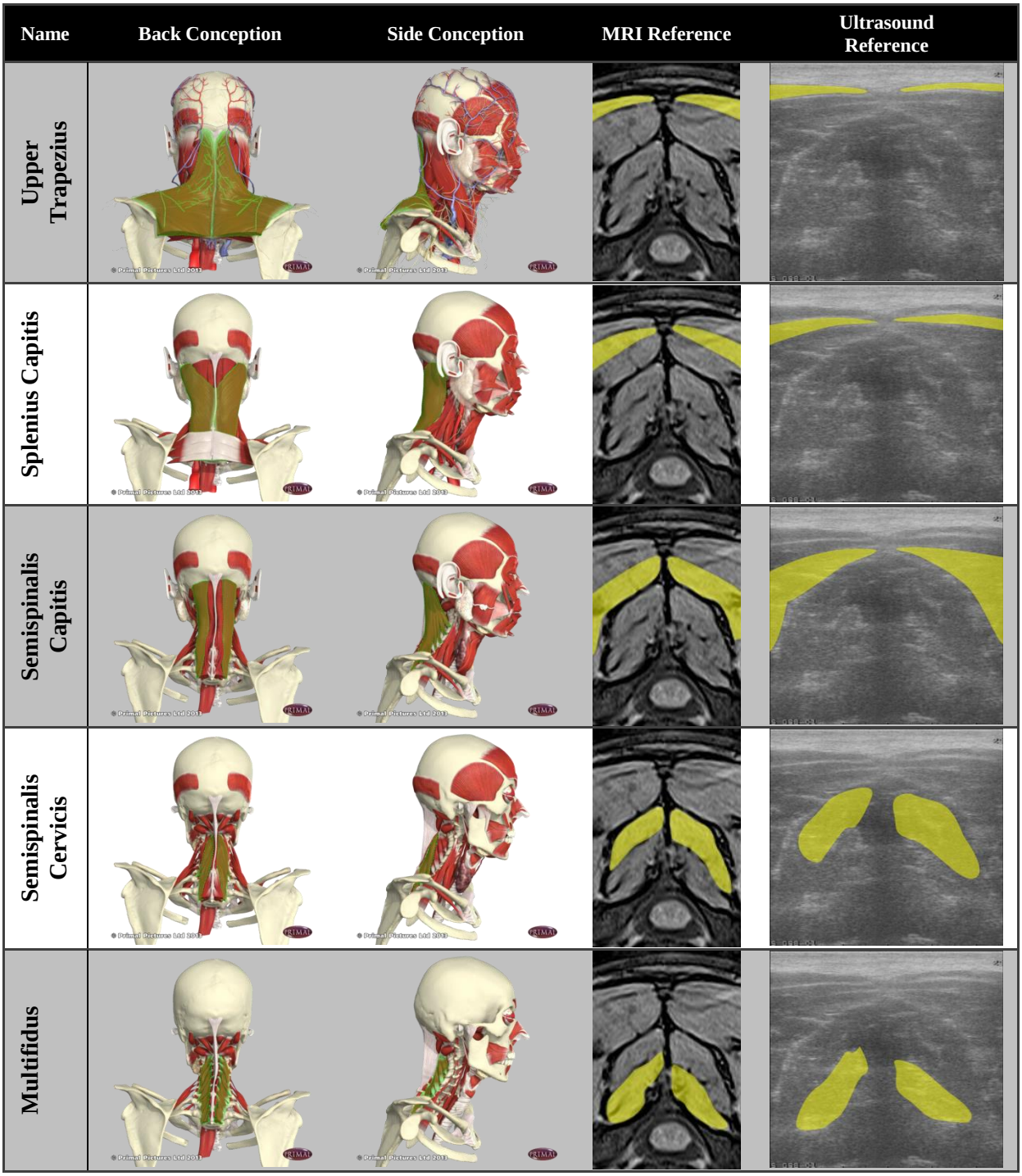


Figure 56. In order of superficial to deep muscles, this figure shows the main muscles of interest for this investigation (shown in green on the back and side conception images, and opaque yellow on the MRI and ultrasound reference images); conceptual images from [111]. The ultrasound image represents a $\approx 5cm^2$ transverse cross-section of the back of the neck where, from the top of each image to the bottom is skin to bone. The MRI image is shown in approximately the same dimensions, and in the same orientation.

6.3 METHODS

This section describes the two proposed methods of annotating ultrasound images of the posterior neck. Method #1 is manual image annotation within the MRI domain, registered to the ultrasound. Method #2 is automatic segmentation of ultrasound images within the ultrasound domain using shape statistics derived from an MRI derived dataset of expert annotations. Before describing the methods in detail, the data collection procedure is explained. Data collection involved the acquisition of ultrasonography and associated MRI of human transverse posterior neck, using image-plane markers to help locate the same cross-sectional areas in both imaging modalities.

6.3.1 DATA COLLECTION PROTOCOL

Figure 57. Ultrasound image acquisition set-up. This figure shows the t-shaped ultrasound probe used to image the muscles in the posterior neck of the participant. In the image on the left, one of the cod liver oil capsules can be seen just to the right of the probe. The participant stands facing forward with their head upright. The participant is shown in two orientations simply to illustrate where the probe is place, and its angle relative to the neck.

Data were collected from 24 participants (14 males, 10 females aged between 18 and 50). Images of their posterior neck muscles were collected via transverse ultrasonography. Firstly, a T-shaped ultrasound probe was held to the back of the neck, in a transverse view, just above C7 in the vertebra (see Figure 57). A single frame was recorded with each participant in a relaxed upright standing position. Two Cod liver oil capsules were placed either side of the probe (taped to the neck) to mark the image-plane. The probe was removed, leaving the capsules in place, and an MRI scan was then performed with the participant lying supine on the scanning bed and their neck positioned central within a cervical imaging coil. Transverse scans were performed in a range from the upper jaw line to

the clavicle, at an angle perpendicular to back of the (sagittal) neck, in 19 even sections (see Figure 58).

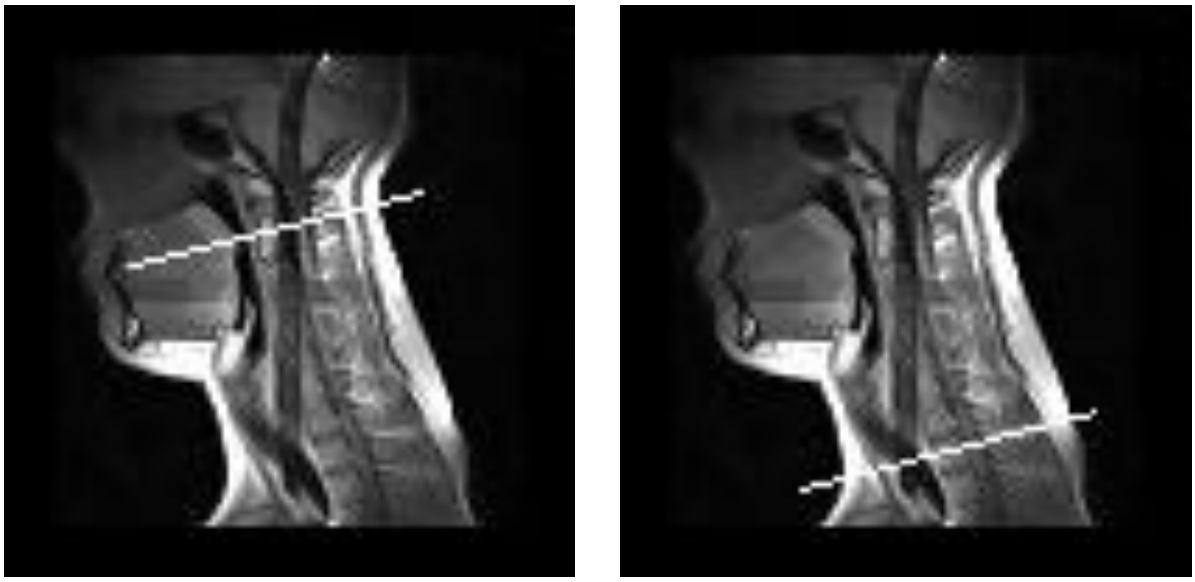


Figure 58. MRI scanning range shows the scanning angle and the upper limit (left) and the lower limit (right).

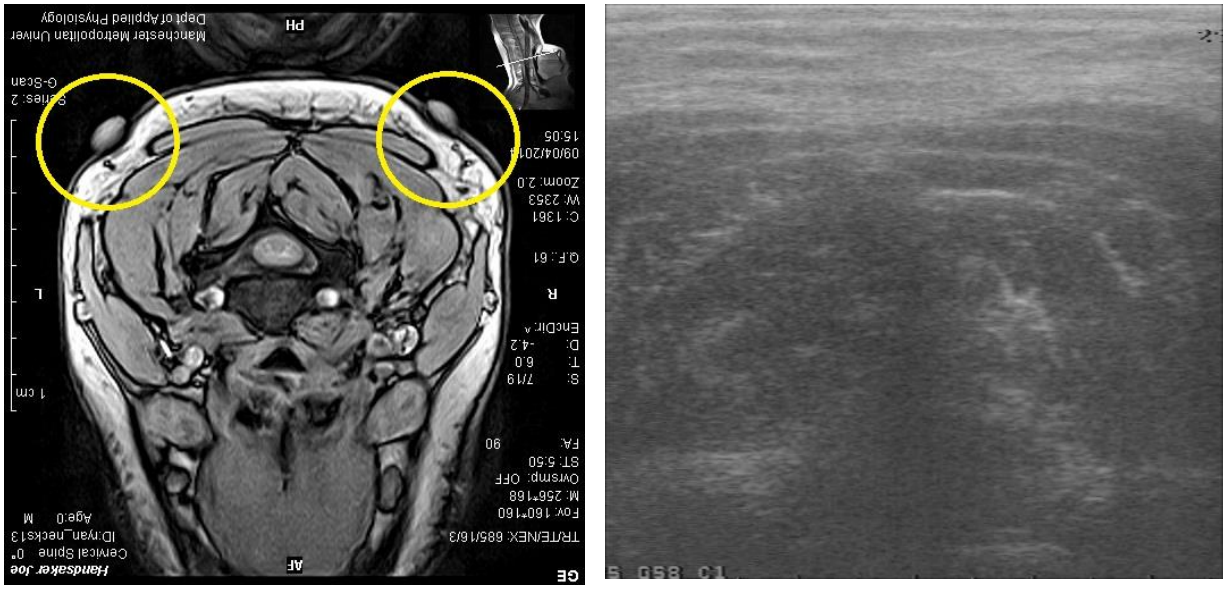


Figure 59. Left: MRI showing Cod liver oil capsules (within yellow circles). Right: associated ultrasound image.

Within the set of 19 MRI images, an expert selected the image showing the two cod liver oil capsules (see Figure 59), which approximately identifies the image plane of the previously recoded ultrasound image. Accepting arbitrary small differences in image-plane orientation and position, these images can now be used to guide annotations of the recorded ultrasound images.

6.3.1.1 Ethical Approval

These experiments were approved by the Research Ethics Committee of the Faculty of Science and Engineering, Manchester Metropolitan University (MMU). Participants gave (written) informed consent to these experiments, which conformed to the standards set by the latest revision of the Declaration of Helsinki. Experiments were performed at the Cognitive Motor Function laboratory, in the School of Healthcare Science at MMU.

6.3.2 IMAGE ANNOTATION PROTOCOL

Following data collection, a human expert then annotated the MRI images, identifying the contours of the following 11 regions in every participant (ordered *superficial* to *deep*):

1. Upper Trapezius (Left|Right)
2. Splenius Capitis (Left|Right)
3. Semispinalis Capitis (Left|Right)
4. Semispinalis Cervicis (Left|Right)
5. Multifidus (Left|Right)
6. Spinal Cord (Centre)

To regulate the annotation process, a simple GUI was implemented using Matlab's keyboard and mouse function window callback interface [80]. Functions were developed that allowed an 11 segment mark-up of manually selected images (MRI images presenting Cod liver oil capsules). The expert was instructed to mark the mid-lateral portion of the neck in order to keep segment boundaries roughly elliptical for shape standardisation (see section 6.3.4.2 and Figure 63). The expert was allowed to cycle through each segment, selecting erroneous markings for removal and subsequent replacement. The MRI images – along with their corresponding ultrasound image – and their annotations were saved. Using the MRI scaling information ($256pixels = 8cm$) and the ultrasound scaling information ($467pixels_x = 5.25cm \mid 440pixels_y = 5cm$) the contours were scaled up from MRI to ultrasound dimensions.

6.3.3 METHOD #1: MRI DEFINED ULTRASOUND IMAGE ANNOTATION

6.3.3.1 Contour Ultrasound Registration Protocol

Following the MRI annotation stage, the expert then went through a contour registration process, in which the task was to manually fit the scaled-up contours to their corresponding ultrasound image. As previously stated, it is expected that when the probe is pressed onto the back of the neck, the muscles directly beneath the probe's transducers (within the image plane) will change shape and deform. It is also noted that this shape change might decay with depth (distance from the probe) due to the resistance arising from the layered structure and stiffness of the muscles. In order to efficiently estimate these parameters (shape change and stiffness), it was decided that a function should be

implemented. The following equations to approximate depth displacement of contour points were created:

First, the data is scaled to have custom asymptotes in the *x*-plane according to,

$$P'_x = \frac{P_x - \Theta_L}{\Theta_U - \Theta_L},$$

and fixed asymptotes of 0 and 1 in the *y*-plane according to,

$$P'_y = \frac{P_y - \min(P_y)}{\max(P_y) - \min(P_y)},$$

where P is a 2D vector of contour mark-ups, $\min(P_y)$ and $\max(P_y)$ are the minimum and maximum of the depth coordinates, and Θ_L and Θ_U are lower (L) and upper (U) asymptotes defined by,

$$\Theta_L = \min(P_x) - \mathrm{W}\big(\max(P_x) - \min(P_x)\big),$$

and

$$\Theta_U = \max(P_x) - \mathrm{W}\big(\max(P_x) - \min(P_x)\big),$$

Equation 9

where W is a tuneable parameter that increases/decreases asymptotes and consequently the wavelength of the sine term in the following equation:

$$P''_y = \big(P'_y + \sin(P'_x\pi) \odot E\big)\Big(\max(P_y) - \min(P_y)\Big) + \min(P_y),$$

Equation 10

where P' is the normalised (min-max) 2D vector of x/y contour coordinates, $\sin(P'_x\pi)$ is a parametric curvature term that represents the curvature of the back of the neck (approximating the shape of the displacement), and E is a parametric stiffness term that represents the resistance to probe pressure as a function of depth. The other terms re-scale the data its original scale. The parametric stiffness term, E is defined as,

$$E = \frac{1}{1 + e^{(2(P'y + B) - 1)S}} \frac{1}{D},$$

Equation 11

where the right hand side is the standard log-sigmoid transfer function with tuneable parameters S, B, and D that control the slope, offset, and magnitude of the log-sig function, respectively.

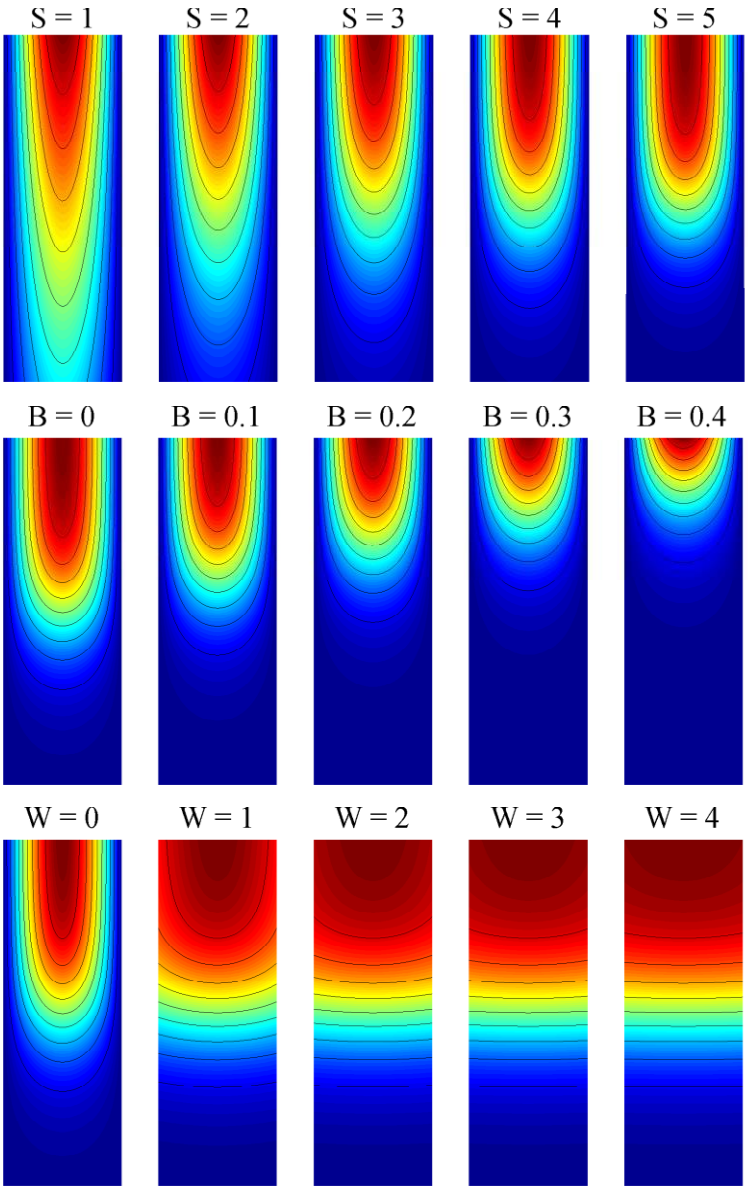


Figure 60. Heat maps show displacement fields for different values of $\boldsymbol{S}$, $\boldsymbol{B}$, and $\boldsymbol{W}$ respectively (all other parameters were constant), where dark blue represents small displacement, and dark red represents large displacement.

With a way to approximate the depth displacement of the muscle contours, another simple GUI was created using Matlab's keyboard and mouse functions window callback interface [80]. Via the GUI, the expert was allowed to translate and rotate the contour to optimise its alignment with the visible deep ultrasound gradients/edges. Restricting the registration to linear transformations regularises the fitting process and retains all the original shape variance over the contour population. Then, the expert was allowed to modify four parameters, W, S, B and D according to Equation 9 and Equation 11, to correct the superficial alignment error. This process gives the expert the power to annotate an ultrasound image by using the MRI shape information and simulating probe placement, maximising the alignment with the visible edges/gradients in the ultrasound image (see Figure 61 and Figure 62).

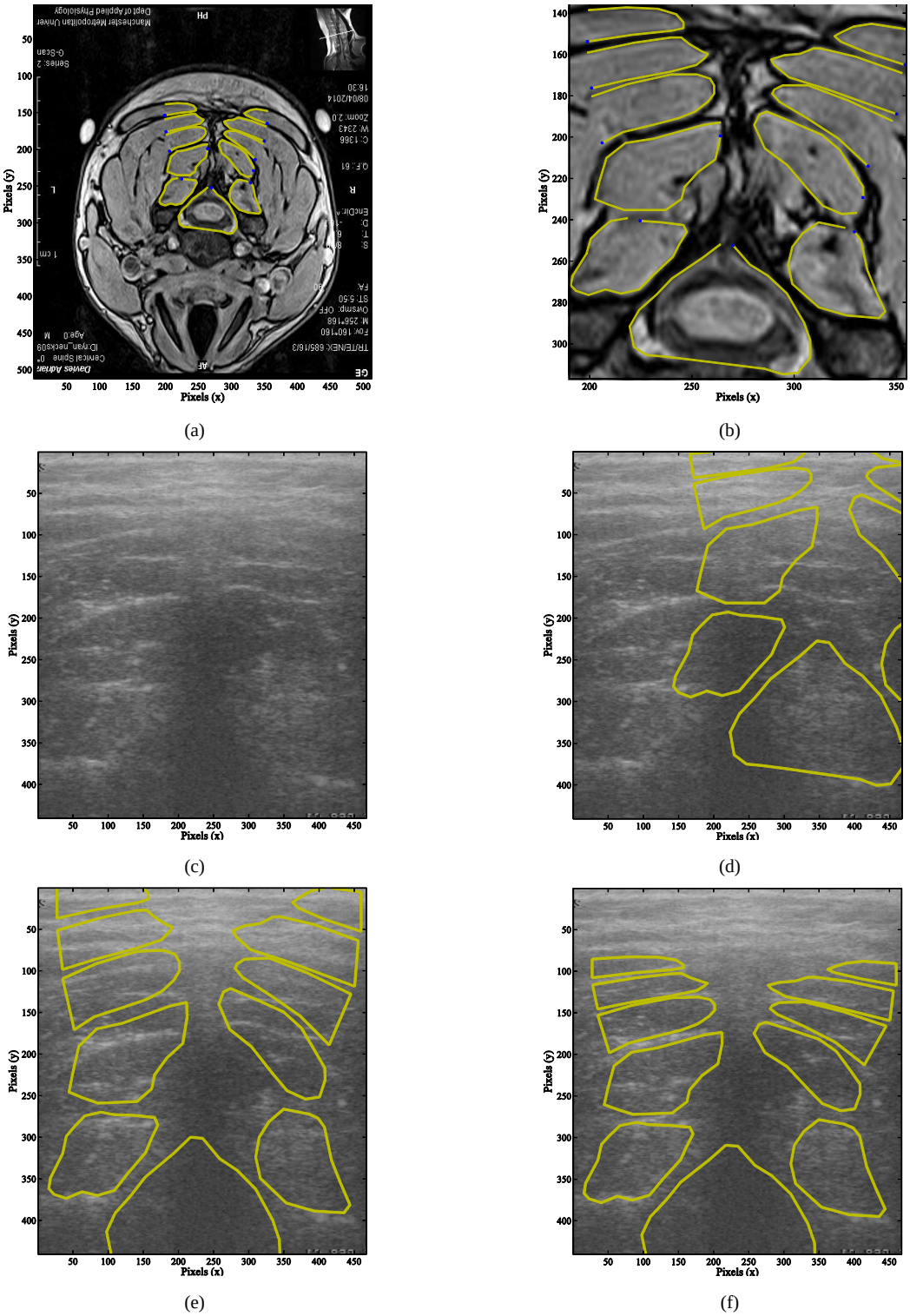


Figure 61. Human expert contour registration process: (a) and (b) expert-marked contour overlays a participant's neck MRI image. (c) participant's corresponding neck ultrasound image. (d) MRI contours scaled to the metric dimensions of ultrasound image. (e) expert translates and rotates the MRI contours to best match deep muscle boundary gradients. (f) expert applies contour squashing function (Equation 10) to match ultrasound image gradients.

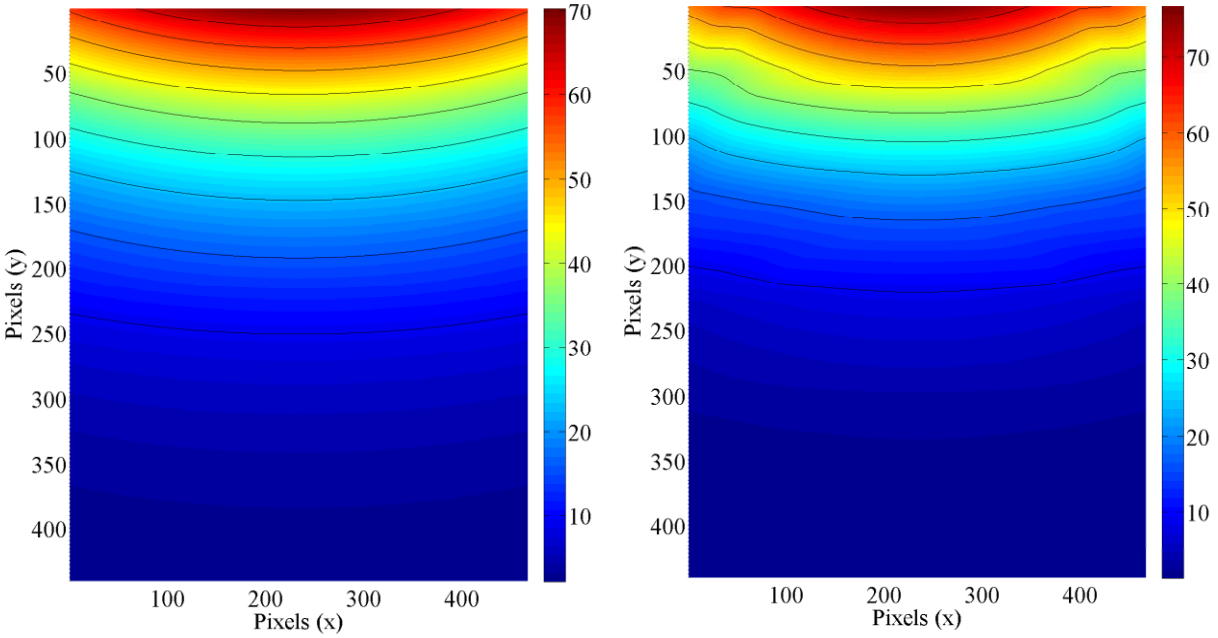


Figure 62. Average pixel displacement over the entire dataset after the completed expert registration process. Left: shows the mean pixel displacement. Right: shows the median pixel displacement. Notice that the gradient of the displacement shows that the superficial muscles are more affected by probe placement than the deeper muscles. Note that the arc of the gradient might relate to the arc of the skin at the back of the neck, though it is not considered paramount to investigate this statement for the immediate goals of this chapter.

6.3.4 Method #2: Ultrasound Defined Automatic Segmentation

6.3.4.1 Modelling Shape Statistics with Principal Component Analysis

Building models of semi-rigid shape from annotations requires standard representation of the distribution of points that make up the shape, so that shapes within the set of annotations are comparable. Previous work on the segmentation of human calf muscles (see chapter 4) standardised annotations by restricting the marking of points at fixed distances across the image (in the *x*-plane). This was adopted because there were no referable horizontal landmarks; the annotations therefore marked examples of how the aponeuroses might curve or slope relatively across the image. This technique cannot be used in the on images of transverse posterior neck muscles because the segments present as closed ellipses within the image, and the segment shape variability is large [6, 5].

6.3.4.2 Shape Contour Point Standardisation

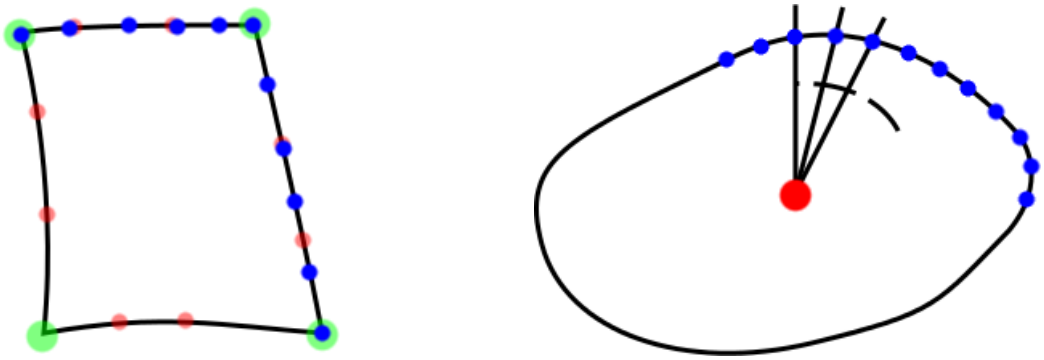


Figure 63. Shows point representations of a rectangular shaped contour and an ellipse shaped contour. Left: the green and red circles over the contour are the expert annotations. The green circles are key points that are used to interpolate over the red circle points to each neighbouring green point at fixed distance intervals, resulting the blue circles (partially filled in). Right: all expert points have been interpolated to form an elliptical polygon. The red circle represents the centroid of the polygon, from which intersections with the polygon edge are computed at fixed angular intervals, resulting in the blue circles (partially filled in). The blue circles in each picture make up a standardised representation of each shape.

For well-defined shapes with obvious landmarks such as rectangles, hands, or faces the standard approach is to identify key points that define important features (e.g. corners in a rectangle), and interpolate between those points. For example, if one was marking a roughly rectangle shape, to standardise the point data one might select the corners and interpolate between those points at a fixed number of intervals. However, for closed circular or elliptical shapes (such as those that present in the transverse posterior neck) there are few obvious landmarks – especially ones that are consistent across people. One approach to solving this issue would be to define a closed polygon with the points, and then from its centroid compute intersections with the edges at fixed angular intervals (see Figure 63). Care should be taken that the orientation of the object is always known; in the case of images of human neck via transverse ultrasonography one can assume the orientation is always the same (the probe always contacts the skin and images the superficial to deep muscle, from top to bottom of the image respectively), and that the only variation is in relative segment location/size and boundary shape.

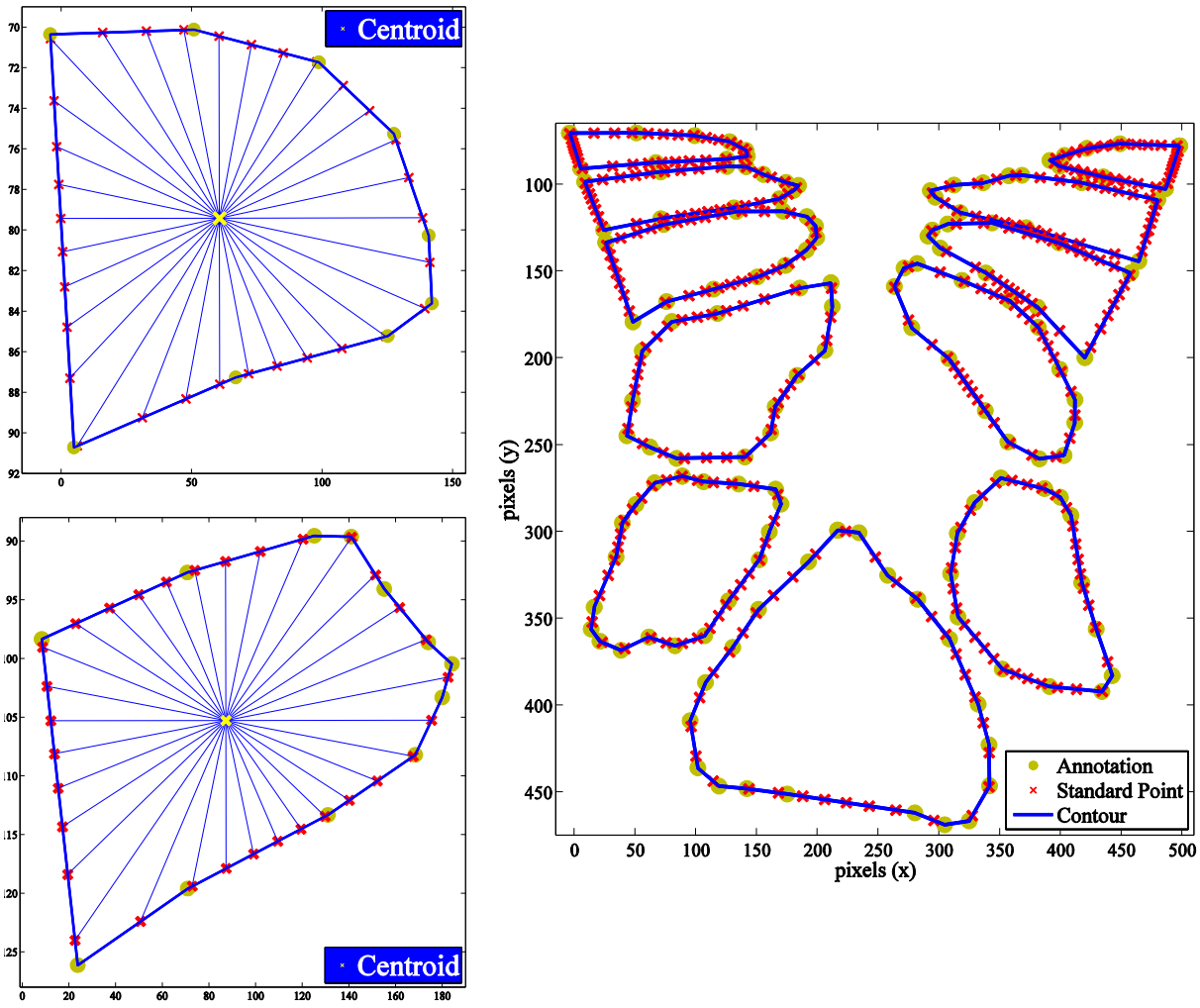


Figure 64. An example expert annotation is standardised by marking fixed angular intervals $\left(\frac{360^o}{32}\right)$ from each segment (polygon) centroid to the segment contour. The green circles represent the original expert annotation, while the red crosses represent the automatically identified intervals according to the centroid (shown in yellow). The two images on the left show a zoom of the upper left two segments froim the image on the right. The red crosses then form a representation of each segment and an overall shape in a way that is comparible with other standardised annotations.

To standardise all of the expert MRI annotations, one first fits a linear polynomial between every point in each segment to construct a closed polygon. Then one computes the centroid (centre of mass) of each polygon according to the following equations:

$$\gamma = P_x \odot P_{y+1} - P_{x+1} \odot P_y,$$

and

$$\varsigma_i = \sum(P_i + P_{i+1}),$$

which compute the cross product between all points P. Then,

$$\alpha = \frac{\sum \gamma}{2},$$

defines the area of the polygon and,

$$C = \frac{\varsigma}{\alpha}\frac{1}{6},$$

Equation 12

computes the centroid (centre of mass) of the polygon. From the centroid, one then computes linear polynomials from the centroid to the edge of the contour at fixed angular intervals $\left(\frac{360^o}{32} = 11.25^o\right)$, marking a point at the intersection of the edge/centroid polynomials (see Figure 64). This was computed using a Mapping Toolbox Matlab function, polyxpoly [112].

6.3.4.3 Shape Modelling with Procrustes Analysis & PCA

To build a model of likely shape appearance in the domain of ultrasound, according to the ASM paradigm, after standardisation one must then register all shapes by linear transformations to some 'golden' example of that shape. This means that, given a perfect example of a shape, one would like to register every other standardised point model of that shape such that its defining features are the minimum Euclidean distance from the same defining features of the 'golden' example. With such a small population of people (23) in the dataset, and no true idea of what might be considered a 'golden' example, the approach taken here was to use the overall mean shape as the 'golden' example. Every shape in the dataset was then registered to that mean (see Figure 65) by minimising the following cost function:

$$J(S_i, G, X) = \left(S_i - x\left(G\begin{bmatrix}\cos(r) & -\sin(r)\\ \sin(r) & \cos(r)\end{bmatrix} + t\right)\right)^2$$

Equation 13

for all shapes S where S_i is a single example shape, G is the overall mean, and t is the translation term, x is the scaling term, and r is the rotation term. All linear transformations were recorded and were later user to guide a global (image-wide) search during segmentation.

Figure 65. Illustrates the shape registration process (often referered to as Procrustes Analysis). Left: shows every example shape overlaid with onanother *before* registration. Right: shows the mean shape in red, and every example shape overlaid with onanother *after* being registered to the mean shape by linear transformations including scaling, rotation, and translation.

When all shapes have been registered to a common shape it is then possible to statistically describe how the contours can possibly vary from that shape. With a comprehensive and accurate point model, and a statistical model of possible variance, it is then possible to guide some heuristic search within the domain of all possible contour configurations, without considering or allowing erroneous contours. PCA is a standard method used for describing the variance of a set of points P and is the standard method used to build an ASM. A principal component model consists of a set of vectors (or components) that describe an N-dimensional direction of variance, where N is the length of the set P. The components are always in rank order of the percentage of the total variance of the set described by each component vector (i.e. the first component is a vector of points that describes the direction of greatest variance, and the last component is a vector of points that describes the direction of least variance). In terms of shape contours, the components will describe the range of allowable variance from the overall mean shape.

There are different methods for extracting the principal components of a dataset; one very common way to do this is by Eigen decomposition of the covariance matrix which is the same method used as in chapter 4:

After mean normalisation, one first computes the covariance matrix with,

$$C = \frac{1}{M} S^T S,$$

where M is the length of the set S, and S is the set of shapes. Then compute by Eigen decomposition the Eigenvalues and Eigenvectors of C satisfying the equation

$$Cv = \lambda v$$

Equation 14

where v is a set of Eigenvectors and λ is a vector of Eigenvalues. Eigenvalues and Eigenvectors of a covariance matrix are exactly equivalent to a set of principal components and their respective variances. In this case Equation 14 was computed using a Statistics Toolbox Matlab function, pca [113]. The resulting Eigenvalues and Eigenvectors along with the mean overall shape now describe every possible mode of variation, and therefore describe every possible shape of human posterior neck (within the samples in the model) via transverse ultrasonography. These statistics can now be used to regularise some segmentation algorithm for any transverse ultrasound image of human neck, provided that there is enough muscle boundary information in the image; if there exists some well-defined set of muscle boundaries that fall within the statistical range of defined shapes in the model, then theoretically some algorithm may be developed that could locate those boundaries. It should be noted that there still remains the possibility of locating erroneous boundaries in the image that also lie within the statistical range of defined shapes in the model.

