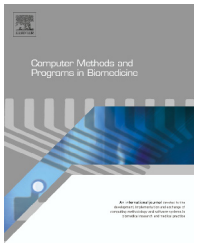
COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE 128 (2016) 111–118

journal homepage: www.intl.elsevierhealth.com/journals/cmpb



UltraTrack: Software for semi-automated tracking of muscle fascicles in sequences of B-mode ultrasound images

*Dominic James Farris**, *Glen A. Lichtwark*

School of Human Movement & Nutrition Sciences, Level 5, Building 26B, Blair Drive, The University of Queensland, Brisbane, QLD 4072, Australia

ARTICLE INFO

Article history:
Received 9 November 2015
Received in revised form
14 January 2016
Accepted 24 February 2016

Keywords:
Fascicle
Tracking
Ultrasound
Automated
Affine optic flow
Matlab

ABSTRACT

Background: Dynamic measurements of human muscle fascicle length from sequences of B-mode ultrasound images have become increasingly prevalent in biomedical research. Manual digitisation of these images is time consuming and algorithms for automating the process have been developed. Here we present a freely available software implementation of a previously validated algorithm for semi-automated tracking of muscle fascicle length in dynamic ultrasound image recordings, "UltraTrack".

Methods: UltraTrack implements an affine extension to an optic flow algorithm to track movement of the muscle fascicle end-points throughout dynamically recorded sequences of images. The underlying algorithm has been previously described and its reliability tested, but here we present the software implementation with features for: tracking multiple fascicles in multiple muscles simultaneously; correcting temporal drift in measurements; manually adjusting tracking results; saving and re-loading of tracking results and loading a range of file formats.

Results: Two example runs of the software are presented detailing the tracking of fascicles from several lower limb muscles during a squatting and walking activity.

Conclusion: We have presented a software implementation of a validated fascicle-tracking algorithm and made the source code and standalone versions freely available for download.

1. Introduction

The in vivo measurement of human skeletal muscle function has long been of interest to scientists and engineers from a variety of backgrounds including: physiology, anatomy, evolutionary biology, ergonomics, biomedical engineering and human movement science. Of fundamental importance in this area is the measurement of muscle architecture and how it dynamically changes with muscle contraction. A key feature of muscle architecture is muscle contractile element length and many important research questions can be answered with knowledge of how contractile elements change length during muscle contraction. The basic contractile element of muscle is the sarcomere and many sarcomeres in series make up muscle fibres, which in turn are bundled into fascicles. Sarcomeres are microscopic in size and so it is common to measure fibre or fascicle length changes as a proxy.

* *Corresponding author*. Tel.: +61 07 3365 6097.
E-mail addresses: d.farris@uq.edu.au (D.J. Farris), g.lichtwark@uq.edu.au (G.A. Lichtwark).
http://dx.doi.org/10.1016/j.cmpb.2016.02.016

In the last twenty years, B-mode ultrasound has risen in popularity as a means of imaging skeletal muscle in static and dynamic conditions. Cross-sectional ultrasound images of skeletal muscle allow identification of muscle fascicles [1]. By sampling images at a suitable rate, the length changes of muscle fascicles during dynamic contractions have been characterised through digitisation of image sequences [2,3]. Initially, the digitisation process was manual, making it extremely time consuming and somewhat subjective. More recently, image processing algorithms to partially or fully automate the digitisation process have been developed [4–6]. These employ a variety of methods including Kanade–Lucas–Tomasi feature tracking and affine transformation extensions to optic flow. The validity and reliability of the affine optic flow algorithm of Cronin et al. [4] was published some time ago [4,7]. The algorithm and the software implementation have since been substantially expanded and in this paper we present the refined algorithm and freely available software package, UltraTrack version 4.1, the source code for which is written in Matlab (The MathWorks Inc., USA). A new 'key-frame correction' algorithm has been added to help remove temporal drift (accumulation of error with time) in the tracking that had previously limited the length of image sequences that could be tracked. A manual correction is also added as an alternative for making adjustments to tracking results. New features of the software include: compatibility with more image file formats, faster implementation of the affine flow algorithm, tracking of multiple fascicles and regions of interest, image auto-cropping function, saving and re-loading of tracking data and a choice of moving or fixed region of interest for the affine flow calculations.

