CONTENTS

OVERVIEW

Code from: Akankshya Shradhanjali*, Brandon D. Riehl*, Bin Duan, Ruiguo Yang, Jung Yul Lim. Spatiotemporal characterizations of spontaneously beating cardiomyocytes with adaptive reference digital image correlation. Sci. Rep. In press. https://doi.org/10.1038/s41598-019-54768-w

We developed an Adaptive Reference-Digital Image Correlation (AR-DIC) method which extends DIC capabilities enabling robust, unbiased, and accurate kinematics and strain measurements of biological samples which lack clear reference frames. Further, innovative tissue mechanical characterization and data visualization may lead to standardized measures of tissue mechanical functioning for lab-grown tissues and in-vivo diagnosis (i.e. photoacoustic imaging, ultrasound speckle tracking, magnetic resonance elastography: MRE). Together the novel DIC methods and tissue characterizations provide researchers and clinicians non-invasive tools for mechanobiology assessment. We applied these concepts to a difficult-to-characterize spontaneously beating cardiomyocyte (CM) tissue model assessing the localization, synchronization, and development of CM beating.

CODE CONTENTS

1. AR-DIC Matlab toolbox
    a. Demo and documentation files
2. Accumulative SpatioTemporal map visualization in Processing
    a. Demo and example files
3. Synthetic Topography visualization in Tangram
    a. Python local HTTP server script
    b. Demo files

CODE AVAILABLILITY

- Github: https://github.com/TheLimLab/AR-DIC

LICENSE

GNU General Public License V 3.0

CITATION

Akankshya Shradhanjali, Brandon D. Riehl, Bin Duan, Ruiguo Yang, Jung Yul Lim. Spatiotemporal characterizations of spontaneously beating cardiomyocytes with adaptive reference digital image correlation. Sci. Rep. In press. https://doi.org/10.1038/s41598-019-54768-w

REPRODUCIBILITY

Download our contracting and non-contracting video samples (1.5GB, links in Supplementary Information). Set the folder paths and run the following scripts to reproduce our main results:
main_AR_DIC.m
main_AR_DIC_process.m
main_AR_DIC_plotting.m

SYSTEM REQUIREMENTS:

Hardware requirements (minimum):

- 2 processing cores at 2 GHz or greater
- 4GB RAM

Software requirements

Tested operating systems:

- Windows 7
- Windows 10
- MacOS Sierra

AR-DIC requirements:

- Matlab R2015a (v8.5) or higher is recommended. Tested on R2015a, R2018b.
- Matlab Image processing toolbox v9.2 or higher.
- ImageJ/FIJI 1.51 or later (https://fiji.sc/)
- MIJ (http://bigwww.epfl.ch/sage/soft/mij/)
- Particle Image Velocimetry ImageJ plugin (https://sites.google.com/site/qingzongtseng/piv)

ASTC Map requirements:

- Processing version 3.3.4 or later with PeasyCam v.202 library or later

Synthetic Topography in Tangram requirements:

- Tangram 0.11.7 or later. A complete simple demo is included with our code.
- HTTP server. We recommend using Python 3.6 or later with the provided http host script.

INSTALLATION GUIDE:

Only AR-DIC toolbox and ImageJ with MIJ and PIV plugin are required to run the AR-DIC methods. Processing is required for the ASTC map visualization. Tangram is required for synthetic topography visualization.

Adaptive Reference Digital Image Correlation

AR-DIC Matlab toolbox (Time required <5 min)

1. Acquire the AR-DIC toolbox (zip package or GitHub repository).
2. Either double click the toolbox file to install or unzip the .m files to a Matlab directory on the Matlab path. If installing manually be sure all folders and subfolders are added to the Matlab path.
3. Unzip and place supplied demo data in a convenient location, preferably on the Matlab path

ImageJ/FIJI

1. Obtain ImageJ/FIJI (https://fiji.sc/)
2. Unzip FIJI to desired location
3. Obtain Particle Image Velocimetry ImageJ plugin and javacv library files from: (https://sites.google.com/site/qingzongtseng/piv)
4. Place the Particle Image Velocimetry files in the FIJI plugin folder

MIJ for Matlab R2017b and later (Time required: 5 min):

For detailed help refer to the MIJ site (http://bigwww.epfl.ch/sage/soft/mij).