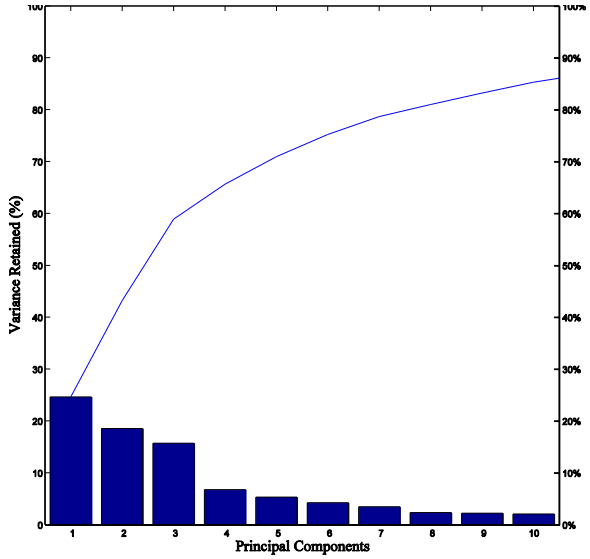


Figure 66. Shows the percentage of the total shape variance described by increasing the number of principal components. Including just the first rank 10 of the total 704 principal components in the model is enough statistical power to account for almost 90% of the total variance in muscle shape in the population of 23 people. The number of components used in the shape model was 33, which retained $> \mathbf{99.9}\%$ of the overall variance.

6.3.4.4 Segmentation Algorithm

In chapter 4 a segmentation algorithm was proposed based on the traditional ASM [17]. That algorithm was applied here to automatically segment the ultrasound images of posterior neck muscles. The traditional ASM captures the transition of pixel intensities over the contour (typically 2 pixels on either side) for each point in the shape, by modelling the gradient over the edge of the shape with PCA. Gradient descent is used to minimise the statistical distance between the current image gradient and the model gradient, by adding weighted linear combinations of retained shape model components to the shape in its current search state. The traditional ASM fitting algorithm is prone to local minima fitting, and does not take advantage of the gradient over the entire edge of the shape. An Active Appearance Model (AAM; [59]) might be more apt for modelling that degree of texture detail however, the technique is notoriously slow.

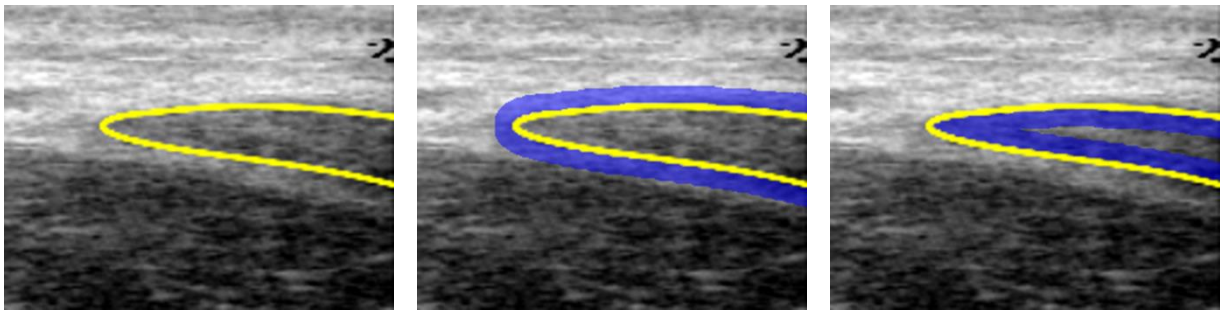


Figure 67. Shows a typical upper trapezius grayscale segment with a generated contour overlay (left), an outer area (middle), and an inner area (right). The difference between outer and inner mean pixel intensities is measured.

In the domain of ultrasound, the pixel intensities characterise the visible tissues and structures to some degree. It is known *a priori* that the boundaries of skeletal muscle (aponeuroses) appear as bright bands of pixel intensity, relative to the internal appearance of the muscle, and this is due to the acoustic impedance properties of muscle and aponeuroses (see chapter 1, subsection 1.3.1). For this reason, it was considered important for robustness to sample the pixel intensities n pixels normal to the contour on the inside and on the outside of the segment (see Figure 67). To avoid convergence to local minima, the fitting technique proposed in chapter 4 was used to segment the previously annotated ultrasound images (see Figure 68). By stochastically searching the space, the proposed algorithm (described below) always has the opportunity to find the global minimum solution, regardless of any previous minima. The term ‘stochastic’ is used rather than random, since randomly searching the space implies that every possible solution is being evaluated, whereas stochastically searching the space implies that the shape statistics are being used to restrict the possible search space.

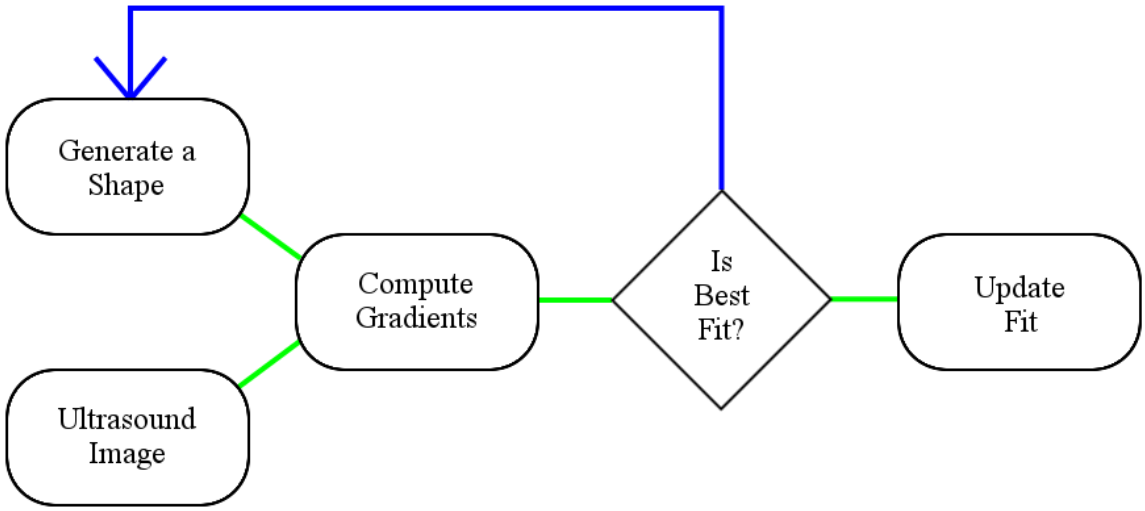


Figure 68. Flow diagram from left to right: Shapes are continually generated from the model and evaluated against the gradients computed from a given image. If by some arbitrary measure of the gradients the fit is considered the best fit in a list of previous fits, that shape is taken as the current best result. This process can run up to some maximum number of iterations in serial, or in parallel, since no random shape evaluation depends on any other shape evaluation. For parallel execution the best fit is determined from a list of costs and associated shapes after execution.

The principal components of the shape model and the mean shape together make it possible to generate realistic example shapes of human posterior transverse neck muscles by adding linear combinations of components, weighted by normally distributed random noise to the mean shape according to,

$$S = \mu + \sum_{i=1}^{i=n} \lambda_i v_i N(0,1),$$

where S is a randomly generated shape, μ is the mean shape, λ_i is the eigenvector of the component i, v_i is the corresponding eigenvalue (variance) and $N(0,1)$ is some normally distributed random noise with 0 mean and unit variance. One can then perform transformations (translation, rotation and scaling) to S within the recorded ranges resulting from the shape registration process (Procrustes Analysis; see Figure 65) for a global (image-wide) search.

A generated shape can be evaluated based on its gradient edge cost (see Figure 67) according to,

$$C(S,I) = \sum_{i=1}^{i=n} (\mu_1 - \mu_2),$$

Equation 15

where $C(S,I)$ is the cost of the fit of shape S, on the image I, n is the number of segments, and μ_1, and μ_2 are the (mean normalised) average (mean) inner and outer pixel intensities, respectively. The minimum of this function therefore locates the shape S with respect to the image I, with the greatest

relative positive difference between the outer pixel region and the inner pixel region, within the constraints of the shape model.

6.3.4.5 Cross-Validation of Segmentation

To assess the performance of the proposed segmentation method a cross-validation approach was used. The cross-validation procedure involves the selection of a subset of the shapes (termed the cross-validation set) after standardisation and normalisation. Then after building a principal component shape model with the remaining data/shapes, the segmentation algorithm is executed on the cross-validation set. Thereafter, the disagreement of the algorithm segmentation with the expert annotation is recorded. The most concise cross-validation method involves repetitions of the process with every shape and annotation taking a turn as the cross-validation set. Due to the stochastic nature of the proposed segmentation method a very high number of iterations (5,000) was used per cross-validation test. The leave-one-out cross-validation algorithm is as follows:

```
1. iterations = 5000;
2. for i = 1 to num_participants
3.         cross_validation_set = participants(i)
4.         test_image = images(i)
5.         training_set = leave_out_participant(participants, i)
6.         model = build_model(training_set)
7.         muscles = segment_image(model, iterations, test_image)
8.         save_results(i, muscles)
9. end
```

6.3.5 TECHNIQUES FOR COMPARISON OF SEGMENTATION METHODS

Four different quantitative techniques were used to compare the two proposed segmentation methods for agreement. The techniques were used to measure agreement in region localisation, overall shape similarity, and majority region classification. The discussion section examines the results of both methods qualitatively. Each comparison technique is defined and justified in a list below:

6.3.5.1 Root Mean Square Error (RMSE)

RMSE is a standard technique for measuring the difference between model-predicted values (either expert annotation, or automatic segmentation) and observed truth values (true segmentation), and is computed with,

$$RMSE = \sqrt{\frac{1}{n}\sum(S - T)^2}$$

Equation 16

where $RMSE$ is a measure of the difference between the boundaries of both the automatic segmentation S, and the expert annotation T, in pixels.

Since both S and T are model-predicted values (i.e. S is predicted by ultrasound defined automatic segmentation, and T is predicted by MRI defined expert annotation), this technique is actually measuring the agreement between S and T, about where the boundaries of the segments lie. The true image segmentation is not known. However, the expert annotations were informed by MRI muscle contour annotations (see Figure 61), and the automatic segmentation uses the gradients of the ultrasound images and a statistical model of shape derived from an anatomically correct dataset (see Figure 68). Thus, a low RMSE implies that ultrasound greyscale gradient information provides a segment boundary identification consistent with the anatomically accurate segmentation defined with the MRI domain.

6.3.5.2 Segment Euclidean centroid distance

The centroid of an arbitrary polygon (shape/segment) is its centre of mass. This technique evaluates the difference between respective expert annotated muscle segments, and automatic segmentation muscle segments. The centroid distances are computed with,

$$CD_i = \sqrt{\sum\left(S_i^{centroid} - T_i^{centroid}\right)^{\circ 2}}$$

Equation 17

where CD is a vector of Euclidean centroid (see Equation 12) distances (in centimetres) between respective muscle segments of the automatic segmentation S and the expert annotation T. $^\circ$ is the Hadamard product symbol. Both $S_i^{centroid}$, and $T_i^{centroid}$ are 2D vectors of $[x\ y]$ position coordinates.

This technique is a measure of agreement between respective muscle segments of the automatic segmentation and the expert annotations, in terms of depth/breadth localisation of muscle segments within the ultrasound image. The expert annotations were informed by MRI muscle contour annotations (see Figure 61), and the automatic segmentation uses the gradients of the ultrasound

images and a statistical model of shape derived from MRI muscle contour annotations (see Figure 68). Thus, a low CD_i for any muscle segment implies that ultrasound grey-scale gradient information provides a region centroid identification consistent with the anatomically accurate segmentation defined with the MRI domain.

6.3.5.3 Segment area agreement

This technique computes a measure of agreement between respective muscle segment areas (pixel classification) of the automatic segmentation and the expert annotations. The 3 superficial muscle layers wrap around the deep muscles to form quasi-crescent structures. However, during MRI annotation the lateral edges of the image were not marked so that crescent shapes could be avoided (see section 6.3.2; where an expert was asked to mark the mid-lateral region of the MRI image to keep segments roughly elliptical). Intuitively, disagreements in the lateral region of these muscle layers should be disregarded when quantifying agreement between expert annotations and automatic segmentations (see Figure 69).

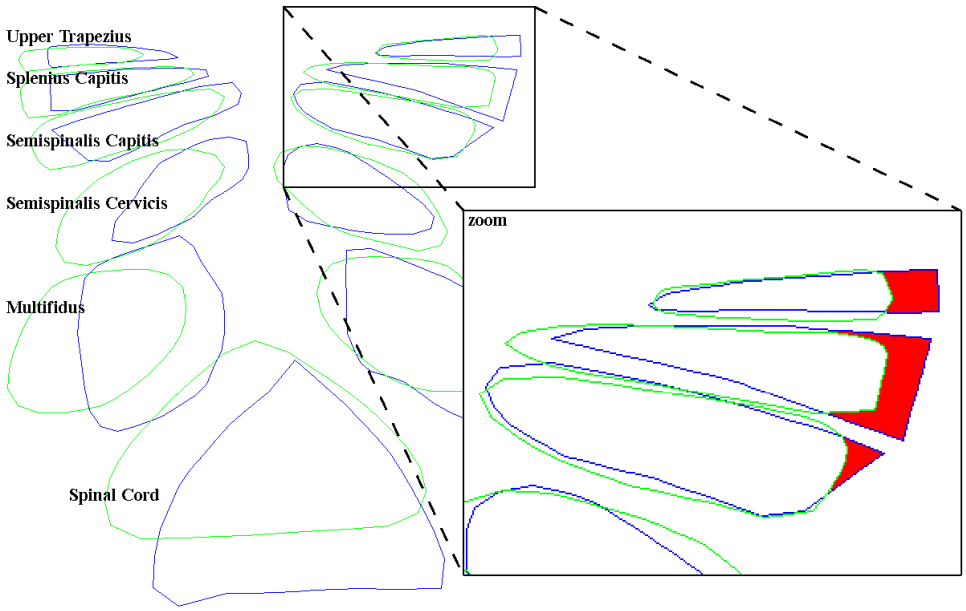


Figure 69. Comparison between expert annotation (blue contour lines) and automatic segmentation (green contour lines) with a zoom view of the 3 most superficial muscle layers on the right. In the zoom view on the right, the 3 red highlighted areas of muscle represent an arbitrary disagreement between the expert annotation and the automatic segmentation.

A primitive way to adjust for this measurement error would be to calculate the proportion of the smallest area segment that intersects with the other,

$$Q = \begin{cases} \mu(S_i) \; if \; \mu(T_i) > \mu(S_i) \\ \mu(T_i) \quad otherwise \end{cases}$$

$$A_i = \frac{S_i \cap T_i}{Q}$$

Equation 18

where A_i is a measure of agreement between the automatic segmentation S and the expert annotation T considering for the segment S_i or T_i that has the smallest area. Q is segment that has the smallest area μ between automatic and annotated segmentations. S_i and T_i are the automatic segmentation and annotation of the segment i, and n is the total number of segments. $\cap$ is an intersection operation.

This technique gives a maximum possible output of 1, which occurs when the smallest segment of S_i and T_i fully intersects with the other. This method gives a primitive measure of agreement between expert annotation and automatic segmentation areas. The technique does not fully account for the comparison error when comparing lateral edges of the top 3 muscle layers. That the expert annotations were informed by MRI muscle contour annotations (see Figure 61), and the automatic segmentation uses the gradients of the ultrasound images and a statistical model of shape derived from MRI muscle contour annotations (see Figure 68). Hence, a high A_i for any muscle segment implies that ultrasound grey-scale gradient information provides an area identification consistent with the anatomically accurate segmentation defined with the MRI domain.

6.3.5.4 Segment classification

This technique quantifies correct-class majority segment classification between the automatic segmentation and the expert annotation. The technique essentially measures the proportion of area agreement between an arbitrary automatic segmentation segment S_i and an expert annotation segment T_j, then takes the highest segment proportion accounted for. If those segments belong to the same class ($i = j$), then the classification is correct. However, if those segments belong to different classes ($i \neq j$), then the classification is incorrect. The segment classification is computed with,

$$f(S_i, T_j) = \frac{S_i \cap T_j}{S_i \cup T_j}$$

$$g(S_i, T) = max\left(f(S_i, T_j)\right) = \begin{cases} 1 \quad if \; i = j \\ 0 \; otherwise \end{cases} \forall j$$

$$\beta = g(S_i, T)$$

Equation 19

where β is a function that computes the correct-class classification between segment S_i of the automatic segmentation and the expert annotation T. $g(S_i, T)$ is a function that computes the largest segment area agreement between segment S_i of the automatic segmentation and all segments of the expert annotation T. The function $g(S_i, T)$ returns 1, meaning that the segments with the largest area of agreement were of the same class, or 0, meaning that the segments with the largest area of agreement were of different classes. $\cap$, and $\cup$ are intersection and union operations respectively. S_i and T_i are segments i and j of the automatic segmentation S and the ecpert annotation T, respectively.

This technique ignores arbitrary area (pixel) and border disagreements, only considering whether the regions classified by both techniques identify the majority of the correct regions and placements over all segments. A value close to 1 implies that ultrasound grey-scale gradient information provides a muscle region identification consistent with the anatomically accurate segmentation defined with the MRI domain.

6.4 RESULTS

The time to complete 5000 iterations of the algorithm with no optimisation, via Matlab, on average (median over all participants) took 306.90 seconds, while the maximum took 310.09 seconds, and the minimum took 306.41 seconds.

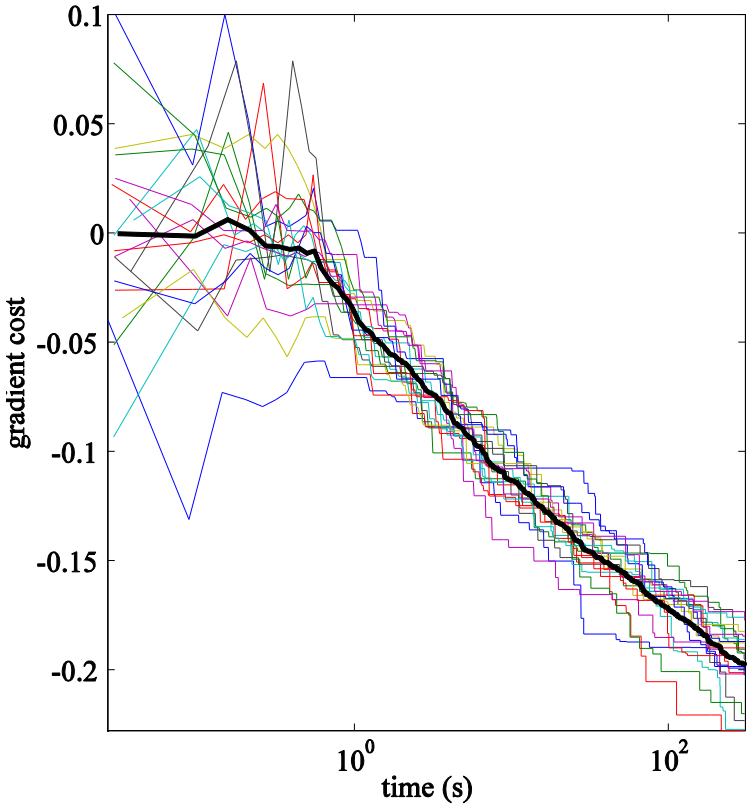


Figure 70. Automatic segmentation edge-gradient costs over time for all participants. The *y* axis shows for each iteration over time the cost of the current best automatic segmentation cost for each participant according to Equation 15 (black line is the mean over participants), while the *x* axis shows time in seconds on a semi-log scale. Over time, stochastically generated shapes find better gradient costs according to Equation 15. Some of the lines 'appear' to rise when intuitively they should only descend; this is because the points on this graph represent averages (median) of the top 10 shapes at each point in time. After enough iterations, the averages of the top 10 can only improve so there are no increases in cost after a few hundred iterations.

These results (see Figure 70) illustrate the process of fitting stochastic shapes from the model to the ultrasound images. The first clear result from looking at the slope of the black line on the figure, is that the minimum gradient does not appear to have been located. This suggests that better gradients (as per Equation 15) could possibly be located in any of the images, and this further suggests that better automatic segmentation solutions may also exist. Secondly, the results also show that the algorithm (see Figure 68) appears to reliably find better gradients, the longer it runs, since the slope of the line is steep and descends towards an asymptotic minimum (the time axis is logarithmic so the descent has diminishing returns on the gradient cost over time). The following subsections will evaluate the

automatic segmentation and expert annotation for agreement on a number of different terms described in section 6.3.5.

6.4.1 ROOT MEAN SQUARE ERROR

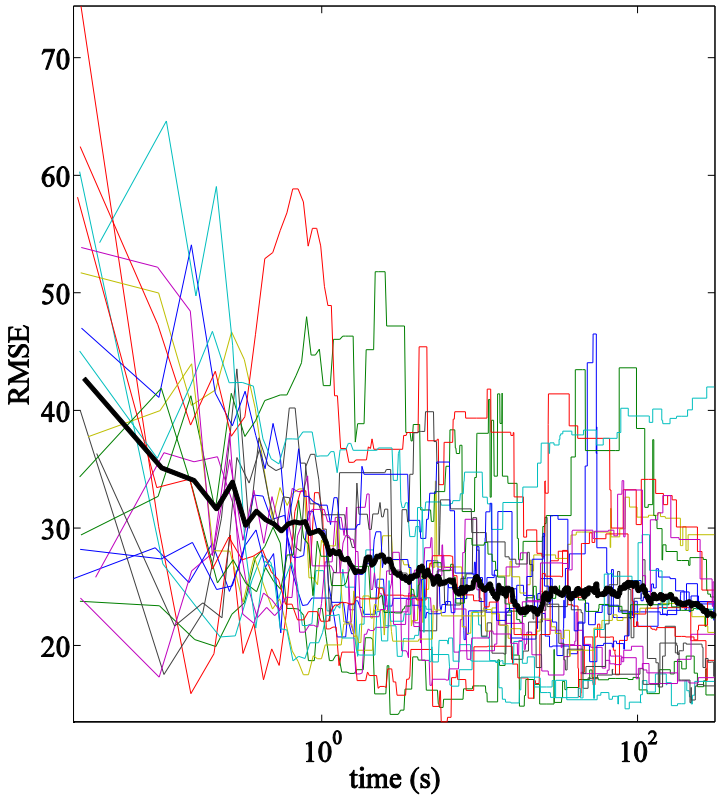


Figure 71. Root mean square error (RMSE) between the expert annotation and the automatic segmentation. The *y* axis shows RMSE between expert annotations and automatic segmentations, in pixels, for every participant (black line is the mean over participants). The *x* axis shows time in seconds on a semi-log scale. Over time, the RMSE between expert annotations and automatics segmentations reduces as the gradient cost function is minimised (see Figure 70).

These results (see Figure 71) illustrate the agreement between the boundary points of the MRI defined segmentation, and ultrasound defined segmentation (see Equation 16). These results show that there are multiple possible agreements arising from the ultrasound defined fitting process. This is shown because the RMS error sometimes rises and sometimes falls while the ultrasound fitting process continues. However, broadly the agreement between MRI and ultrasound defined information improves over time and converges to the best answer or set of best answers.

6.4.2 *Segment Euclidean Centroid Distance*

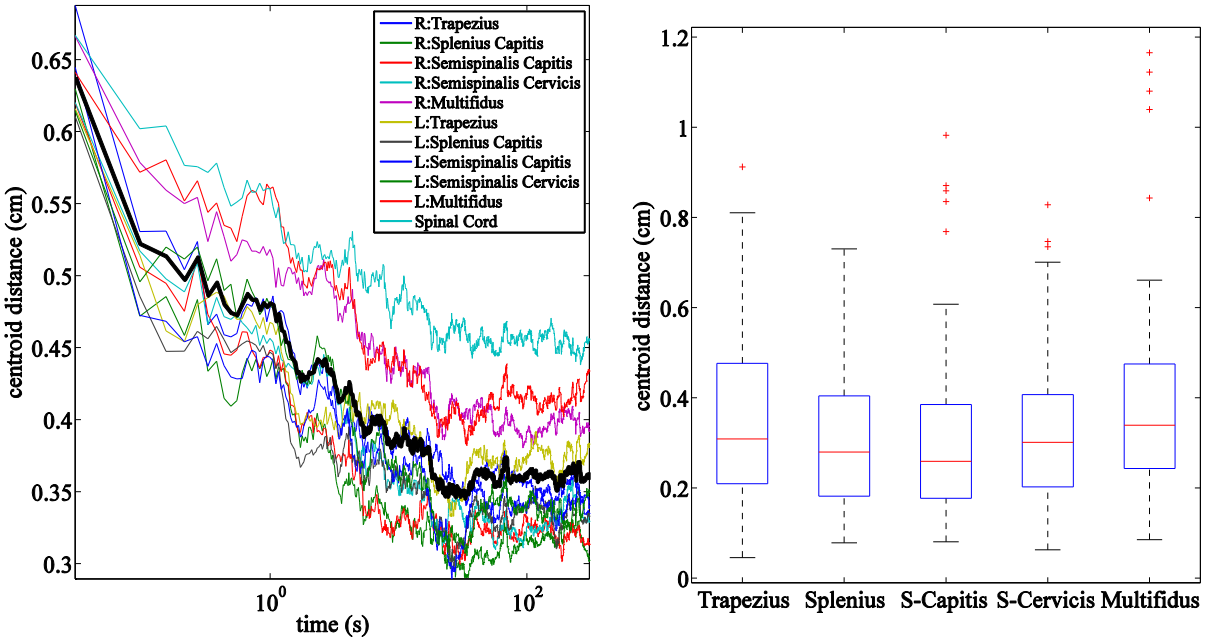


Figure 72. Left: Segment Euclidean centroid distances over time, averaged over all participants per time-step (overall average shown in black). Right: Segment Euclidean centroid distances for all participants after 5000 iterations, where group represents an average of the left and right side muscles.

These results (see Figure 72) illustrate the agreement between the segment centroids of the MRI defined segmentation, and ultrasound defined segmentation (see Equation 17). There is a clear initial rapid ($< 50s$) decrease in the distance between the segment centroids of both MRI and ultrasound methods, followed by somewhat slower decrements thereafter. There is some variability between segments but, in general the centroid distances all approximately appear to follow the same trend. Similar to the RMSE results, it is clear that there are multiple possible agreements arising from the ultrasound defined fitting process. However, broadly the agreement between MRI and ultrasound defined information improves over time and converges to the best answer or set of best answers. The final results (box and whisker plot in Figure 72) show that the average (median) distance between centroids is $\approx 3mm$. This result empirically suggests that the gradients of the ultrasound are descriptive enough to automatically and manually localise the relative muscle segment centroids to within a $3mm$ segment localisation error.

6.4.3 *Segment Area Agreement*

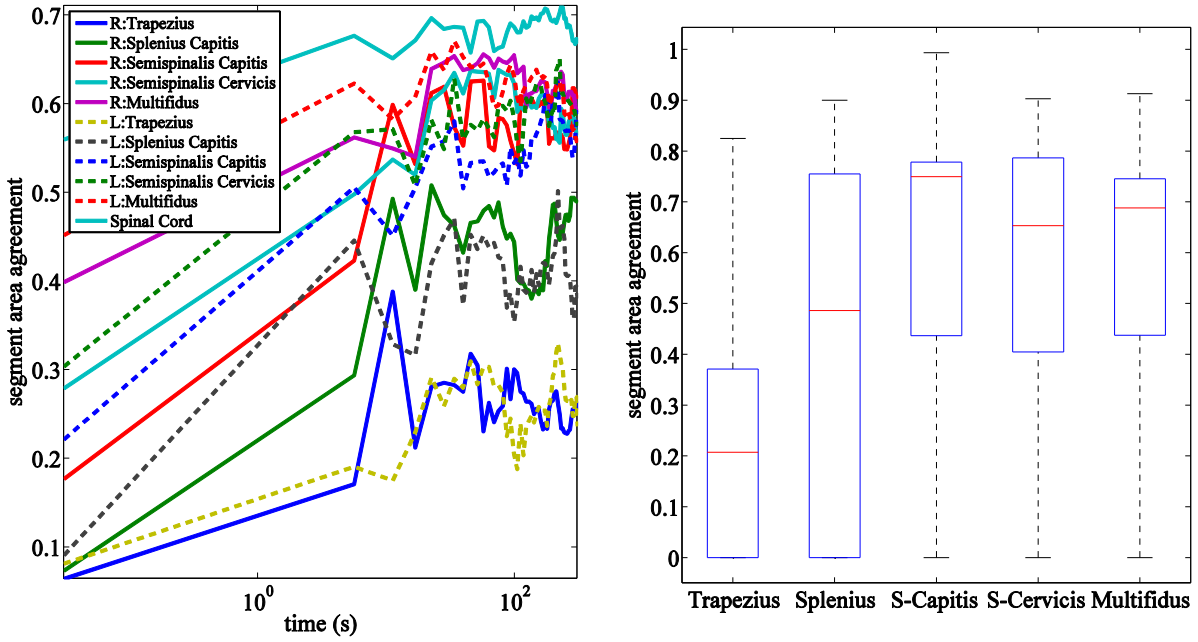


Figure 73. Left: Segment area agreement over time, averaged (mean) over all participants per time-step (51 evenly spaced discrete time-steps are linearly interpolated from 0 to end). Right: Segment area agreement for all participants after 5000 iterations, where each group represents an average of the left and right side muscles.

These results (see Figure 73) illustrate the agreement between the segment areas of the MRI defined segmentation, and ultrasound defined segmentation (see Equation 18). Results show that in terms of pixel area classification, the deeper muscle layers (semispinalis capitis/cervicis, and multifidus) and the spine agree better ($> 60\%$) between segmentation methods (automatic/expert segmentation). The superficial layers show very little relative agreement ($> 20\%$), while all other layers show modest agreement ($> 40\%$). However, these results also show that as the gradients improve with each iteration the segment area agreement increases. This in turn suggests that the algorithm has not converged, and these results are not representative of the true potential of the automatic segmentation technique.

6.4.4 SEGMENT CLASSIFICATION

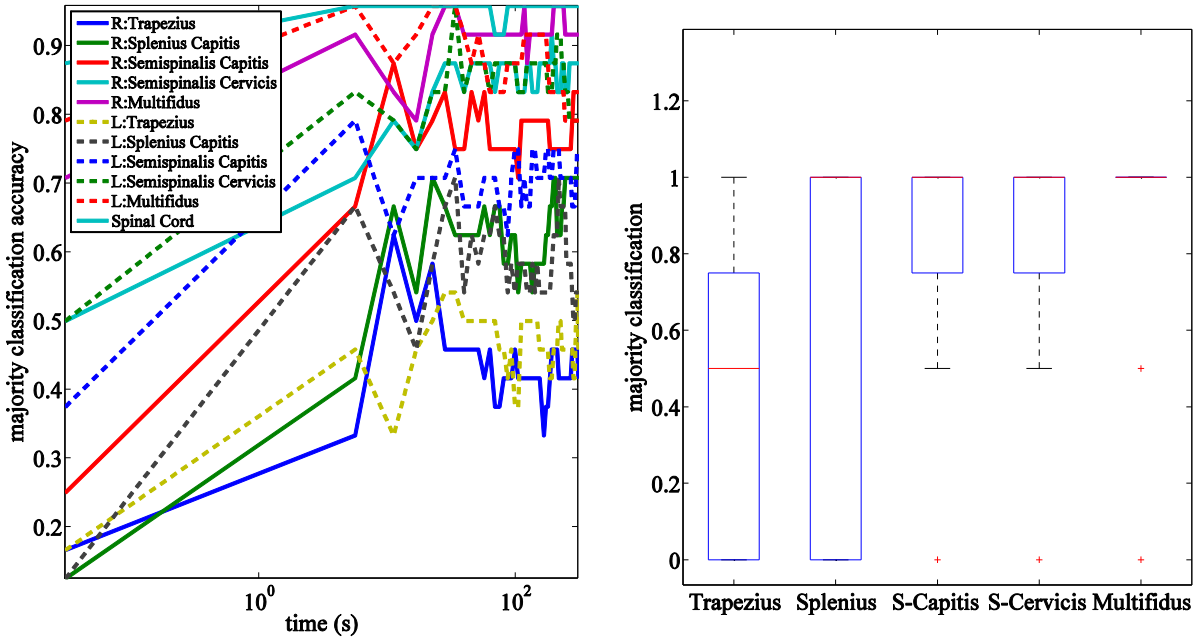


Figure 74. Left: Majority segment classification over time, averaged (mean) over all participants per time-step (51 evenly spaced discrete time-steps are linearly interpolated from $\boldsymbol{t = 0}$ to $\boldsymbol{t = end}$). Right: Majority segment classification for all participants after 5000 iterations, where each group represents an average of the left and right side muscles.

These results (see Figure 74) illustrate the agreement between the majority image region classification of the MRI defined segmentation, and ultrasound defined segmentation (see Equation 19). These results empirically show that the deepest 4 layers (semispinalis capitis/cervicis, multifidus and the spine) are more accurately classified/located by both methods (MRI defined, and ultrasound defined). These results demonstrate that the gradients of the ultrasound are detailed enough to accurately (automatically/manually) classify the relative muscle regions.

In support of these results, a confusion matrix of these results is presented below in Table 9 and Figure 73. The confusion matrix shows the results of the majority segment classification, including all misclassifications. The confusion matrix also shows that the technique works well over a population across all muscle layers, with the worst results in the superficial muscle layers. The trapezius exhibits the lowest classification accuracy with a maximum of 10 misclassifications out of 24. However, the trapezius was most often misclassified as the splenius capitis (next layer down).

Table 9. Confusion matrix of 12 classes over all 24 participants after 5000 iterations. The class names corresponding to the col/row labels can be found in Table 10. The diagonal (top left to bottom right) shows the number of correct-class classifications ($\boldsymbol{row = column}$ as per automatic and expert segmentation agreement; Equation 19). Any other value ($\boldsymbol{column \neq row}$) shows the number of incorrectly classified regions (e.g. row 6 and column 7 shows that the right semispinalis capitis (**#6**) of the automatic segmentations, has incorrectly classified the right splenius capitis (**#7**) of the expert annotations **9** times). High numbers in the diagonal and low values elsewhere equates to a good classification rate. See Figure 75 for a graphical representation of this matrix.

		Automatic Segmentation Class											
Expert Annotation Class		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
	0	0	4	0	0	0	0	3	3	0	0	0	0
	1	0	10	3	1	0	0	0	0	0	0	0	0
	2	0	10	17	2	0	0	0	0	0	0	0	0
	3	0	0	4	20	3	0	0	0	0	0	0	0
	4	0	0	0	1	21	2	0	0	0	0	0	0
	5	0	0	0	0	0	22	0	0	0	0	0	1
	6	0	0	0	0	0	0	11	1	2	0	0	0
	7	0	0	0	0	0	0	9	13	1	1	0	0
	8	0	0	0	0	0	0	1	7	18	2	0	0
	9	0	0	0	0	0	0	0	0	3	19	4	0
	10	0	0	0	0	0	0	0	0	0	2	19	0
	11	0	0	0	0	0	0	0	0	0	0	1	23

Table 10. Accompanying list of class values and corresponding muscle names for Table 9.