2. Computational methods

2.1. Overview

The software requires an image sequence as input (see Section 3.1 for details) and the user must manually define region(s) of interest and the representative fascicle(s) that they wish to track (see Section 3.3 for details) in the image frame from which they wish to start tracking. Once region(s) and fascicle(s) are defined, the user can process the image sequence, which implements an algorithm based on the affine optic flow model described in Section 2.2 to track fascicle length in the subsequent frames of the image sequence. The algorithm steps through the image sequence one frame at a time, computing the optic flow between consecutive images and applying the affine transformation to calculate the new position and length of the fascicle(s). Because this is an iterative process, any tracking errors in individual frames may accumulate over time and be compounded, causing the fascicle position and length to 'drift' as the algorithm progresses through image sequences. To correct for this, we have developed the '*key-frame correction*' method that is described in Section 2.3.

2.2. The affine optic flow model

The main algorithm in the software implements an affine optic flow model to compute the optic flow and the affine transformation between consecutive images in the sequence. Once the model parameters have been computed for a pair of images, the affine transformation matrix can be used to determine the new Cartesian coordinates of any point defined in the first image, in the second image. This process is applied to compute the displacement of the fascicle endpoints from one image to the next. The model and algorithm have been described and their use validated previously [4,7], but for completeness we will also describe it here.

The model is based upon an affine extension of the well-established Lucas–Kanade method [8]. The affine optic flow model has six parameters: v_{xt} – optic flow at the image origin (top left corner) in the *x*-direction; v_{yt} – optic flow at the origin (top left corner) in the *y*-direction; d – rate of dilation; r – rate of rotation; s_1 – shear along the main image axis; s_2 – shear along the diagonal image axis. These parameters can be used to estimate the flow vector (change in position vx, vy) at specific points in the image (x, y) by applying the following first order model:

$$(v_x, v_y) = \begin{bmatrix} x & t & 1 \end{bmatrix} \times \begin{bmatrix} d + s_1 & s_2 + r \\ s_2 - r & d - s_1 \\ v_{xt} & v_{yt} \end{bmatrix} \quad (1)$$

The Matlab algorithm used here (contributed by Dr. David Young and available from Matlab Central – see acknowledgments) implements a least squares fit of the above parameters to estimates of the spatial and temporal grey-level gradients on a rectilinear grid within the defined region of interest. To calculate the spatial and temporal gradients, the images are first smoothed in both the x and y components by convoluting the image with a 1D Gaussian mask. In the implementation used here, the Matlab function "fspecial" is used to create the mask with a width of 2.6 times the parameter "sigma," which is the standard deviation of the distribution. To calculate the spatial gradients, the symmetric local difference of the average of the two smoothed images is calculated by two-dimensional convolution in the *x* and *y* directions using a simple linear gradient function. The temporal gradients are calculated as the difference between the two smoothed images. The affine flow parameters are then solved by a least squares fit of the parameters to achieve the given spatial and temporal gradients. The x and y grids along with the spatial and temporal gradients are then resampled (3 pixels width) to reduce the data used in the least squares solution and improve processing time. Only data within the defined region of interest are used to obtain the least squares solution. Once the parameters for the model are determined, the flow at individual points can be determined using Eq. (1). Using the model it is possible to calculate the change in position of any x–y point from one image to the next using the affine transformation. This includes points outside of the region of interest and thus, the fascicle endpoints can be outside the region of interest. In our algorithm, the calculated affine transformation is applied to the Cartesian coordinates of the defined fascicle end points from the first image frame of each pair to calculate the new coordinates in the subsequent image. This iterative approach allows

fascicle length to be defined for each ultrasound image in a sequence.

2.3. *Key-frame correction algorithm*

Once the affine optic flow algorithm has been run, the length of the fascicle(s) has been calculated in each frame of the image sequence. It is not uncommon for image quality to be lower at some point during the sequence, causing tracking errors in those frames. Because the algorithm is iterative, these errors can accumulate or be compounded with each iteration and introduce a low-frequency 'drift' to the tracking results. To remove this drift from the signal, we have introduced a post-processing algorithm, the 'keyframe correction'.

