This is a brief overview of functionality contained in the SCUMS library. We also invite the reader to go through the provided sample scripts, where the functionality is demonstrated and commented upon using examples of real-world data (briefly summarized below). The source codes for the SCUMS library are commented and all the parameters are explained there; the comments describing the functions are also reproduced in this document. The function names below are identical for Matlab and Python. Input/output parameters are given in a table for each function.

Sample scripts

- **sampleScript_1_CVcomparison** - This script takes two recordings of voltage mapping in the same heart, before and after application of isoproterenol, showing an increase in conduction velocity under isoproterenol.
- **sampleScript_2_singleTraceProcessing** - This script shows how to extract properties such as APD from single-trace recordings. While such a trace here was extracted as a spatial average of imaging experiment, electrophysiological recordings may be naturally processed in a similar fashion.
- **sampleScript_3_alternans** - This script shows how to analyse calcium transient (CaT) alternans in a rapidly-paced heart. It visualizes and quantifies the discordant alternans present.
- **sampleScript_4_S1S2 -** This script demonstrates how to use custom comb to process S1S2 recordings (3 S1 stimuli, 1 S2 stimulus), producing a calcium transient restitution curve (Figure 6 in the article). For the purposes of this script, we will take the average of the field of view as a single trace on which we measure the restitution. In addition, it is demonstrated at the end of the code how the processing may be ran using parallel for-loop, accelerating the runtime.
- **sampleScript_5_cellCultureMask** - This script demonstrates how to a) load a stack of data stored in a .mat file, rather than in image files, b) apply binary mask to it to discard image background. In this case, this is applied to remove the surroundings of a Petri dish with a cell culture. The script ultimately plots the conduction map (based on calcium mapping data).
- **sampleScript_6_videoSpiral** - This scripts shows how to a) read data of a spiral wave mapping using off-axis illumination imaging from an mp4 video, b) plot the activation map and to show how standard deviation of duration may be used to get rid of spurious signal areas.

Reading data

Here is described how image/video data may be read using SCUMS.

We note, that ultimately, the stack should be converted to double, so that the smoothing function does not crash.

SCUMS.readFolder

This reads a folder with an image sequence corresponding to a recording (one image corresponding to a single frame); common formats such as .tif, .jpg, or .png are supported. The function returns a 3D stack, where rows and columns correspond to image rows and columns, and the third dimension corresponds to the index of the frame in the sequence (i.e. corresponding to time in the recording). In Python, a completely analogical 3D matrix is used to store data.

IN	
folderName	The folder containing a sequence of images corresponding to frames.
varargin	Optional parameteter, which can contain a structure of parameters (called e.g. readerParameters), with the following fields: 1) readerParameters.extension - extension of images in the given folder 2) readerParameters.fromTo - a 2-by-1 vector [from to], determining from which to which frame is the recording to be processed 3) readerParameters.binningFactor - spatial binning parameter (must be a power of two) - binningFactor x binningFactor pixels are aggregated into a single one.
OUT	
imStack	The resulting 3D stack, where rows and columns correspond to frame rows and columns, and 3rd dimension to index of image in the sequence (i.e. time).
avgTrace	A trace corresponding to spatially averaged stack (useful for a quick look at what the pattern of activation was, etc.).

SCUMS.readTifStack

An analogy of the function above, but reading a TIFF stack, where all the frames are stored in a single .tif file.

IN	
fname	The path to the tif stack (including extension).
varargin	Optional parameteter, which can contain a structure of parameters (called e.g. readerParameters), with the following fields: 1) readerParameters.extension - extension of images in the given folder 2) readerParameters.fromTo - a 2-by-1 vector [from to], determining from which to which frame is the recording to be processed 3) readerParameters.binningFactor - spatial binning parameter (must be a power of two) - binningFactor x binningFactor pixels are aggregated into a single one.
OUT	
imStack	The resulting 3D stack, where rows and columns correspond to frame rows and columns, and 3rd dimension to index of image in the sequence (i.e. time).
avgTrace	A trace corresponding to spatially averaged stack (useful for a quick look at what the pattern of activation was, etc.).