Class Value	**Class Name**
0	Unclassified
1	Left Upper Trapezius
2	Left Splenius Capitis
3	Left Semispinalis Capitis
4	Left Semispinalis Cervicis
5	Left Multifidus
6	Right Upper Trapezius
7	Right Splenius Capitis
8	Right Semispinalis Capitis
9	Right Semispinalis Cervicis
10	Left Multifidus
11	Spinal Cord

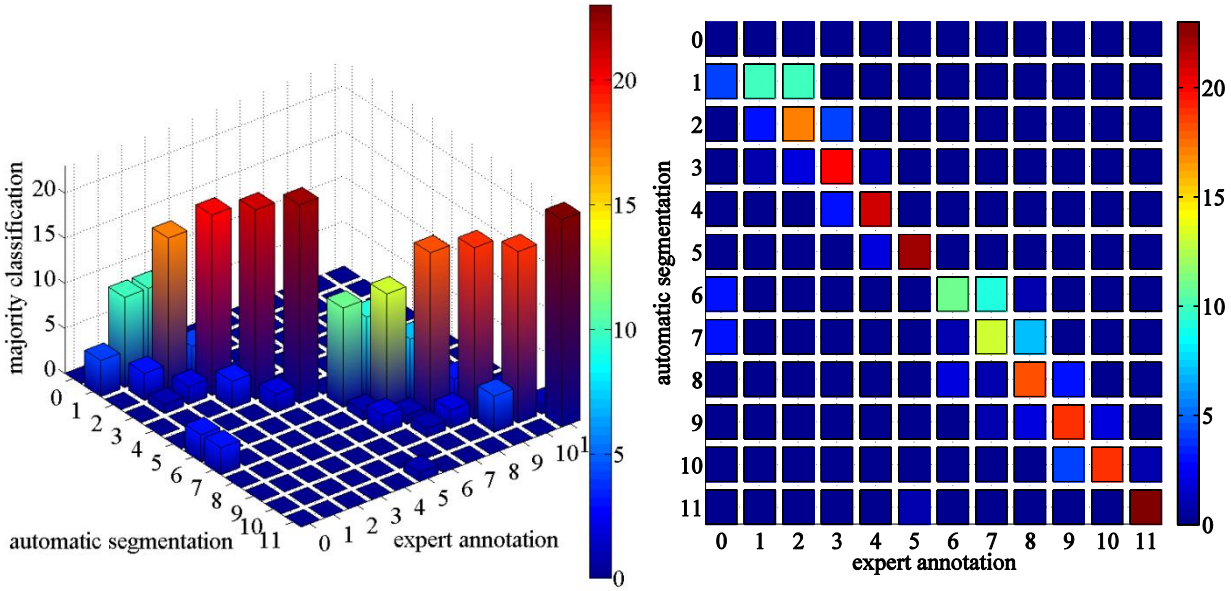
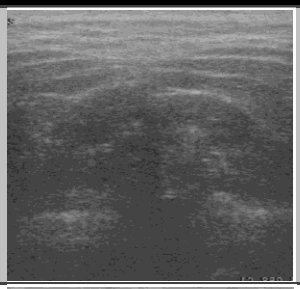
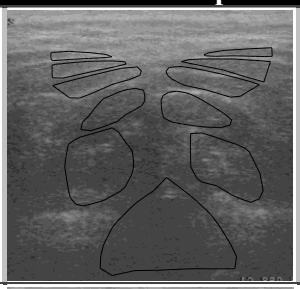
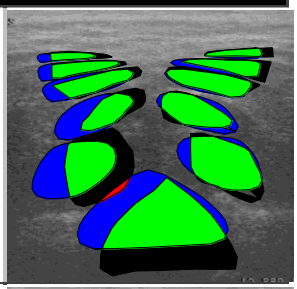
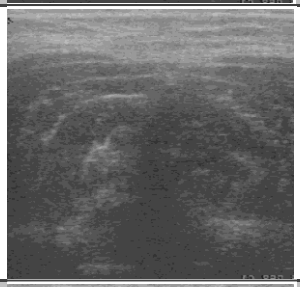
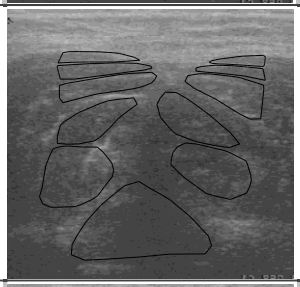
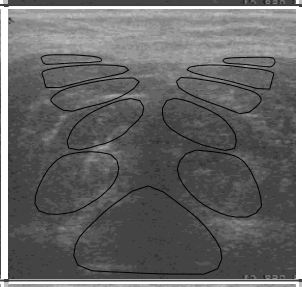
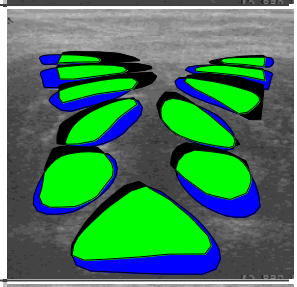
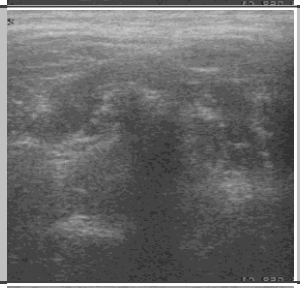
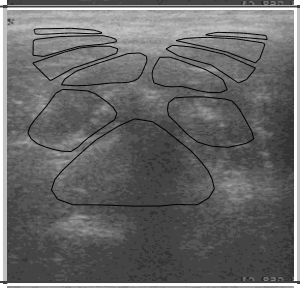
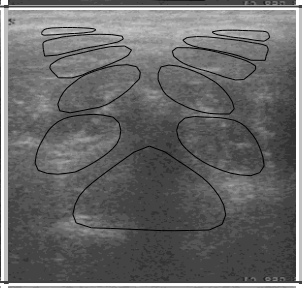
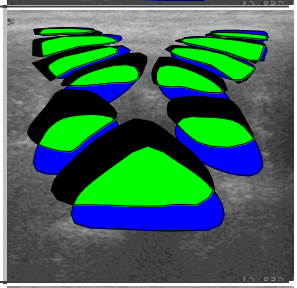
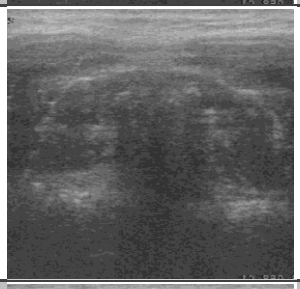
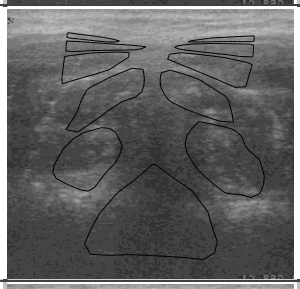
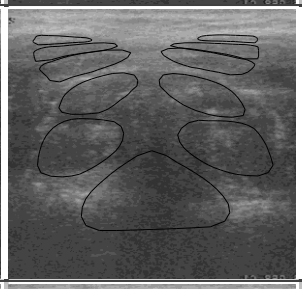
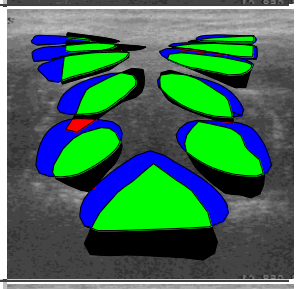
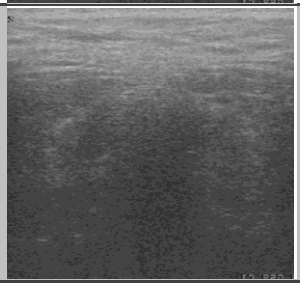
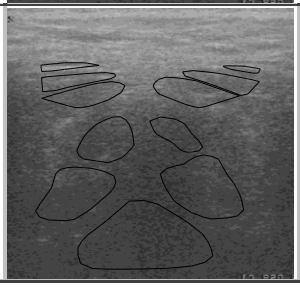
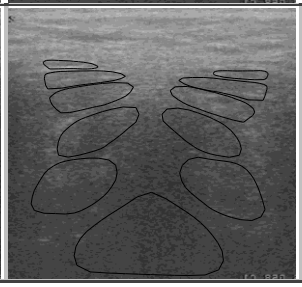
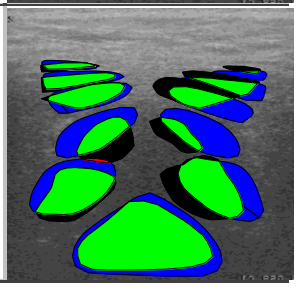


Figure 75. Confusion matrix in 2 view-points of all 12 classes over all 24 participants after 5000 iterations, corresponding to the values in Table 9. The *y*-axis shows the automatic segmentation class, while the *x*-axis shows the expert annotation class. The diagonal of the matrix (top left to bottom right) shows the number correct-class classifications ($\boldsymbol{row = column}$ as per automatic and expert segmentation agreement; Equation 19). Any other bar ($\boldsymbol{column \neq row}$) shows the number of incorrectly classified regions (e.g. row 6 and column 7 shows that the right semispinalis capitis (**#6**) of the automatic segmentations, has incorrectly classified the right splenius capitis (**#7**) of the expert annotations $\approx$ **9** times). High numbers in the diagonal and low values elsewhere equates to a good classification rate.

6.4.5 *Qualitative Analysis*

Table 11. Top 5 segmentation results according to Equation 18 (full table can be found in Appendix 8.5). From left to right: raw ultrasound image, expert annotation, automatic segmentation, comparison between expert/automatic methods. The image on the right shows the classification according to Equation 19, where green is agreement, red is disagreement, blue is unclassified automatic segmentation, and black is unclassified expert annotation.

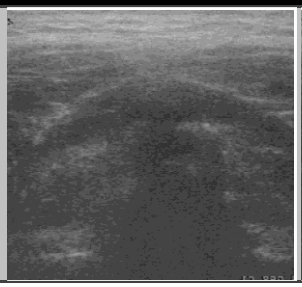
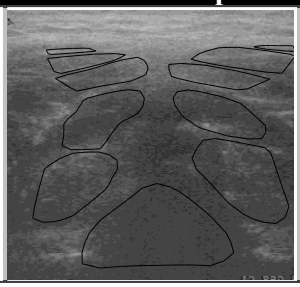
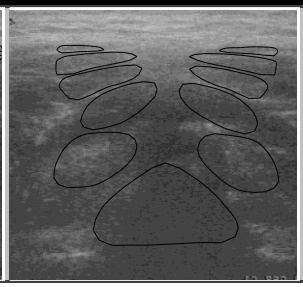
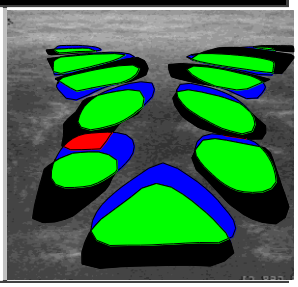
	Ultrasound Image	Registered MRI Mark-up	Segmentation	Classification
1				
2				
3				
4				
5				

The results in Table 11 show that there is a remarkable agreement between both methods; even when there are few well-defined muscle regions in the ultrasound image (see result #5). What is also clear is that even though these selected examples represent the top 5 results, there is still a reasonably high degree of disagreement. However, it should be noted that this disagreement is mainly manifested in unclassified regions (blue or black colour code) rather than misclassified regions (red colour code). The blue and black regions in the images can be thought of not as disagreement but, as regions that have been identified as muscle by one technique or the other. It is also clear from these results that much of the blue and black regions result from seemingly correct extra lateral edge identification of more muscle in the top 3 muscle layers (upper trapezius, splenius capitis, and semispinalis capitis).

6.5 Discussion & Conclusions

This study has shown that MRI can be used to image all 5 skeletal muscle layers and the spine in the human posterior neck. MRI can also be used to label ultrasound images by approximating the superficial muscle deformation (see section 6.3.3.1). Results show that it is possible to use the MRI shape statistics in combination with primitive methods of evaluating the grey-scale content of ultrasonography, to automatically segment muscles within ultrasound images from unseen participants. The technique has been scrutinised heavily and has held up the assertion that it is possible to automatically segment complex muscle groups from ultrasound using intuitive methods (see section 6.3.5). These results empirically demonstrate that ultrasonography can be used to image all 5 skeletal muscle layers in the human posterior neck, and that there is sufficient detail to locate each muscle region to within $3mm$ accuracy. Additional novelty lies in the MRI-to-ultrasound annotation of muscle segment boundaries, and the stochastic grey-level automatic segmentation technique.

Table 12. An example of poor landmark matching from MRI to ultrasound. Here the automatic method (image #3) appears to be discredited by the expert annotation (image #2), even though the expert annotation appears to contain error. Due to heavy regularisation during expert annotation (and possibly poor image-plane marker identification), this was the best solution according to the expert. The image on the right shows the classification according to Equation 19 (rank of 24 in the left column), where green is agreement, red is disagreement, blue is unclassified automatic segmentation, and black is unclassified expert annotation.

	Ultrasound Image	Registered MRI Mark-up	Segmentation	Classification
8				

The accuracy of either expert annotation or automatic segmentation is difficult to measure in isolation, as there is currently no way to know for certain where the muscles are in the ultrasound image. Much of the disagreement between measures can be accounted for by ultrasound and/or MRI acquisition of poor quality data, use of primitive image-plane markers, poor matching of scan landmarks, and difficulty in recognising muscle boarders in the MRI image and particularly in the ultrasound image (see Table 12). This is supported in the literature with the work of Dupnot, et al [106]. However, therein lays the benefit of the automated segmentation method; if the technique can be optimised for real-time use as operator feedback, the quality and confidence in the data can be improved by regularising probe placement via real-time muscle identification. Higher-quality data may in turn lead

to better models of human posterior neck muscle, and in turn may lead to better segmentation methods.

Future work will investigate optimisation of the segmentation technique for real-time operation. This method essentially presents an atomic search routine (see Figure 68), such that speed optimisation via Graphical Processing Units (GPU) is a trivial matter of implementation; GPUs compute simultaneous atomic commands all in parallel on thousands of dedicated processing cores, typically in application to image processing and computer vision problems [81]. More data should be collected using the optimised automatic segmentation as a real-time feedback guide, then more comprehensive models should be created from the optimised quality data; models should include full muscle segments, and could possibly be extended to the 3D domain. By annotating every slice of the MRI, a robust 3D statistical model could be constructed in the same manner as described (see section 6.3.4). A 3D shape model would give more degrees of freedom during fitting by allowing automatic optimisation of an image-plane parameter during automatic segmentation.

Region extraction for the purposes of modelling functional muscle state raises the question of annotation of functional muscle via ultrasonography. Not only has it not been demonstrated that this technique holds over a sequence of images containing contractions and changes in head orientation, it would also not be possible to evaluate performance of any method without annotation. It would be possible to annotate various poses via MRI and customised neck braces that impose certain head orientations but, this would be time-consuming. Participants could be instructed to apply a constant contractile pressure while in certain discrete head orientations in order to annotate images of muscle experiencing active contraction. Another method would be to obtain a static high-quality MRI and then acquire regular images using functional MRI, which has already been used to measure muscle thickness in the multifidii [105].

In conclusion, it is possible to use MRI with primitive annotation/registration techniques to estimate the soft structure deformation when under applied pressure from an ultrasound probe. It is therefore possible to annotate ultrasound images of skeletal muscle via image-plane markers with confidence, even when the clarity of the image is lacking, and/or when segment boundaries are either not present or not complete. It is also possible to automatically estimate the location of the boundaries of 10 muscles and the spinal cord in the human posterior neck via ultrasonography. The presented segmentation technique promises optimisation by GPU implementation and therefore is potentially a real-time segmentation and operator feedback technique.

7 CONCLUSIONS: THE CAPACITY OF ULTRASOUND AS A TOOL FOR ANALYSIS OF HUMAN SKELETAL MUSCLE

7.1 CONCLUSIONS

The purpose of this study was to investigate the use of b-mode ultrasound for analysis of human skeletal muscle. That required an in-depth investigation into advanced image analysis techniques that are either well-established or emerging and showing promise. There is currently no gold standard method to analyse ultrasound images of dynamic skeletal muscle and retrieve useful physiological information. There are many promising techniques which show that there is meaningful information in such data [96, 85, 1, 34, 28, 106, 60, 32, 31, 30] however, these methods were either not robust and/or failed to exploit the full information present within the ultrasound images. This thesis has presented investigations of emerging machine learning algorithms in comparison with existing ultrasound techniques (see chapter 2), an argument for using new methods over existing methods (see chapter 3) novel approaches to modelling complex skeletal muscle anatomy (see chapters 0 and 6) and function (see chapter 5) of multilayer skeletal muscle simultaneously via ultrasound. The overwhelming conclusion is that there is a strong future in ultrasound research and analysis of dynamic skeletal muscle. The spatial and temporal resolution of ultrasonography is increasing with time, allowing for the possibility of deeper investigation into muscle function (fast twitches, micro contractions, etc…). Massively parallel deep neural networks are an emerging powerful image analysis technology which currently is only limited by computing power, therefore with the increase in computing power comes a greater potential to further exploit the information content of ultrasound.

The following sections of this chapter revisit and summarise the relevant preceding chapters, extracting the main questions, points of discussion and conclusions with respect to the aims of this thesis. The thesis aims can be referred to in chapter 1, section 1.4. The final section of this thesis draws the main conclusions of the entire study together, and also explores some of the possibilities and recommendations for future research and development of the techniques presented here.

7.1.1 Evaluation of Existing and Emerging Techniques

This thesis is about the application of b-mode ultrasound for analysis of human skeletal muscle; therefore a comprehensive review of the development of currently available techniques was required. The first purpose of chapter 2 was to understand the development of many of the current analysis techniques, as well as understand what information is desirable/possible to extract from ultrasound images of skeletal muscle. The secondary purpose of that chapter was to investigate the feasibility of using emerging state of the art techniques in the field of deep learning, which is currently under-utilised in medical imaging. The chapter then ended with a detailed review of a selection of 5 publications which were considered the seminal studies on segmentation and image processing with respect to ultrasound, skeletal muscle analysis via ultrasound, and unsupervised image feature extraction.

Regarding the task of ultrasound image segmentation there was a general consensus in the literature that it is extremely challenging. Early attempts were hindered by numerous factors such as the quality of ultrasound images, lack of established techniques, and poor problem domain knowledge [12, 13, 14, 16]. Since those early attempts the spatial quality of ultrasound images has improved, computational power has increased (both in general and with the introduction of GPUs), and the techniques have advanced. A comprehensive review and evaluation of ultrasound image segmentation techniques revealed that the most powerful and successful techniques made use of shape and texture priors (models) in order to regularise heuristic segmentation algorithms [11], where regularisation allows for commonly occurring region boundary dropout and poor contrast between regions. This approach was shown to be successful in segmenting ultrasound images of skeletal muscle [1]. The main conclusion of the segmentation review was that provided a good model of shape can be defined (usually from annotations), automatic segmentation of complex skeletal muscle groups such as those in the neck may be possible.

Initial investigations into analysis techniques revealed an overwhelming interest in the use of ultrasound to estimate strain generated by muscles during contraction. The main paradigm used in attempts to achieve this was local speckle tracking (feature tracking) and initial techniques showed that motion fields produced by speckle tracking algorithms during isometric contractions were physiologically meaningful [20]. Studies showed that speckle tracking was ultimately flawed [2, 3] due to signal drift resulting from a combination of the artefacts inherent to ultrasound images such as speckle noise, signal dropout, and trans-planar motion. However, the paradigm was developed and subsequently used to show that ultrasound sequences of dynamic skeletal muscle do contain information relating to passive joint rotation and isometric contraction [1]. Their study also presented an automatic segmentation algorithm and as a result set the standard for automated skeletal muscle

analysis. While the tracking paradigm is flawed, there was enough evidence in the literature to support the argument for development of new methods of extracting that information.

Discounting relatively non-complex muscle parameters such as muscle thickness, the other main paradigm discovered in the literature was founded on attempts to understand the internal mechanics of muscle by estimating local and global orientations of fascicles directly from a single image [30]. Their approach evades the possibility of signal drift because no information is propagated between frames; only the current frame is considered when estimating fascicle orientations. The technique was further developed such that individual fascicles could be traced over the image in order to extract regional fascicle curvature [32]. This technique assumed the presence of well-defined fascicles in the image, and their persistence throughout a sequence. In reality, fascicles are not always well-defined and frequently move in/out of the image plane over a dynamic sequence of skeletal muscle. However, in ideal cases this approach is powerful and may lead to understanding muscle mechanics by relating regional texture information to muscle function, provided the regional information can be more consistently extracted.

A review of feature learning techniques demonstrated that descriptive features of images could be automatically extracted from raw image data, and that those features can be used to infer the content of a given image. These methods were found to be the state of the art in image segmentation [114] and image recognition [67]. Feature learning allows for the possibility of modelling the appearance of fascicles and other structures in an image of skeletal muscle directly from data, without a prior definition of the appearance of fascicles, and without prior assumption to the information content of an ultrasound image of skeletal muscle. For this reason the main conclusion derived from the review of analysis techniques was that feature tracking is an unlikely candidate technique for further development. Obtaining single frame measures with greater regional and sequential consistency should also be investigated further with respect to the application of feature learning as an alternative to the wavelet approach. This approach would not limit the investigation to modelling fascicle orientation, and where fascicle orientation is concerned feature learning allows for modelling the regional variances in appearance of fascicles without manual optimisation of wavelet kernels.

7.1.2 *An Argument for a New Approach to Analysis of Dynamic Human Skeletal Muscle via Ultrasonography*

Prior to this project feature tracking was the paradigm that had been used to relate skeletal muscle movements – observed via ultrasound – to external factors such as joint rotation and muscle contraction measured by EMG [1]. Evidence from the literature revealed that feature tracking is inappropriate for tracking feature movement via ultrasound due to unavoidable feature drift, even with attempts to regularise the motion field [2, 3]. The authors of [1] demonstrated that stable tracking can be achieved in short sequences (<40 seconds), although their evidence revealed that tracking started to fail beyond the first 20 seconds. Therefore the purpose of chapter 3 was to provide an introduction to tracking failure and supporting evidence for a new approach to analyse dynamic skeletal muscle via ultrasound. To achieve this, a lengthy sequence (>3 minutes) of isometric muscle contraction was analysed with optimisation of the KLT parameters, feature width and feature height.

The results supported the argument for development of a new approach by showing that feature drift results from numerous factors; the main factors being speckle noise and non-linear/affine transformations (or trans-planar motion). In the discussion it was mentioned that speckle noise would pose less of a problem if a non-adaptive tracking technique was used. This is because speckle noise introduces small errors in the feature window however, it does not alter the underlying feature description in an adaptive manner (does not accumulate noise/error). Therefore selecting an initial feature and tracking its movement in every successive frame without updating its appearance (template) would likely be more stable and less susceptible to drift. It was noted that the reason this could not be implemented is that muscle ultrasound contains many similar (in appearance) features and during contraction those features transform nonlinearly (affine) and can move out of the image plane (trans-planar motion), meaning that adaptive tracking is required for stability.

Chapter 3 also revealed that tracking drift was regionally dependant due to regional variations in affine transformation during contraction. It was found that the superficial region of the muscle exhibited the most feature drift. It was noted that this might have resulted from affine transformations such as regional variations in fascicle curvature, and this statement is supported with evidence that fascicles indeed curve more in the superficial region than any other [32]. It was suggested that another contributing factor might be greater trans-planar motion in that region. Although that was difficult to demonstrate/prove, it was supported by demonstrating that tracking also failed in the deeper regions of the muscle. Comparisons against tracking drift resulting from additive speckle noise revealed that in general the dynamic contraction sequence resulted in more drift than the static speckle noise sequence. The drift of the motion field from the dynamic contraction sequence was not uniformly distributed which also contributed to evidence that features may or may not leave the image plane and therefore

tracking reliability depends on unpredictable regional variations in affine, and trans-planar feature motion.

The main conclusion derived from the work in this chapter was that a new approach was required that is time-invariant and therefore does not allow for drift. Speckle noise was much less a contributing factor towards the problem of the feature tracking paradigm, namely tracking drift. Furthermore, the affine transformation or trans-planar motion inherent to sequences of this nature makes local feature tracking a poor approach for analysing dynamic muscle. A consequence of these conclusions was that since speckle noise was a relatively negligible problem, feature learning techniques could be used to extract time-invariant feature templates representative of dynamic muscle states. Then these features could be used to analyse muscle states via linear models of measured activity and joint rotation. Chapter 3 then concluded with the suggestion that a very accurate segmentation algorithm was required before feature learning algorithms could be deployed, since feature learning requires normalised (scaled and centred image regions) inputs. This led directly to the work in the following chapter on automatic calf muscle region extraction which was based on the approach presented in [1].

7.1.3 Development of a New Segmentation Technique

The immediate purpose of chapter 4 was to develop a robust and accurate human calf segmentation technique that was capable of segmenting sequences of dynamic calf muscle. That chapter was to be the preceding stages of the development of a new approach for information extraction via feature learning. More generally this chapter was an initial investigation of the application and use of shape models and simple edge gradient cost functions with a parallel segmentation algorithm, which may be applicable in the general sense to other multilayer skeletal muscle groups. The development of a generic approach to segmenting multiple muscle regions would make feature learning a feasible possibility for different muscle groups, if feature learning was shown to be a good approach. This chapter presented the eventual solution as a tool for the Matlab API which will promote the development of other techniques such as the fascicle curvature technique [30, 32], which required manual region annotation for every frame that was processed.

The main conclusion of this chapter was that, provided a good model of muscle shape can be acquired, there is a generic edge gradient cost function that allows boundary detection of regions of interest, even when the boundary is partially visible or has a poor gradient. Importantly, model statistics were recorded in metric dimensions and the tool was implemented such that if the metric dimensions of a video sequence are known, the tool can be used to segment that video sequence. In addition to the metric shape model dimensions, the edge gradient cost function simply maximises the relative intensity (attenuation) differences between muscle boundaries. This would theoretically allow segmentation of sequences from entirely different machines where the gradients can be very different

to the gradients in the images used to build the model. A suggestion was also put forward that this method could be used to segment raw radio frequency data, based on the same premise. Those reasons make the presented algorithm a generic approach to segmentation of ultrasound image sequences of skeletal muscle, and make the tool a practical solution for automatic region segmentation, regardless of machine settings, or the type of machine used. When compared against the previous technique [1] upon which this method was based, results were shown to be more accurate and stable over time, validated on a vastly larger dataset containing more participants, and many more images. This work solved a practical problem and could be used to process vast amounts of data which is usually required for unsupervised feature learning, which was investigated in the following chapter.

7.1.4 Exploring the Limits of the Information Content of Ultrasound with Restricted Boltzmann Machines

The main purpose of chapter 5 was to develop the hypothesis that feature learning techniques can be used to model time-invariant muscle states via ultrasound, such that information may be extracted from that model. The premise was that ultrasound images of skeletal muscle are complex and information rich, and by reducing such images to a lower-dimensional space representative of generic muscle states (rather than a matrix of attenuation values) could make information extraction a simpler task. More generally this approach was being used to evaluate whether the information content of ultrasound allows for the delineation of states influenced by completely different factors such as passive and active muscle shortening. To answer this question a comprehensive dataset of combined muscle functions was acquired by experimental design, such that this information could be exploited. By experimental design of muscle inputs a rich dataset was acquired which contained thousands of ultrasound images of the human calf with associated muscle input labels, which were recorded externally by EMG and joints fixed to pedals with known angle.

An intermediate study using feature tracking empirically showed that muscle shortening that was caused by active contraction is different from muscle shortening that was caused by passive joint rotation. This work was published and can be found in Appendix 8.9.1. Leaving out the question of inter-participant generalisation, the results of this chapter showed that muscle states are unique and can be delineated into a space representative of the external factors that caused those states, without optimising the model with the data labels. This means that a change in muscle state brought about by joint rotation is distinct from a change in muscle state brought about by active contraction. Linear neural networks were then used to map those states to known external signals (EMG, angle) which resulted in a time-invariant prediction of the muscle inputs which caused a given image (muscle state). With regards to ultrasound, that result means that the spatial information of ultrasonography encodes enough information about muscle function such that it can be extracted directly from an image with no knowledge of external influences of muscle state. With regards to physiology, that result also means

that muscle intrinsically encodes information about joint angles and muscle length potentially available as sensory input (proprioception) which may be processed by the brain and used for motor control.

The main conclusion of this chapter was that feature learning methods can be used to model time-invariant dynamic skeletal muscle states directly from ultrasound images by experimental design of the muscle input space. It is inconclusive if this approach can be used on more challenging and complex muscle groups such as the neck. However, given a robust and accurate segmentation algorithm and a way to manipulate/design the input space, it may be possible to apply the same methods used here to other muscle groups. Therefore the following chapter was about modelling the architecturally complex muscles in the human posterior neck, such that the algorithm presented in chapter 4 could be used to automatically extract the muscle regions from ultrasound images. This would allow for the possibility of using the approach presented in chapter 5 to model muscle states in the posterior neck.

7.1.5 Development of a New Approach to Modelling Ultrasound-Observed Architecturally Complex, Multilayer Skeletal Muscle Shapes via MRI and Segmentation of Human Posterior Neck Images

The purpose of chapter 6 was to assess the extent to which the muscles in the human posterior neck could be automatically segmented via ultrasound, extending the approach developed in chapter 4. Because of the challenging nature of ultrasound images, this required the development of a new approach to modelling ultrasound-observed muscle shape. Ultrasound has relatively poor spatial resolution and region contrast, which makes manual image annotation very challenging and in some cases not possible. Magnetic Resonance Imaging (MRI) has much improved spatial resolution and region contrast compared to ultrasound, which could make the task of muscle boundary annotation far simpler. It was decided that, if the shape modelling approach detailed in chapter 4 was to be adopted a methodology was required which allowed annotation of MRI images and conversion of annotations into ultrasound-observed shape statistics, which could then be used with the previously presented segmentation algorithm (see chapter 4).

It was demonstrated that MRI annotations could be registered to the correct anatomical regions in an ultrasound image (where the imaging planes were approximately aligned via MRI-visible markers placed during collection of the ultrasound images). Converting an MRI annotation into an ultrasound annotation also required the development of a depth displacement function which would approximate the magnitude and decay of the displacement of muscle boundaries when an ultrasound probe is pressed on to the skin. With that function it was then possible to register the annotations to the ultrasound images which could then be used to create shape statistics (used for segmentation) in the

ultrasound domain. Results showed that a large amount of displacement results from probe placement in the superficial layers of muscle, with less influence on the deeper muscles.

Ultrasound registered annotations were used to create a statistical shape model which would then guide the generic stochastic segmentation routine detailed in chapter 4. It was previously stated that the segmentation algorithm is entirely parallelisable and GPUs can be used to accelerate computation. However, in this study no GPU acceleration was used due to time constraints and therefore positive potential of the algorithm has not been fully explored. Irrespective of this, comparisons between the automatic segmentations and the registered annotations showed a high level of agreement for the majority of images. During a qualitative analysis, examples of poor segmentation were shown to have a speculative annotation. This was due to heavy restrictions during the annotation process (i.e. not allowing the expert to modify individual contour points/markers), which were required to ensure that muscle shapes were physiologically accurate as defined by the MRI.

The main conclusion of this study was that ultrasound images of human posterior neck contain sufficient spatial information to allow for automatic segmentation of 10 muscles and the spine. While the segmentation technique may not need some development, in its simplest form it allowed successful boundary detection of every muscle and the spine via ultrasound, even when the boundaries of some muscles were missing or incomplete. Another conclusion is that shape models of multilayer muscles with complex architecture can be created via MRI and an estimate of tissue deformation resulting from ultrasound probe placement. Furthermore, the segmentation technique developed in chapter 4 was shown to be possibly applicable to many ultrasound image segmentation problems where the boundaries between regions can be defined by a simple gradient measure. This study also provides the initial stages of work for investigations into analysis of the information content of ultrasound of dynamic posterior human neck muscles.

7.2 Suggestions for Further Work

This thesis has worked towards developing standard techniques for analysing dynamic skeletal muscle via ultrasound. There are many possibilities for extending this work, and some of which will now be discussed. What will not be discussed is use of alternative segmentation methods such as Convolutional Neural Networks CNN [114], or the application of the proposed segmentation technique (see chapter 4) with existing techniques such as estimating fascicle curvature [30, 32], or fascicle tracking [1, 34]. Although these methods promote information extraction from ultrasound, they are separate avenues of investigation which answer different questions. This thesis is promoting a general approach to modelling and extracting the information from ultrasound image sequences for interpretation and contribution to knowledge of the musculoskeletal system.

There are possibilities for optimisation of the proposed segmentation algorithm via parallel GPU implementation. This may allow real-time operator feedback in order that data acquisition can be regulated for higher quality, standardised data, or at least quicker back processing of thousands of images. There is also the possibility of using optimised segmentation for real-time participant feedback of non-complex muscle parameters such as muscle thickness. This would allow the option of designing an experiment in which the control parameters were individual muscle properties, instead of joint angles or muscle activity (which is difficult to acquire in deep neck muscles) as in chapter 5. There are also possibilities of developing the feature learning approach such that the muscle state inputs (joint angle, EMG) could be predicted on unseen participants' ultrasound images. Currently the model only generalised within a participant, and due to project time constraints was not extended to generalise between participants. Initial work was conducted towards this aim however; there was some difficulty in creating a generalised model. A generalised model would allow the use of ultrasound to extract force, angle and EMG measures directly from ultrasound images without the need for invasive methods like intramuscular EMG. Both optimisation of the segmentation algorithm and development of a generalised model of muscle function will be explained briefly in the following two subsections.

7.2.1 Optimisation of the Segmentation Algorithm

The segmentation algorithm was developed as an iterative algorithm, such that each iteration would execute an atomic instruction that could be executed in parallel. GPU devices have been used to execute atomic instruction all in parallel and when used to optimise atomic instruction algorithms they are only limited by the number of computing cores, the speed of the GPU clock, and the memory transfer latency. Typically a GPU can be used to optimise an algorithm by a factor of 30 however; multiple GPUs can be used simultaneously (up to 8 on a GPU compute server) for a far greater optimisation. All that is required to implement the segmentation algorithm proposed in chapter 4 is the following:

1. Transfer ultrasound image to GPU
2. Evaluate N stochastic shapes on N parallel cores
3. Transfer N shapes with costs back to host machine
4. Perform a linear search of the costs to find the minimum

For a GPU with 3000 cores, 3000 shapes can be evaluated simultaneously. In comparison with the results of the neck segmentation in chapter 6, only 5000 shapes were evaluated, which took 300 seconds; in just two GPU iterations this number could be exceeded in much less time given an efficient C++ CUDA (GPU programming language) implementation.

7.2.2 *Development of a Generalised Predictive Model of Muscle Sates*

There are some methodological issues with the model proposed in chapter 5 which should be rectified before developing the approach. In chapter 5 the approach was to normalise the muscle region to have fixed input dimensions regardless of the thickness of the muscle segment. The way this was done was to use the segmentation to fit an oriented (about the dominant orientation of the muscle segment) bounding box, where the texture within the bounding box was extracted and subsequently resized to fixed dimensions, 80×80. The process of resizing the texture within the bounding box was flawed because it introduced a non-linearity in the texture, which is of course bad for generalisation since this non-linearity needs to be delineated prior to modelling/extracting information about muscles. To rectify this issue the bounding box should be of fixed dimensions about the centre of the muscle region, which would include varying amounts of information depending on muscle thickness of individual participants however, the structures (fascicles) within the ultrasound will still be representative of a general muscle structure.

Upon creation of the new model, to better exploit the information within the image, it would be beneficial to train a larger GBRBM model. The model trained in chapter 5 did not over-fit the training data at any point which usually means that the GBRBM model is not large enough to extract all the descriptive features of the data. Furthermore, GBRBMs were developed as a module that can be used to model many layers of non-linear features. In chapter 5 only a single layer of features was extracted from the data. If a generalised mode is to be created in the future it would be recommended that a larger feature vector should be used. It is difficult to know how many features to use prior to modelling however; it is recommended that the largest possible model should be built within the available time. Assuming a time scale of 6 months, one should calculate the time to complete approximately 2000 iterations in that time with varying size feature vectors.

The final practical methodological issue also resulted from time restrictions and that was the use of only 22,000 training samples from a set of over 700,000. Certainly if a larger model is to be constructed, a much larger proportion of the data should be included. This will not only contribute to a more comprehensive and robust model, it will also help with regularisation of the larger feature vectors. However, the proportion used should also be evaluated against available model training time, since including more examples will take longer to complete cycles over the training set.

Once the basic methodological issues have been resolved, a multilayer GBRBM model can be constructed in exactly the same way as in chapter 5 (hold out validation and testing data, stopping training when the validation set error plateaus or increases). Then there should be some consideration given towards including historical states when mapping from the model to externally measured muscle inputs. It would be counter-intuitive to assume that a generalised model of joint angles could be

created with no prior knowledge of the appearance of the muscle in its neutral state. Although muscle force output has been linked to fascicle curvature, the same statement could be said about predicting EMG from a single image without prior knowledge of the muscle in its neutral state. The recommended approach would be to include the feature activations of top-level GBRBM features resulting from an image of the muscle in its neutral state, alongside the current feature activations of the GBRBM resulting from the muscle in its current state. This would allow construction of a model of how muscles change between neutral and dynamic states, and may lead to a generalised model of muscle activity.

There is a possibility that the model will not generalise between participants simply by building a larger model. Another very common way to improve model generalisation is to acquire additional data from additional participants. In that case, the same experiment should be completed for additional participants, exploring the same joint angle and EMG data ranges. One must also consider the possibility that inter-participant generalisation is simply not possible without participant-specific priors, such as knowledge of participant muscle architecture (muscle thickness, fascicle density, etc…). In this case, an entirely new approach may be required. However, the recommendation is that larger models should be trained in order to identify the number of free parameters (weights or hidden units) required to achieve model over-fitting. Once over-fitting has been achieved, it will be known that with the current dataset, it is not be possible to build a more complex model; therefore that option can be eliminated from future research.