1. Download mij.jar from http://bigwww.epfl.ch/sage/soft/mij/
    a. Move mij.jar to Matlab java folder.
2. Update FIJI for use with java8 (fresh install may be easiest)
    a. Enable ImageJ-Matlab update site in FIJI (Help >Update> Manage update sites> check ImageJ-Matlab)
3. Copy ij-x.jar from imagej/fiji installation (where 'x' is a version number) to Matlab java folder
4. In Matlab use command addpath to add path of ImageJ-Matlab scripts in Fiji (For example: addpath C:\Users\username\Documents\Fiji.app\scripts)
5. In Matlab: add mij.jar to path: javaaddpath 'C:\Program Files\MATLAB\R2015a\java\mij.jar'
6. To avoid having to add the paths every time Matlab is initialized it may be beneficial to add the paths from step 4 and 5 in a startup.m file.

Depending on dataset input, increase Java heap memory size:

Home tab >Environment>Preferences>MATLAB > General > Java Heap Memory.

MIJ for Matlab R2015a to R2017a (Time required: 5-20 min):

MIJ takes advantage of the underlying Java framework in both Matlab and ImageJ. For this purpose both ImageJ and Matlab should run compatible versions of Java. If R2017b or later is used the Java8 configuration step is not necessary.

1. Download Java8 (e.g. jre-8u131-windows-x64.exe)
    a. Install Java8
2. Download mij.jar from http://bigwww.epfl.ch/sage/soft/mij/
    a. Move mij.jar to Matlab java folder.
3. Update FIJI for use with java8 (fresh install may be easiest)

    a. Enable ImageJ-Matlab update site in FIJI (Help >Update> Manage update sites> check ImageJ-Matlab)
4. Set or add system variable: MATLAB_JAVA (In Windows7: Right click on Computer>properties>advanced system settings>Advanced>Environment Variables>New system variable)
    a. Set variable value to path of parent folder of jr.jar (For example: C:\Program Files\Java\jre1.8.0_131)
5. Copy ij-x.jar from imagej/fiji installation (where 'x' is a version number) to Matlab java folder
6. In Matlab use command addpath to add path of ImageJ-Matlab scripts in Fiji (For example: addpath C:\Users\username\Documents\Fiji.app\scripts)
7. In Matlab: add mij.jar to path: javaaddpath 'C:\Program Files\MATLAB\R2015a\java\mij.jar'
8. To avoid having to add the paths every time Matlab is initialized it may be beneficial to add the paths from step 6 and 7 in a startup.m file.
9. Restart computer after install

At startup Matlab may print an error to the console caused by use of the unexpected Java version. This error message does not affect the program operation.

Depending on dataset input, increase Java heap memory size:
Home tab >Environment>Preferences>MATLAB > General > Java Heap Memory.

Accumulative Spatiotemporal Contraction (ASTC) Map

Processing Installation (Required time <5 min)

1. Obtain Processing 3.3.4 or later from https://processing.org/download/ and run installer.
2. Install PeasyCam library using processing menus: Sketch > Import > Library… > Add Library… and search for PeasyCam.

Synthetic Topography in Tangram

Tangram and http host (Required time <5 min)

1. We have provided a mini demo of Tangram, move this demo folder to desired location
    a. Alternatively, obtain full Tangram from (https://github.com/tangrams/simple-demo)
2. Obtain and install Python 3.6 or later (https://www.python.org/downloads/)
3. In our supplied python script, start_server.py, set folder path to Tangram demo folder:
        a. os.chdir("C:\\Users\\username\\Synthetic_topography\\Tangram_mini_demo")

DEMO

AR-DIC demo

The entire AR-DIC demo should take about 5 minutes to run.

1. Unzip the supplied demo data and set folder paths as directed in the demo scripts
2. Run demo scripts in the order:

Demo number	Demo script	Required demo data
(1)	AR_DIC_Demo.m	Demo_CM_contract.avi
(2)	AR_DIC_Demo_Process.m	AR_DIC_demo_output folder or saved output from Demo (1)
(3)	AR_DIC_Demo_Plotting.m	demo_data.mat and demo_frame.tif OR saved workspace output from Demo (2)

Expected demo output is provided in the Toolbox documentation under 'Examples' and at the end of this document in Appendix B.