The key-frame correction requires that the data be from either a periodic activity or an activity that has some recurring reference state, such that the fascicle length can be assumed to be at a similar length at the same fraction of consecutive activity cycles or at each occurrence of the reference. For example, the muscle fascicles should be at similar lengths if the configuration of the joints that the muscle crosses (e.g. knee and ankle for the gastrocnemius muscle) and muscle forces are similar at two different time points. For instance, during steady state walking, the muscle fascicles should be at similar lengths at the same fraction of the gait cycle (e.g. foot contact, toe-off). The user must identify 'key frames' in the image sequence that occur where the fascicles should be considered to be at similar lengths. Once key frames are identified, the key-frame images are treated as a separate image sequence and the optic flow and affine transformation are computed between each consecutive pair of key-frame images. The first key-frame is assumed to be error free because it should be near the start of the image sequence and therefore the fascicle endpoint coordinates and lengths are not corrected for this frame. For the subsequent key-frames, the coordinates of the fascicle endpoints are recomputed by applying the new affine transformation to the coordinates from the previous key-frame. The horizontal and vertical displacements between the original endpoint coordinates and the new endpoint coordinates are taken to be the magnitudes of the errors in each direction at that point in the full image sequence. The measurement drift between each pair of key-frames is assumed to accumulate linearly with time and the key-frame error values are linearly interpolated to compute horizontal and vertical error values at the time of each image frame between each pair of key-frames. The error value is subtracted from the fascicle endpoint coordinates to compute the new position of each fascicle endpoint in each image frame. These are referred to as corrected fascicle endpoints and are used to compute a corrected fascicle length for each frame (see Section 4 for examples).

The key-frame correction may not always provide satisfactory results. This may be because the assumption that errors accumulate linearly is not always appropriate. As noted previously errors are often due to a sub-sequence of images where tracking is poor, so manually correcting (Section 3.5) during such sub-sequences may produce better results than the key-frame correction.

3. Program description

3.1. *Input files and file conversion*

Version 4.1 of UltraTrack handles any format of input file that is supported by the inbuilt Matlab 'VideoReader' (post R2010a) or 'mmreader' (pre R2010a) functions for the image sequences as well as specifically formatted MAT, B8 and B32 files. B8 and B32 file types are generated by Ultrasonix ultrasound systems (BK Ultrasound, USA) and are read using custom Matlab functions. The formats supported by the 'VideoReader' and 'mmreader' functions are platform dependent with the exception of AVI, which is supported across platforms and has been most extensively tested with the UltraTrack software. For details of other supported file types, visit the support documentation of the MathWorks website (www.mathworks.com). It should be noted that the loading of files that require 'VideoReader' or 'mmreader' functions is considerably slower than loading a MAT file containing the image sequence.

To use a MAT file as the input file, it must contain a Matlab structure named 'TVDdata' with the following fields: 'Fnum' – a numeric value for the number of frames in the image sequence; 'Width' – a numeric value specifying how many columns of pixels the images consist of; 'Height' – a numeric value specifying how many rows of pixels the images consist of; 'Time' – a column vector of time stamps corresponding to each frame in the image sequence; 'Im' – a three dimensional array where each 'page' is a grey scale image of data type uint8 values. The MAT input file content was specifically developed from the conversion of TVD files that are generated by EchoWave II ultrasound imaging software (Telemed, Lithuania). With the UltraTrack software, we provide a Matlab script (TVD_2_MAT.m) and functions that convert TVD files to MAT files with the necessary content. The conversion script interfaces with the latest releases of EchoWave II (Telemed, Lithuania) that must be installed and running when the script is executed. Also, the user must be running Matlab as an administrator for the script to execute.

3.2. *Loading, cropping and cutting the image sequence*

Loading, cropping and cutting the image sequence are the first three steps in the process of tracking a file. Loading is simply a case of selecting 'Open Video File' from the 'File' drop-down menu and you will be prompted to select the appropriate file. When the file has finished loading, the first image frame will be displayed. In this software, 'cropping' refers to removing any surrounding borders from the image and it is particularly important to remove any borders above or below the image. This is because the scale of the image (pixels/mm) is calculated by dividing the number of rows of pixels by the specified image depth. The image depth refers to the scanning depth in mm and is input either by the editable text box on the user interface or from the 'Settings' drop-down menu. If the user does not wish to crop the image but knows the scale factor for the image and the image height (in pixels), an appropriate image depth can be calculated and used. For images produced by EchoWave II software (Telemed, Lithuania), auto crop functionality is available from the 'Image' drop-down

menu that will automatically remove all borders from the image.

'Cutting' the sequence refers to removing frames from the start or end of the video sequence. It is imperative that any cutting of the image sequence is done before any of the steps in the tracking process described below are undertaken. The sequence can be cut by navigating to the appropriate frame and clicking on the 'Cut Before' or 'Cut After' buttons on the 'Setup Tracking' panel of the user interface. Cutting the image sequence down to the necessary number of frames will speed up the tracking proportionally. Cutting or cropping does not affect the original input file in any way.