7.3 Final Conclusions

The studies presented within this thesis have explored the capacity of ultrasound as a skeletal muscle analysis tool, showing that it is possible and feasible to extract meaningful information from skeletal muscle ultrasound images. Currently the information limit of ultrasound is bound by computational and algorithmic capability. New and emerging feature learning techniques have been explored that are theoretically applicable to any ultrasound-observable muscle given successful extraction of the region of interest. These techniques have provided strong evidence that muscle states uniquely encode muscle activation, force output and joint angles. A generic segmentation algorithm has been presented that has demonstrated successful segmentation of architecturally simple and complex muscle layers even in suboptimal circumstances (poor image quality). The segmentation algorithm is theoretically applicable to any well-defined muscle shapes given that it was demonstrated that MRI can be used to model shapes in difficult circumstances. That result means that models of muscle state can be created and evaluated using the feature learning approach regardless of the shape and internal structure of the given muscle. Further work has been suggested which may lead to clinical technology such as real-time segmentation of skeletal muscle, and generalised evaluation of muscle activity. The final conclusion of this thesis is therefore, ultrasound is an extremely powerful technology which

overcomes the limits of EMG, and kinematic analysis, and that future efforts based on this work may lead to advanced insights into the internal mechanics of muscle, and neuromusculoskeletal function/dysfunction.

BIBLIOGRAPHY

[1] J. Darby, E. Hodsen-Tole, N. P. Costen and I. D. Loram, "Automated regional analysis of B-mode ultrasound images of skeletal muscle movement," *Applied Physiology,* vol. 112, no. 2, pp. 313-327, 2012.

[2] F. Yeung, S. F. Levinson, D. Fu and K. J. Parker, "Feature-adaptive motion tracking of ultrasound image sequences using a deformable mesh," *IEEE Transactions on Medical Imaging,* vol. 17, no. 6, pp. 945-956, 1998.

[3] F. Yeung, S. F. Levinson and K. J. Parker, "Multilevel and motion-based ultrasonic speckle tracking algorithms," *Ultrasound in Medicine & Biology,* vol. 24, no. 3, pp. 427-441, 1998.

[4] "What are muscles, what are they made of & how many?," 2015. [Online]. Available: http://www.musclesused.com/. [Accessed 31 01 2015].

[5] L. K. Kamibayashi and F. J. Richmond, "Morphometry of human neck muscles," *Spine,* vol. 23, no. 12, pp. 1314-1323, 1998.

[6] G. Rankin, M. Stokes and D. J. Newham, "Size and shape of the posterior neck muscles measured by ultrasound imaging: normal values in males and females of different ages," vol. 10, no. 2, pp. 108-115, 2005.

[7] A. N. Vasavada, B. W. Peterson and S. L. Delp, "Three-dimensional spatial tuning of neck muscle activation in humans," *Experimantal Brain Reseach,* vol. 147, no. 4, pp. 437-448, 2002.

[8] M. Kramer, I. Schmid, S. Sander, J. Hogel, R. Eisele, L. Kinzl and E. Hartwig, "Guidelines for the intramuscular positioning of EMG electrodes in the semispinalis capitis and cervicis muscles," *Electromyography and Kinesiology,* vol. 13, no. 3, pp. 289-295, 2003.

[9] M. O. Culjat, D. Goldenberg, P. Tewari and R. S. Singh, "A review of tissue substitutes for ultrasound imaging," *Ultrasound in medicine & biology,* vol. 36, no. 3, pp. 861-873, 2010.

[10] 2014. [Online]. Available: http://dan.corlan.net/medline-trend.html. [Accessed August August 2014].

[11] A. J. Noble and D. Boukerroui, "Ultrasound image segmentation: a survey," *Medical Imaging,* vol. 25, no. 8, pp. 987-1010, 2006.

[12] R. F. Chang, W. J. Wu, W. K. Moon and D. R. Chen, "Multiresolution texture segmentation with application to diagnostic ultrasound images," *IEEE Transactions on Medical Imaging,* vol. 17, no. 4, pp. 108-123, 1993.

[13] E. A. Ashton and K. J. Parker, "Multiple bayesian segmentation of ultrasound images,"

Ultrasonic Imaging, vol. 12, no. 1, pp. 291-304, 1995.

[14] M. Sonka, M. Zhang, M. Siebes, M. S. Bissing, S. C. DeJong, S. M. Collins and C. R. McKay, "Segmentation of intravascular ultrasound images: a knowledge based approach," *IEEE Transactions on Medical Imaging*, vol. 14, no. 4, pp. 719-732, 1995.

[15] M. Isard and A. Blake, Active contours, New york: Springer-Verlag, 1998.

[16] B. Levienaise-Obadia and A. Gee, "Adaptive segmentation of ultrasound images," vol. 17, no. 8, pp. 583-588, 1999.

[17] T. F. Cootes, C. J. Taylor, D. H. Cooper and J. Graham, "Active shape models-their training and application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38-59, 1995.

[18] A. C. Hodge, A. Fenster, D. B. Downey and H. M. Ladak, "Prostate boundary segmentation from ultrasound images using 2D active shape models: optimisation and extension to 3D," *Computer Methods and Programs in Biomedicine*, vol. 82, no. 2, pp. 99-113, 2006.

[19] L. W. Weiss and F. C. Clark, "Ultrasonic protocols for separately measuring subcutaneous fat and skeletal muscle thickness in the calf area," *Physical Therapy*, vol. 65, no. 4, pp. 477-481, 1985.

[20] M. Bertrand, J. Meunier, M. Douchet and G. Ferland, "Ultrasonic biomechanical strain gauge based on speckle tracking," in *IEEE Ultrasonics Symposium*, 1989.

[21] R. G. Lopata, N. M. Maartje, I. H. Gerrits, J. M. Thijssen, L. Kapusta, F. N. van de Vosse and C. L. de Korte, "2H-1 in vivo 3D cardiac and skeletal muscle strain estimation," in *IEEE Ultrasonics Symposium*, 2006.

[22] C. N. Maganaris, "Image-based estimates of moment arm length in intact human muscle-tendons," *European Journal of Applied Physiology*, vol. 91, no. 2-3, pp. 130-139, 2004.

[23] L. Li and K. Y. Tong, "Musculotendon parameters estimation by ultrasound measurement and geometric modelling: application on brachialis muscle," in *IEEE Engineering in Medicine and Biology Society*, 2006.

[24] P. W. Hodges, L. H. Pengel, R. D. Herbert and S. C. Gandevia, "Measurement of muscle contraction with ultrasound imaging," *Muscle & Nerve*, vol. 27, no. 6, pp. 682-692, 2003.

[25] Y. H. Liu, S. M. Chen, C. Y. Lin, C. L. Huang and Y. N. Sun, "Motion tracking on elbow tissue from ultrasonic image sequence for patients with lateral epicondylitis," in *IEEE Engineering in Medicine and Biology Society*, 2007.

[26] J. Shi, Y. P. Zheng, X. Chen and Q. H. Huang, "Assessment of muscle fatigue using sonomyography: muscle thickness change detected from ultrasound images," *Medical Engineering & Physics*, vol. 29, no. 4, pp. 472-479, 2007.

[27] T. Miyoshi, T. Kihara, H. Koyama and S.-I. Yamamoto, "Automatic detection method of muscle fibre movement as revealed by ultrasound images," *Medical Engineering & Physics,* vol. 31, no. 5, pp. 558-564, 2009.

[28] J. Farron, T. Varghese and D. G. Thelen, "Measurement of tendon strain during muscle twitch contractions using ultrasound elastography," *IEEE Transactions on Ultrasonics, Ferroelectronics and Frequency Control,* vol. 56, no. 1, pp. 27-35, 2009.

[29] Y. Zhou and Y.-P. Zheng, "Estimation of muscle fibre orientation in ultrasound images using revoting hough transform (RVHT)," *Ultrasound in Medicine & Biology,* vol. 34, no. 9, pp. 1474-1481, 2008.

[30] M. Rana, G. Hamarneh and J. M. Wakeling, "Automated tracking of fascicle orientation in B-mode ultrasound images," *Journal od Biomechanics,* vol. 42, no. 13, pp. 2068-2073, 2009.

[31] M. Rana and J. M. Wakeling, "In vivo determination of 3D muscle architecture of human muscle using free hand ultrasound," *Journal of Biomechanics,* vol. 44, no. 11, pp. 2129-2135, 2011.

[32] A. I. Namburete, M. Rana and J. M. Wakeling, "Computational methods for quantifying in vivo muscle fascicle curvature from ultrasound images," *Journal of Biomechanics,* vol. 44, no. 14, pp. 2538-2543, 2011.

[33] L. Qiaoliang, X. Chen, D. Ni, S. Chen and T. Wang, "A statistical approach for muscle fascicle orientation estimation in ultrasound images," in *Biomedical Engineering and Informatics*, 2012.

[34] J. Darby, B. Li, N. Costen, I. Loram and E. Hodson-Tole, "Estimating Skeletal Muscle Fascicle Curvature from B-Mode Ultrasound Image Sequences," *IEEE Transactions on Biomedical Engineering,* vol. 60, no. 7, pp. 1935-1945, 2013.

[35] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics,* vol. 5, no. 4, pp. 115-133, 1943.

[36] D. Hebb, "The organization of behavior," vol. 44, 1949.

[37] F. Rosenblatt, "Frank. The perceptron, a perceiving and recognizing automaton Project Para," Cornell Aeronautical Laboratory, 1957.

[38] M. Minsky and S. Papert, Perceptrons, Cambridge, Ma: Press, 1969.

[39] P. Werbos, *Beyond regression: New tools for prediction and analysis in the behavioral sciences,* 1974.

[40] S. E. Fahlman, "An empirical study of learning speed in back-propagation networks," 1988.

[41] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning,* vol. 20, no. 3, pp. 273-297, 1995.

[42] Y. Bengio, “"Deep Learning of Representations for Unsupervised and Transfer Learning,” in *ICML Unsupervised and Transfer Learning*, 2012.

[43] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel, “Handwritten digit recognition with a back-propagation neural network,” in *Advances in neural information processing systems*, 1990.

[44] “MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges,” 2014. [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed 28 12 2014].

[45] “Learning methods for generic object recognition with invariance to pose and lighting,” in *Computer Vision and Pattern Recognition*, 2004.

[46] K. Fukushima, “Cognitron: A self-organizing multilayered neural network,” *Biological cybernetics*, vol. 20, no. 3-4, pp. 121-136, 1975.

[47] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological cybernetics,* vol. 36, no. 4, pp. 193-202, 1980.

[48] B. A. Olshausen, “Emergence of simple-cell receptive field properties by learning a sparse code for natural images,” *Nature,* vol. 381, no. 6583, pp. 607-609, 1996.

[49] G. Karol and Y. LeCun, “Learning fast approximations of sparse coding,” in *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[50] J. Pearl, Probabilistic reasoning in intelligent systems: networks of plausible inference, Morgan Kaufmann, 1988.

[51] R. Neal, “Connectionist learning of belief networks,” *Artificial intelligence,* vol. 56, no. 1, pp. 71-113, 1992.

[52] S. E. Fahlman, G. E. Hinton and T. J. Sejnowski, “Massively parallel architectures for Al: NETL, THISTLE, and BOLTZMANN machines,” in *Proc. National Conf. Mifidai Intelligence*, 1983.

[53] D. H. Ackley, G. E. Hinton and T. J. Sejnowski, “A learning algorithm for boltzmann machines*,” *Cognitive Science,* vol. 9, no. 1, pp. 147-169, 1985.

[54] P. Smolensky, Information processing in dynamical systems: Foundations of harmony theory, 1986, p. 194–281.

[55] G. Hinton, S. Osindero and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527-1554, 2006.

[56] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” vol. 14, no. 8, pp. 1771-1800, 2002.

[57] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science,* vol. 313, no. 5786, pp. 504-507, 2006.

[58] Y. Lecun, “Learning invariant feature hierarchies,” in *Computer vision–ECCV 2012. Workshops and demonstrations*, 2012.

[59] T. F. Cootes, G. J. Edwards and C. J. Taylor, “Active appearance models,” in *Computer Vision*, Springer Berlin Heidelberg, 1998.

[60] I. D. Loram, C. N. Maganaris and M. Lakie, “Use of ultrasound to make noninvasive in vivo measurements of continuous changes in human muscle contractile length,” *Applied Physiology,* vol. 100, no. 4, pp. 1311-1323, 2006.

[61] “Tune control-point locations using cross correlation - MATLAB cpcorr,” 2014. [Online]. Available: http://uk.mathworks.com/help/images/ref/cpcorr.html. [Accessed 30 12 2014].

[62] C. Tomasi and T. Kanade, Detection and tracking of point features, Pittsburgh: Pittsburgh: School of Computer Science, Carnegie Mellon Univ, 1991.

[63] C. Tomasi and J. Shi, “Good features to track,” in *CVPR94*, 1994.

[64] J. Hangyi, P. van Zijl, J. Kim and G. D. Pearlson, “DtiStudio: resource program for diffusion tensor computation and fiber bundle tracking,” *Computer methods and programs in biomedicine,* vol. 81, no. 2, pp. 106-116, 2006.

[65] C. Farabet, C. Couprie, L. Najman and Y. LeCun, “Learning hierarchical features for scene labeling,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 35, no. 8, pp. 1915-1929, 2013.

[66] B. Fulkerson, A. Vedaldi and S. Soatto, “Class segmentation and object localization with superpixel neighborhoods,” in *Computer Vision, IEEE 12th International Conference on*, 2009.

[67] A. Krizhevsky, I. Sutskever and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012.

[68] N. Srivastava, G. Hinton, A. Krizhevsky and I. Sutskever, “Dropout: A simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research,* vol. 15, no. 1, pp. 1929-1958, 2014.

[69] L. Breiman, “Random forests,” *Machine Learning,* vol. 45, no. 1, pp. 5-32, 2001.

[70] D. G. Lowe, “Object recognition from local scale-invariant features,” in *The proceedings of the seventh IEEE international conference on Computer vision,* 1999.

[71] F. Perronnin and C. Dance, “Fisher kernels on visual vocabularies for image categorization,” in *Computer Vision and Pattern Recognition*, 2007.

[72] Mathworks, “Plot velocity vectors as cones in 3-D vector field - MATLAB conep,” [Online]. Available: http://www.mathworks.co.uk/help/matlab/ref/coneplot.html. [Accessed 10 10 2014].

[73] G. E. Hinton and M. Guy, “Recognizing handwritten digits using hierarchical products of experts,” *Patern Analysis and Machine Intelligence,* vol. 24, no. 2, pp. 189-197, 2002.

[74] J. G. Bosch, S. C. Mitchell, B. P. Leliveldt, F. Nojland, O. Kamp, M. Sonka and J. H. Reiber, “Automatic segmentation of echocardiographic sequences by active appearance motion models,” *Medical Imaging,* vol. 21, no. 11, pp. 1374-1383, 2002.

[75] S. C. Mitchell, J. G. Bosch, B. P. Lelieveldt, R. J. van der Geest, J. H. Reiber and M. Sonka, “3-D active appearance models: segmentation of cardiac MR and ultrasound images,” *Medical Imaging,* vol. 21, no. 9, pp. 1167-1178, 2002.

[76] D. Shen, Y. Zhan and C. Davatzikos, “Segmentation of prostate boundaries from ultrasound images using statistical shape model,” *Medical Imaging,* vol. 22, no. 4, pp. 539-551, 2003.

[77] T. Finni, J. A. Hodgson, A. L. Lai, V. R. Edgerton and S. Sinha, “Mapping of movement in the isometrically contracting human soleus muscle reveals details of its structural and functional complexity,” *Applied Physiology,* vol. 95, no. 5, pp. 2128-2133, 2003.

[78] S. Sinha, U. Sinha and V. R. Edgerton, “In vivo diffusion tensor imaging of the human calf muscle,” *Magnetic Resonance,* vol. 24, no. 1, pp. 182-190, 2006.

[79] P. Han, Y. Chen, L. Ao, G. Xie, H. Li, L. Wang and Z. Yongjin, “Automatic thickness estimation for skeletal muscle in ultrasonography: evaluation of two enhancement methods,” *Biomed Eng Online,* vol. 12, 2013.

[80] Mathworks, “Callbacks - Programmed Response to User Action - MATLAB & Simulink,” [Online]. Available: http://www.mathworks.co.uk/help/matlab/creating_plots/callbacks-programmed-response-to-user-action.html. [Accessed 04 11 2014].

[81] “Using graphics devices in reverse: GPU-based image processing and computer vision,” in *2008 IEEE International Conference on Multimedia and Expo*, 2008.

[82] D. A. Winter and H. J. Yack, “EMG profiles during normal human walking: stride-to-stride and inter-subject variability,” *Electroencephalography and clinical neurophysiology,* vol. 67, no. 5, pp. 402-411, 1987.

[83] A. L. Hof, J. P. Van Zandwijk and M. F. Bobbert, “Mechanics of human triceps surae muscle in walking, running and jumping,” *Acta Physiologica Scandinavica,* vol. 174, no. 1, pp. 17-30, 2002.

[84] I. D. Loram, C. N. Maganaris and M. Lakie, “Human posturel sway results from frequent, ballistic bias impulses by soleus and gastrocnemius,” *Physiology,* vol. 564, no. 1, pp. 295-311, 2005.

[85] N. J. Cronin, J. Avela, T. Finni and J. Peltonen, “Differences in contractile behaviour between the soleus and medial gastrocnemius muscles during human walking,” *Experimental Biology,* vol. 216, no. 5, pp. 909-914, 2013.

[86] M. V. Narici, T. Binzoni, E. Hiltbrand, J. Fasel, F. Terrier and P. Cerretelli, “In vivo human gastrocnemius architecture with changing joint angle at rest and during graded isometric contraction,” *Physiology,* vol. 496, no. 1, pp. 287-297, 1996.

[87] L. Benedikt, G. A. Lichtwark and A. G. Cresswell, “Reciprocal activation of gastrocnemius and soleus motor units is associated with fascicle length change during knee flexion,” *Physiological reports,* vol. 2, no. 6, 2014.

[88] J. L. Van Leeuwen and C. W. Spoor, “Modelling mechanically stable muscle architectures,” *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences,* vol. 336, no. 1277, pp. 275-292, 1992.

[89] K. L. Kwah, R. Z. Pinto, J. Diong and R. D. Herbet, “Reliability and validity of ultrasound measurements of muscle fascicle length and pennation in humans: a systematic review,” *Applied Physiology,* vol. 114, no. 6, 2013.

[90] A. Nickos, E. Giannakou, K. Albracht and A. Arampatzis, “Reproducibility of fascicle length and pennation angle of gastrocnemius medialis in human gait in vivo,” *Gait & Psture,* vol. 31, no. 1, pp. 73-77, 2010.

[91] G.-Q. Zhou, P. Chan and Y.-P. Zheng, “Automatic Measurement of Pennation Angle and Fascicle Length of Gastrocnemius Muscles Using Real-time Ultrasound Imaging,” *Ultrasonocs,* vol. 57, pp. 72-83, 2014.

[92] Y. Kawakami, Y. Ichinose and T. Fukunaga, “Architectural and functional features of human triceps surae muscles during contraction,” *Applied Physiology,* vol. 85, no. 2, pp. 398-404, 1998.

[93] G. Hinton, S. Osindero and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” vol. 18, no. 7, 2006.

[94] G. Hinton, “A practical guide to training restricted boltzmann machines,” *Momentum,* vol. 9, no. 1, p. 926, 2010.

[95] M. Welling, R. S. Zemel and G. E. Hinton, “Self supervised boosting,” in *Advances in Neural Information Processing Systems*, 2002.

[96] R. S. Chow, M. K. Medri, D. C. Martin, R. N. Leekam, A. M. Agur and N. H. McKee, “Sonographic studies of human soleus and gastrocnemius muscle architecture: gender variability,” *European journal of applied physiology,* vol. 82, no. 3, pp. 236-244, 2000.

[97] “Linear layer - MATLAB linearlayer,” 2014. [Online]. Available: http://uk.mathworks.com/help/nnet/ref/linearlayer.html. [Accessed 09 12 2014].

[98] B. L. Suderman, B. Krishnamoorthy and A. N. Vasavada, “Neck muscle paths and moment arms are significantly affected by wrapping surface parameters,” *Computer methods in biomechanics and biomedical engineering parameters*, vol. 15, no. 7, pp. 735-744, 2012.

[99] E. Volkan and M. Acar, “Viscoelastic finite element analysis of the cervical intervertebral discs in conjunction with a multi-body dynamic model of the human head and neck,” *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of engineering in medicine*, vol. 223, no. 2, pp. 249-262, 2009.

[100] C. M. Sommerich, S. Joines, V. Hermans and S. D. Moon, “Use of surface electromyography to estimate neck muscle activity,” *Electromyography and Kinesiology*, vol. 10, no. 6, pp. 377-398, 2000.

[101] M. M. Benhamou, M. Revel and C. Vallee, “Surface electrodes are not appropriate to record selective myoelectric activity of splenius capitis muscle in humans,” *Experimental Brain Research*, vol. 105, no. 3, pp. 432-438, 1990.

[102] K. S. Turker, “Electromyography: some methodological problems and issues,” *Physical Theraphy*, vol. 73, no. 10, pp. 698-710, 1993.

[103] J.-P. Lee, C. L. Wang, Y. W. Shau and S. F. Wang, “Measurement of cervical multifidus contraction pattern with ultrasound imaging,” *Electromyography and Kinesiology*, vol. 19, no. 3, pp. 391-397, 2009.

[104] J.-P. Lee, W.-Y. Tseng, Y.-W. Shau, C.-L. Wang, H.-K. Wang and S.-F. Wang, “Measurement of segmental cervical multifidus contraction by ultrasonography in asymptomatic adults,” *Manual Therapy*, vol. 12, no. 3, pp. 286-294, 2007.

[105] B. Cagnie, N. Dickx, I. Peeters, J. Tuytens, E. Achten, D. Cambier and L. Danneels, “The use of functional MRI to evaluate cervical flexor activity during different cervical flexion exercises,” *Applied Physiology*, vol. 104, no. 1, pp. 230-235, 2008.

[106] A.-C. Dupont, E. E. Sauerbrei, P. V. Fenton, P. C. Shragge, G. E. Loeb and F. J. Richmond, “Real-time sonography to estimate muscle thickness: Comparison with MRI and CT,” *Clinical Ultrasound*, vol. 29, no. 4, pp. 230-236, 2001.

[107] E. Kristjansson, “Reliability of ultrasonography for the cervical multifidus muscle in asymptomatic and symptomatic subjects,” *Manual Therapy*, vol. 9, no. 2, 2004.

[108] J. Khodabakhsh, M. Amiri, M. A. Moheseni-Bandpei, A. Rezasoltani and C. Fernandez-de-las-Perias, “Ultrasonography of the cervical muscles: a critical review of the literature,” *Manipulative and physiological therapeutics*, vol. 33, no. 8, pp. 630-637, 2010.

[109] I. E. Dana, C. Duffy, L. Kavanagh, A. Stanton and P. F. Whelan, “Fully automated segmentation and tracking of the intima media thickness in ultrasound video sequences of the common carotid artery,” *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 60, no. 1, 2013.

[110] A. K. Hamou and M. R. El-Sakka, “A novel segmentation technique for carotid ultrasound images.,” in *Acoustics, Speech, and Signal Processing*, 2004.

[111] “Anatomy.tv | 3D Human Anatomy | Primal Pictures,” 2014. [Online]. Available: https://www.anatomy.tv/. [Accessed 22 12 2014].

[112] Mathworks, “Intersection points for lines or polygon edges - MATLAB polyxpoly,” [Online]. Available: http://www.mathworks.co.uk/help/map/ref/polyxpoly.html. [Accessed 04 11 2014].

[113] Mathworks.co.uk, “Principal component analysis of raw data - MATLAB pca,” [Online]. Available: http://www.mathworks.co.uk/help/stats/pca.html. [Accessed 06 11 2014].

[114] F. Clement, C. Couprie, L. Najman and Y. LeCun, “Scene parsing with multiscale feature learning, purity trees, and optimal covers,” vol. 1202, no. 2160, 2012.

[115] “Data - Digit Recognizer | Kaggle,” 2015. [Online]. Available: https://www.kaggle.com/c/digit-recognizer/data. [Accessed 02 01 2015].

[116] I. D. Loram, M. Lakie, I. Di Giulio and C. N. Maganaris, “The consequences of short-range stiffness and fluctuating muscle activity for proprioception of postural joint rotations: the relevance to human standing,” *Neurophysiology,* vol. 102, no. 1, pp. 460-474, 2009.

[117] P. W. Hodges, L. H. M. Pengel, R. D. Herbert and S. C. Gandevia, “Measurement of muscle contraction with ultrasound imaging,” *Muscle & Nerve,* vol. 20, no. 6, pp. 682-692, 2003.

[118] R. Cunningham, P. Harding, I. D. Loram and N. P. Costen, “Automated measurement of human skeletal calf muscle contraction via b-mode ultrasound imaging,” in *Medical Image Understanding and Analysis*, 2013.

[119] Mathmetica, “Singular value decomposition - MATLAB svd,” [Online]. Available: http://www.mathworks.co.uk/help/matlab/ref/svd.html. [Accessed 05 11 2014].

[120] C. F. Njeh, C. W. Kuo, C. M. Langton, H. I. Atrah and C. M. Boivin, “Prediction of human femoral bone strength using ultrasound velocity and BMD: an in vitro study,” *Osteoperosis,* vol. 7, no. 5, pp. 471-477, 1997.

[121] L. Klimek, J. Schreiber, R. G. Amedee and W. J. Mann, “Three-dimensional ultrasound evaluation in the head and neck,” *Otolarynology-Head and Neck Surgery,* vol. 118, no. 2, pp. 267-271, 1998.

[122] Y. Ichinose, Y. Kawakami, M. Ito, H. Kaneshia and T. Fukunaga, “In vivo estimation of contraction velocity of human vastus lateralis muscle during “isokinetic” action,” *Journal of Applied Physiology,* vol. 88, no. 3, pp. 851-856, 2000.

[123] M. Ishikawa, T. Finni and P. V. Komi, “Behaviour of vastus lateralis muscle–tendon during high intensity SSC exercises in vivo,” *Acta physiologica scandinavica,* vol. 178, no. 3, pp. 205-213, 2003.

[124] F. Shao, K. V. Ling, W. S. Ng and R. Y. Wu, "Prostate boundary detection from ultrasonographic images," *Journal of ultrasound in medicine*, vol. 22, no. 6, pp. 605-623, 2003.

8 APPENDIX

8.1 RBM FEATURES – GASTROCNEMIUS

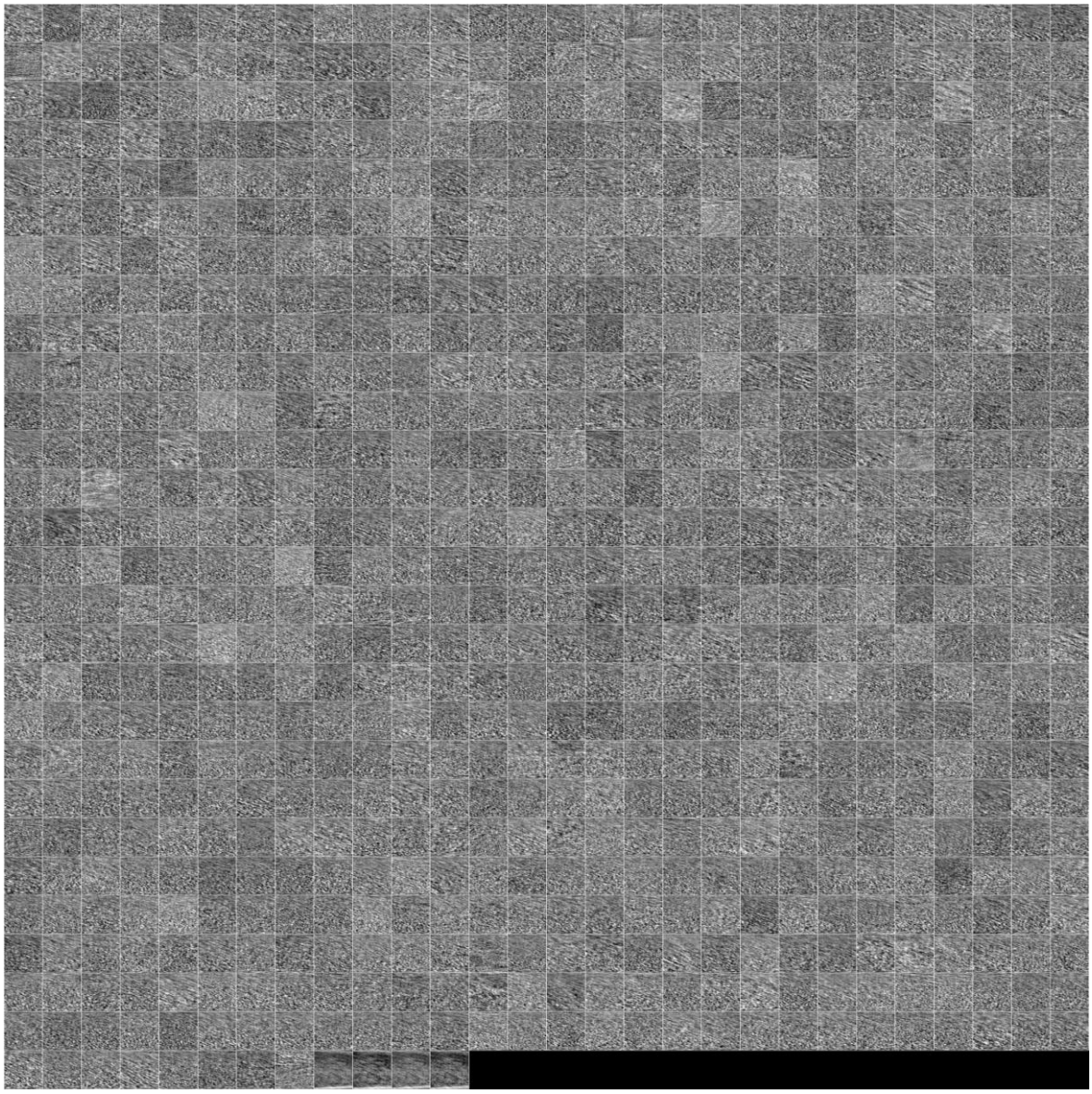


Figure 76. GBRBM features. Each square in the graphic is a matrix representation of the coefficients (min-max normalised) of a node in the MG GBRBM. The squares are ranked from left to right, top to bottom, in order of absolute Spearman rank correlation coefficient when correlating feature responses with sEMG over all sequences for a representative participant. This picture is therefore an image representation of the entire MG GBRBM.

8.2 RBM Features – Soleus

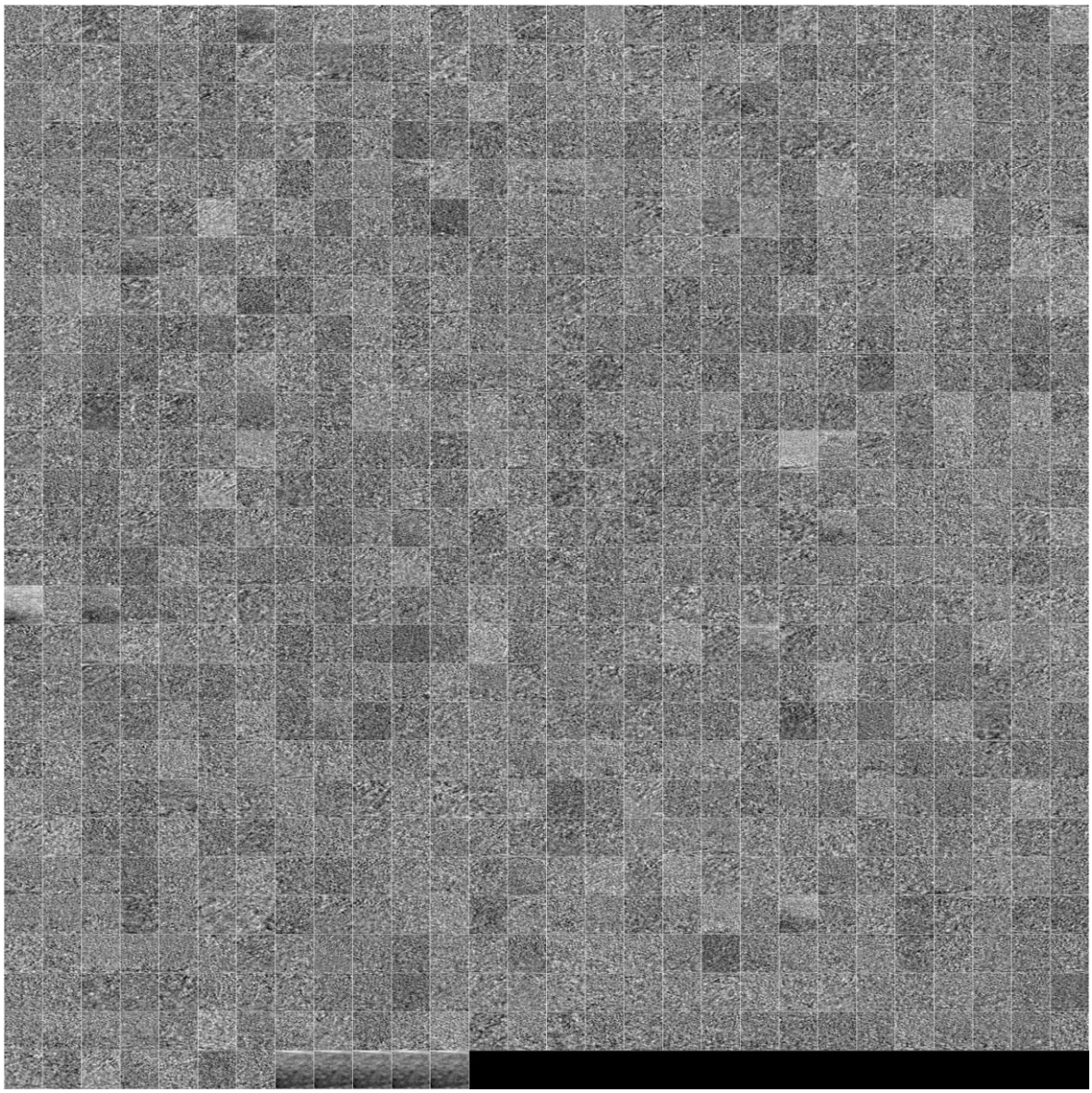
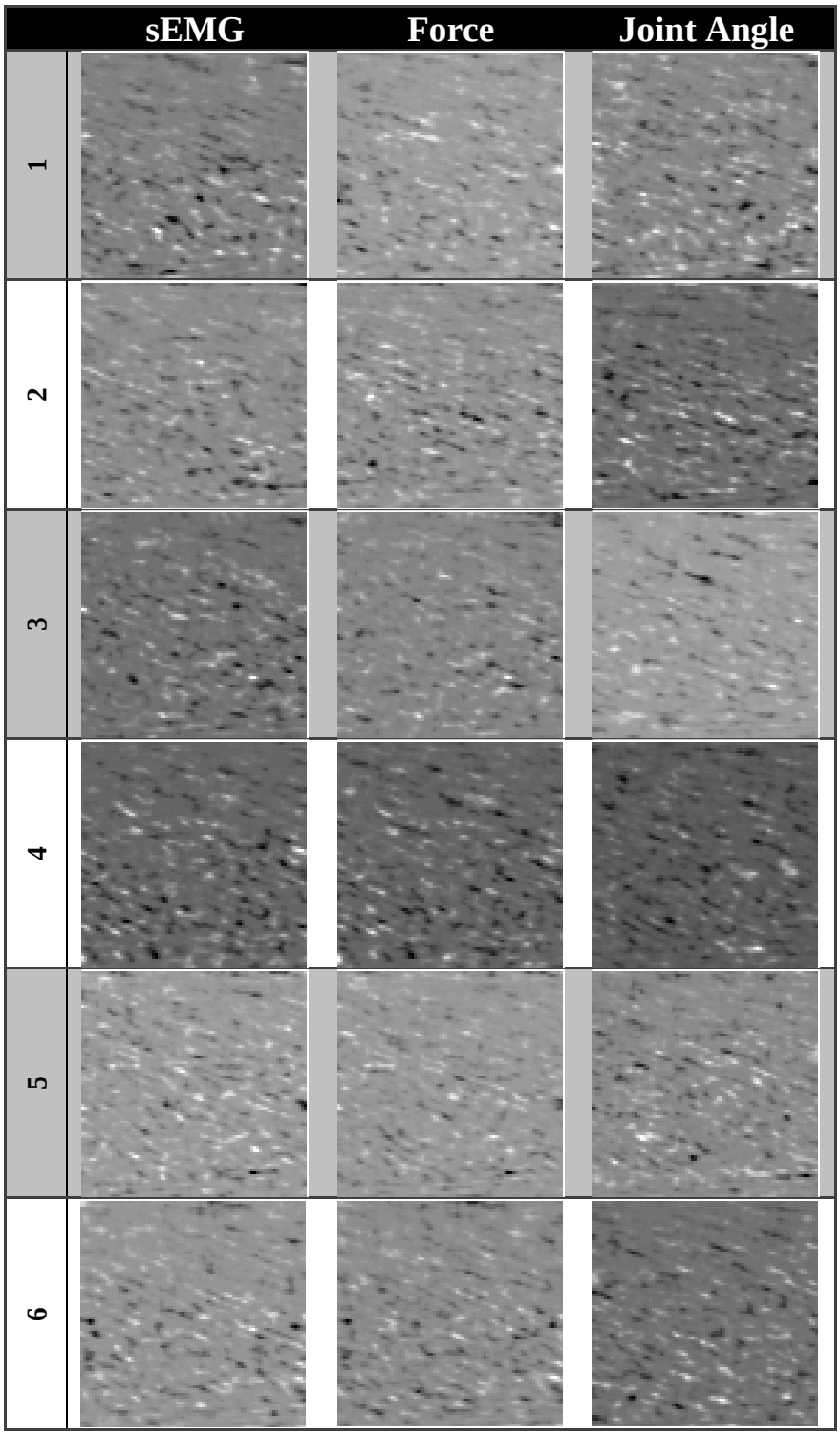
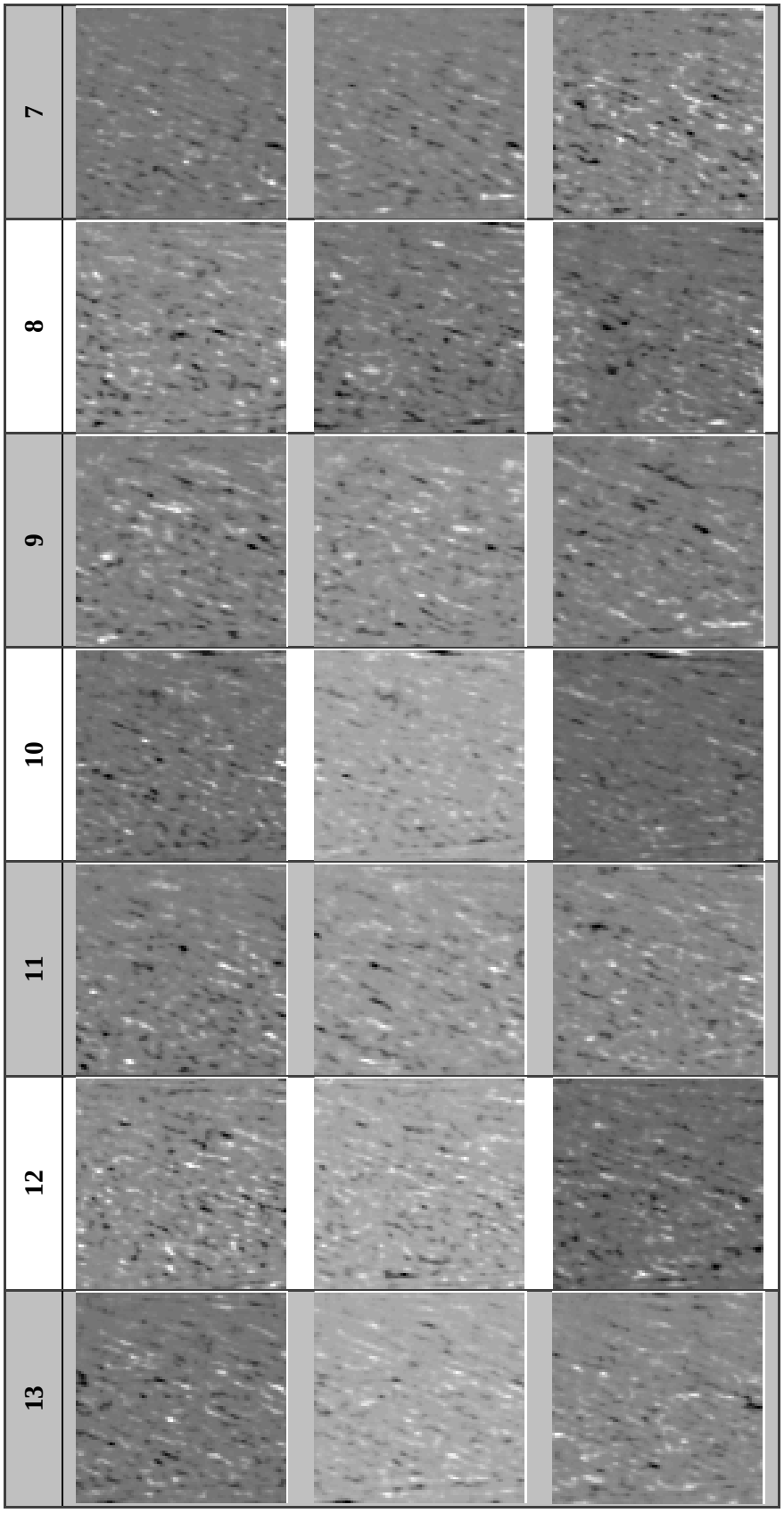