Alternative ways of reading data

If your data are in a format that cannot be read using the two functions above, but you know how to read them in Matlab/Python otherwise, just load them into the 3D matrix format described above (dimensions being rows, columns, time), and this may be seamlessly used as the input to the data analysis functions.

Analysing data

Here are described functions which extract features (duration, amplitude, activation times, etc.) from 3D stacks of data.

SCUMS.analyseRegularPacing

This function extracts the features of multiple waves passing through the recording, returning the information on each wave pass, as well as their average.

IN	
imageStack	A stack representing the recording.
bcl	Basic cycle length of the recording (in frames per second)
parameters	A structure of processing parameters, with the following fields: 1) parameters.baselineDefinition – whether baseline of a wave pass (action potential or calcium transient) is taken as the first element in each segmented activation ('first'), or as the average of the first and last element('firstlast'). The default is 'first'. 2) parameters.baselineSubtractionOrder – the order of polynomial subtraction of signal baseline/drift. Use -1 if no baseline subtraction is to be done (or just leave the parameter field undefined). The default is '-1'. 3) Parameters.customComb – if provided, a custom-shaped comb to segment action potentials/calcium transients is used, rather than a comb with teeth that are regularly spaced (as a consequence, the parameter 'bcl' is not used in the analysis, although it still has to be provided). The custom comb is provided as a vector where distances between consecutive elements correspond to intervals between minima to be found using the comb, and the first element of the vector determines until which frame the first minimum is searched for (i.e. corresponding to the position of the first comb tip). For example, the vector [15, 115, 215, 265] will produce a comb to segment a recording where we know that the first minimum occurs within first 15 frames, and is followed by two minima after 100 frames each, and then another one after 50 more frames (this could be e.g. S1S2 protocol with S1=100 ms and S2=50 ms). See sampleScript_5_S1S2 for a real-world demonstration of the application of a custom comb. 4) parameters.durationLevel – level at which duration is extracted, scaled to 0-1 (e.g. use 0.8 for APD80). The default is 0.75. 5) parameters.objectDetection – when multiple activations are discovered in a single signal segment (while only one can be true calcium transient/action potential), this parameter determines how the correct activation is detected. 'first' - first object found in the segment. 'largest' - the largest object is picked (with most frames). 'augmented' - information on derivative of the signal is used, see the publication. For voltage mapping with multiple wave passes, we recommend using 'augmented' (which is not great for calcium, given that there

	are no sharp upstrokes). Otherwise, 'largest' is usually more robust than 'first'. The default is ‘largest’. 6) parameters.smoothingParameter - width of Savitzky-Golay filtering for signal smoothing. It should be an odd number (if an even number is given, 1 will be added). Given that 4th order smoothing is used, this parameter has to be at least 5 when provided (when smaller, no filtering is done). Th default is 11. 7) parameters.spikesPointDown – if true, signal activation manifests as reduction in signal intensity (e.g. some voltage dyes such as RH237, using which action potentials “point down”). By default, this parameter is set to ‘false’. 8) parameters.verbose – if true, the code reports when there is a problem with segmentation and/or processing of a pixel trace (which may happen when a trace contains pure noise, or is saturated). The default is ‘false’. 9) parameters.waveProcessing – if 'perbeat', each wave is processed separately, and the output structures contain a map for each complete wave pass. If 'hybrid', a recording clock is used to chop the recording into sub-stacks, which are then averaged, and a single-wave processing is applied to this subsequently. The value 'hybrid' is good for very noisy recordings and activation mapping, but is not suggested to be used for APD mapping or amplitude measurements. If ‘hybrid’ is used, parameter.objectDetection should be set to ‘largest’. The default is ‘perbeat’.
OUT	
baseline	A structure describing signal baseline (e.g. diastolic calcium levels). See below for structure fields.
amplitude	A structure describing signal amplitude (e.g. amplitude of a calcium transient).
duration	A structure describing signal duration (e.g. APD).
activationMaps	A structure describing activation pattern in the recording – this is taken relative to the recording clock; if you want to have a minimum of 0 in the provided maps, just subtract the map minimum.
recoveryMaps	A structure describing the recovery pattern (e.g. the time when APD80 is reached); again, relative to the recording clock.
recordingClock	The recording clock determining global synchronization of single segmented activations.
GENERAL STRUCTURE	The structures **baseline**, **amplitude**, and **duration** have the following fields (written as for **duration**): 1) duration.maps - a 3D stack where each slice corresponds to map of the feature in a single wave pass (i.e. for four processed wave passes, the stack will have four layers). 2) duration.mapMean - the average of the previous maps (averaged over the slices) 3) duration.data – a vector of spatial averages of maps in duration.maps. 4) duration.mapMeanEven - mean map for even beats (useful for inspection of alternans). 5) duration.mapMeanOdd - mean map for odd beats