3.3. *Defining regions of interest and fascicles*

The affine optic flow computations are applied to a specified region of interest (ROI) within the images. This ROI is defined using the 'Define ROI' button on the 'Setup Tracking' panel that allows the user to define the vertices of a polygon by clicking on the image before double clicking inside the completed polygon to finalise the ROI, which will display as a red dashed outline when finalised. Typically, the ROI is defined as the outline of the cross section of the muscle of interest. Our experience has taught us that this should usually include the thick muscle fascia, or aponeurosis, that borders the muscle. However, finding the ROI definition that works best for your image sequences may require some trial and error. Whether or not to include aponeurosis may depend on whether it can be distinguished separately from that of adjacent muscles and any relative movement that occurs between muscles (or other tissues).

In version 4.1 of UltraTrack, it is now possible to have multiple ROI's such that more than one muscle can be concurrently tracked. ROI's can be added by clicking on the add button next to the drop-down list of region numbers. The number displayed on the region number list is that of the currently selected region. The user has the option to select a fixed ROI by checking the 'Fixed ROI' check box. This means that the ROI will remain the same throughout the image sequence as the algorithm steps though the images. If this box is not checked, then the ROI vertices have the affine flow transformation applied to them in the same way as the fascicle endpoints and the ROI will move and change shape to account for movement of the muscle. This latter option is only recommended if there is significant vertical displacement of the muscle during the sequence. It will also tend to reduce the width of the ROI as the sequence progresses, which results in the tracking region changing across a video sequence and therefore may influence the tracking results.

Within each ROI, at least one fascicle must be defined. To define a fascicle, click on the 'Define Fascicle' button and you will be able to click on or around the image to define the endpoints of the fascicle. It is possible to define multiple fascicles within a ROI and fascicles can be added to the currently selected ROI by clicking on the add button next to the drop-down list of fascicle numbers. Fascicle numbering starts from one for each ROI. For example, if you wish to track two ROI's with one fascicle in each, the ROI list should have the numbers one and two available and the fascicle list has only the number one. In this case, which fascicle is selected depends only on which region is selected. Regions and fascicles should be defined in the image from which you wish to start the tracking and this should be the first frame of the sequence. Therefore users should cut the image sequence so that it starts from the desired start frame.

3.4. *Running the tracking algorithm*

Having defined the appropriate ROI(s) and fascicle(s) the user can then run the affine flow algorithm by clicking the 'Process All' button on the user interface or selecting that option from the 'Tracking' menu. This uses the affine optic flow model described in Section 2.2 to track the defined fascicles throughout the image sequence. Once the algorithm has finished, fascicle lengths will be plotted in the axes at the top right of the user interface and the tracked fascicle(s) and ROI(s) will be overlaid on the image sequence. A more manual approach to tracking can be performed by clicking through the sequence frame-by-frame rather than selecting "process all".

3.5. *Making manual adjustments*

It may be desirable to make adjustments to the tracking results if there are frames in which the tracking algorithm has been unable to suitably track the fascicle. Potential causes of poor tracking are: large movements between frames (often caused by insufficient sampling rates) and large changes in image brightness caused by movement not in the plane of the image (constant image brightness is a key assumption behind optic flow calculations). As described elsewhere in this paper, a key-frame correction has been developed to remove drift in recordings of periodic activities or activities with a common set point (e.g. a muscle that is fully relaxed before and after a contraction). However, there may be occasions for which it is more appropriate simply to make a manual correction in certain frames. For this purpose there is the manual correction panel on the user interface that allows the user to adjust the position of the upper (U button) or lower (L button) endpoint of the fascicle. Upper or lower refers to the vertical position within the image of that fascicle endpoint. Having clicked on the U or L button, the user can then use the cursor to select a new position for the fascicle endpoint and the affine transformation will be re-applied to the new coordinates and all subsequent frames to account for the adjustment. Corrected fascicle lengths will be calculated and displayed in the fascicle length plot and the corrected fascicle overlay will be displayed on the image sequence in yellow.