ASTC Map

1. Run the ASTC_Demo script in Processing
2. Click and drag the map to rotate, use the middle mouse wheel to zoom. Click and drag the middle button to pan.

Synthetic Topography mini demo in Tangram

(Tested with Chrome version 69.0.3497.100 and Firefox 62.0)

1. Start the Python http server as set up in the installation instructions
2. Open web browser and navigate to http://localhost:8000/
3. Explore synthetic topography environment using the mouse

INSTRUCTIONS

Each object has usage and method documentation in the AR-DIC toolbox and in Appendix A. See the demo scripts in the AR-DIC Toolbox and Appendix B for example usage.

AR-DIC Toolbox overview

The AR-DIC Toolbox is divided into two main sections: (1) the Adaptive Reference section, and (2) the post-processing tools. The adaptive reference section operates procedurally and may be configured easily to a variety of scoring methods and frame selection logic. The post-processing tools are object oriented and provide for efficient processing, storage, and visualization of DIC results. The following sections briefly highlight these functions.

(1) AR-DIC loop

The main AR-DIC loop has three main processes:

1. Digital Image Correlation (DIC)
2. Scoring of current reference frame
3. Selection of reference frame

The AR-DIC toolbox accomplishes these processes succinctly:

1. The function **MIJ_DIC** controls DIC processing in ImageJ.
2. The current frame displacement data is obtained with **DIC_disp** and scored with **calcrefscore**
3. **Selection logic** is applied to select a new reference frame. Depending on the setup this can be a simple 'IF' statement as shown in the AR_DIC_Demo example.

(2) Object-oriented processing of AR-DIC data

Processing of AR-DIC data occurs in objects which perform computations, organize the data, and provide visualization functionality. The most basic processing occurs by calling a **vidobj** for each data set. The vidobj imports and organizes data from the AR-DIC loop and performs displacement and velocity field calculations. Other vidobj methods can then be called to perform more computationally expensive calculations such as morphology measurements, strain calculations, and organization into data trees. For example, to create a data tree first call the **mask_threshold** method to create a binary mask and then call the **construct_tree** method to create a data tree. These functions are demonstrated in the AR_DIC_Demo_Process script.

Object overview

The **vidobj** contains properties relevant to the data set and video as a whole. Creation of the **vidobj** automatically creates a **frameobj** for each input DIC frame. The **frameobj** contains data relevant to each frame such as the displacement field, strain fields, and area masks. Calling the **vidobj/construct_tree** method automatically creates a **region_tree** object which manages the data tree. The **region_tree** organizes each region into a **nodobj**, an object that points to data and parent/child nodes of the data tree. In general, most processing functions can be accessed through the **vidobj** methods. The hierarchy of objects is shown below:

1. **vidobj**
   i. **frameobj**
   ii. **region_tree**
      a. **nodobj** (has access to frameobj data)

Accumulative SpatioTemporal (ASTC) Map

The ASTC Map is exported from Matlab using the **export_tree_centroids** function. The resulting csv file can then be visualized in Processing. To visualize a new ASTC Map simply set the filename of the csv file in loadTable. The paths are visualized in gray by default. Adjust the vector "nums" to select which rows from the csv file should be plotted with distinct colors. See the Demo section in this manual and Appendix B for more instructions and expected output.

Synthetic Topography

Synthetic topography is viewed in the open source map rendering software Tangram. Map tiles may be exported using the **export_to_map** function in the AR-DIC toolbox. This function exports PNG map tiles in both grayscale and surface normal RGBα format. In this format the surface normal are encoded to the red, green, and blue color channels and the height data is encoded in the alpha channel. Copy these map tiles to the Tangram folder to view in the mapping software. The map view can be configured for multiple tiles and the scene style can be set by editing the index.html file and the YAML scene file. The YAML scene file may be edited to use different scene shaders, environment maps, and color schemes. Refer to the Tangram documentation for detailed instructions on configuring scene files: https://github.com/tangrams/tangram. To explore the synthetic topography start the HTTP server as directed in the Demo section and browse to http://localhost:8000/ in a web browser.

APPENDIX A: CODE DOCUMENTATION

Documentation from the AR-DIC Matlab toolbox is copied here for convenience.