	6) duration.dataMeanEven - spatial averages of even maps of the feature (this gives a sort of “average alternans” in each even wave pass) 7) duration.dataMeanOdd – as above, but for odd wave passes. The structures **activationMaps** and **recoveryMaps** have the same fields, except the ones starting with 'data' (there is not much point in spatially averaged activation or recovery).

SCUMS.analyseSinglePass

This analyses a stack containing a single pass of the cardiac wave in a similar fashion to SCUMS.analyseRegularPacing, except it does not split the recording into single wave passes (as there is only one). It is up to the user to guarantee that the stack provided to this function contains a single wave pass, there is no additional check.

Inputs/outputs are as in SCUMS.analyseRegularPacing, except the ‘bcl’ input is not provided. The output structures (baseline, amplitude, duration, …) contain only two fields: mapMean[1] (2D map of the feature) and dataMean (a single number which is an average of mapMean).

Postprocessing/visualizing data

These functions are helpful for analyzing the maps produced by the function above (e.g. measuring alternans or conduction velocity from such maps).

SCUMS.getAlternans

The function measures alternans in any feature extracted above (e.g. amplitude, duration, or activation). When called on a vector, it returns a single number (giving alternans between even/odd values), when called on stack of spatial maps, it produces a single spatial map (giving alternans between even/odd slices).

IN	
data	Either a vector of numbers or a stack of spatial maps of the feature on which alternans is to be measured (e.g. ‘amplitude.maps‘ produced by SCUMS.analyseRegularPacing).
varargin	An optional parameter which may determine the method for alternans estimation. In both, average for even and odd values/slices is computed. Then, 'largerToSmaller' (default) measures ratio of larger to smaller, and 'sMAPE' computes abs(odd-even)/(odd+even).
OUT	
alternans	If data is a vector, this gives a single number, if data is a stack of spatial maps, it produces a single spatial map.

[1] The use of ‘Mean’ in the name may sound illogical at first (there is just one map in the recording, so doing temporal average doesn’t do anything). However, when using this function in our analyses, the outputs of the function were used in a similar context when mapMean/dataMean would be used in analyseRegularPacing. Calling the outputs the same means there is less code rewriting needed when one switches between analyseRegularPacing and analyseSinglePass.

SCUMS.getCV

Analyses conduction velocity (CV) between pair (or pairs) of points, using a given activation map. By default, the function returns cv in pixels/frame, but if spatial and temporal resolution are provided, it provides it in the standard unit of cm/s.

IN	
activationMap	A single activation map.
XY	a matrix of size n-by-4 encoding pairs of points between which CV is measured using the provided activation map. Each row corresponds to an origin and target point (columns are: rowFrom, columnFrom, rowTo, columnTo)
varargin	The optional parameter may contain a two-element vector allowing specification of spatial (how many mm is a single pixel side) and temporal resolution (in frames per second). If provided, the CV is given in cm/s, otherwise in pixels/frame.
OUT	
cv	A vector of conduction velocities, one per row of XY.

SCUMS.getLocalCV

Performs local estimation of CV using Bayly's method (doi: 10.1109/10.668746), producing a vector field and optionally its plot.

IN	
activationMap	A single activation map.
baylyNeighbourhood	Thr distance around a point that is considered when fitting the Bayly polynomial.
varargin{1}	If defined, gives maximum length of a velocity vector (the longer ones are discarded).
varargin{2}	If defined, contains the index of figure in which the CV field is drawn. If not defined, no figure is produced.
varargin{3}	If defined, the output path of storage of varargin{2}.
OUT	
xyuv	A n-by-4 matrix, where n is number of pixels and columns correspond to x,y,u,v: x,y gives indices of row and column, with u,v corresponding to dx,dy. Mind that this is in row/column coordinates - if plotting via quiver (in standard x-y coordinates), this needs to be shuffled slightly, see the code at the end of the function.