3.6. *Making a key-frame correction*

The key-frame correction and its purpose have been described in detail in Section 2.3. It can only be performed once the affine optic flow algorithm has been run and can be run on data that has had manual or previous key-frame corrections made to it (although the latter is not recommended). To make a correction, the user must first create a list of key-frames by navigating to each key-frame using the video controls and clicking 'Add' button on the key-frame correctionpanel. The

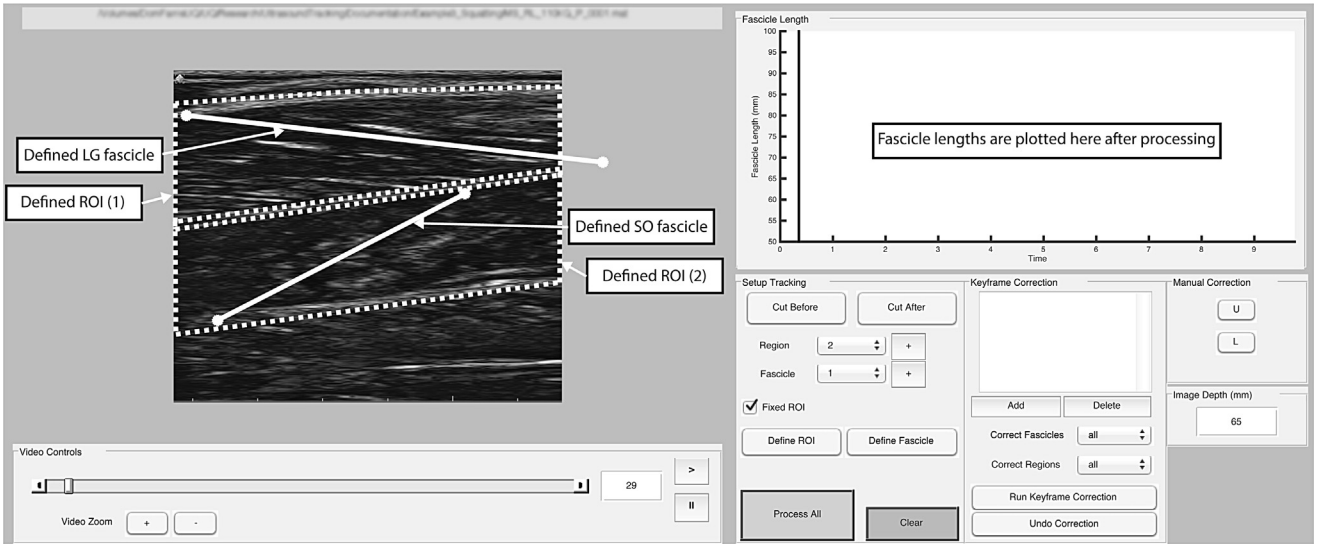


Fig. 1 – Snapshot of the UltraTrack GUI after the image sequence has been loaded and an ROI and fascicle have been defined (note: colours have been adjusted for grey-scale display).

current frame number should be added to the list of key-frames. To remove a key-frame from the list, the user simply selects that frame in the list and clicks the 'Delete' button underneath the list. When the key-frames have been selected, the user can choose what ROI and fascicle to apply the correction to using the drop-down lists. Most often the user will simply select 'all' and correct all ROI's and fascicles. Having set these selections, the user can click the 'Run Key-frame Correction' button and the correction algorithm described in Section 2.3 will be applied. If the results of the correction are unsatisfactory, the user can click the 'Undo Correction' button and the tracking will be reset to its previous state. If more than one correction has been applied, the user can only undo the most recent correction.

3.7. *Saving tracking results*

Two options exist for saving tracking results to file: MAT file and TXT file. MAT file is recommended as this saves all the necessary information to reload the tracking results along with the image sequence if necessary. Saving to a TXT file will only save the fascicle length and pennation angle data. Tracking data can be saved using the 'Save Tracking Data' or 'Save Tracking Data as..' options from the 'File' menu. When saved to a MAT file, the 1-D structure 'Fdat' is saved in the file. Fdat contains the field 'Region' which is a $1 \times n$ structure where n is the number of ROI's. The Region structure contains the fields: 'FL' – fascicle length; 'PEN' – pennation angle and 'Time' – time stamps corresponding to each of the data points in FL and PEN. FL and PEN are $m \times n$ 2-D arrays where m is the number of tracked image frames and n is the number of fascicles defined for that ROI. It should be noted that pennation angle is not a true pennation angle and is in fact the angle of the fascicle relative to the image horizontal axis. There is also an option in the Tracking ≫ Fascicle Tracking menu to 'Save Fascicle' that saves the tracking data for current frame to a MAT file.

3.8. *Loading tracking data*

If tracking results were saved to a MAT file, they can be reloaded after the image sequence has been loaded with the 'Load All Tracked Frames' option in the 'Fascicle Tracking' sub-menu of the 'Tracking' menu. Doing so will cut the image sequence to the number of previously tracked frames. If a single frame of data has been saved with 'Save Fascicle' then it can be reloaded as the current frame's data using Tracking ≫ Fascicle Tracking ≫ Load Fascicle.