Main scripts:

main_AR_DIC

Adaptive reference digital image correlation main script
Script sections:

i. User input
ii. Adaptive Reference Digital Image Correlation (loop)
    a. Digital image correlation
    b. Adaptive reference evaluation
iii. Reference score plotting

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also
main_AR_DIC_process main_AR_DIC_plotting MIJ_DIC DIC_disp

main_AR_DIC_process

Process output of AR-DIC method: 'main_AR_DIC.m'

Organizes data structures, obtains mechanics measurements, and builds a data tree from the results. Data can be explored manually by viewing the vidobjs in the variable viewer. The script main_AR_DIC_plotting handles plotting of the results.

Script sections:

i. User input
ii. Data processing
    a. Initial processing in vidobjs
    b. Secondary/specialized processing
iii. Data exploration

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also
main_AR_DIC main_AR_DIC_plotting

main_AR_DIC_plotting

Plot output of 'main_AR_DIC_process.m'

Run this script after main_AR_DIC_process.m to plot results.
Run in sections (Ctrl + Enter) to view plots
Basic plots:

- Displacement magnitude map
- Velocity vector field
- Maximum and minimum displacement vs. time
- Trace of velocity across frame through maximum velocity
- Contraction volume vs. time
- Contraction frequency heat map
- Contraction magnitude heat map
- Centroids of large contracting areas
- Accumulative spatial contraction map
- Smallest independent regions
- Raw strain vs. time at maximum strain location
- Principal strain vs. time at maximum strain location
- Raw strain field
- Principal strain field
- Power spectrum of reference image
- Synthetic topography image to export

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

main_AR_DIC main_AR_DIC_process

Objects:

frameobj

frameobj Is a handle class for managing and analyzing DIC mechanics data.
In general, frameobjs are intended to be created automatically through
the vidobj. Each frame of the vidobj is organized into a frameobj.

Usage: obj=frameobj(varargin)

Inputs: filename (string), scale (scalar), timestep (scalar), xprev (nxm
array of x-displacements) , yprev (nxm array of y-displacements,
refmethod (optional, currently unused input to set reference scoring
method).

frameobj Properties:
binary_mask - Binary mask from threshold
CC - Connected components in binary frame from bwconncomp
contraction_volume - Displacement magnitude times thresholded area
disp_mat - Displacement magnitude matrix
disp_max_index - Index of maximum displacement
disp_min_index - Index of minimum displacement
file - Input filename and paths
max_disp - Maximum displacement in frame
max_location_x - Maximum displacement x location
max_location_y - Maximum displacement y location
max_vel - Maximum velocity in frame
min_disp - Minimum displacement in frame
min_location_x - Minimum displacement x location
min_location_y - Minimum displacement y location
morphology_props - Morphology measurement structure
    note: Centroid, BoundingBox, and ConvexHull are stored unscaled.
percent_area - Percent of frame above displacement threshold
refscore - refscore is currently unused in frameobj
strain_tensor - Structure containing strain tensor
u_mat - x direction velocity matrix
v_mat - y direction velocity matrix
vel_mat - Velocity magnitude matrix
xdisp - x direction displacement matrix
ydisp - y direction displacement matrix

frameobj Methods:
calc_contraction_volume - Calculates contraction volume for frame and sets in object properties
calc_morphology - Process morphology data from binary masks and sets morphology properties of frameobj
calc_principal_strain - Calculate principal strains from raw strain data
calc_tensor - Calculates strain tensor and sets in object properties
find_mask - Calculate binary mask based on threshold
frameobj - Object constructor
get_max_morphology - Retrieves maximum value from morphology data

Object hierarchy:

1. vidobj
   i. **frameobj**
   ii. region_tree
      a. nodobj (has access to frameobj data)

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

vidobj nodeobj region_tree

frameobj Methods:

frameobj/calc_contraction_volume

calc_contraction_volume Calculates contraction volume for frame and sets in object properties

Usage: **calc_contraction_volume**(obj)

Method Details

Access	public
Sealed	false
Static	false

frameobj/calc_morphology

calc_morphology Process morphology data from binary masks and sets morphology properties of frameobj
vidobj/measure_morphology calls **calc_morphology** for each frame in
vidobj

Usage: **calc_morphology**(obj,scale,type)

Inputs: scale (scalar) is the pixel scale (e.g. px/um). type
(string) either 'box' or 'convex_hull'. Specifies if BoundingBox
or ConvexHull should be calculated by regionprops.