SCUMS.plotActivationMap

A function which plots a contour map of activation pattern.

IN	
activationMap	A single activation map.
varargin{1}	Optionally, the index of figure used for this purpose may be given - if not specified, a new figure is opened and used.
varargin{2}	Optional parameter defining the 'levels' parameter of @contourf (basically, a contour line is drawn each varargin{2} ms).

Other helper functions

SCUMS.applyMask

This function may be used to apply a binary mask to each frame of a 3D stack, setting pixels-to-be-discarded to NaN. This may be useful to get rid of empty space around the image of the heart, space around a Petri dish for cultures, etc.

IN	
imStackIn	A 3D stack representing a recording.
mask	A binary mask of the same size as a single frame of imStackIn. Where mask==0, the corresponding pixels are set to NaN.
OUT	
imStackOut	imStackIn where zero elements in mask are set to NaN for each frame.

SCUMS.binningVec/SCUMS.binning_vec

A function which carries out spatial binning for an image (replacing each bin with the average of the pixels found in it).

IN	
img	An image to be spatially binned.
logFactor	A base-two logarithm of the binning factor - this must be apositive integer (i.e., logfactor of 1 leads to 2-by-2 binning, 2 to 4-by4, 3 to 8-by-8, etc.). Note that while this helper function requires a logarithm of the binning factor, the data-reading functions readFolder and readTifStack take the binning factor directly, computing its logarithm internally, so that the user doesn't have to take care of that.
OUT	
binnedImage	The source image after spatial binning.

SCUMS.combGetMinima

A function searching for minima in traces from cardiac preparations with known activation pattern. The function can be naturally also used to extract signal maxima when the source trace is inverted.

IN	
signalTrace	A vector containing signal, such as calcium transients or action potentials (e.g., intensity of a pixel in optical mapping, or an electrophysiological recording).
bcl	The basic cycle length (number of frames between two activations).
varargin{1}	The first optional parameter is the refinementWidth parameter for comb algorithm (in ms/samples - the radius of local search around comb teeth). Default is 10.
varargin{2}	The second optional parameter is the custom comb that is used instead of the regularly placed one. It should be an increasing vector of numbers, where the first element determines the last possible position of the first minimum to be searched for, and thesubsequent elements give further indices of minima. E.g. using [30 180, 330, 400] means that the algorithm will search for 3 minima that are 150 frames apart, and one that is further 100 frames after the last previous one, and the first minimum is to be placed between frame 1 and frame 30. See sampleScript_5_S1S2 for an example of custom comb. If this parameter is defined, then the parameter 'bcl' is not used within the code (it still has to be provided, but it can be any number).

OUT	
vectorMinima	The vector of local minima in the signal, which are approximately bcl ms (or samples) apart.

SCUMS.getAPD

The function returns the duration of an action potential (or calcium transient) at the desired level of repolarization. The function assumes the action potential/calcium transient to be upward-pointing (i.e. peak activation is more positive than resting value) - if used on downward-pointing signal, this needs to be inverted first.

IN	
time	The time vector for the activation trace[2].
activationSignal	The trace of a single action potential or calcium transient (or any similar signal).
level	The level of recovery at which the duration is to be measured. This is scaled between 0 and 1 (i.e., for APD80, the value 0.8 is to be used).
varargin{1}	A string encoding the method of baseline estimation; see the documentation of SCUMS.analyseRegularPacing, the 'parameters.baselineDefinition' parameter.
varargin{2}	A string encoding the method of object detection/selection; see the documentation of SCUMS.analyseRegularPacing, the 'parameters.objectDetection' parameter.
varargin{3}	When 'augmented' objectDetection is used, this parameter is used to pass the time of peak upstroke time within the single activation provided in 'activationSignal'. When more objects are above the threshold determined by 'level', the one closest to this parameter is used.
OUT	
apd	The duration of the action potential/calcium transient at the given level of recovery. The function uses interpolation to get sub-frame resolution.
timeRecovery	The time of the end of the action potential/calcium transient at the given repolarization level. The function uses interpolation to get sub-frame resolution.