4. Sample runs

4.1. *Gastrocnemius and soleus fascicles during squatting*

As an example run of the software we present the tracking of a gastrocnemius fascicle and a soleus muscle fascicle during a repeated squatting movement. B-mode ultrasound imaging was performed using a 96-element linear array ultrasound transducer (LV7.5/60/96Z, Telemed, Lithuania) operating at a central frequency of 6 MHz, a depth of 65 mm and sampling images at 80 frames per second. The participant was a healthy young male athlete and he performed two repetitions of a squatting movement (for details see [9]). The image sequence was recorded using EchoWave II software as a TVD file and converted to MAT file using our custom TVD_2_MAT script described in Section 3.1. The sequence was loaded into UltraTrack 4.1 and auto-cropped to remove image borders before the sequence was cut down to the relevant frames for analysis (using 'cut before' and 'cut after' buttons and synchronously collected ground reaction force data). We then defined the first ROI (surrounding the gastrocnemius muscle) in the first frame by selecting 'Define ROI' button and defining a polygon around the portion of the image where gastrocnemius was visualised (Fig. 1). Here, we used a fixed ROI by checking the 'Fixed ROI'

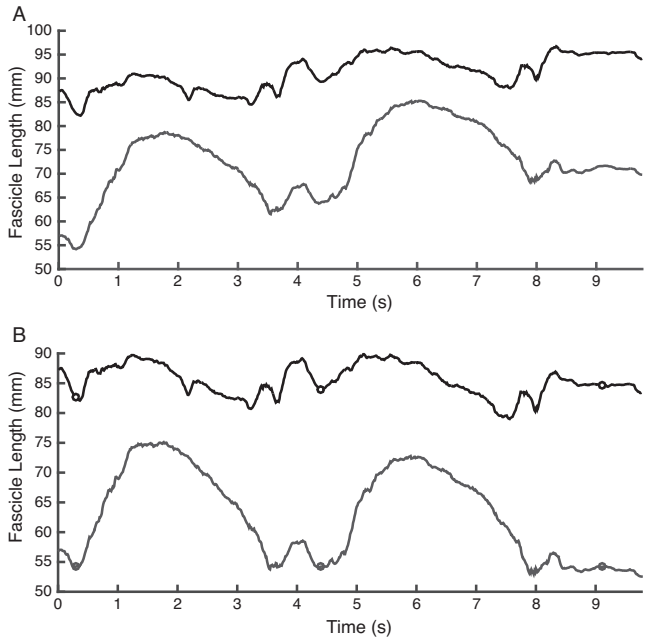


Fig. 2 – Lateral gastrocnemius (black) and soleus (grey) muscle fascicle length over two consecutive squats before (A) and after (B) the key-frame correction was used to remove drift in the signal. The timings of key-frames (which correspond to quiet standing between squats) are highlighted with circles (B). A characteristic lengthening and shortening of the fascicles of both muscles was observed as the participant flexed and then extended their lower limb.

check box (Fig. 1). We then selected the 'Define Fascicle' button and clicked on the endpoints of a clearly visible gastrocnemius fascicle in the centre of the ROI in the first image (Fig. 1). A second ROI was added by selecting the '+' button next to the ROI drop-down list. The second ROI was then made the current ROI by selecting '2' from the ROI drop-down list and subsequently defined by clicking 'Define ROI' and outlining a polygon surrounding the soleus muscle. The soleus fascicle was then defined (Fig. 1) using 'Define Fascicle'. Note that the fascicle count restarts for each region, so you do not need to add a fascicle to the drop-down list. We then clicked the 'Process All' button to run the affine optic flow algorithm.

The fascicle length data output from the affine optic flow algorithm are shown in Fig. 2A. As one might predict, the muscle fascicles of both muscles lengthened during the downward phase of the squat motions and shortened during the upward phase. The length change was more pronounced in soleus than lateral gastrocnemius due to soleus spanning only the ankle joint whereas gastrocnemius spans the ankle and knee joints. The presence of low-frequency drift is evidenced by the progressive drift to longer lengths, but with a constant pattern of length changes in each squat. We therefore opted to run a key-frame correction on this data. Using synchronously collected ground reaction force data, we identified the times in between squats when the participant was standing still and selected the corresponding image frames as key-frames using the video controls and 'Add' button on the Key-frame Correction panel (Fig. 1). We then clicked the 'Run Key-frame Correction' button and the resulting corrected fascicle length plot is shown in Fig. 2B. A video of the image sequence with the tracked fascicles and ROIs overlaid was exported using the 'File ≫ Save Video' menu item and is provided in the electronic supplementary information of this manuscript. Further examples will be provided via the software web repository (see Section 6).