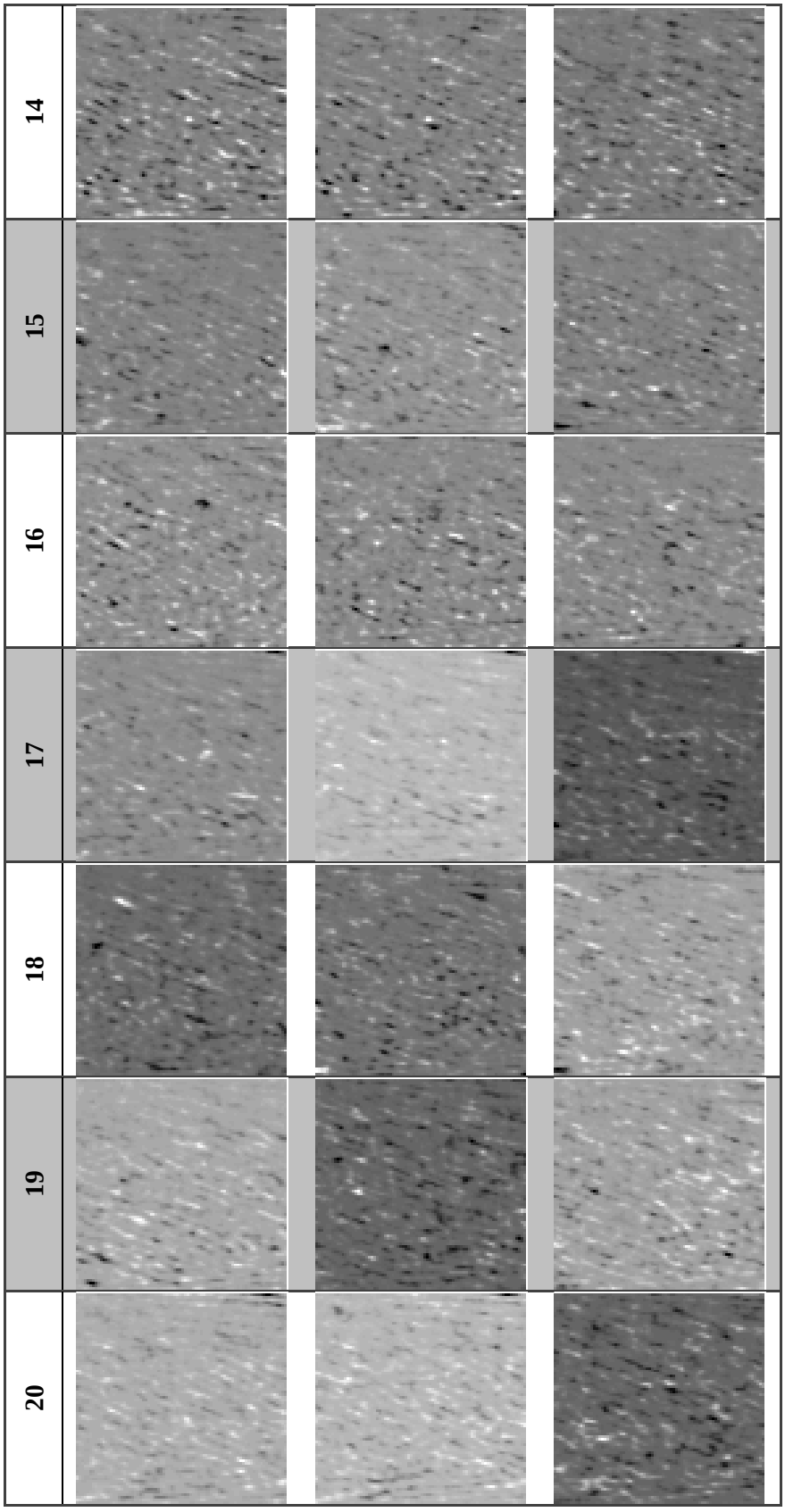
Figure 77. GBRBM features. Each square in the graphic is a matrix representation of the coefficients (min-max normalised) of a node in the SO GBRBM. The squares are ranked from left to right, top to bottom, in order of absolute Spearman rank correlation coefficient when correlating feature responses with sEMG over all sequences for a representative participant. This picture is therefore an image representation of the entire SO GBRBM.

8.3 Table of Optimised RBM Features – Medial Gastrocnemius

Table 13. This table shows the weighted linear combination (optimised by linear neural network) of feature coefficients from the MG GBRBM model for each participant. To get each image, the receptive fields of all GBRBM hidden nodes was added together, weighted by the coefficients of the linear model that was trained to predict each of the 3 signals (sEMG, Force, and Joint Angle respectively). Each row therefore represents the features/coefficients specific to each participant, each of which linearly predict each of the quantities in each column.

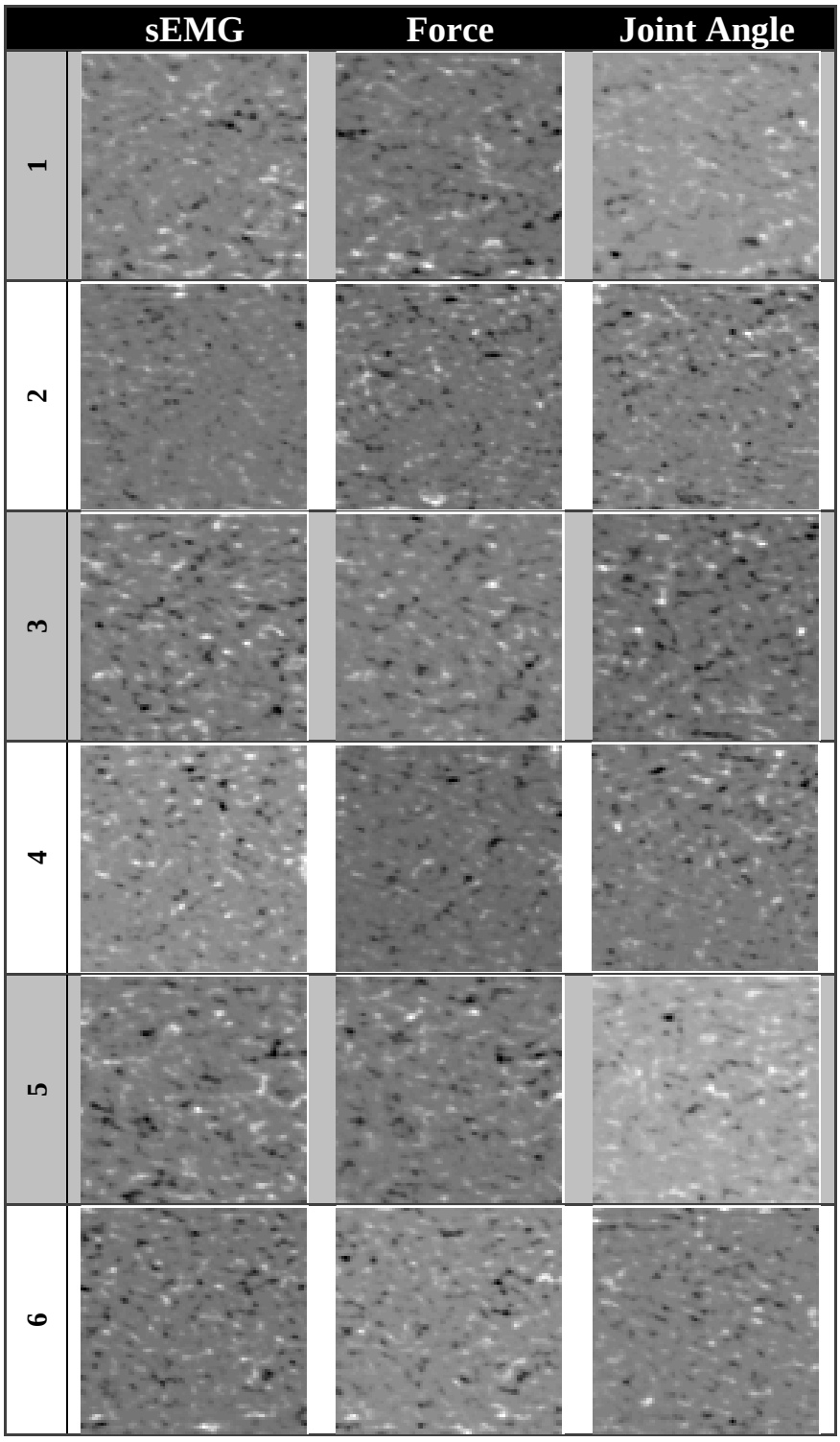


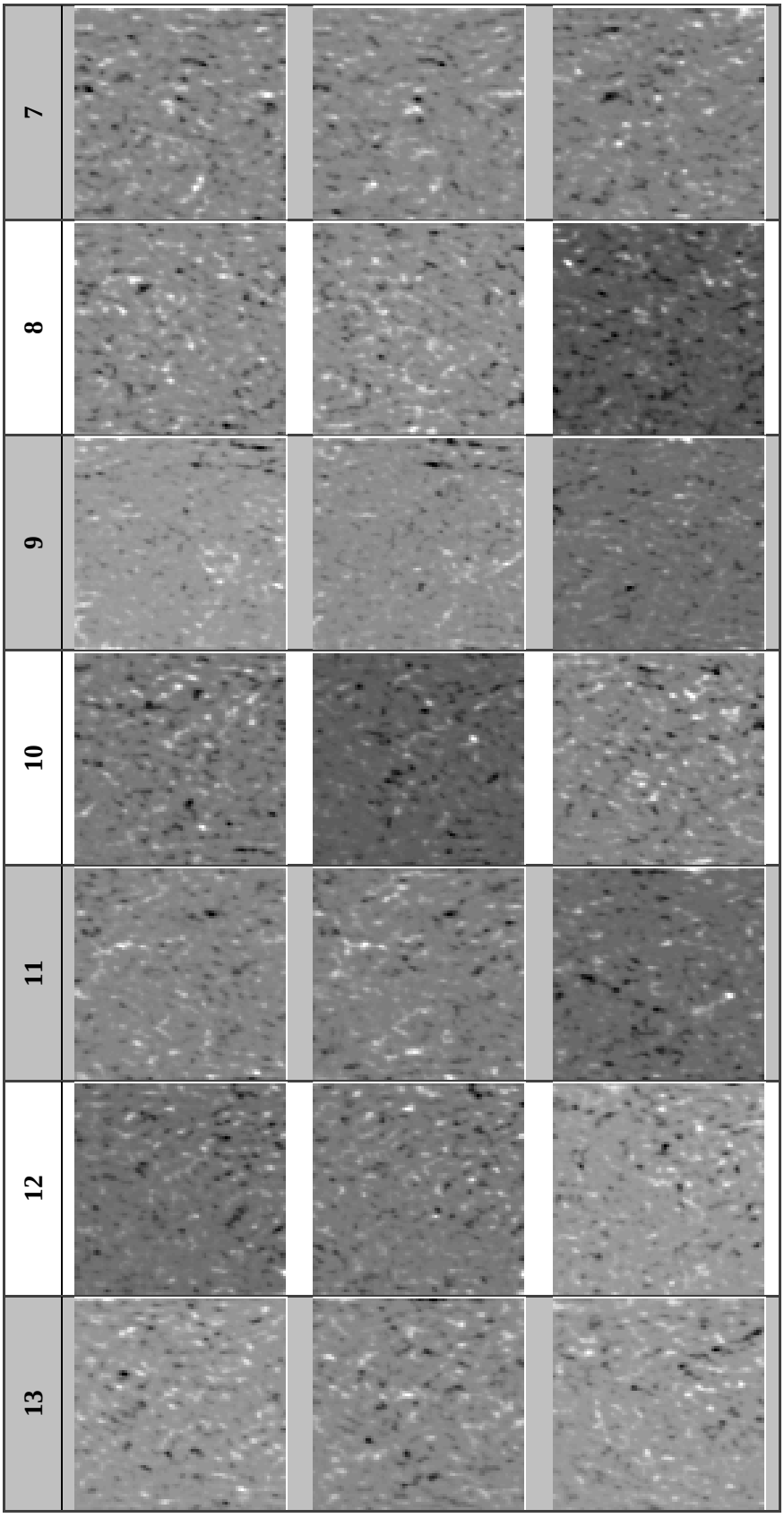

7
8
9
10
11
12
13

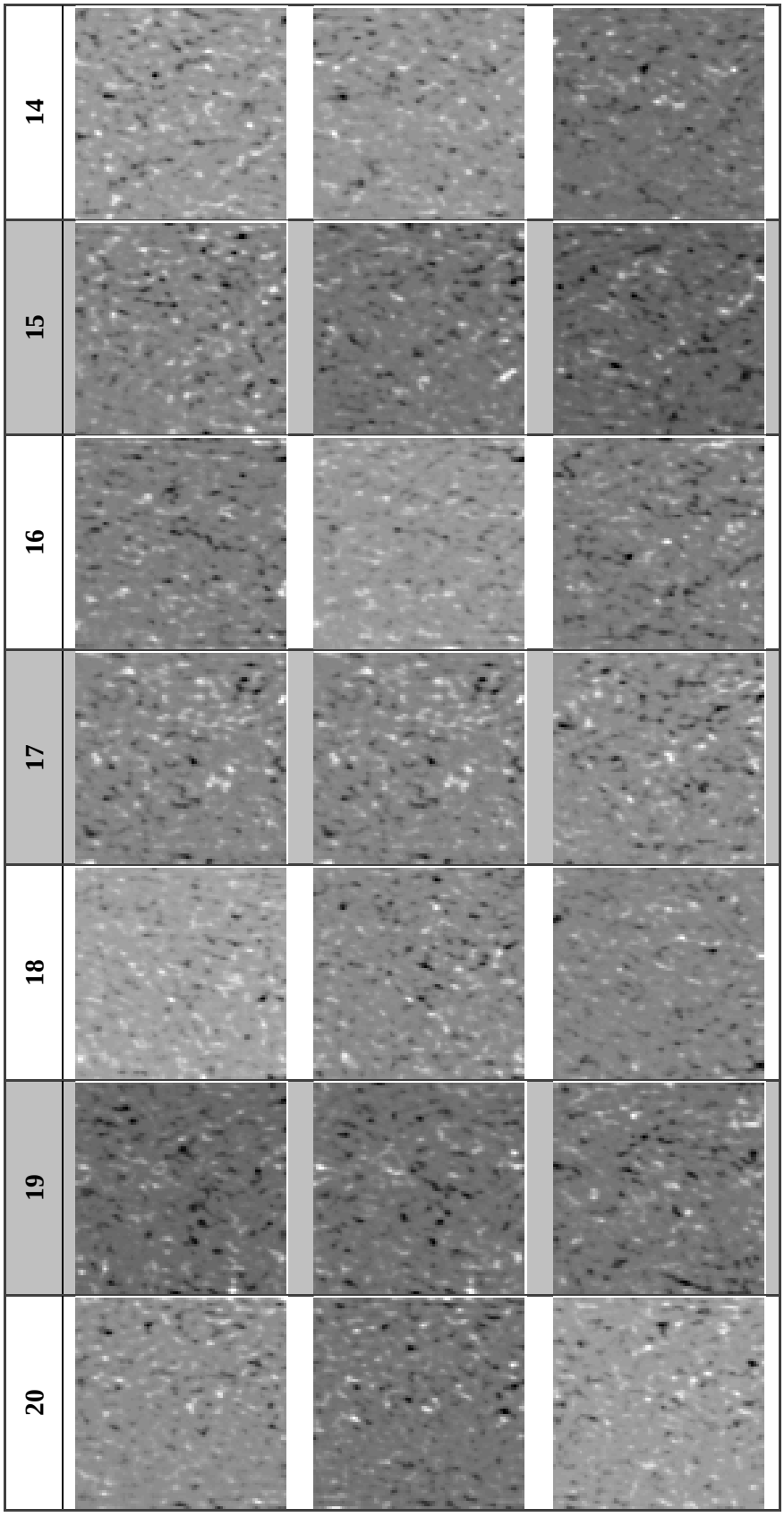

14
15
16
17
18
19
20

8.4 Table of Optimised RBM Features – Soleus

Table 14. This table shows the weighted linear combination (optimised by linear neural network) of feature coefficients from the SO GBRBM model for each participant. To get each image, the receptive fields of all GBRBM hidden nodes was added together, weighted by the coefficients of the linear model that was trained to predict each of the 3 signals (sEMG, Force, and Joint Angle respectively). Each row therefore represents the features/coefficients specific to each participant, each of which linearly predict each of the quantities in each column.

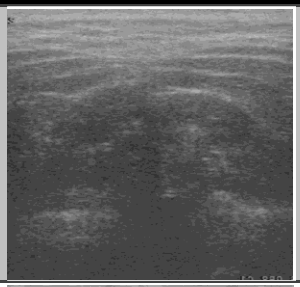
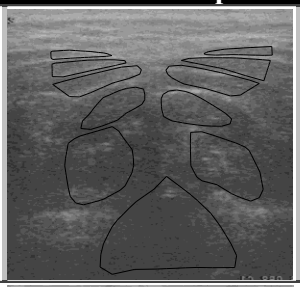
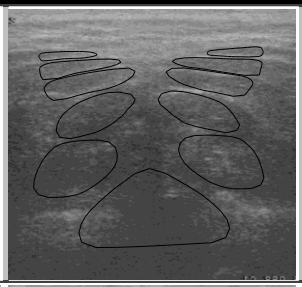
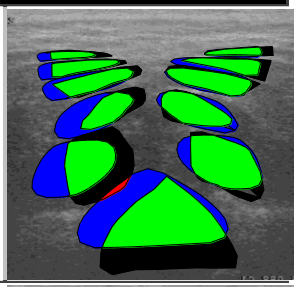
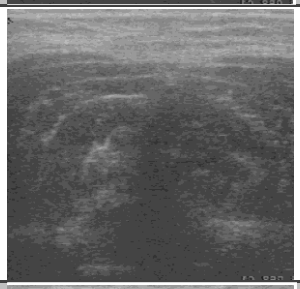
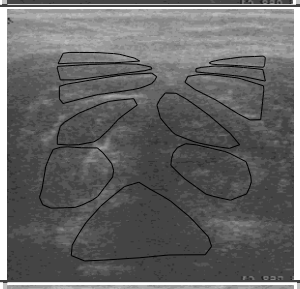
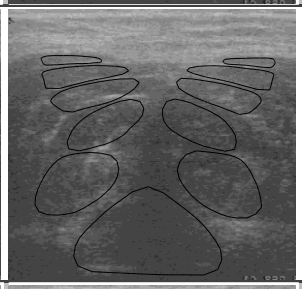
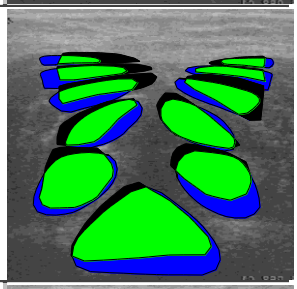
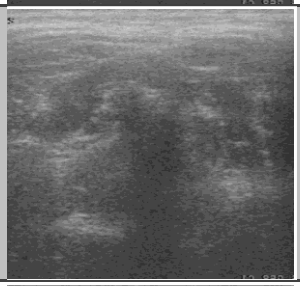
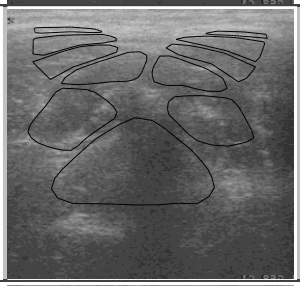
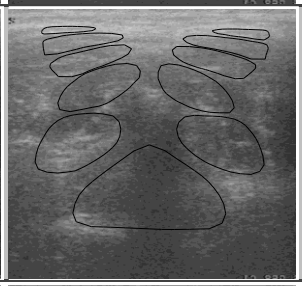
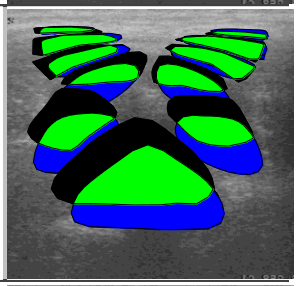
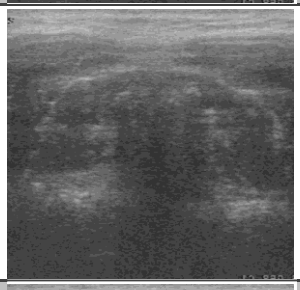
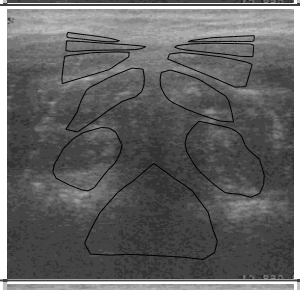
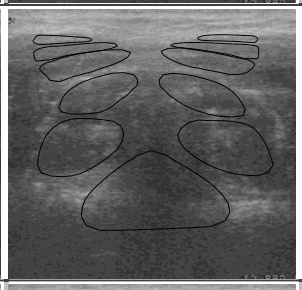
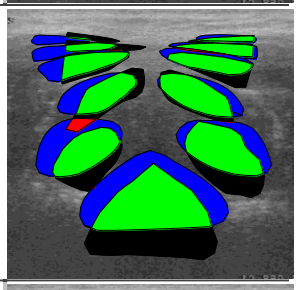
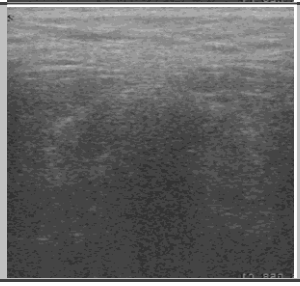
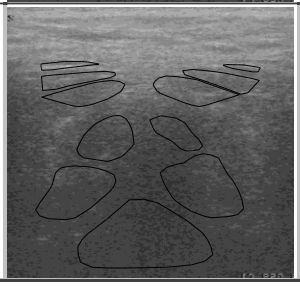
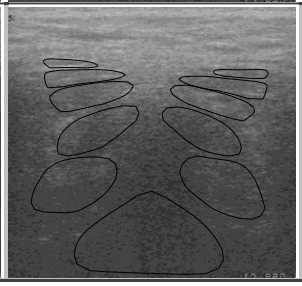
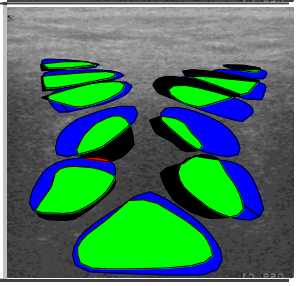


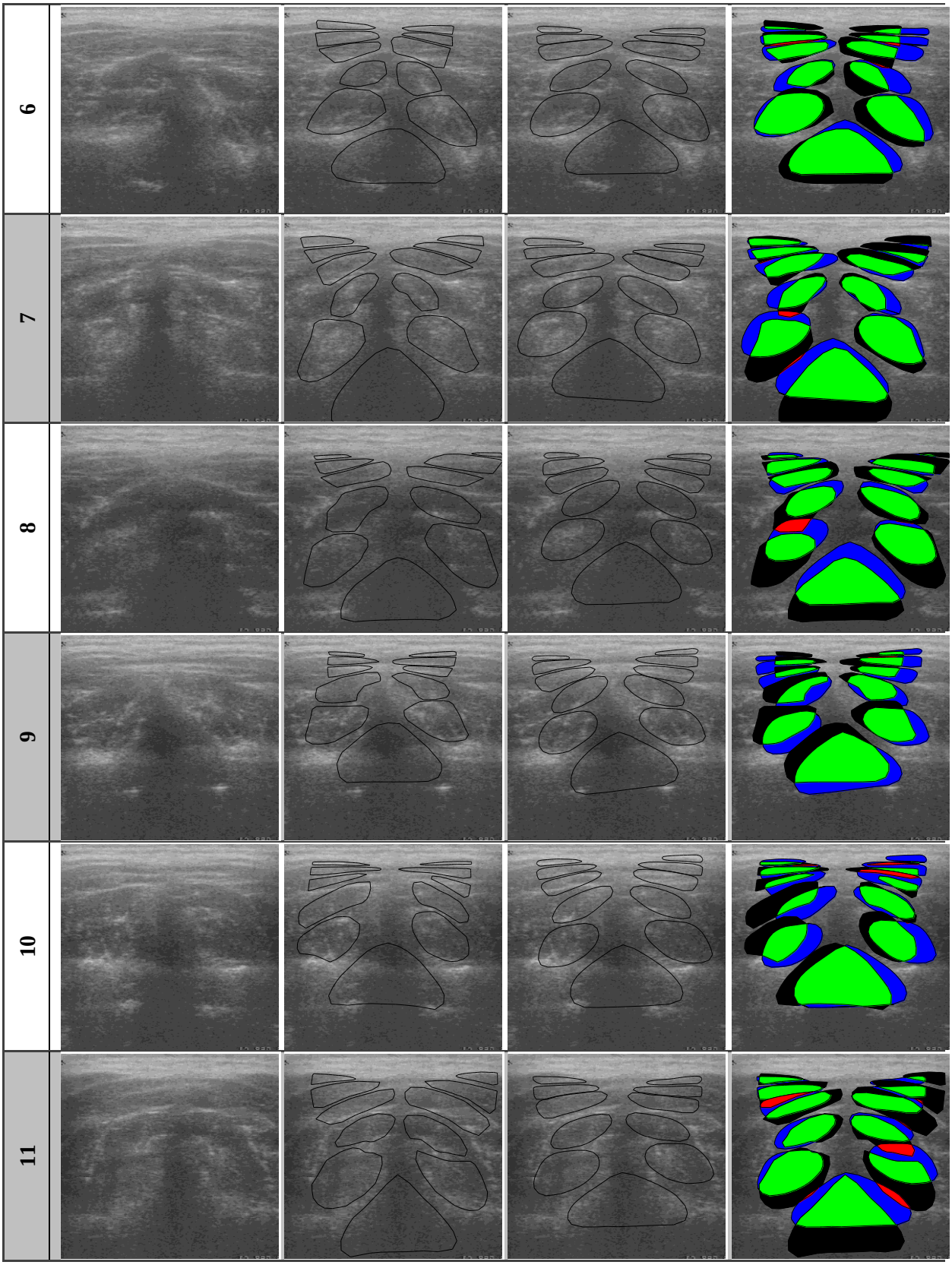

7
8
9
10
11
12
13

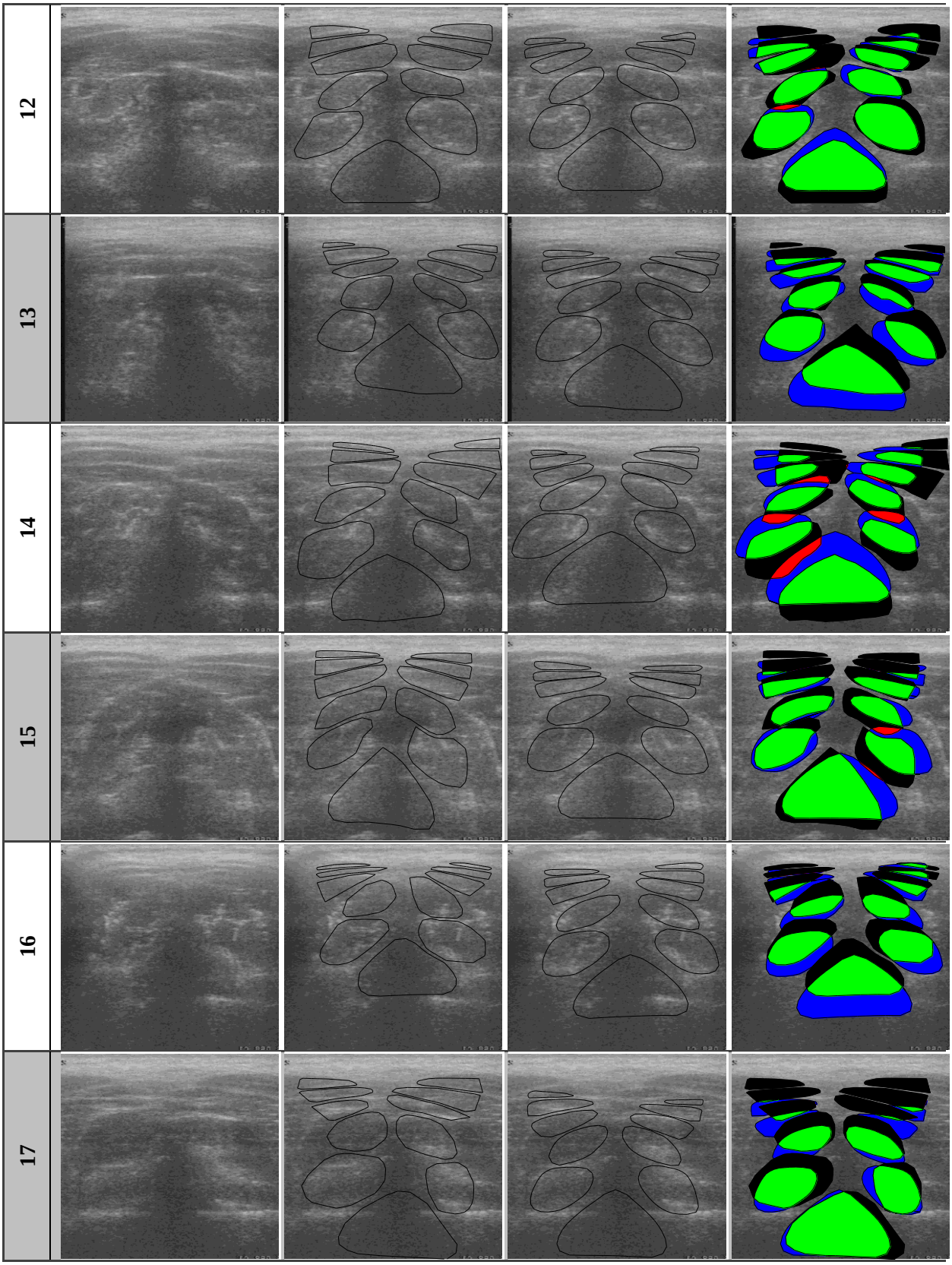

14
15
16
17
18
19
20

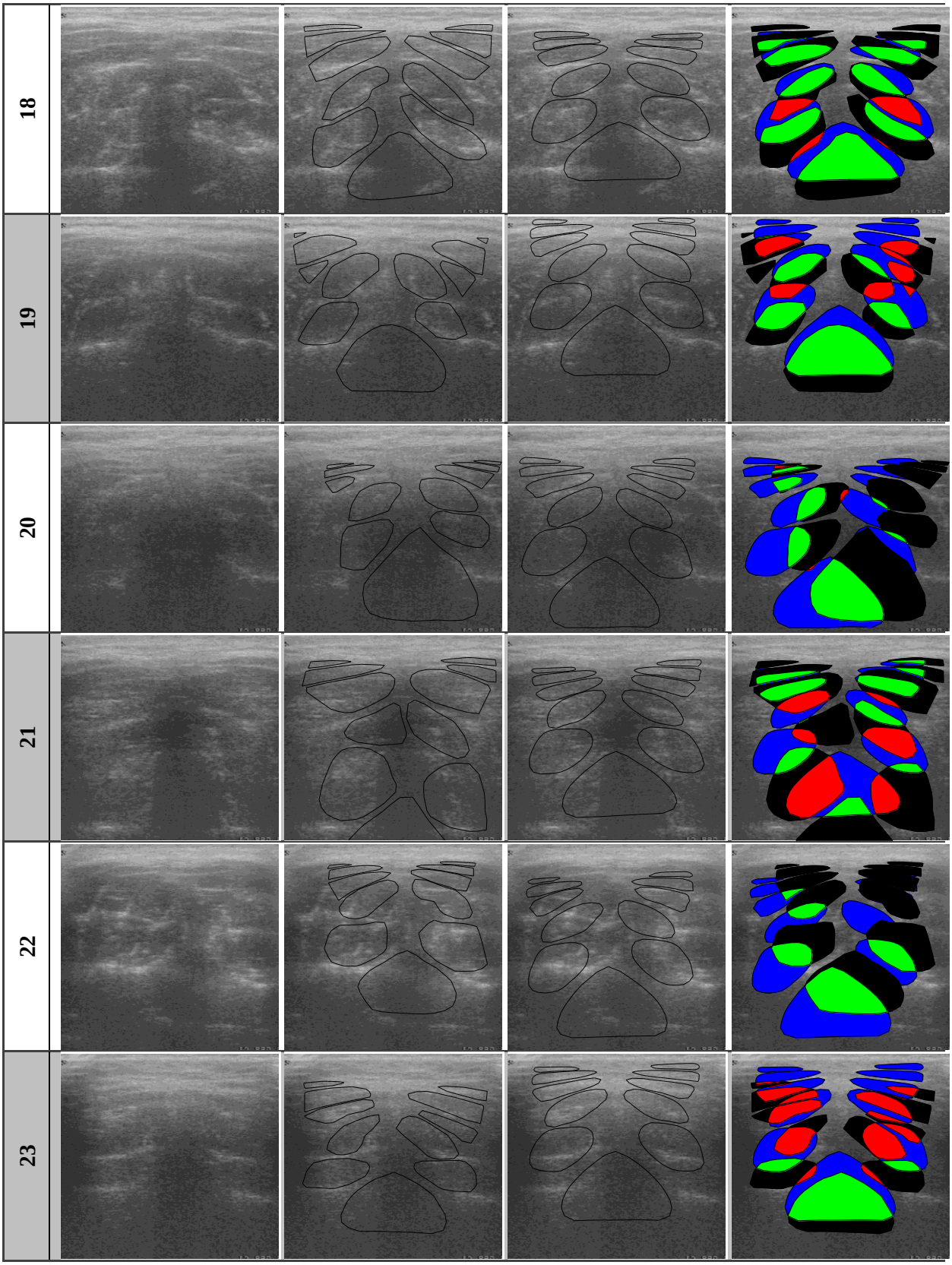
8.5 Table of Neck Segmentation Results

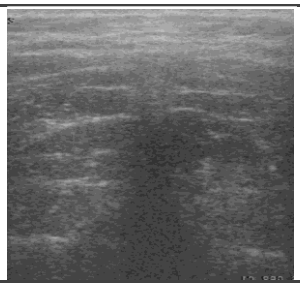
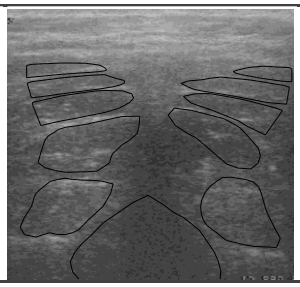
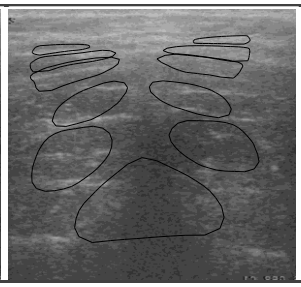
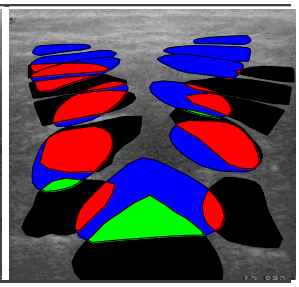
Table 15. The table below shows the raw ultrasound images of the human posterior neck, a registered MRI mark-up, an automatic segmentation, and a classification. The number in the column on the left represents a rank from 1 to 23 (number of participants) based on Equation 18. The image on the right shows the classification according to Equation 19, where green is agreement, red is disagreement, blue is unclassified automatic segmentation, and black is unclassified expert annotation.