SCUMS.getHalfActivationTime

The function returns the time at "half-activation" of an action potential or calcium transient; taken with regards to amplitude (i.e. it corresponds to the time when APD50 would be measured from when applied to action potentials). It uses linear interpolation to get this half-activation time "accurately" (as in not just the previous/subsequent frame). The function assumes the action potential/calcium transient to be upward-pointing (i.e. peak activation is more positive than resting value) - if used on downward-pointing signal, this needs to be inverted first.

IN	
activationSignal	The trace of a single action potential or calcium transient (or any

[2] This is the only function which takes a time vector (other functions assume the sampling rate of the recording is constant, and it works with frames as time units). The reason for this is purely historical, rather than a smart design – we were using this function in other projects to also analyse results of computer simulations in which the "sampling rate" is irregular when using adaptive solvers, and we needed to be able to work with this.

	similar signal).
level	The level of recovery at which the duration is to be measured. This is scaled between 0 and 1 (i.e., for APD80, the value 0.8 is to be used).
varargin{1}	A string encoding the method of baseline estimation; see the documentation of SCUMS.analyseRegularPacing, the 'parameters.baselineDefinition' parameter.
varargin{2}	A string encoding the method of object detection/selection; see the documentation of SCUMS.analyseRegularPacing, the 'parameters.objectDetection' parameter.
varargin{3}	When 'augmented' objectDetection is used, this parameter is used to pass the time of peak upstroke time within the single activation provided in 'activationSignal'. When more objects are above the threshold determined by 'level', the one closest to this parameter is used.
OUT	
halfActivationTime	The time of half-activation of the signal (i.e. time when 50% of the signal amplitude is achieved). The function uses interpolation to get sub-frame resolution.

SCUMS.getImageForMask

This function gives a high-contrast average image of the stack (averaged over time). This can be used to draw a mask (manually or automatically).

IN	
imStack	A stack representing the recording. The input can be of type uint8, uint16, or double (that one is expected to be scaled between 0 and 1 - if not, please convert the stack to the uint types).
OUT	
imOut	A maximum-contrast image of the time-average of the stack.

SCUMS.processSingleActivation

Processes a single activation (action potential or calcium transient), returning its properties/features - this is mainly a helper function for SCUMS.analyseRegularPacing or SCUMS.analyseSinglePass, and the parameters it uses are the same as there.

SCUMS.smoothTrace

This function smooths (~denoises) a trace using Savitzky-Golay filter of a given width. It may also remove drift/trend in data using polynomial fitting.

IN	
tracePixel	The trace to be smoothed.
filterWidth	the width of the Savitzky-Golay filter (should be an odd number - if an even one is provided, one is addedto it).
varargin{1}	the optional parameter may contain the order of the polynomial used to detrend the signal (useful for removing drift etc.). Unlike standard methods which subtract the polynomial making the signal approximately zero-centered on the y-axis, here we re-add the average of the original trace to the zero-centered trace, so that the information

	on signal baseline is not lost.
	OUT
traceSmoothed	tracePixel after smoothing.
traceSmoothedDetrended	tracePixel after smoothing and baseline subtraction.

SCUMS.traceToStack

A minor helper function which converts a vector (either n-by-1 or 1-by-n) to a “fake” 3D stack of 1-by-1-by-n – in this way, it is the same format as a stack representing a video (albeit with 1x1 resolution) and it can be fed to analyseRegularPacing. One can therefore use the functionality of SCUMS with regards to baseline, amplitude, or duration features even on single traces (coming e.g. from electrophysiological recordings).

	IN
signalTrace	The trace to be converted into a "stack".
	OUT
imStack	The resulting stack.

External function

SCUMS.natsortfiles

This is an external library by Stephen Cobeldick which has been embedded within SCUMS. It is used when reading image series via SCUMS.readFolder, using natural sorting for the image files found in the folder (i.e., these are ordered 1.png, 2.png, 3.png,… rather than 1.png, 10.png, 11.png…,19.png, 2.png, 20.png,…).