The image sequence for this example was 9.2 s long, with 801 grey-scale image frames of dimensions 645 × 745 pixels. Stored as a Matlab data file, the file size was 336 MB. UltraTrack was running on a MacBook Pro with a 2.5 GHz Intel core i5 processor, 8 GB of RAM and Mac OS X 10.9.5 (Mavericks). The Matlab profiler indicated that loading the file took a total time of 17.1 s and processing with the affine flow algorithm took 110.3 s.

4.2. *Fascicles in the superficial and deep compartments of tibialis anterior during walking*

For a second example we present data for the bipennate tibialis anterior (TA) muscle from one young healthy individual as they walked at 4.5 km/h on an instrumented treadmill (AMTI, USA). The same ultrasound system as in Section 4.1 was used with the same settings, but positioned on the anterior leg to image TA. We followed the exact same process for tracking as described in Section 4.1 and the defined fascicles and ROI's are shown in Fig. 3. The fascicles extended outside the field of view of the transducer and thus, the endpoints had to be extrapolated outside the image. Despite this, the algorithm uses information from within the ROI only. That is to say that the affine transformation matrix is computed from the optic flow calculated within the ROI, but applied to the coordinates of the endpoints that may be outside the ROI. The initial tracking

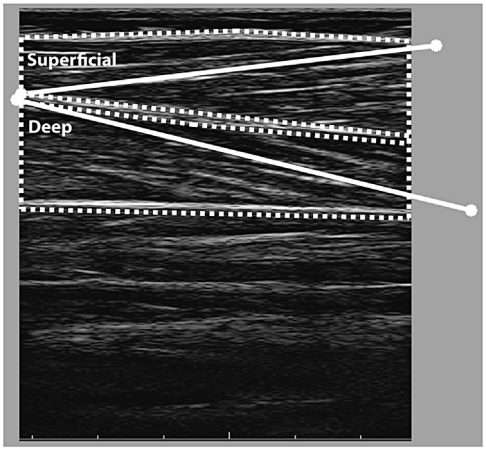


Fig. 3 – A B-mode ultrasound image of tibialis anterior from the sample run described in Section 4.2. Fascicles and ROI's for the superficial and deep compartments of tibialis anterior were defined as shown. Note that the fascicles extend beyond the field of view of the image and thus are extrapolated to where they would insert on the muscle fascia/aponeurosis. Only image data within the ROI's is used to track the extrapolated fascicle endpoints.

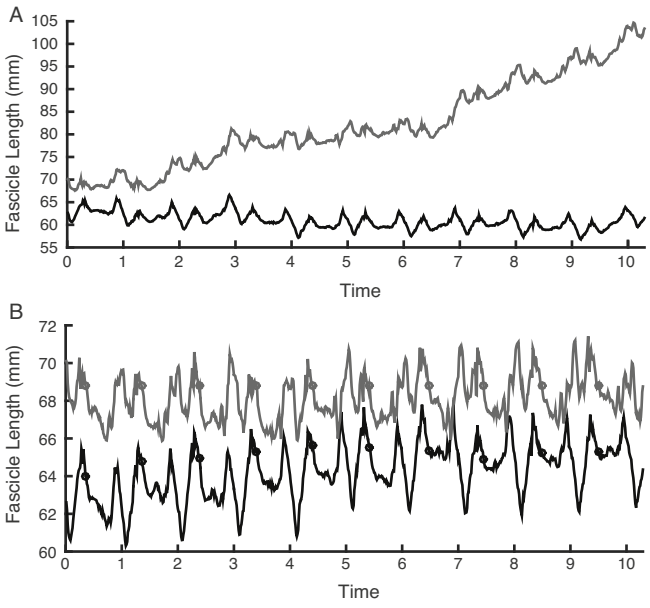


Fig. 4 – Tracking results for fascicles from the superficial (black) and deep (grey) compartments of the tibialis anterior during walking on a treadmill before (A) and after (B) key-frame correction (timings of key-frames indicated with circles in panel B). A video showing the tracking overlaid on the image sequence is available as supplementary material.

results are shown in Fig. 4a and drift is clearly present, especially for the superficial compartment. Using synchronously collected ground reaction force data we identified the timing of the first active peak in vertical force in each step led with the right foot and used the corresponding frames as key-frames in a key-frame correction applied to both fascicles. The corrected lengths are shown in Fig. 4b and a video of the tracking is available as supplementary material.