	Ultrasound Image	Registered MRI Mark-up	Segmentation	Classification
1				
2				
3				
4				
5				


6
7
8
9
10
11


12
13
14
15
16
17


18
19
20
21
22
23

24				

8.6 HUMAN CALF SEGMENTATION: AN ORIGINAL IMPLEMENTATION – MATLAB SOURCE CODE

8.6.1 USAGE

To use the code, all scripts and functions below must be loaded into the MATLAB path. Then you must have a video file, cropped to include only the ultrasound grey-scale image of a gastrocnemius and soleus muscle. Then type the following in the MATLAB command line:

load('ASM_Gastrocnemius')

to load the gastrocnemius statistical model or,

load('ASM_Soleus')

to load the soleus statistical model. Then, simply type,

ASM(model, filename, xmm, ymm);

to begin segmentation of every frame of the video, where filename is the path and filename of the video you with to segment, and xmm and ymm are the respective x and y dimensions in millimetres that the video resolution represents.

8.6.2 ASM.M

```
function [ASM] = ASM(model, filename, xmm, ymm)
    %
    % begin processing
    %
        %load video
    videoFile = VideoReader(filename);
    NF = videoFile.NumberOfFrames;

        %get the first frame
    img = read(videoFile, 1);

    %
    % model preprocessing - model scaling
    %
        % model multipliers from mm to pixels
    old_imgx = model.params.sx * model.params.xmm;
    old_imgy = model.params.sy * model.params.ymm;
    new_imgx = size(img, 2);
    new_imgy = size(img, 1);
    a = new_imgx / old_imgx * xmm / model.params.xmm;
    b = new_imgy / old_imgy * ymm / model.params.ymm;
    sx = old_imgx / model.params.xmm * a;
    sy = old_imgy / model.params.ymm * b;
        % model positioning offset proportional to image width
    ox = new_imgx * model.params.ox;
    oy = new_imgy * model.params.oy;

        %location model
```

```
    model.locationModel.eigenVectors = model.locationModel.eigenVectors *
sy * model.params.imScale;
    model.locationModel.yMu = (model.locationModel.yMu * sy + oy) *
model.params.imScale * model.params.sy;
        %shape model
    model.shapeModel.eigenVectors = model.shapeModel.eigenVectors * sy *
model.params.imScale;
    model.shapeModel.yMu = (model.shapeModel.yMu * sy + oy) *
model.params.imScale * model.params.sy;
    model.shapeModel.xMu = (model.shapeModel.xMu * sx + ox) *
model.params.imScale * model.params.sx;
        %parameters
    model.params.k = ceil(model.params.k * model.params.imScale);

    %process all frames
    ASM.yMu = cell(NF-1,1);
    ASM.xMu = model.shapeModel.xMu / model.params.imScale;
    ASM.t = cell(NF-1,1);

    start = tic; %initialise timer
    model.shapeModel.yMu = ASMfit2(imresize(img, model.params.imScale),
model, model.params.maxIt);

    idx = [1:model.shapeModel.contourLen; model.shapeModel.contourLen +
1:model.shapeModel.contourLen + model.shapeModel.contourLen];

        %reduce the search range of the shape model
    model.locationModel.sig = model.locationModel.sig2;
    model.shapeModel.sig = model.shapeModel.sig2;

    for f = 1:NF
        img = read(videoFile, f);
        model.shapeModel.yMu = ASMfit2(imresize(img, model.params.imScale),
model, model.params.maxIt2);
        ASM.t{f} = toc(start); %get time stamp
        ASM.yMu{f} = model.shapeModel.yMu / model.params.imScale;

        %draw shape model
        sxMu = model.shapeModel.xMu / model.params.imScale;
        syMu = model.shapeModel.yMu / model.params.imScale;
        figure(2); clf; hold on; axis([0 size(img, 2) 0 size(img, 1)]);
set(gca, 'YDir', 'reverse');
        imshow(img); hold on; title([num2str(f),' of ',num2str(NF)]),
        line(sxMu(idx(1, :)), syMu(idx(1, :)), 'Color', 'y');
        line(sxMu(idx(2, :)), syMu(idx(2, :)), 'Color', 'y');

        display(['segmented ', num2str(f),' of ',num2str(NF), 'frames']);
    end
end
```

8.6.3 ASMFIT2.M

```
function [best] = ASMfit2(img, model, iterations)
    best = model.shapeModel.yMu;

    idx = [1:model.shapeModel.contourLen; model.shapeModel.contourLen +
1:model.shapeModel.contourLen + model.shapeModel.contourLen];

    score = [-Inf -Inf];
    for c = 1:iterations
        if c > 1
            Lsample = getRandomVector(model.locationModel);
            sample = model.shapeModel.yMu +
getRandomVector(model.shapeModel);
            sample(idx(1, :)) = sample(idx(1, :)) + Lsample(1);
            sample(idx(2, :)) = sample(idx(2, :)) + Lsample(2);
        else
            sample = model.shapeModel.yMu;
        end

        sum = [0 0];
        for k = 1:model.shapeModel.contourLen - 1
            sum(1) = sum(1) + getMedianDerivative(img, ...
                [sample(k), ...
                    model.shapeModel.xMu(k)], ...
                [sample(k + 1), ...
                    model.shapeModel.xMu(k + 1)], ...
                model.params.k, ...
                model.params.cMult(1:model.shapeModel.contours) ...
            ) * model.shapeModel.contourSign(k);

            sum(2) = sum(2) + getMedianDerivative(img, ...
                [sample(k + model.shapeModel.contourLen), ...
                    model.shapeModel.xMu(k + model.shapeModel.contourLen)],
...
                [sample(k + model.shapeModel.contourLen + 1), ...
                    model.shapeModel.xMu(k + model.shapeModel.contourLen +
1)], ...
                model.params.k, ...
                model.params.cMult(model.shapeModel.contours+1:end) ...
            ) * model.shapeModel.contourSign(k +
model.shapeModel.contourLen);
        end

        if sum(1) > score(1)
            best(idx(1, :)) = sample(idx(1, :));
            score(1) = sum(1);
        end

        if sum(2) > score(2)
            best(idx(2, :)) = sample(idx(2, :));
            score(2) = sum(2);
        end

%         figure(1); clf; hold on; axis([0 size(img, 2) 0 size(img, 1)]);
set(gca, 'YDir', 'reverse'); title([mat2str(c), ' of ',
mat2str(iterations)]);
%         imshow(img); hold on;
%         line(model.shapeModel.xMu(idx(1, :)), sample(idx(1, :)), 'Color',
```

```
'b');
%         line(model.shapeModel.xMu(idx(2, :)), sample(idx(2, :)), 'Color',
'b');
%         line(model.shapeModel.xMu(idx(1, :)), best(idx(1, :)), 'Color',
'y');
%         line(model.shapeModel.xMu(idx(2, :)), best(idx(2, :)), 'Color',
'y');
%         drawnow;
    end
end
```

8.6.4 GETANGLE.M

```
function [alpha] = getAngle(X, Y)
    dy = Y(2) - Y(1);
    dx = X(2) - X(1);
    alpha = atand(dy/dx);
end
```

8.6.5 GETMEDIANDERIVATIVE.M

```
function [d] = getMedianDerivative(img, yx, yx1, my, s)
    try
        [c, r] = makePoly(yx(2), yx(1), yx1(2), yx1(1), my);
        roi1 = roipoly(img, c, r);
        med1 = median(im2double(img(roi1)));

        [c, r] = makePoly(yx(2), yx(1), yx1(2), yx1(1), -my);
        roi2 = roipoly(img, c, r);
        med2 = median(im2double(img(roi2)));

        d = med1*s(1) - med2*s(2);
    catch err
        d = 0;
    end
end
```

8.6.6 GETRANDOMVECTOR.M

```
function [sample] = getRandomVector(model)
    x = model.contourLen * model.contours;
    sample = zeros(1, x, 1);

    for i = x:-1:x-model.PC + 1
        v = model.sig * model.eigenValues(i, i)^0.5;
        w = -v + (v - -v)*rand(1);
        sample = sample + w*model.eigenVectors(:, i)';
    end
end
```

8.6.7 MAKEPOLY.M

```
function [c, r] = makePoly(x, y, x1, y1, s)
    dx = x1 - x;
    dy = y1 - y;
    t = atand(dy/dx) - 90;
    o = s*sind(t);
    a = s*cosd(t);
    c = [x, x + a, x1 + a, x1];
    r = [y, y + o, y1 + o, y1];
end
```

8.7 Gaussian Bernoulli Restricted Boltzmann Machine: An Original Implementation – C++/CUDA Source code

8.7.1 main.cpp

```
#include <iostream>
#include <algorithm>
#include <time.h>
#include "cudaRBM.h"

using namespace std;

//note: linker->system Heap size is in bytes 2^29 MAX! (times 4 if 64 bit)
void main(int argc, char *argv[]) {
	int gpu = 0;
	cudaSetDevice(gpu);

	cudaDeviceProp device_properties;
	cudaGetDeviceProperties(&device_properties, gpu);
	cout<<"device: "<<device_properties.name<<"\n";
	cout<<"L2 cache: "<<device_properties.l2CacheSize<<"\n";
	cout<<"warp size: "<<device_properties.warpSize<<"\n";
	cout<<"const mem: "<<device_properties.totalConstMem<<"\n";
	cout<<"global mem: "<<device_properties.totalGlobalMem<<"\n";
	cout<<"memory clock rate: "<<device_properties.memoryClockRate<<"\n";
	cout<<"clock rate: "<<device_properties.clockRate<<"\n";
	cout<<"concurrent kernels: "<<device_properties.concurrentKernels<<"\n";
	cout<<"multi-processors: "<<device_properties.multiProcessorCount*192<<"\n";
	cout<<"compute capability: "<<device_properties.major<<"\n\n";

	clock_t t1, t2;

	int H = 5625;//75^2
	int V = 7744;
	float mu = 0.0001;
	float noise = 1;
	float wcoeff = 0.0001;
	float lambda = 0;

	cudaRBM rbm(V, H, wcoeff);
	rbm.SetParams(mu, noise, lambda);
	//rbm.load("E:/GBRBM_GASTROCNEMIUS/gbrbm999.rbm");

	cout<<"rbm initialised...\n";

	float error;
	int epochs = 2000;
	int counter = 0;

	//training data
	int trainr = 633830, trainc = V;//633830
	float **training;
	importCSV_2("E:/GBRBM_GASTROCNEMIUS/GM_data_train.csv", &training, trainr,
trainc);

	int validr = 33447, validc = V;//33447
	float **validation;
	importCSV_2("E:/GBRBM_GASTROCNEMIUS/GM_data_train.csv", &validation, validr,
validc);
```

```
	int testr = 28374, testc = V;//28374
	float **testing;
	importCSV_2("E:/GBRBM_GASTROCNEMIUS/GM_data_validation.csv", &testing, testr,
testc);

	//training order pointers
	int *order = (int*)malloc(sizeof(int)*trainr);
	for (int i = 0; i <= trainr; i++) order[i] = i;

	cout<<"data loaded...\n";

	for (int i = 0; i < epochs; i++) {
		//shuffle training order
		randomOrder(order, trainr);
		cout<<"data shuffled...\n";

		t1 = clock();
		for (int j = 0; j < trainr; j++) {
			rbm.Data(training[order[j]]);
			rbm.CD(1);

			//if (realRandom(0, 1) > 0.9) {
				rbm.UpdateW();
				cout<<100.0f/trainr*j<<"%\n";
			//}

			//***************
			//* ERROR CALC (approximately every 10000 iterations)
			//***************
			if ((realRandom(0, 1) > 0.99 && realRandom(0, 1) > 0.99) || (i ==
0 && j == 0)) {
				std::ofstream errorFile;

	errorFile.open("E:/GBRBM_GASTROCNEMIUS/save_data/GM_gbrbm_errors.csv",
std::ios_base::app);

				//switch off noise during reconstruction
				rbm.SetParams(mu, 0, lambda);

				//training set error
				error = 0; counter = 0;
				for (int k = 0; k < trainr; k++) {
					if (realRandom(0, 1) > 0.999) {
						rbm.Data(training[k]);
						rbm.Reconstruct();
						error += rbm.MSError();
						counter++;
					}
				}
				cout<<i<<": train "<<error/counter<<"\n";
				errorFile<<error/counter<<",";

				//validation set error
				error = 0; counter = 0;
				for (int k = 0; k < validr; k++) {
					if (realRandom(0, 1) > 0.99) {
						rbm.Data(validation[k]);
						rbm.Reconstruct();
						error += rbm.MSError();
						counter++;
					}
```

```
                        }
                        cout<<i<<": validation "<<error/counter<<"\n";
                        errorFile<<error/counter<<",";

                        //testing set error
                        error = 0; counter = 0;
                        for (int k = 0; k < testr; k++) {
                                if (realRandom(0, 1) > 0.99) {
                                        rbm.Data(testing[k]);
                                        rbm.Reconstruct();
                                        error += rbm.MSError();
                                        counter++;
                                }
                        }
                        cout<<i<<": test "<<error/counter<<"\n";
                        errorFile<<error/counter<<"\n";

                        errorFile.close();

                        //save boltzmann machine
                        string filename =
"E:/GBRBM_GASTROCNEMIUS/save_data/GM_gbrbm_";
                        filename.append(string(to_string((long long)i)));
                        filename.append("_");
                        filename.append(string(to_string((long long)j)));
                        filename.append(".rbm");
                        rbm.save(filename);

                        //add noise back to RBM
                        rbm.SetParams(mu, noise, lambda);
                }
        }
        t2 = clock();
        cout<<"epoch "<<i<<" computed in "<<((float)t2-
(float)t1)/CLOCKS_PER_SEC<<" seconds...\n";
    }

    system("pause");
}
```

8.7.2 CUDABACKPROP.H

```
#ifndef ANN_CUDA_BACKPROP_KERNEL
#define ANN_CUDA_BACKPROP_KERNEL

#include "cudaRBM.h"

using namespace std;

class cudaBackprop {
public:
	cudaBackprop();
	cudaBackprop(int _l, int *_s1, int *_s2, float _wcoeff);
	~cudaBackprop();

	void Data(float *_data_i, float *_data_t);
	void Clamp();
	void FeedForward();
	void FeedbackError();
	void UpdateW();
	float MSError();

	void SetParams(float _mu, float _noise, float _lambda);
	void GetParams(float *_mu, float *_noise, float *_lambda);
	void ToDevice();
	void FromDevice();
	void Reset();
	void create(int _l, int *_s1, int *_s2, float _wcoeff);
	void destroy();
	void save(string filename);
	void load(string filename, int _l);

	int Size();
	cudaRBM *GetCudaRBM(int _k);
	void LinkLayers();

private:
	int net;
	cudaRBM **crbm;
	float *data_i, *data_t;
};

#endif
```

8.7.3 *CUDARBM.H*

```
#ifndef ANN_CUDA_RBM
#define ANN_CUDA_RBM

#include <iostream>
#include "RBM.h"
#include "RBMkernel.cuh"
#include <fstream>
#include <string>
#include <sstream>

using namespace std;

class cudaRBM : public RBM {
public:
	cudaRBM();
	cudaRBM(int _n, int _m, float _wcoeff);
	~cudaRBM();

	void ResizeV(int _n);
	void ResizeH(int _m);
	void Data(float *_data);
	void Clamp();
	void BinaryH();
	void ConstructH();
	void ConstructHLinear();
	void ConstructV();
	void ConstructVLinear();
	void UpdateDW();
	void UpdateW();
	void CopyStats();
	void CD(int _n); //Contrastive Divergence _n steps in the Markov Chain
	void Construct();
	void Reconstruct();

	void SetParams(float _mu, float _noise, float _lambda);
	void GetParams(float *_mu, float *_noise, float *_lambda);

	void ToDevice();
	void FromDevice();

	void Reset();
	void Alloc();
	void Clean();

	void save(string filename);
	void load(string filename);

	//backprop modifications
	void SetDataT(float *data_t);
	void ClampH();
	void ComputeError();
	void Backpropagate();
	void UpdateBackpropagation();
	void Link(cudaRBM *_next);
	myStruct *CU() {return &cu;};

private:
	myStruct cu;//cuda device variables
};
```

```
#endif
```

8.7.4 LAYER.H

```
#ifndef ANN_LAYER
#define ANN_LAYER

#include <iostream>

using namespace std;

class Layer {
public:
	Layer(int _n);
	~Layer();

	void Reset();
	void Resize(int _n);
	void S(float _si, int _i);
	void Copy(Layer *_l);

	float S(int _i);
	float *S();
	int Size();

private:
	int size;
	float *s;
};

#endif
```

8.7.5 RBM.H

```
#ifndef ANN_RBM
#define ANN_RBM

#include <iostream>
#include "Layer.h"
#include "Weight.h"
#include "util.h"
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

using namespace std;

class RBM {
public:
	RBM();
	RBM(int _n, int _m, float _wcoeff);
	~RBM();

	void ResetW();
	void ResizeV(int _n);
	void ResizeH(int _n);
	void Data(float *_data);
	void Clamp();
	void BinaryH();
	void ConstructH();
	void ConstructV();
	void UpdateDW();
	void UpdateW();
	void CopyStats();
	void CD(int _n); //Contrastive Divergence _n steps in the Markov Chain
	void Construct();
	void Reconstruct();
	float MSError();

	void create(int _n, int _m);
	void destroy();
	void save(string filename);
	void load(string filename);

	void SetParams(float _mu, float _noise, float _lambda) {mu = _mu; noise =
_noise; lambda = _lambda;};
	void GetParams(float *_mu) {*_mu = mu;};

	float Mu() {return mu;};

	Layer *V();
	Layer *H();
	Layer *PV();
	Layer *PH();
	Weight *W();
	Weight *DW();

	//backprop modifications
	void Link(RBM *_next);
	void LinkedV(bool _linked) {linked_v = _linked;};

private:
	Layer *v, *h;
```

```
        Layer *pos_v, *pos_h;
        Weight *w, *dw;
        float *data;

        float lambda;
        float mu;
        float noise;
        float wcoeff;

        int cd;

        //backprop modifications
        bool linked_h, linked_v;
};

#endif
```

8.7.6 RBMKERNEL.CUH

```
#ifndef ANN_RBM_KERNEL
#define ANN_RBM_KERNEL

#include <cuda.h>
#include <cuda_runtime.h>
#include <curand_kernel.h>

struct myStruct {
	float *d_w, *d_dw, *d_bv, *d_dbv, *d_bh, *d_dbh;
	float *d_v, *d_h, *d_pv, *d_ph;
	float *d_data, *d_data_t;
	curandState *d_s;
	int d_randlen;
	float d_mu;
	float d_noise;
	float d_lambda;
	int d_cd;
	int d_size_v, d_size_h;
	bool linked_v, linked_h;
	int d_n_blocks_v, d_n_blocks_h, d_n_blocks_rand;
	int d_block_size_v, d_block_size_h, d_block_size_rand;
	int d_n_cores;
};

void cudaBinaryH(myStruct *cu);
void cudaConstructH(myStruct *cu);
void cudaConstructHLinear(myStruct *cu);
void cudaConstructV(myStruct *cu);
void cudaConstructVLinear(myStruct *cu);
void cudaUpdateDW(myStruct *cu);
void cudaUpdateW(myStruct *cu);
void cudaCopyStats(myStruct *cu);
void cudaClamp(myStruct *cu);
void cudaInit(myStruct *cu);

void cudaSetData(myStruct *cu, float *_data);
void cudaSetParams(myStruct *cu, float _mu, float _noise, float _lambda);
void cudaSetV(myStruct *cu, float *_v, float *_pv, float *_bv);
void cudaSetH(myStruct *cu, float *_h, float *_ph, float *_bh);
void cudaSetW(myStruct *cu, float *_w);

void cudaGetParams(myStruct *cu, float *_mu, float *_noise, float *_lambda);
void cudaGetV(myStruct *cu, float *_v, float *_pv, float *_bv);
void cudaGetH(myStruct *cu, float *_h, float *_ph, float *_bh);
void cudaGetW(myStruct *cu, float *_w);

void cudaAlloc(myStruct *cu, int _n, int _m);
void cudaClean(myStruct *cu);

//backprop mods
void cudaSetDataT(myStruct *cu, float *_data_t);
void cudaClampH(myStruct *cu);
void cudaComputeError(myStruct *cu);
void cudaBackpropagate(myStruct *cu);
void cudaUpdateBackpropagation(myStruct *cu);
void cudaLink(myStruct *cu, myStruct *cu_next);

#endif
```

8.7.7 *UTIL.H*

```
#ifndef ANN_UTIL
#define ANN_UTIL

#include <random>
#include <time.h>
#include <math.h>
#include <fstream>
#include <string>
#include <sstream>
#include <iostream>

using namespace std;

float realRandom(float _l, float _u);
float gaussRandom(float _l, float _u);
float e(float x);
void importCSV(string filename, float ***data, int *r, int *c);
void importCSV_2(string filename, float ***data, int rows, int cols);
void randomOrder(int *order, int size);

#endif
```

8.7.8 WEIGHT.H

```
#ifndef ANN_WEIGHT
#define ANN_WEIGHT

#include <iostream>
#include "util.h"

using namespace std;

class Weight {
public:
	Weight(int _n, int _m); //create a weight matrix of size nm
	~Weight(); //destroy weight matrix and destruct class

	void Reset(); //reset all weights to 0
	void Reset(float _u, float _l); //reset all weights to random vals between u
and l
	void Resize(int _n, int _m); //resize weight vector to size nm, preserving all
current weights
	void W(float _wij, int _i, int _j); //set value of ijth weight
	void Bv(float _bi, int _i); //set value of ith bias
	void Bh(float _bj, int _j);
	void Bv(float *_bv) {delete bv; bv = _bv; linkedbv = true;};
	void Bh(float *_bh) {delete bh; bh = _bh; linkedbh = true;};

	float W(int _i, int _j); //get value of ijth weight
	float Bv(int _i); //get value of ith hidden bias
	float Bh(int _j); //get value of ith visible bias
	float *W(); //get the weight matrix
	float *Bv(); //get the visible bias vector
	float *Bh(); //get the hidden bias vector
	int *Size(); //return the vector size

private:
	int size[2]; //length of weight vectors
	float *w; //weight matrix
	float *bv, *bh; //bias vector

	//backprop mods
	bool linkedbv, linkedbh;
};

#endif
```

8.7.9 CUDABACKPROP.CPP

```
#include "cudaBackprop.h"

cudaBackprop::cudaBackprop() {
	int s1[2], s2[2];
	s1[0] = 2; s1[1] = 100; s2[0] = 100; s2[1] = 1;
	create(2, s1, s2, 0.0001);
}

cudaBackprop::cudaBackprop(int _l, int *_s1, int *_s2, float _wcoeff) {
	create(_l, _s1, _s2, _wcoeff);
}

cudaBackprop::~cudaBackprop() {
	destroy();
}

void cudaBackprop::Data(float *_data_i, float *_data_t) {
	crbm[0]->Data(_data_i);
	data_t = _data_t;
}

void cudaBackprop::Clamp() {
	crbm[0]->Clamp();
}

void cudaBackprop::FeedForward() {
	for (int k = 0; k < net; k++) {
		if (k < net-1) crbm[k]->BinaryH();
		else crbm[k]->ConstructHLinear();
	}
}

void cudaBackprop::FeedbackError() {
	//clamp the supervised learning vector
	crbm[net-1]->SetDataT(data_t);

	//compute the top layer error
	crbm[net-1]->ComputeError();

	//backpropagate the errors
	for (int i = net-1; i >= 0; i--) crbm[i]->Backpropagate();//updates deltas
during backprop
}

void cudaBackprop::UpdateW() {
	for (int i = 0; i < net; i++) crbm[i]->UpdateBackpropagation();
}

float cudaBackprop::MSError() {
	crbm[net-1]->FromDevice();
	float sum = 0;
	for (int i = 0; i < crbm[net-1]->H()->Size(); i++) sum += pow(crbm[net-1]->H()-
>S(i) - data_t[i], 2);
	return sum/crbm[net-1]->H()->Size();
}

void cudaBackprop::SetParams(float _mu, float _noise, float lambda) {
	for (int k = 0; k < net; k++) crbm[k]->SetParams(_mu, _noise, lambda);
}
```

```
void cudaBackprop::GetParams(float *_mu, float *_noise, float *_lambda) {
	for (int k = 0; k < net; k++) crbm[k]->GetParams(_mu, _noise, _lambda);
}

void cudaBackprop::ToDevice() {
	for (int k = 0; k < net; k++) crbm[k]->ToDevice();
}

void cudaBackprop::FromDevice() {
	for (int k = 0; k < net; k++) crbm[k]->FromDevice();
}

void cudaBackprop::create(int _l, int *_s1, int *_s2, float _wcoeff) {
	net = _l;
	crbm = (cudaRBM**)malloc(sizeof(cudaRBM*)*_l);
	for (int i = 0; i < _l; i++) crbm[i] = new cudaRBM(_s1[i], _s2[i], _wcoeff);
}

void cudaBackprop::destroy() {
	for (int k = 0; k < net; k++) delete crbm[k];
	delete crbm;
}

void cudaBackprop::save(string filename) {
	string fname;
	for (int k = 0; k < net; k++) {
		fname = filename;
		string str = string(to_string((long long)k));
		fname.append("_L"); fname.append(str); fname.append(".rbm");

		crbm[k]->save(fname);
	}
}

void cudaBackprop::load(string filename, int _l) {
	string fname;
	net = _l;
	crbm = (cudaRBM**)malloc(sizeof(cudaRBM*)*_l);

	for (int k = 0; k < net; k++) {
		fname = filename;
		string str = string(to_string((long long)k));
		fname.append("_L"); fname.append(str); fname.append(".rbm");

		crbm[k] = new cudaRBM();
		crbm[k]->load(fname);
	}
}

int cudaBackprop::Size() {
	return net;
}

cudaRBM *cudaBackprop::GetCudaRBM(int _k) {
	return crbm[_k];
}

void cudaBackprop::LinkLayers() {
	for (int i = 0; i < net-1; i++) crbm[i]->Link(crbm[i+1]);
}
```

8.7.10 CUDARBM.CPP

```
#include "cudaRBM.h"

cudaRBM::cudaRBM() : RBM() {
	Reset();
}

cudaRBM::cudaRBM(int _n, int _m, float _wcoeff) : RBM(_n, _m, _wcoeff) {
	Reset();
}

cudaRBM::~cudaRBM() {
	Clean();
}

void cudaRBM::ResizeV(int _n) {
	RBM::ResizeV(_n);
	Clean();
	Reset();
}

void cudaRBM::ResizeH(int _m) {
	RBM::ResizeH(_m);
	Clean();
	Reset();
}

void cudaRBM::Data(float *_data) {
	RBM::Data(_data);
	cudaSetData(&cu, _data);
}

void cudaRBM::Clamp() {
	cudaClamp(&cu);
}

void cudaRBM::BinaryH() {
	cudaBinaryH(&cu);
}

void cudaRBM::ConstructH() {
	cudaConstructH(&cu);
}

void cudaRBM::ConstructHLinear() {
	cudaConstructHLinear(&cu);
}

void cudaRBM::ConstructV() {
	cudaConstructV(&cu);
}

void cudaRBM::ConstructVLinear() {
	cudaConstructVLinear(&cu);
}

void cudaRBM::UpdateDW() {
	cudaUpdateDW(&cu);
}

void cudaRBM::UpdateW() {
```

```
	cudaUpdateW(&cu);
}

void cudaRBM::CopyStats() {
	cudaCopyStats(&cu);
}

void cudaRBM::CD(int _n) {
	//positive statistics
	Clamp(); //clamp data vector
	BinaryH(); //stochastic activation of hdden unit

	CopyStats(); //record hidden & visible positive statistics

	//negative statistics
	for (int k = 0; k < _n; k++) {
		ConstructV();
		BinaryH(); //stochastic activation of hdden unit
	}
	UpdateDW();
}

void cudaRBM::Construct() {
	Clamp();
	ConstructH();
	cudaGetH(&cu, H()->S(), PH()->S(), W()->Bh());
}

void cudaRBM::Reconstruct() {
	Clamp();
	BinaryH(); //stochastic activation of hdden unit
	ConstructV();
	cudaGetV(&cu, V()->S(), PV()->S(), W()->Bv());
}

void cudaRBM::SetParams(float _mu, float _noise, float _lambda) {
	RBM::SetParams(_mu, _noise, _lambda);
	cudaSetParams(&cu, _mu, _noise, _lambda);
}

void cudaRBM::GetParams(float *_mu, float *_noise, float *_lambda) {
	cudaGetParams(&cu, _mu, _noise, _lambda);
}

void cudaRBM::ToDevice() {
	cudaSetV(&cu, V()->S(), PV()->S(), W()->Bv());
	cudaSetH(&cu, H()->S(), PH()->S(), W()->Bh());
	cudaSetW(&cu, W()->W());
}

void cudaRBM::FromDevice() {
	cudaGetV(&cu, V()->S(), PV()->S(), W()->Bv());
	cudaGetH(&cu, H()->S(), PH()->S(), W()->Bh());
	cudaGetW(&cu, W()->W());
}

void cudaRBM::Reset() {
	Alloc();
	ToDevice();
	cudaInit(&cu);
}
```

```
void cudaRBM::Alloc() {
	cudaAlloc(&cu, V()->Size(), H()->Size());
}

void cudaRBM::Clean() {
	cudaClean(&cu);
}

void cudaRBM::save(string filename) {
	FromDevice();
	RBM::save(filename);
}

void cudaRBM::load(string filename) {
	Clean();
	RBM::load(filename);
	Reset();
}

void cudaRBM::SetDataT(float *data_t) {
	cudaSetDataT(&cu, data_t);
}

void cudaRBM::ClampH() {
	cudaClampH(&cu);
}

void cudaRBM::ComputeError() {
	cudaComputeError(&cu);
}

void cudaRBM::Backpropagate() {
	cudaBackpropagate(&cu);
	cudaGetV(&cu, V()->S(), PV()->S(), W()->Bv());
}

void cudaRBM::UpdateBackpropagation() {
	cudaUpdateBackpropagation(&cu);
}

void cudaRBM::Link(cudaRBM *_next) {
	RBM::Link((RBM*)_next);//dynamic cast and link host data structures
	cudaLink(&cu, _next->CU());
}
```

8.7.11 LAYER.CPP

```
#include "Layer.h"

Layer::Layer(int _n) {
	s = (float*)malloc(sizeof(float)*_n);
	size = _n;
}

Layer::~Layer() {
	free(s);
	size = 0;
}

void Layer::Reset() {
	for (int i = 0; i < size; i++) s[i] = 0;
}

void Layer::Resize(int _n) {
	s = (float*)realloc(s, sizeof(float)*_n);
	size = _n;
}

void Layer::S(float _si, int _i) {
	s[_i] = _si;
}

void Layer::Copy(Layer *_l) {
	if (size == _l->Size()) for (int i = 0; i < size; i++) _l->S(s[i], i);
}

float Layer::S(int _i) {
	return s[_i];
}

float *Layer::S() {
	return s;
}

int Layer::Size() {
	return size;
}
```