See also

vidobj/measure_morphology

Method Details

Access	public
Sealed	false
Static	false

frameobj/calc_principal_strain

calc_principal_strain Calculates principal strains from raw strain data.

Usage: out=**calc_principal_strain**(obj)

Returns: out, structure with fields s1, s2, theta, and max_eng_shear containing matrices of maximum principal strain, minimum principal strain, transform angle, and maximum engineering strain respectively.

Method Details

Access public
Sealed false
Static false

frameobj/calc_tensor

calc_tensor Calculates strain tensor and sets in object properties.

Usage: **calc_tensor**(obj,h,type)

Inputs: h (scalar) is spacing between sample points. type (string) determines calculation method: 'cauchy' or 'green_lagrangian'.

Method Details

Access public
Sealed false
Static false

frameobj/find_mask

find_mask Calculates binary mask based on threshold.
Values above the threshold are assigned a 1 and everything else is filled with 0.

Usage: **find_mask**(obj,threshold)

Input: threshold (scalar)

Method Details

Access public
Sealed false
Static false

frameobj/frameobj

frameobj constructor

Usage: obj=frameobj(varargin)

Input order: filename, scale, timestep, xprev(optional),
yprev(optional), method(optional). frameobj can be created with either
0 inputs, the first 3 inputs, or all inputs.

Returns: a new frameobj.

frameobj/get_max_morphology

get_max_morphology Retrieves maximum value of 'morph_prop' from morphology data.

Usage: max_prop=**get_max_morphology**(obj,morph_prop)

Input: morph_prop (string) specifies property to measure. Any of
the morphology structure parameters: 'Area', 'Perimeter',
'MajorAxisLength','MinorAxisLength','Eccentricity','Orientation',
'EquivDiameter','Centroid'.

Returns: max_prop (scalar), value of maximum morphology value.

Method Details

Access	public
Sealed	false
Static	false

frameobj local function:

frameobj>scale_morphology

scale_morphology Called automatically from calc_morphology.
Scales morphology measurements: Area, major axis length, minor axis length, equivalent diameter, and perimeter by scale.

Usage: scaled_st=scale_morphology(input_st,scale)

Inputs: input_st is 1xn output of regionprops function. scale (scalar) defines scale to correct morphology measurements.

Returns: scaled_st, 1xn array of scaled morphology values

See also:
frameobj/calc_morphology

nodeobj

nodeobj stores data and points to children and parent nodes.
Suitable for building data trees. Tracks both parent and child node to
facilitate top-down building from object region_tree. If constructed with
no inputs then ID is set to 0. nodeobj is called automatically from
region_tree to build a data tree.

Usage: obj=nodeobj(parent, data, ID)

Inputs: (variable number of inputs): parent (nodeobj), data (frameobj),
ID ([frame_number, morphology_index]).

nodeobj Properties:
data - Stores data here
children - Cell array pointing to children handles, if node has no children then cell array is empty
ID - Node identification, of form [frame_number, morphology_index]
parent - Points to parent node
parent_ID - Identification of parent

nodeobj Methods:
nodeobj - Object constructor. varargin order: parent, data, ID
trace_path - Traces path from the input node obj to root
plot_region - Plots binary mask region associated with node

Object hierarchy:
1. vidobj
 i. frameobj
 ii. region_tree
  a. **nodeobj** (has access to frameobj data)

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

region_tree frameobj vidobj

nodeobj Methods:

nodeobj/nodeobj

nodeobj Constructor.
nodeobj is intended to be called automatically from region_tree
which constructs a data tree and creates nodeobjs on the fly.

Usage: obj=nodeobj(varargin)

Input: varargin order: parent, data, ID.

Returns: a new nodeobj

nodeobj/trace_path

trace_path Traces path from the input node obj to root.

Usage: out_array=**trace_path**(obj,sorttype)

Input: sorttype (string) specifies which sort method to use.
Current option: 'time_sort'.

Returns: out_array, a 1xn cell array containing nodeobjs in path order from obj to root of data tree.

Method Details

Access	public
Sealed	false
Static	false

nodeobj/plot_region

plot_region Plots binary mask region associated with node.
Contracting regions are plotted in white on black background.

Usage: **plot_region**(obj,mag)

Input: mag (scalar) sets initial image magnification.