The second example was a 10.55 s clip, with 844 grey-scale image frames of dimensions 645 × 583 pixels. Stored as a Matlab data file, the file size was 270 MB. UltraTrack was running on the same computer described above. The Matlab profiler recorded that file loading took 14.9 s and processing with the affine flow algorithm took 97.7 s. It should be noted that loading an AVI file as opposed to a MAT file takes significantly longer. As an example, a similar an AVI with only 441 frames, but similar dimensions took 96.3 s to load.

5. Hardware/software specifications

5.1. *Standalone version*

A standalone application installer has been created for MacOS and Windows platforms. The standalone application requires Matlab Runtime 8.5 to be installed and the installer for this is packaged with the application installer.

5.2. *Source code*

For users wishing to run UltraTrack 4.1 using the source code, the Matlab M and FIG files required are available. Relevant folders must be added to the Matlab path and the in-built image processing toolbox is required. UltraTrack v4.1 has been tested in Matlab R2015a. There are known graphics problems running UltraTrack 4.1 with R2014a and b but prior releases have no such issues. This is likely related to problems with updates to Matlab graphics objects in R2014a. Separate versions are available for MacOS and Windows platforms, but the only differences are cosmetic – the source code is essentially the same.

5.3. *File conversion*

M files are provided with the software for the conversion of TVD files (created by EchoWave II software, Telemed, Lithuania) to MAT files. A beta version of EchoWave II (freely available for download from pcultrasound.com) must be installed and running when executing these files in Matlab, which must be run as an administrator.

5.4. *System requirements*

The requirements for installing and running Matlab Runtime 8.5 and Matlab 2015a are: Windows – Windows XP or later; 3–4 GB of disk space (typical installation), 2 GB of RAM. Mac – Mac OS X 10.9.5 (Mavericks) or later, 3–4 GB of disk space (typical installation), 2 GB of RAM.

6. Mode of availability

UltraTrack 4.1 is available free of charge for non-commercial use from the 'files' page of the website (https://sites.google.com/site/ultratracksoftware/file-cabinet). The website has additional materials and examples.

Acknowledgements

The authors would like to acknowledge Dr David Young of The University of Southampton (UK) who wrote the Matlab classes for implementing the affine optic flow algorithm (available at http://au.mathworks.com/matlabcentral/fileexchange/27093-affine-optic-flow).

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.cmpb.2016.02.016.

REFERENCES

[1] T. Fukunaga, Y. Kawakami, S. Kuno, K. Funato, S. Fukashiro, Muscle architecture and function in humans, J. Biomech. 30 (1997) 457–463.

[2] T. Fukunaga, Y. Ichinose, M. Ito, Y. Kawakami, S. Fukashiro, Determination of fascicle length and pennation in a contracting human muscle in vivo, J. Appl. Physiol. 82 (1997) 354–358.

[3] G.A. Lichtwark, K. Bougoulias, A.M. Wilson, Muscle fascicle and series elastic element length changes along the length of the human gastrocnemius during walking and running, J. Biomech. 40 (2007) 157–164.
[4] N.J. Cronin, C.P. Carty, R.S. Barrett, G. Lichtwark, Automatic tracking of medial gastrocnemius fascicle length during human locomotion, J. Appl. Physiol. 111 (2011) 1491–1496.
[5] J. Darby, E.F. Hodson-Tole, N. Costen, I.D. Loram, Automated regional analysis of B-mode ultrasound images of skeletal muscle movement, J. Appl. Physiol. 112 (2012) 313–327.
[6] G.-Q. Zhou, P. Chan, Y.-P. Zheng, Automatic measurement of pennation angle and fascicle length of gastrocnemius muscles using real-time ultrasound imaging, Ultrasonics 57 (2015) 72–83.
[7] J.G. Gillett, R.S. Barrett, G.A. Lichtwark, Reliability and accuracy of an automated tracking algorithm to measure controlled passive and active muscle fascicle length changes from ultrasound, Comput. Methods Biomech. Biomed. Eng. 16 (2013) 678–687.
[8] B.D. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, in: Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, Canada, 1981.
[9] D.J. Farris, G.A. Lichtwark, N.A. Brown, A.G. Cresswell, Deconstructing the power resistance relationship for squats: a joint-level, Scand. J. Med. Sci. Sports (2015), http://dx.doi.org/10.1111/sms.12508.