8.7.12 RBM.CPP

```
#include "RBM.h"

RBM::RBM() {
	wcoeff = 0.0001;
	create(1, 1);
}

RBM::RBM(int _n, int _m, float _wcoeff) {
	wcoeff = _wcoeff;
	create(_n, _m);
}

RBM::~RBM() {
	destroy();
}

void RBM::ResetW() {
	w->Reset(-1*wcoeff, 1*wcoeff);
	dw->Reset();
}

void RBM::ResizeV(int _n) {
	v->Resize(_n);
}

void RBM::ResizeH(int _m) {
	h->Resize(_m);
}

void RBM::Data(float *_data) {
	data = _data;
}

void RBM::Clamp() {
	for (int i = 0; i < v->Size(); i++) v->S(data[i], i);
}

void RBM::BinaryH() { //binarise the states of the hidden vector
	float x;
	for (int j = 0; j < h->Size(); j++) {
		x = 0;
		for (int i = 0; i < v->Size(); i++) x += v->S(i)*w->W(i, j); //weighted
sum of visible states
		x = 1/(1+e(-(x + w->Bh(j))));
		if (realRandom(0, 1) <= x) h->S(1, j); //activate the hiden unit
		else h->S(0, j); //deactivate the hidden unit
	}
}

void RBM::ConstructH() { //construct hidden states
	float x;
	for (int j = 0; j < h->Size(); j++) {
		x = 0;
		for (int i = 0; i < v->Size(); i++) x += v->S(i)*w->W(i, j); //weighted
sum of visible states
		h->S(1/(1+e(-(x + w->Bh(j)))), j); //set hidden state via sigmoid
transfer function of x + the bias
	}
}
```

```
void RBM::ConstructV() { //construct visible states
	float x;
	for (int i = 0; i < v->Size(); i++) { //construct visible states
		x = 0;
		for (int j = 0; j < h->Size(); j++) x += h->S(j)*w->W(i, j); //weighted
sum of states
		v->S(w->Bv(i) + noise*gaussRandom(0, 1), i); //set linear state of the
visible unit plus gaussian noise
	}
}

void RBM::UpdateDW() {
	for (int i = 0; i < v->Size(); i++)
		for (int j = 0; j < h->Size(); j++) dw->W(dw->W(i, j) + ( (pos_v-
>S(i)*pos_h->S(j)) - (v->S(i)*h->S(j)) ), i, j);
	for (int i = 0; i < v->Size(); i++) dw->Bv(dw->Bv(i) + ( pos_v->S(i) - v->S(i)
), i);
	for (int j = 0; j < h->Size(); j++) dw->Bh(dw->Bh(j) + ( pos_h->S(j) - h->S(j)
), j);
	cd++;
}

void RBM::UpdateW() {
	for (int i = 0; i < v->Size(); i++)
		for (int j = 0; j < h->Size(); j++) {
			w->W(w->W(i, j) + mu*dw->W(i, j)/cd, i, j);
			dw->W(0, i, j);
		}
	for (int i = 0; i < v->Size(); i++) {
		w->Bv(w->Bv(i) + mu*dw->Bv(i)/cd, i);
		dw->Bv(0, i);
	}
	for (int j = 0; j < h->Size(); j++) {
		w->Bh(w->Bh(j) + mu*dw->Bh(j)/cd, j);
		dw->Bh(0, j);
	}
	cd = 0;
}

void RBM::CopyStats() {
	v->Copy(pos_v);
	h->Copy(pos_h);
}

void RBM::CD(int _n) {
	//positive statistics
	Clamp(); //clamp data vector
	BinaryH(); //stochastic activation of hdden unit

	CopyStats(); //record hidden & visible positive statistics

	//negative statistics
	for (int k = 0; k < _n; k++) {
		ConstructV();
		BinaryH(); //stochastic activation of hdden unit
	}
	UpdateDW();
}

void RBM::Construct() {
	Clamp();
```

```
	ConstructH();
}

void RBM::Reconstruct() {
	Clamp();
	BinaryH(); //stochastic activation of hdden unit
	ConstructV();
}

float RBM::MSError() {
	float sum = 0;
	for (int i = 0; i < v->Size(); i++) sum += pow(v->S(i) - data[i], 2);
	return sum/v->Size();
}

void RBM::create(int _n, int _m) {
	v = new Layer(_n);
	h = new Layer(_m);
	pos_v = new Layer(_n);
	pos_h = new Layer(_m);
	w = new Weight(_n, _m);
	dw = new Weight(_n, _m);
	cd = 0;
	ResetW();
	v->Reset();
	pos_v->Reset();
	h->Reset();
	pos_h->Reset();
	linked_h = false;
	linked_v = false;
}

void RBM::destroy() {
	delete v;
	if (!linked_h) delete h;
	delete pos_v;
	if (!linked_h) delete pos_h;
	if (!linked_v) delete w;
	if (!linked_v) delete dw;
	cd = 0;
}

void RBM::save(string filename) {
	ofstream file(filename);

	//write network size
	file<<v->Size()<<"\n";
	file<<h->Size()<<"\n";

	//write weight matrix
	for (int i = 0; i < w->Size()[0]; i++) {
		for (int j = 0; j < w->Size()[1]; j++) {
			file<<w->W()[i*w->Size()[1] + j];
			if (i*w->Size()[1] + j < w->Size()[0]*w->Size()[1] - 1)
file<<",";
		}
	}
	file<<"\n";

	//write visible bias
	for (int i = 0; i < w->Size()[0]; i++) {
```

```
                file<<w->Bv()[i];
                if (i < w->Size()[0] - 1) file<<",";
        }
        file<<"\n";

        //write hidden bias
        for (int j = 0; j < w->Size()[1]; j++) {
                file<<w->Bh()[j];
                if (j < w->Size()[1] - 1) file<<",";
        }
        file<<"\n";

        file.close();
}

void RBM::load(string filename) {
        string input;
        istringstream line;
        int _n, _m;

        ifstream file(filename);

        //get visible vector size
        getline(file, input);
        sscanf(input.c_str(), "%i", &_n);

        //get hidden vector size
        getline(file, input);
        sscanf(input.c_str(), "%i", &_m);

        //build the netowrk in memory
        destroy();
        create(_n, _m);

        //get weight matrix
        getline(file, input);
        line = istringstream(input);
        for (int i = 0; i < _n; i++) for (int j = 0; j < _m; j++) if (getline(line,
input, ',')) sscanf(input.c_str(), "%f", &(w->W()[i*w->Size()[1] + j]));

        //get visible bias
        getline(file, input);
        line = istringstream(input);
        for (int i = 0; i < _n; i++) if (getline(line, input, ','))
sscanf(input.c_str(), "%f", &(w->Bv()[i]));

        //get hidden bias
        getline(file, input);
        line = istringstream(input);
        for (int j = 0; j < _m; j++) if (getline(line, input, ','))
sscanf(input.c_str(), "%f", &(w->Bh()[j]));

        file.close();
}

Layer *RBM::V() {
        return v;
}

Layer *RBM::H() {
        return h;
```

```
}

Layer *RBM::PV() {
	return pos_v;
}

Layer *RBM::PH() {
	return pos_h;
}

Weight *RBM::W() {
	return w;
}

Weight *RBM::DW() {
	return dw;
}

void RBM::Link(RBM *_next) {
	delete h;//delete existing h vector
	delete pos_h;//delete existing h pos stats vector
	h = _next->V();//link h vector
	pos_h = _next->PV();//link h pos stats vector
	_next->W()->Bv(w->Bh());//link bias vectors
	_next->DW()->Bv(dw->Bh());//link bias vectors
	linked_h = true;
	_next->LinkedV(true);
}
```

8.7.13 RBMKERNEL.CU

```
#include "util.h"
#include "RBMkernel.cuh"
#include <iostream>

__global__ void binaryH(curandState *d_s, float *_v, float *_w, float *_bh, float *_h,
                        int _size_v, int _size_h) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	float x = 0;
	for (int i = 0; i < _size_v; i++) x += _v[i]*_w[i*_size_h + idx];
	_h[idx] = 1/(1+expf(-(x + _bh[idx])));

	if (curand_uniform(&d_s[idx]) <= _h[idx]) _h[idx] = 1;
	else _h[idx] = 0;
}

__global__ void constructH(float *_v, float *_w, float *_bh, float *_h,
                        int _size_v, int _size_h) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	float x = 0;
	for (int i = 0; i < _size_v; i++) x += _v[i]*_w[i*_size_h + idx];
	_h[idx] = 1/(1+expf(-(x + _bh[idx])));
}

__global__ void constructHLinear(float *_v, float *_w, float *_bh, float *_h,
                        int _size_v, int _size_h) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	float x = 0;
	for (int i = 0; i < _size_v; i++) x += _v[i]*_w[i*_size_h + idx];
	_h[idx] = x + _bh[idx];
}

__global__ void constructV(curandState *d_s, float *_h, float *_w, float *_bv, float
*_v,
                        float _noise, int _size_v, int _size_h) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	int h_idx = idx*_size_h;
	float x = 0;
	for (int j = 0; j < _size_h; j++) x += _h[j]*_w[h_idx + j];
	_v[idx] = _bv[idx] + x + _noise*curand_normal(&d_s[idx]);
}

__global__ void constructVLinear(float *_h, float *_w, float *_bv, float *_v,
                        int _size_v, int _size_h) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	int h_idx = idx*_size_h;
	float x = 0;
	for (int j = 0; j < _size_h; j++) x += _h[j]*_w[h_idx + j];
	_v[idx] = x + _bv[idx];
}

__global__ void updateDW(float *_dw, float *_pv, float *_ph, float *_v, float *_h,
float *_dbv,
                        int _size_h) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	int h_idx = idx*_size_h;
	for (int j = 0; j < _size_h; j++) _dw[h_idx + j] += ( _pv[idx]*_ph[j] -
_v[idx]*_h[j] );
	_dbv[idx] += ( _pv[idx] - _v[idx] );
}
```

```
__global__ void updateDBH(float *_dbh, float *_ph, float *_h) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	_dbh[idx] += ( _ph[idx] - _h[idx] );
}

__global__ void updateW(float *_w, float *_dw, float *_bv, float *_dbv,
				float _mu, int _cd, int _size_h) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	int h_idx = idx*_size_h;
	for (int j = 0; j < _size_h; j++) {
		_w[h_idx + j] += _mu*_dw[h_idx + j]/_cd;// + _lambda*_w[h_idx + j];
		_dw[h_idx + j] = 0;
	}
	_bv[idx] += _mu*_dbv[idx]/_cd;
	_dbv[idx] = 0;
}

__global__ void updateBH(float *_bh, float *_dbh,
				float _mu, int _cd) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	_bh[idx] += _mu*_dbh[idx]/_cd;
	_dbh[idx] = 0;
}

__global__ void resetDW(float *_dw, float *_dbv,
				int _size_h) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	int h_idx = idx*_size_h;
	for (int j = 0; j < _size_h; j++) _dw[h_idx + j] = 0;
	_dbv[idx] = 0;
}

__global__ void resetDBH(float *_dbh) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	_dbh[idx] = 0;
}

__global__ void cudaInitRand(curandState *d_s) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	curand_init(clock() + idx, idx, 0, &d_s[idx]);
}

__global__ void copyV(float *_pv, float *_v) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	_pv[idx] = _v[idx];
}

__global__ void copyH(float *_ph, float *_h) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	_ph[idx] = _h[idx];
}

__global__ void computeError(float *_h, float *_ph, float *_data_t, float *_dbh) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	_ph[idx] = _h[idx]-_data_t[idx];
	_dbh[idx] += _ph[idx];
}

__global__ void backpropagate(float _lambda, float *_v, float *_h, float *_pv, float
*_ph, float *_w, float *_dw, float *_dbv, int _size_h) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
```

```
	int h_idx = idx*_size_h;
	_pv[idx] = 0;
	for (int j = 0; j < _size_h; j++) {
		_pv[idx] += _ph[j]*_w[h_idx + j];//sum of errors*weights
		_dw[h_idx + j] += _ph[j]*_v[idx] + _lambda*_w[h_idx + j];//error*input + 
regularization (1 is the regularization term)
	}
	_pv[idx] *= _v[idx]*(1-_v[idx]);//*derivative
	_dbv[idx] += _pv[idx];
}

__global__ void updateBackpropagation(float *_w, float *_bh, float *_dw, float *_dbh,
		float _mu, float _cd, int _size_v, int _size_h) {
	int idx = blockIdx.x * blockDim.x + threadIdx.x;
	for (int i = 0; i < _size_v; i++) {
		_w[i*_size_h + idx] += -_mu*_dw[i*_size_h + idx]/_cd;
		_dw[i*_size_h + idx] = 0;
	}
	_bh[idx] += -_mu*_dbh[idx]/_cd;
	_dbh[idx] = 0;
}

void cudaBinaryH(myStruct *cu) {
	binaryH<<<cu->d_n_blocks_h, cu->d_block_size_h>>>(cu->d_s, cu->d_v, cu->d_w, 
cu->d_bh, cu->d_h, cu->d_size_v, cu->d_size_h);
}

void cudaConstructH(myStruct *cu) {
	constructH<<<cu->d_n_blocks_h, cu->d_block_size_h>>>(cu->d_v, cu->d_w, cu-
>d_bh, cu->d_h, cu->d_size_v, cu->d_size_h);
}

void cudaConstructHLinear(myStruct *cu) {
	constructHLinear<<<cu->d_n_blocks_h, cu->d_block_size_h>>>(cu->d_v, cu->d_w, 
cu->d_bh, cu->d_h, cu->d_size_v, cu->d_size_h);
}

void cudaConstructV(myStruct *cu) {
	constructV<<<cu->d_n_blocks_v, cu->d_block_size_v>>>(cu->d_s, cu->d_h, cu->d_w, 
cu->d_bv, cu->d_v, cu->d_noise, cu->d_size_v, cu->d_size_h);
}

void cudaConstructVLinear(myStruct *cu) {
	constructVLinear<<<cu->d_n_blocks_v, cu->d_block_size_v>>>(cu->d_h, cu->d_w, 
cu->d_bv, cu->d_v, cu->d_size_v, cu->d_size_h);
}

void cudaUpdateDW(myStruct *cu) {
	updateDW<<<cu->d_n_blocks_v, cu->d_block_size_v>>>(cu->d_dw, cu->d_pv, cu-
>d_ph, cu->d_v, cu->d_h, cu->d_dbv, cu->d_size_h);
	updateDBH<<<cu->d_n_blocks_h, cu->d_block_size_h>>>(cu->d_dbh, cu->d_ph, cu-
>d_h);
	cu->d_cd++;
}

void cudaUpdateW(myStruct *cu) {
	updateW<<<cu->d_n_blocks_v, cu->d_block_size_v>>>(cu->d_w, cu->d_dw, cu->d_bv, 
cu->d_dbv, cu->d_mu, cu->d_cd, cu->d_size_h);
	updateBH<<<cu->d_n_blocks_h, cu->d_block_size_h>>>(cu->d_bh, cu->d_dbh, cu-
>d_mu, cu->d_cd);
	cu->d_cd = 0;
```

```
}

void cudaCopyStats(myStruct *cu) {
	copyV<<<cu->d_n_blocks_v, cu->d_block_size_v>>>(cu->d_pv, cu->d_v);
	copyH<<<cu->d_n_blocks_h, cu->d_block_size_h>>>(cu->d_ph, cu->d_h);
}

void cudaClamp(myStruct *cu) {
	cudaMemcpy(cu->d_v, cu->d_data, sizeof(float)*cu->d_size_v,
cudaMemcpyDeviceToDevice);
}

void cudaInit(myStruct *cu) {
	cudaInitRand<<<cu->d_n_blocks_rand, cu->d_block_size_rand>>>(cu->d_s);
	resetDW<<<cu->d_n_blocks_v, cu->d_block_size_v>>>(cu->d_dw, cu->d_dbv, cu-
>d_size_h);
	resetDBH<<<cu->d_n_blocks_h, cu->d_block_size_h>>>(cu->d_dbh);
	cu->d_cd = 0;
}

void cudaSetData(myStruct *cu, float *_data) {
	cudaMemcpy(cu->d_data, _data, sizeof(float)*cu->d_size_v,
cudaMemcpyHostToDevice);
}

void cudaSetParams(myStruct *cu, float _mu, float _noise, float _lambda) {
	cu->d_mu = _mu; cu->d_noise = _noise; cu->d_lambda = _lambda;
}

void cudaSetV(myStruct *cu, float *_v, float *_pv, float *_bv) {
	cudaMemcpy(cu->d_v, _v, sizeof(float)*cu->d_size_v, cudaMemcpyHostToDevice);
	cudaMemcpy(cu->d_pv, _pv, sizeof(float)*cu->d_size_v, cudaMemcpyHostToDevice);
	cudaMemcpy(cu->d_bv, _bv, sizeof(float)*cu->d_size_v, cudaMemcpyHostToDevice);
}

void cudaSetH(myStruct *cu, float *_h, float *_ph, float *_bh) {
	cudaMemcpy(cu->d_h, _h, sizeof(float)*cu->d_size_h, cudaMemcpyHostToDevice);
	cudaMemcpy(cu->d_ph, _ph, sizeof(float)*cu->d_size_h, cudaMemcpyHostToDevice);
	cudaMemcpy(cu->d_bh, _bh, sizeof(float)*cu->d_size_h, cudaMemcpyHostToDevice);
}

void cudaSetW(myStruct *cu, float *_w) {
	cudaMemcpy(cu->d_w, _w, sizeof(float)*cu->d_size_v*cu->d_size_h,
cudaMemcpyHostToDevice);
}

void cudaGetParams(myStruct *cu, float *_mu, float *_noise, float *_lambda) {
	*_mu = cu->d_mu; *_noise = cu->d_noise; *_lambda = cu->d_lambda;
}

void cudaGetV(myStruct *cu, float *_v, float *_pv, float *_bv) {
	cudaMemcpy(_v, cu->d_v, sizeof(float)*cu->d_size_v, cudaMemcpyDeviceToHost);
	cudaMemcpy(_pv, cu->d_pv, sizeof(float)*cu->d_size_v, cudaMemcpyDeviceToHost);
	cudaMemcpy(_bv, cu->d_bv, sizeof(float)*cu->d_size_v, cudaMemcpyDeviceToHost);
}

void cudaGetH(myStruct *cu, float *_h, float *_ph, float *_bh) {
	cudaMemcpy(_h, cu->d_h, sizeof(float)*cu->d_size_h, cudaMemcpyDeviceToHost);
	cudaMemcpy(_ph, cu->d_ph, sizeof(float)*cu->d_size_h, cudaMemcpyDeviceToHost);
	cudaMemcpy(_bh, cu->d_bh, sizeof(float)*cu->d_size_h, cudaMemcpyDeviceToHost);
}
```

```
void cudaGetW(myStruct *cu, float *_w) {
	cudaMemcpy(_w, cu->d_w, sizeof(float)*cu->d_size_v*cu->d_size_h,
cudaMemcpyDeviceToHost);
}

void cudaAlloc(myStruct *cu, int _n, int _m) {
	cudaDeviceProp device_properties;
	cudaMalloc((void**)&cu->d_v, sizeof(float)*_n);
	cudaMalloc((void**)&cu->d_h, sizeof(float)*_m);
	cudaMalloc((void**)&cu->d_pv, sizeof(float)*_n);
	cudaMalloc((void**)&cu->d_ph, sizeof(float)*_m);
	cudaMalloc((void**)&cu->d_w, sizeof(float)*_n*_m);
	cudaMalloc((void**)&cu->d_dw, sizeof(float)*_n*_m);
	cudaMalloc((void**)&cu->d_bv, sizeof(float)*_n);
	cudaMalloc((void**)&cu->d_bh, sizeof(float)*_m);
	cudaMalloc((void**)&cu->d_dbv, sizeof(float)*_n);
	cudaMalloc((void**)&cu->d_dbh, sizeof(float)*_m);
	cudaMalloc((void**)&cu->d_data, sizeof(float)*_n);
	cudaMalloc((void**)&cu->d_data_t, sizeof(float)*_m);
	if (_n > _m) cu->d_randlen = _n;
	else cu->d_randlen = _m;
	cudaMalloc((void **)&cu->d_s, sizeof(curandState)*cu->d_randlen);
	cu->d_size_v = _n;
	cu->d_size_h = _m;
	cu->linked_v = false;
	cu->linked_h = false;
	cu->d_n_cores = 1536;
	cu->d_block_size_v = 4;
	cu->d_n_blocks_v = cu->d_size_v/cu->d_block_size_v + (cu->d_size_v%cu-
>d_block_size_v == 0 ? 0:1);
	cu->d_block_size_h = 4;
	cu->d_n_blocks_h = cu->d_size_h/cu->d_block_size_h + (cu->d_size_h%cu-
>d_block_size_h == 0 ? 0:1);
	cu->d_block_size_rand = 4;
	cu->d_n_blocks_rand = cu->d_randlen/cu->d_block_size_rand + (cu->d_randlen%cu-
>d_block_size_rand == 0 ? 0:1);
}

void cudaClean(myStruct *cu) {
	cudaFree(cu->d_v);
	if (!cu->linked_h) cudaFree(cu->d_h);
	cudaFree(cu->d_pv);
	if (!cu->linked_h) cudaFree(cu->d_ph);
	cudaFree(cu->d_w);
	cudaFree(cu->d_dw);
	cudaFree(cu->d_bv);
	if (!cu->linked_v) cudaFree(cu->d_bh);
	cudaFree(cu->d_dbv);
	if (!cu->linked_v) cudaFree(cu->d_dbh);
	cudaFree(cu->d_data);
	cudaFree(cu->d_data_t);
	cudaFree(cu->d_s);
	cu->d_size_v = 0;
	cu->d_size_h = 0;
	cu->d_randlen = 0;
	cu->d_block_size_v = 0;
	cu->d_n_blocks_v = 0;
	cu->d_block_size_h = 0;
	cu->d_n_blocks_h = 0;
	cu->d_block_size_rand = 0;
```

```
	cu->d_n_blocks_rand = 0;
	cu->d_n_cores = 0;
	cu->d_cd = 0;
}

//backprop mods

void cudaSetDataT(myStruct *cu, float *_data_t) {//clamp a supervised learning vector
to the latent vector
	cudaMemcpy(cu->d_data_t, _data_t, sizeof(float)*cu->d_size_h,
cudaMemcpyHostToDevice);
}

void cudaClampH(myStruct *cu) {
	cudaMemcpy(cu->d_h, cu->d_data_t, sizeof(float)*cu->d_size_h,
cudaMemcpyDeviceToDevice );
}

void cudaComputeError(myStruct *cu) {//compute top level error
	computeError<<<cu->d_n_blocks_h, cu->d_block_size_h>>>(cu->d_h, cu->d_ph, cu-
>d_data_t, cu->d_dbh);
}

void cudaBackpropagate(myStruct *cu) {//backpropagate errors
	backpropagate<<<cu->d_n_blocks_v, cu->d_block_size_v>>>(cu->d_lambda, cu->d_v,
cu->d_h, cu->d_pv, cu->d_ph, cu->d_w, cu->d_dw, cu->d_dbv, cu->d_size_h);
	cu->d_cd++;
}

void cudaUpdateBackpropagation(myStruct *cu) {//update backpropagated delta weights
	updateBackpropagation<<<cu->d_n_blocks_h, cu->d_block_size_h>>>(cu->d_w, cu-
>d_bh, cu->d_dw, cu->d_dbh, cu->d_mu, cu->d_cd, cu->d_size_v, cu->d_size_h);
	cu->d_cd = 0;
}

void cudaLink(myStruct *cu, myStruct *cu_next) {
	cudaFree(cu->d_h);
	cudaFree(cu->d_ph);
	cudaFree(cu_next->d_bv);
	cudaFree(cu_next->d_dbv);
	cu->d_h = cu_next->d_v;
	cu->d_ph = cu_next->d_pv;
	cu->linked_h = true;
	cu_next->d_bv = cu->d_bh;
	cu_next->d_dbv = cu->d_dbh;
	cu_next->linked_v = true;
}
```

8.7.14 UTIL.CPP

```
#include "util.h"
#include <algorithm>

default_random_engine generator(time(NULL));

float realRandom(float _l, float _u) {
	uniform_real_distribution<float> distribution(_l, _u);
	return distribution(generator);
}

float gaussRandom(float _l, float _u) {
	normal_distribution<float> distribution(_l, _u);
	return distribution(generator);
}

float e(float x) {
	return exp(x);
}

void importCSV(string filename, float ***data, int *r, int *c) {
	*data = (float**)malloc(sizeof(float*));

	//open file
	ifstream file(filename);

	//line and csv from file
	string input, value;

	//initialize row/col counters
	*r = 0;

	while (getline(file, input)) {
		//read a line
		istringstream line(input);

		//reallocate line memory
		data[0] = (float**)realloc(*data, sizeof(float*)*((*r)+1));
		data[0][*r] = (float*)malloc(sizeof(float));

		//initialize cols
		*c = 0;

		//read csvs
		while (getline(line, value, ',')) {
			//reallocate column memory (extra slot to store column lengths)
			data[0][*r] = (float*)realloc((*data)[*r],
sizeof(float)*((*c)+1));

			//convert and store value at offset of 1 for extra slot
			data[0][*r][*c] = atof(value.c_str());

			//incrament column counter
			(*c)++;
		}

		//incrament row counter
		(*r)++;
	}

	file.close();
```

```
}

void importCSV_2(string filename, float ***data, int rows, int cols) {
	*data = (float**)malloc(sizeof(float*));

	//open file
	ifstream file(filename);

	//line and csv from file
	string input, value;

	//initialize row/col counters
	int r = 0, c = 0;

	//allocate memory a priory
	data[0] = (float**)realloc(*data, sizeof(float*)*rows);

	while (getline(file, input)) {
		//read a line
		istringstream line(input);

		//reallocate line memory
		data[0][r] = (float*)malloc(sizeof(float)*cols);

		//initialize cols
		c = 0;

		//read csvs
		while (getline(line, value, ',')) {
			//convert and store value at offset of 1 for extra slot
			data[0][r][c++] = atof(value.c_str());
		}

		//incrament row counter
		cout<<100.0f/rows*r++<<"%\n";

		if (r > rows) break;
	}

	file.close();
}

void randomOrder(int *order, int size) {
	random_shuffle(&order[0], &order[size + 1]);
}
```

8.7.15 WEIGHT.CPP

```
#include "Weight.h"

Weight::Weight(int _n, int _m) {
	w = (float*)malloc(sizeof(float)*_n*_m);
	bv = (float*)malloc(sizeof(float)*_n);
	bh = (float*)malloc(sizeof(float)*_m);

	size[0] = _n;
	size[1] = _m;
	linkedbv = false;
	linkedbh = true;
}

Weight::~Weight() {
	free(w);
	if (!linkedbv) free(bv);
	if (!linkedbh) free(bh);

	size[0] = 0;
	size[1] = 0;
}

void Weight::Reset() {
	for (int i = 0; i < size[0]; i++)
		for (int j = 0; j < size[1]; j++) w[i*size[1] + j] = 0;
	for (int i = 0; i < size[0]; i++) bv[i] = 0;
	for (int j = 0; j < size[1]; j++) bh[j] = 0;
}

void Weight::Reset(float _l, float _u) {
	for (int i = 0; i < size[0]; i++)
		for (int j = 0; j < size[1]; j++) w[i*size[1] + j] = realRandom(_l, _u);
	for (int i = 0; i < size[0]; i++) bv[i] = realRandom(_l, _u);
	for (int j = 0; j < size[1]; j++) bh[j] = 0;//realRandom(_l, _u);
}

void Weight::Resize(int _n, int _m) {
	w = (float*)realloc(w, sizeof(float)*_n*_m);
	bv = (float*)realloc(bv, sizeof(float)*_n);
	bh = (float*)realloc(bh, sizeof(float)*_m);

	size[0] = _n;
	size[1] = _m;
}

void Weight::W(float _wij, int _i, int _j) {
	w[_i*size[1] + _j] = _wij;
}

void Weight::Bv(float _bi, int _i) {
	bv[_i] = _bi;
}

void Weight::Bh(float _bj, int _j) {
	bh[_j] = _bj;
}

float Weight::W(int _i, int _j) {
	return w[_i*size[1] + _j];
}
```

```
float Weight::Bv(int _i) {
	return bv[_i];
}

float Weight::Bh(int _j) {
	return bh[_j];
}

float *Weight::W() {
	return w;
}

float *Weight::Bv() {
	return bv;
}

float *Weight::Bh() {
	return bh;
}

int *Weight::Size() {
	return size;
}
```

8.7.16 MNIST HANDWRITTEN DIGITS TEST

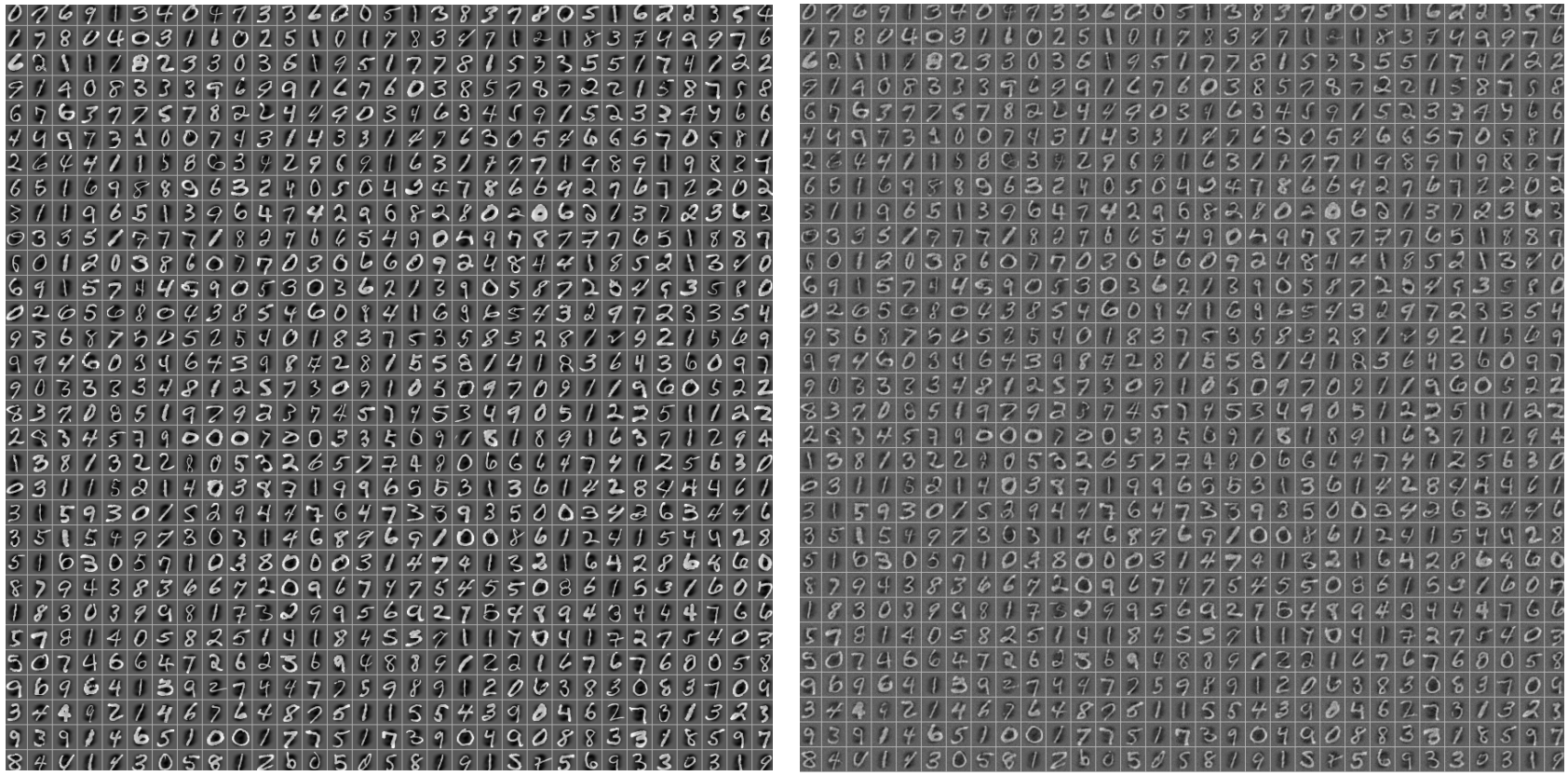


Figure 78. MNIST digit reconstructions from the trained original implementation of a GBRBM (see Appendix 8.7). Left: a matrix of 961 (randomly selected) of the original images of handwritten digits taken from [115]. Right: the corresponding reconstructions of the digits on the left after training a GBRBM with 961 hidden variable for 1000 epochs of $\boldsymbol{CD_1}$. The overall mean square error was $\approx \mathbf{0.1479}$ (the error can vary because the reconstructions are based on the activation of stochastic hidden units, i.e. units do not always 'fire').

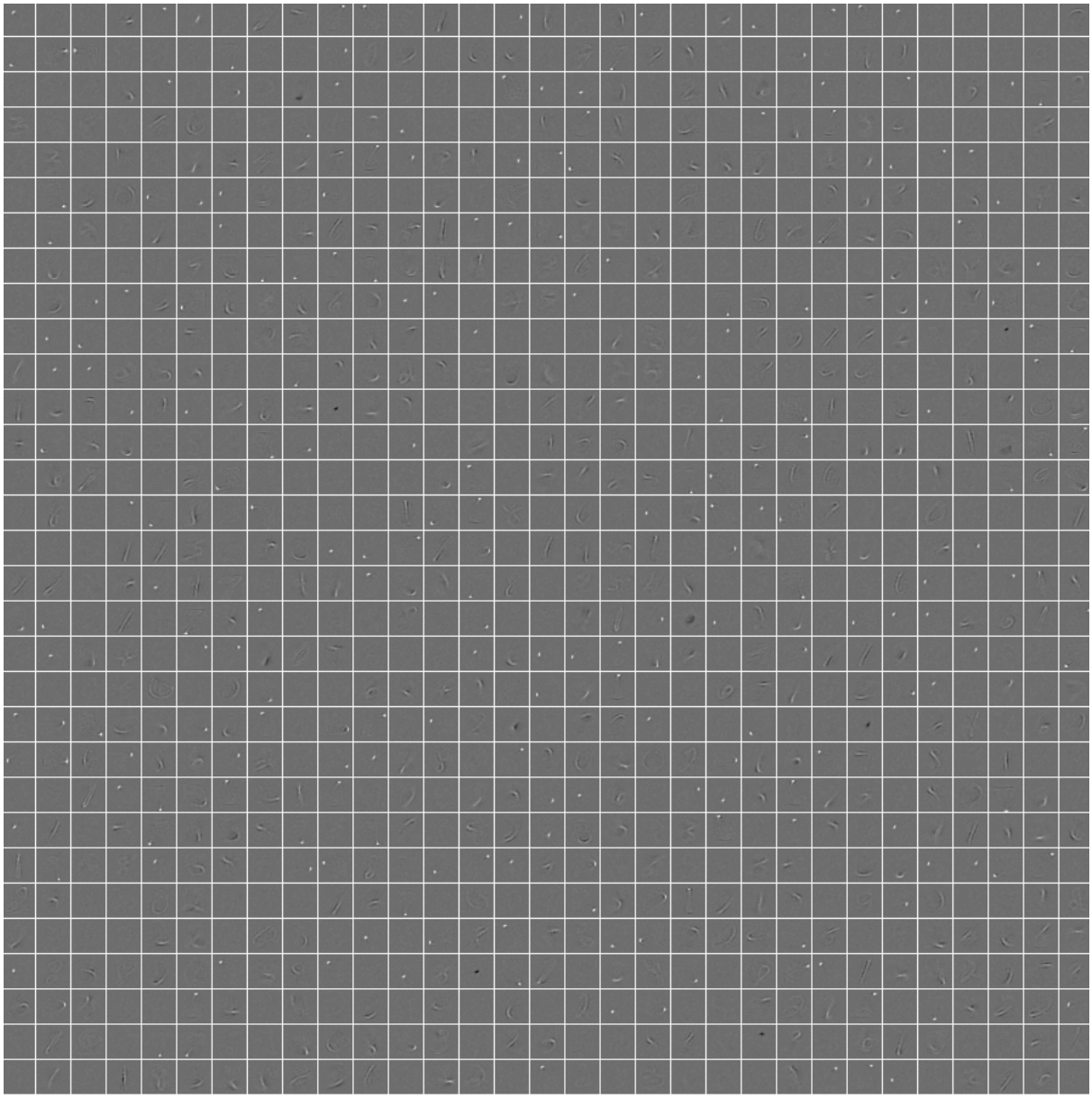


Figure 79. $\mathbf{31 \times 31}$ matrix of the receptive fields of all 961 hidden variables of the trained GBRBM. Figure 78 shows that linear combinations of these weights are capable of accurately reconstructing any digit in the training set.

8.8 List of Abandoned Ultrasound Image Segmentation Techniques

This section of the appendix is a list of just some of the many lines of investigation into the complex issue of ultrasound image segmentation of dynamic skeletal muscle. These were abandoned prior to obtaining results due to development of other methods and poor outlook. A figure and a brief explanation accompany each technique in the list.

8.8.1 Gaussian Mixture Model for Automatic Classification of Intersecting Oriented Edges – Segmentation of Human Posterior Neck Muscles via Ultrasonography

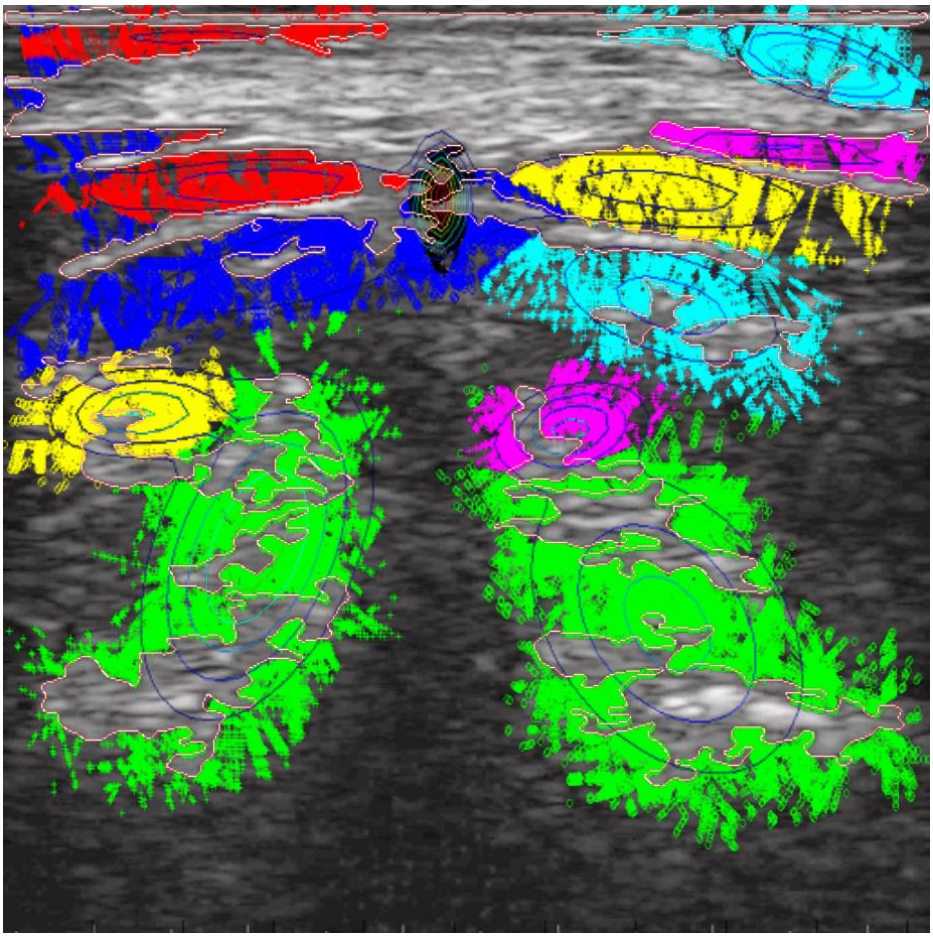


Figure 80. A linear threshold ($\boldsymbol{pixel > 2\sigma = 1}$) extracts the bright regions within the image (white highlight). Then a density map is constructed from intersecting lines perpendicular to the edges (white lines around the threshold region) of the threshold regions. Then, a Gaussian Mixture Model with 12 kernels automatically classifies the density map (different classes are colour coded in the image). This technique was very slow ($\approx$ **15** minutes to segment a single frame) and relied heavily on the presence of edges which define the boundary of muscle segments.