Method Details

Access	public
Sealed	false
Static	false

nodeobj local function:

nodeobj>sort_path_time

sort_path_time Sorts cell array of nodeobjs based on frame number (obj.ID(1)).
Function is called from trace_path to sort node path in ascending order based on time.

Usage: time_nodes= **sort_path_time** (path_cell)

Input: path_cell (1xn cell array) containing nodeobjs to be sorted.

Returns: sorted 1xn cell array of nodeobjs.

See also:
nodeobj/trace_path

region_tree

region_tree Constructs data tree by organizing segmented areas from vidobj.
Expects to use binary masks obtained from vidobj method mask_threshold.
Establishes parent/child relationship between all areas.

Usage: obj=**region_tree**(in_vidobj,type)

Input: in_vidobj is a vidobj, type (string) either 'box' or
'convex_hull'. Specifies method to use to check if child bounded by
parent region. 'box' may be significantly faster but may not deal with
complex shapes well. 'convex_hull' is more accurate but may be
significantly slower.

region_tree Properties:
leaf_index - (Dependent property) Stores index to every node in the **region_tree**
node_holder - Stores tree structure with linked nodes
root - For bookkeeping, root of tree is the video (a vidobj)

region_tree Methods:
ASC_map - Visualize region tree in either 2D or 3D
check_lineage - Finds parent of input node, assigns parent-child data, returns level of child node
get_node - Retrieves node(s) from **region_tree** at 2xn array node_index
region_tree - Object constructor

Object hierarchy:
1. vidobj
 i. frameobj
 ii. **region_tree**
  a. nodobj (has access to frameobj data)

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

nodeobj frameobj vidobj vidobj/mask_threshold

region_tree/ASC_map

ASC_map Visualizes accumulative spatial contraction map in either 2D or 3D.
The 3D option produces the Accumulative Spatial Contraction (ASC)
map.

Usage: **ASC_map**(obj,view_type,pause_time)

Inputs: view_type (string) can be '2D' or '3D' for 2D or 3D
visualization respectively. pause_time (scalar) and is for
animation purposes only and can be set to zero.

Method Details

Access	public
Sealed	false
Static	false

region_tree/check_lineage

check_lineage Given a region_tree object, finds parent of nodeobj test_node.
Also assigns parent-child data and returns level of child node.

Usage: [level]=**check_lineage**(obj,test_node,type)

Input: test_node, the nodeobj to find parents for, type (string)
either 'box' or 'convex_hull' which specifies which method to use
to check if the child is bounded by the parent region. 'box' may
be significantly faster but may not deal with complex shapes well.
'convex_hull' is more accurate but may be significantly slower.
Future releases will add more options.

Returns: level (scalar) of child node.

Method Details

Access public
Sealed false
Static false

region_tree/get_node

get_node Retrieves node(s) from region_tree at 2xn array node_index.

Usage: out=**get_node**(obj,node_index)

Input: node_index has the form [level_index;cell_index].

Returns: 1xn cell array containing specified nodeobjs.

Method Details

Access public
Sealed false
Static false

region_tree/region_tree

region_tree Constructor. Builds data tree from areas contained in vidobj. Establishes parent/child relationship between all areas. Expects to use binary masks obtained from vidobj method mask_threshold. A **region_tree** is constructed automatically when construct_tree is called on a vidobj.

Usage: obj=**region_tree**(in_vidobj,type)

Input: in_vidobj is a vidobj, type (string) either 'box' or 'convex_hull' specifies method to use to check if child bounded by parent region. 'box' may be significantly faster but may not deal with complex shapes well. 'convex_hull' is more accurate but may be significantly slower. Future releases will add more options.

Returns: a new **region_tree** object

region_tree local function:

region_tree>in_region

in_region Checks if centroid, cent, is bounded by region bound_region. Output true if centroid is in bounding box, otherwise output is false.

Usage: binary_out= **in_region** (bound_region,cent,type)

Input: bound_region is either a 1x4 array as returned from regionprops BoundingBox property or the convex hull returned by the regionprops ConvexHull property. BoundingBox format: [upperleft x-coordinate, upperleft y-coordinate, x-width, y-width]. cent (1x2 array) containing the [x,y] coordinates of the centroid to test. type (string) either 'box' or 'convex_hull'. Specifies method to use to check if child bounded by parent region. 'box' may be significantly faster but may not deal with complex shapes well. 'convex_hull' is more accurate but may be significantly slower. Future releases will add more options.