8.8.2 *Linear Hough Transform for Automatic Detection of Aponeurotic Sheaths – Segmentation of Human Calf Muscles via Ultrasonography*

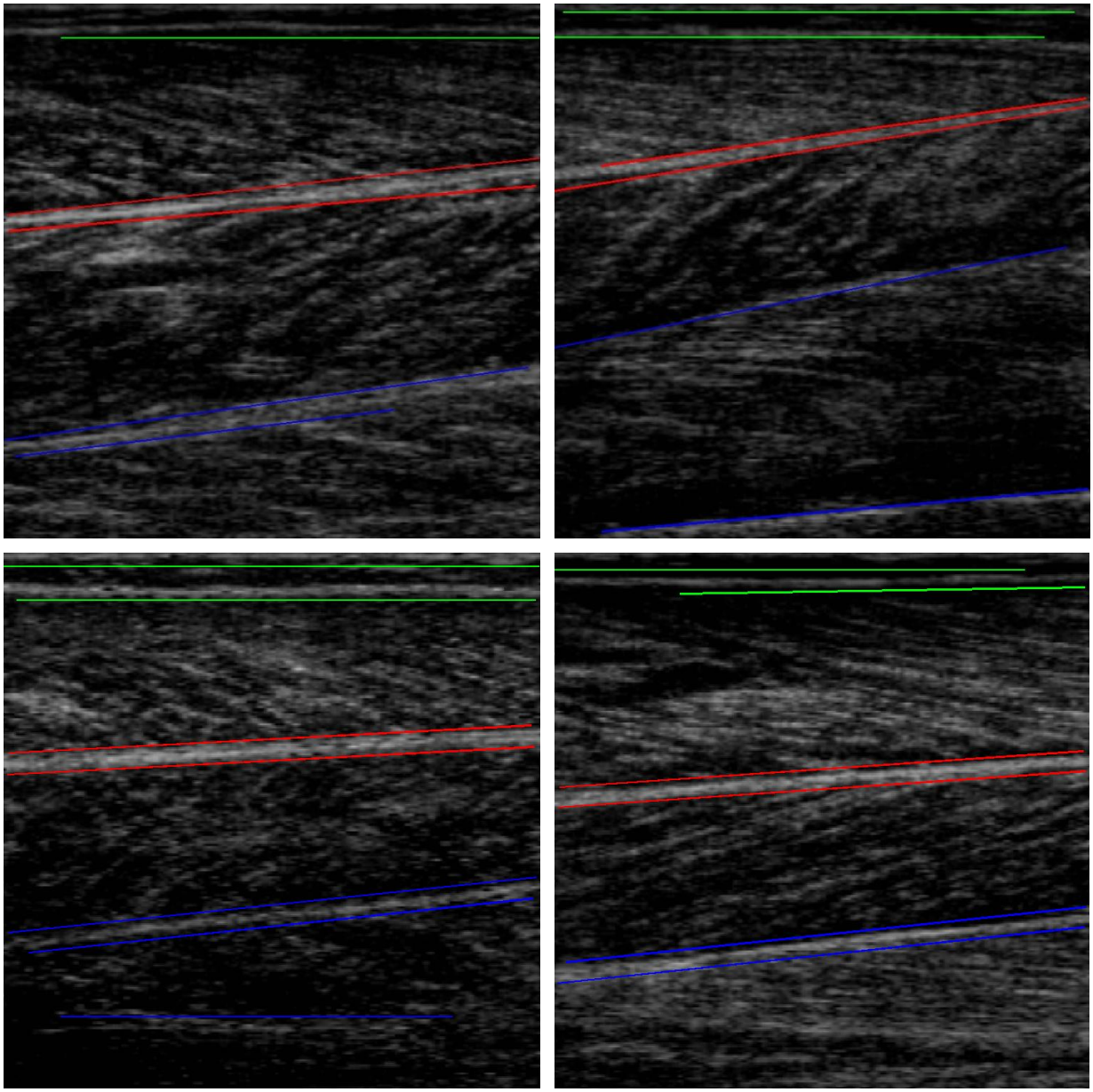


Figure 81. The Linear Hough Transform was used to automatically find lines with large canny edge gradients ($> 2\sigma$). The coloured lines represent an automatic depth classification using the k-means clustering algorithm. This technique was abandoned doe to other developments, poor edge tracing, too many erroneous lines, and too many arbitrary parameters to improve results.

8.8.3 *Histograms of Orientated Gradients for Muscle Segmentation via Support Vector Machines – General Ultrasound Segmentation*

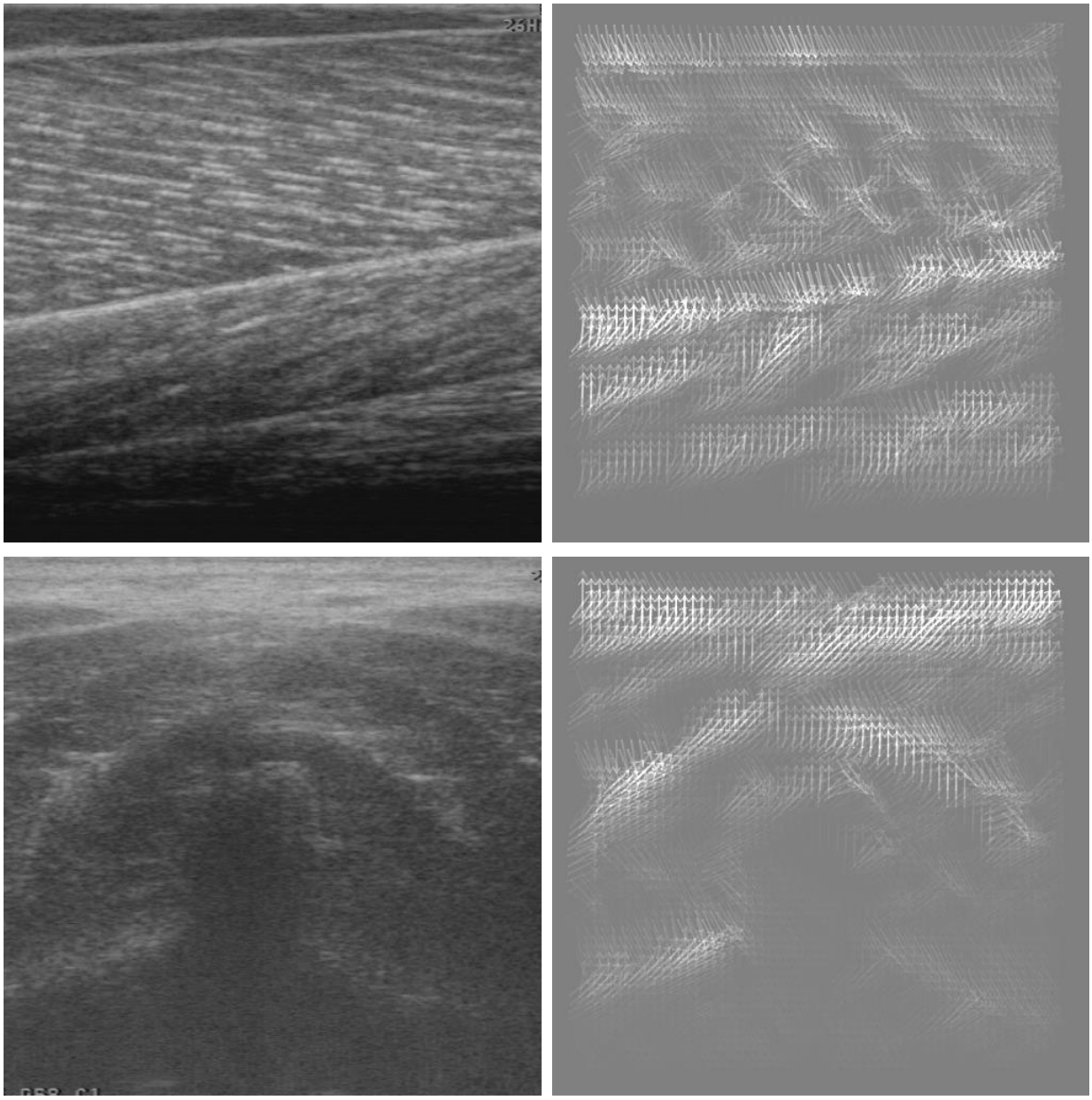


Figure 82. A custom Histogram of Orientated Gradients (HOG) implementation computes a $\mathbf{64 \times 64}$ orientation grid from 90 bin histograms of $\mathbf{32 \times 32}$ image patches. Top: human calf muscles. Bottom: human posterior neck muscles. The image on the left shows the original ultrasound. The image on the right shows the orientation vector field, where the arrow indicates the direction of local gradients, and the intensity of the arrow indicates the magnitude of the local gradient. This project was abandoned due to development of other techniques, and due to difficulty in obtaining enough labelled data, with which to train a multi-class Support Vector Machine (SVM) to classify regions of the image.

8.9 PUBLICATIONS

This section presents full prints of first author publications. The first publication was published in Medical Image Understanding and Analysis (MIUA) which is a peer reviewed publication for a conference in Birmingham, UK. The following is two publications (not peer reviewed) which were submitted to the International Society of Electrophysiology and Kinesiology (ISEK) in Rome, Italy.

8.9.1 AUTOMATED MEASUREMENT OF HUMAN SKELETAL CALF MUSCLE CONTRACTION VIA B-MODE ULTRASOUND IMAGING

This publication present initial work on classifying skeletal muscle states with a Support Vector Machine using a hand engineered descriptor of the KLT motion field generated during isotonic contraction and passive muscle movement.

Authors: Cunningham, R.J [1, 2], Harding, P.J [2], Loram, I.D [2], Costen, N.P [1]

[1] School of Computing, Maths & Digital Tech, MMU, Manchester, England

[2] School of Healthcare Science, MMU, Manchester, England

E-mail: ryan.j.cunningham@stu.mmu.ac.uk

Conference: MIUA

Year: 2013

Location: Birmingham

Abstract

Traditionally surface electromyography (EMG) has been used to non-invasively identify the activation of superficial skeletal muscle. We propose that the automated analysis of ultrasound images can provide an alternative technique by which active and passive muscle movement may be classified. We present a method by which the change in shape can be extracted from pairs of sequential ultrasound images and used to classify whether that shape change was caused by active or passive influences. Results are presented which show that our method can correctly classify active or passive movements with greater than 95% accuracy and is less affected by change in contraction strength than EMG.

Introduction

There are many methods available that provide ways by which muscle activity can be analysed - most notably EMG (surface) or IEMG (intramuscular). Electrodes placed on the skin (EMG) can measure activation of superficial muscle but there is an inherent level of noise which can make it difficult to identify activation at small forces/velocities; this is particularly true if a small activation co-occurs with a larger one. Filtering an EMG metric can help in the extraction of meaningful physiological information about muscle activity, but often the signal to noise ratio is too low to reveal anything at low force exertion [24]. Thin wire electrodes inserted into the muscle through the skin (IEMG) can measure contraction in both superficial and deep muscle, and is generally considered a more accurate representation of activity than EMG. IEMG is more susceptible to external electrical noise, measures a small volume within the muscle, and is invasive (IEMG requires a sterile environment; there is also an inherent risk factor when measuring motor neurons near the spine and neck). Ultrasound has been considered by many as a possible alternative method of measuring activity in skeletal muscle [1, 24, 60, 116].ultrasound is entirely non-invasive, risk free, cost effective, and has been shown to be more sensitive to architectural changes (such as pennation angle, cross-sectional area) resulting from contraction than EMG at low force exertion [116, 24].ultrasound can also analyse a much greater cross-sectional area of muscle tissue.

Recent work [1] has demonstrated that with the application of computer vision techniques (such as ASM [17], and KLT feature tracking [63, 62]), useful information about muscle architectural changes can be automatically derived from US. We propose that further to this, measures derived from architectural changes of muscle can be used to correctly identify whether the shape change was caused by external forces (i.e. passively) or by voluntary contraction (i.e. actively).

Methods

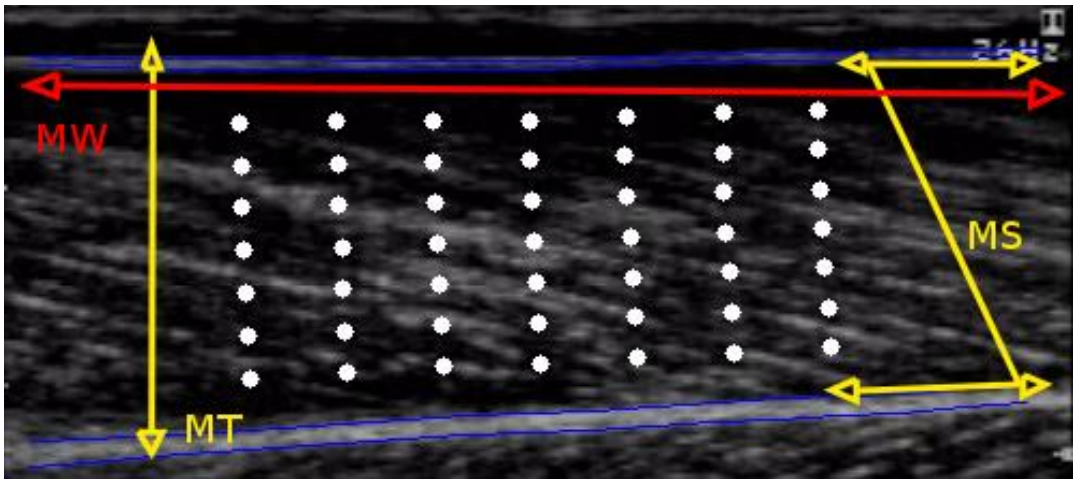


Figure 1. ASM segmentation (blue contours) of medial gastrocnemius (MG) and matrix of point features (white dots) over MG. Changes in fascicle shearing, muscle thickness and muscle width (MS, MT, MW respectively) are calculated as: the mean difference in $\boldsymbol{x}$ displacement of the top and bottom row of point features, the mean difference in $\boldsymbol{y}$ displacement of the top and bottom rows of point features, and the mean difference in $\boldsymbol{x}$ displacement in the leftmost and rightmost columns.

An ASM was used to segment $25Hz$ ultrasound video sequences of MG, and then KLT features were selected within the intramuscular area on a 7×7 grid arrangement and tracked into the next frame. The ASM segmentation was then updated and features re-selected to prevent tracking drift beyond the image boundary. A square KLT feature window size of $55px$ was used across all trials (Figure 1 illustrates the tracking process). MG contours in 450 images (describing range of motion for each participant) were marked up and used to train the ASM. The principal component model was constructed at runtime from the mark-up database.

A non-standard initialisation step was used with this model: At the start of each video sequence, the most accurate mean shape was chosen from all known participant shape means. This is done by iterating over each participant mean and calculating the Mahalanobis distance from each contour point to the statistical models of its edge gradient, calculated during training. At this point the best fit from the participant i.e. that with the lowest total Mahalanobis distance, was used as the "mean shape" for the duration of that video sequence. Other than this change, the ASM used was as defined in [17]. This method of segmentation and tracking is described in detail in previous work [1].

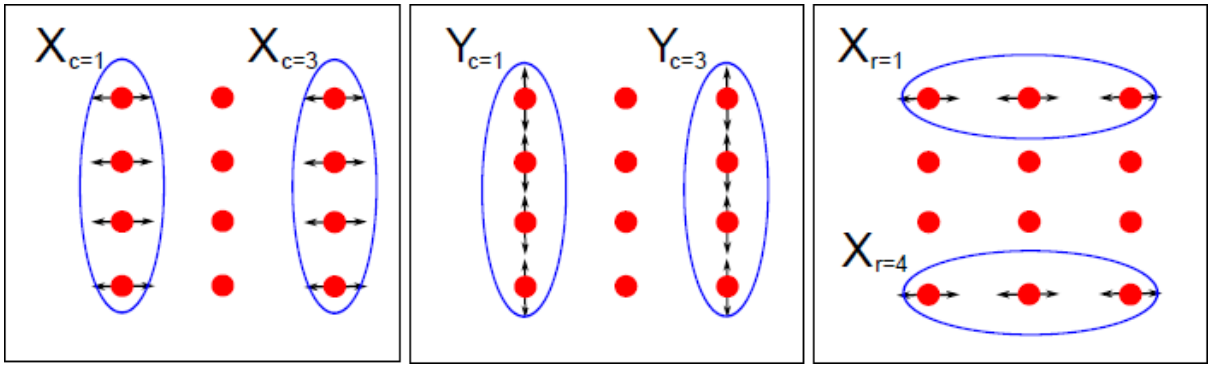


Figure 2. Left: A 3-item vector of x-direction displacements per column (shearing). Centre: A 3-item vector of y-direction displacements per column (thickening). Right: A 4-item vector of x-direction displacements per row (widening).

Muscle Shape Change Modelling

Previous measures used to assess muscle shape change have included muscle thickness and muscle shearing [117, 116], in addition to these measures muscle widening is also considered in our analysis (see Figure 2). Rather than distilling the muscle shape change between two frames into 2 or 3 scalar metrics, the use of grid sampling over the entire cross-sectional area of the muscle provides additional information about the active or passive nature of that shape change. All of the 49 point features were considered on a frame by frame basis, in the analysis. Figure 2 shows examples of how information is extracted that can be used to approximate width, thickness and shearing from the point features. To use the entirety of the data would result in a 98-dimensional feature vector, where the sample size is 6,672. For reduction of dimensionality the mean shearing per column, S_c, mean thickening per column, T_c, and mean widening per row, W_r were computed. Since the magnitude of active movement over passive movement is $\approx$ 1 order of magnitude greater [116], in order to avoid solving an ill-posed problem, velocity was removed from the data by normalising on a per frame basis, giving a relative velocity (shape of motion). Shearing, thickening and widening measures (S, T, W) are created as,

$$S = \bar{S_c} \forall c,$$

$$T = \bar{T_c} \forall c,$$

$$W = \overline{W_r} \forall r.$$

Where S is the mean shearing over all feature columns, T is the mean thickening over all feature columns, and W is the mean widening over all feature rows. This leaves 3 vectors containing 7 mean displacement values which are used as a 21-dimensional descriptor of shape change between two frames.

Activity Discrimination

A Support Vector Machine (SVM) [41] was used for the classification of active and passive shape changes. The training data was labelled using force exerted in active trails per participant to define when the muscle is being moved actively, and corresponding frames of each following passive trial to define when the muscle was being moved passively. The segmentation was defined as,

$$s(v,t) = \left((f_{vt} > \sigma_v) \vee \left(\frac{df_{kt}}{dt} > \sigma_k\right)\right) \neg \left(\frac{df_{kt}}{dt} < -\sigma_k\right),$$

where, v is the trial, t is the frame, k is a constant ($k = 1$), σ_v is the standard deviation of force in trial v, and f_{vt} is the force in trial v at time t.

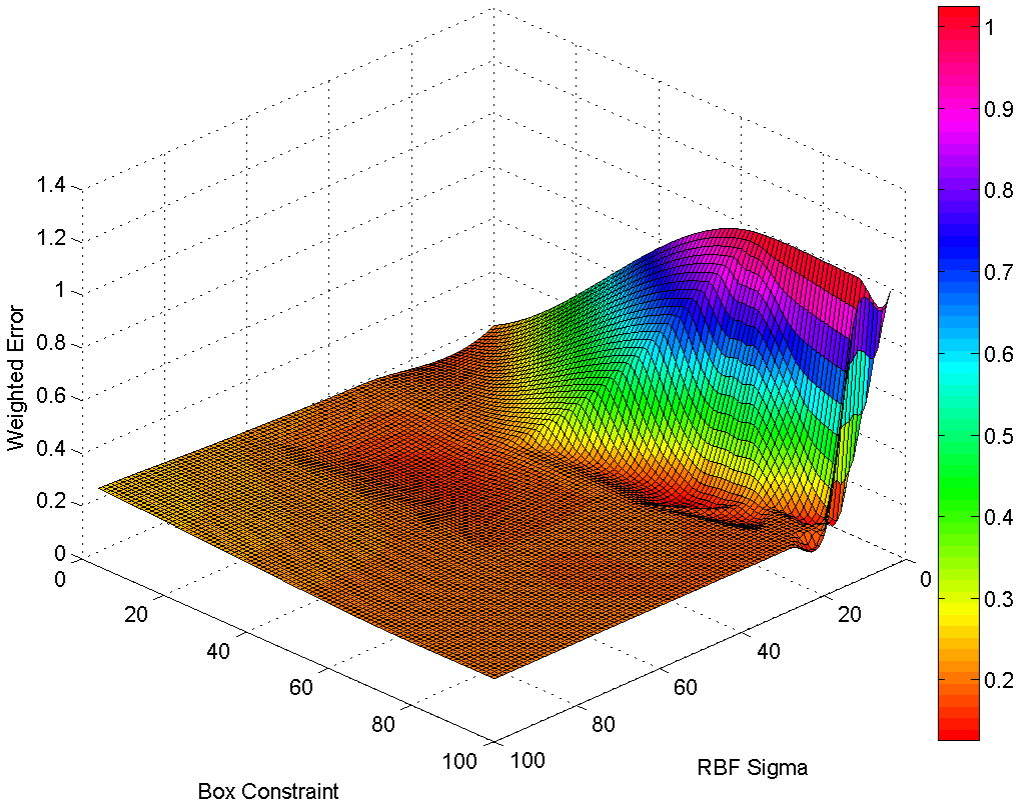


Figure 3. SVM optimisation error surface. The cost function is plotted against the $\boldsymbol{\sigma}$ parameter of the Radial Basis Function (RBF), and the SVM regularisation parameter (box constraint), C. The error is almost uniform beyond small values of RBF$\boldsymbol{\sigma}$, while the optimum error lies within a range of $\mathbf{20} > \boldsymbol{\sigma} > \mathbf{10}$.

Leave one participant out cross validation was used to train the SVM, where a single participant's data was used to validate and all remaining participants' data was used to train. Both the training and validation data were created from the shear, thickness and width measures according to,

$$D = \{|S_{vt}, T_{vt}, W_{vt}|\} \forall vt.$$

The SVM used a Gaussian Radial Basis Function (RBF) kernel and was trained with the binary training signal, $s(v,t)$ of all participants, to differentiate between active and passive shape changes. The SVM was optimised during training by varying the margin regularisation parameter, C, and the RBFσ parameter. The optimisation function – where minimal distance between two errors reduces the bias on the training samples – is defined as,

$$E(D^{-},C,\sigma) = \frac{S_{\sigma C}(D^{-})_{error} + S_{\sigma C}(D)_{error}}{2} + |S_{\sigma C}(D^{-})_{error} - S_{\sigma C}(D)_{error}|$$

$$Minimise{:}\, E(D^{-},C,\sigma)\ \forall C\ \in\ \mathbb{R} : 0 \rightarrow e^{4.5}, \forall \sigma\ \in\ \mathbb{R} : 0 \rightarrow e^{4.5}, \forall D^{-} \in D,$$

Where D^{-} is the validation set $D^{-} \in D$, D is the training set $D \in \{x_1, y_1, \dots, x_n, y_n\} - D^{-}$, and $S_{\sigma C}(D)_{error}$ is the error of the SVM, $S_{\sigma C}$, on the set D.

Data Collection

Simultaneous ultrasound and EMG measurements were made over 3 sets of 2 trials (20 seconds duration) with 12 participants. Participants were positioned upright with their backs against a stiff backboard, standing on programmable foot pedals. For the first of each set (trial *A*) participants rotated the foot pedals in a plantar flexion (increasing the angle between shank and foot) motion (with a diminished level of force from trials 1–3), while maintaining their body posture; the pedals automatically returned to a neutral angle if no force was exerted. The force exerted (Nm) and foot pedal angle (degrees) were recorded at 1000Hz. The force was used to actuate the motors on the foot pedals which caused an ankle plantar flexion rotation. For the second of each set (trial *B*) participants allowed their ankle angle to rotate freely with the pedals while maintaining posture; the recorded angle from each participant's trial *A* was used to drive the motors in trial *B*, resulting in passive muscle length changes occurring within the corresponding time frame to the active muscle length changes in trial *A*. Data were recorded via EMG over the MG muscle at $1000Hz$, and an US probe secured to MG. All US video sequences were collected at a static temporal resolution of $25Hz$.

Data Processing

EMG was filtered with a sixth order high pass Butterworth filter, followed by rectification and a second order, low pass filter. Cubic interpolation was used to re-sample the recorded ultrasound measures from $25Hz - 1000Hz$. The data were realigned temporally via cross-correlation of ultrasound and external measures (angle, EMG), with a maximum possible realignment of 15 frames (assuming that ultrasound measures will always lag external measures). On average the ultrasound lags the external measures by ≈ 13 frames ($0.52s$).

Results

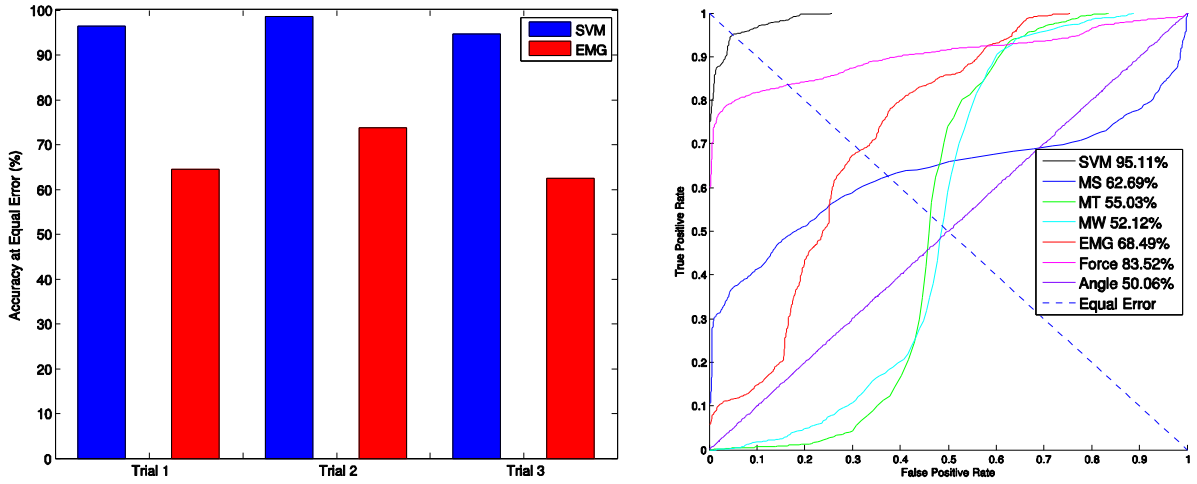


Figure 4. Left: SVM/EMG comparison over all participants for trials 3A-3B (lowest active and passive trials, respectively), showing accuracy at equal error according to ROC. Right: ROC curves for all participants over trials 3A-3B (lowest active and passive trials, respectively), showing accuracy at equal error (dashed black line).

Receiver Operating Characteristics (ROC:Figure 4) were computed on the optimised, cross-validated SVM and all other measures (external: EMG, angle, force, and ultrasound: MS, MT, MW). After filtering the SVM time series output with a low pass Butterworth filter, the SVM shows a highly reliable classification accuracy of 95.11% at equal error. Independent ultrasound metrics (MS, MT, MW) report reasonable accuracy, with fascicle shearing operating at 62.69% accuracy at equal error. EMG does prove reliable in the majority of cases however, for some participants the noise threshold was enough to being the accuracy down to 68.49% at equal error. The significance of these results is in the fact that they represent the classification accuracy of low force muscle activations (see Figure 4). The average peak force exertion over all participants for trial 3*A* (lowest active trial) was $21.47Nm$, and the average peak foot pedal angle was 4.37^o. The average peak force exertion over all participants for trial 1*A* (highest active trial) was $64.35Nm$, and the average peak foot pedal angle was 8.51^o.

Conclusions

It has long been established that there is a non-linear relationship of sensitivity to physiological change under force between ultrasound and EMG, with ultrasound being more sensitive to change at smaller activations in the isometric case [24]. It has previously been established that these changes can be measured automatically [1, 116]. We have shown that it is possible to correctly classify an active or passive muscle shape change from automated analysis of temporal skeletal muscle ultrasound, even when the joint rotation angle is identical. The technique presented here has also been shown to offer a more accurate classification between active and passive muscle shape change than surface EMG on this data set. This method establishes a starting point in the construction of a comprehensive model of human muscle function, which combines muscle length change, activation and joint angle. Future work will explore the wider application of this technique to other, less accessible, muscles such as deep muscles near the spinal cord and cervical muscles in the neck.

8.9.2 SKELETAL MUSCLE STATES PREDICTED FROM SEQUENTIAL B-MODE ULTRASOUND IMAGES

This is an abstract which contains some of the preliminary work on the calf muscle state modelling via b-mode ultrasonography, using a Restricted Boltzmann Machine (RBM). Specifically, this abstract was about the predictive power of the RBM model, regarding the joint angle and activity as measured by EMG.

Authors: Cunningham, R.J [1, 2], Harding, P.J [2], Costen, N.P [1], Loram, I.D [2]

[1] School of Computing, Maths & Digital Tech, MMU, Manchester, England

[2] School of Healthcare Science, MMU, Manchester, England

E-mail: ryan.j.cunningham@stu.mmu.ac.uk

Conference: ISEK

Year: 2014

Location: Rome

Appendix

AIM: Ultrasonography (US) is capable of non-invasively imaging large cross-sectional areas of muscle tissue, revealing fascicle curvature/orientation, and strong indicators of activation, segment length change and tendon stiffness. Standard analysis techniques track fascicle motion and measure differentials between muscle boundaries, but these methods fail over arbitrarily long sequences (~60") due to unavoidable tracking error. Presented here is a robust graphical model of the human Gastrocnemius Medialis (GM) which has the capacity to estimate ankle joint angle and muscle activity, directly from two sequential frames of ultrasound. A Restricted Boltzmann Machine (RBM) and a Multilayer Neural Network (MLNN) was used to model pairs of images from US recordings of muscle function from 17 participants, and was validated on a single held-out participant. Initial results show significant correlations when predicting ankle angle and activation (EMG); r=0.724 and r=0.772, respectively.

METHODS: 18 Participants were asked to stand on programmable foot pedals while strapped upright to a flat backboard. They then performed a series of ankle rotations designed to exploit skeletal muscle properties under 3 conditions: isometric contraction, passive ankle rotation, and combined contraction and ankle rotation. Ankle angle (1000*Hz*), EMG (1000*Hz*), and US (25*Hz*) of the calf were simultaneously recorded. For isometric and combined tasks, rectified filtered EMG feedback was used to guide the participants in matching their muscle activity to a pre-designed tracking target presented via oscilloscope. For passive and combined tasks, a pre-designed signal was used to drive the actuators on the foot pedals, which created a rotation about the participants' ankle joint (±10º).

An Active Shape Model was used to automatically extract images of GM from each frame of US, and a large database of image pairs (frames t and t+1) was created. Each image was down-sampled to an 80x80 matrix. Then a RBM was used to create a graphical model of the muscle from that database, leaving out a random participant's data for validation. The RBM yields a 1536 point feature vector. That feature vector was then used as an input to a MLNN, which was trained to predict ankle angle and EMG, using the Error Backpropagation algorithm.

RESULTS: Initial results from the model empirically demonstrate the capacity to estimate the current joint angle (r>0.71, p<0.001), and the current activity (filtered EMG: r>0.75, p<0.001) in all participants. The model was validated on a single, held-out participant, with r values of 0.772 and 0.724 for estimating joint angle and activity respectively, from a sequence containing combined activity and ankle joint changes.

CONCLUSION: This method is a significant departure from standard methods, which rely on tracking approximations of fascicle motion in the 2D US plane. By modeling fascicle motion directly from the data, our method has eliminated the possibility of tracking drift, and gives a good global estimate of activity and muscle length.

8.9.3 A B-MODE ULTRASOUND MUSCLE MODEL FOR HYPOTHESIS TESTING

This is an abstract which contains some of the preliminary work on the calf muscle state modelling via b-mode ultrasonography, using a Restricted Boltzmann Machine (RBM). Specifically, this abstract was about using the RBM as a generative model of images of muscle undergoing specified changes in muscle states.

Authors: Cunningham, R.J [1, 2], Harding, P.J [2], Costen, N.P [1], Loram, I.D [2]

[1] School of Computing, Maths & Digital Tech, MMU, Manchester, England

[2] School of Healthcare Science, MMU, Manchester, England

E-mail: ryan.j.cunningham@stu.mmu.ac.uk

Conference: ISEK

Year: 2014

Location: Rome

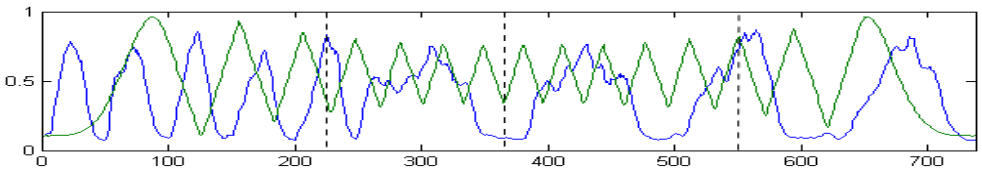
AIM: Ultrasonography (US) is capable of non-invasively imaging large cross-sectional areas of muscle tissue, revealing fascicle curvature, and strong indicators of activation. While methods exist which aim to extract these properties automatically, little is known regarding how to interpret them physiologically. A Restricted Boltzmann Machine (RBM) is used to build a US texture model of human Gastrocnemius Medialis (GM). Via Principal Components Analysis, the model is capable of generating probabilistic images of GM under specified isometric and/or passive strain, which can be used to test physiological hypotheses.

METHODS: 18 Participants were asked to stand on programmable foot pedals while strapped upright to a flat backboard. They then performed a series of ankle rotations designed to exploit skeletal muscle properties under 3 conditions: isometric contraction, passive ankle rotation, and combined contraction and ankle rotation, while US of the GM was recorded.

An Active Shape Model was used to automatically extract images of GM from each frame of US, and a large database of images was created. Each image was down-sampled to an 80x80 matrix. Then a RBM was used to create a graphical model of the muscle from that database, leaving out a random participant's data for validation. The RBM yields a 1536 point feature vector, from which the RBM can approximately reconstruct the original image.

Given the feature vectors of images from two sequences (isometric and passive) in the model data-set we can capture the covariance using PCA. Then we can take a static frame from the validation data-set, produce a feature vector, and add a linear combination of the eigenvectors to the feature vector. Given the feature vector, the RBM can construct probabilistic examples of images depicting how the muscle would likely look under some passive ankle rotation, and/or isometric contraction. This can be achieved by constructing isometric/passive signals to be used as coefficients to each respective eigenvector.

RESULTS: Figure 1 shows generated images (bottom) using the validation participant's data. These images were generated using the signals (top: isometric coefficient is blue, passive coefficient is red).



These coefficients multiply the eigenvectors of their respective classes, which are then added to the feature vector generated from the image at $t = 0$.

$t = 0$ $t = 224$ $t = 365$ $t = 550$ $t = 737$

Figure 1: Generated US images from the model.

CONCLUSION: This paper sets groundwork for testing new hypotheses about muscle tissue behavior under strain. By taking the accumulated difference between a generated image for a desired activation, and relaying this information back to a patient, and since US is known to be more sensitive to activation than EMG, we also have a potential method of biofeedback for patients - for example - with spinal cord injury.