Returns: logical 'true' if centroid is bounded by region otherwise returns 'false'.

vidobj

vidobj is a handle class that processes and stores data from digital image correlation files.
vidobj uses stringtext and folderpath to open DIC output files
sequentially. vidobj creates a frameobj for each DIC file.

Usage: obj=vidobj(stringtext, folderpath, scale, timestep).

Inputs: stringtext (string) specifies the text to search for to input DIC
files. For example, for files from ImageJ PIV plugin set stringtext to
PIV3. vidobj sorts thse filenames in numerical order for processing.
folderpath (string) specifies the directory containing DIC files. scale
(scalar) specifies the pixel scale (e.g. px/um). timestep (scalar)
specifies the time between image frames.

vidobj Properties:
x_mat - x location matrix
y_mat - y location matrix
xdim - x dimension of frame
ydim - y dimension of frame
num_frame - Number of frames
scale - Scale of frame (e.g. px/um)
vect_space - Spacing of DIC sample vector
timestep - Time between frames
frame_holder - Cell array for holding frameobjs
max_disp - Maximum displacement from all frames
max_disp_frame - Frame number with maximum displacement
min_disp - Minimum displacement from all frames
min_disp_frame - Frame number with minimum displacement
max_velocity - Maximum velocity from all frames
max_velocity_frame - Frame with maximum velocity
tree - Tree data structure

vidobj Methods:
vidobj - Object constructor
Plotting methods
plot_contour - Creates either displacement or velocity contour plot
plot_quiver - Creates either displacement or velocity quiver plot
plot_threshold_contour - Creates contour plot of displacement values above specified threshold
heatmap_frequency - Creates heatmap of contraction frequency
heatmap_magnitude - Creates heatmap of contraction magnitude
surfplot - Visualize displacement in 3D plot using z axis to encode magnitude
plot_contraction_volume - Plots the contraction volume vs. time
plot_binary_mask - Animation formed by plotting each thresholded binary mask of vidobj
xy_quiver - Animates quiver plot with displacement in x direction plotted with blue arrows and displacement in y direction with red arrows
plot_centroid_areathreshold - Plots centroids for areas greater than specified area threshold
plot_max_disp_each_frame - Plots the maximum displacement in each frame vs. time
plot_disp_location - Plots displacement for each frame at location specified by index

Calculation methods
percent_area_disp_thresh - Thresholds displacement to obtain percent contracting area above threshold
mask_threshold - Obtains binary mask for contracting areas with displacements greater than threshold
contraction_volume - Calculates the contraction volume for each frame
measure_morphology - Calculates morphology properties by passing each frame to calc_morphology
calc_strain - Calculates strain tensor for all video frames
transform_strain - Transforms the raw strain data using the specified method
strain_to_cell - Collects the strain tensor from each frameobj and returns in a cell array
sort_morphology - Sorts the morphology measurements from largest to smallest
sort_all_areas - Sorts all areas in descending order
construct_tree - Passes vidobj to region_tree which constructs a data tree from the areas
get_disp - Obtains displacement at index for every frame
get_contraction_volume - Get contraction volume from all frames
get_max_velocity - Get maximum velocity from each frame

Object hierarchy:
1. **vidobj**
  i. frameobj
  ii. region_tree
    a. nodobj (has access to frameobj data)

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

frameobj nodeobj region_tree

vidobj Methods:

vidobj/vidobj

vidobj constructor

Usage: obj=vidobj(varargin)

Input: stringtext (string) specifies the text string to search for in DIC filenames. For example, from ImageJ PIV plugin set stringtext to '*PIV3*'. folderpath (string) specifies the directory containing DIC files. scale (scalar) specifies the pixel scale (e.g. px/um). timestep (scalar) specifies the time between image frames.

Returns: a new vidobj.

vidobj/plot_contour

plot_contour Creates contour plot of displacement or velocity from vidobj.

Usage: contour_frame=**plot_contour**(obj,select_string,ncont,mapstyle)

Inputs: select_string (string) specifies plotting of displacement
'disp' or velocity 'vel' data. ncont (scalar) specifies the number
of contours to use in the plot. mapstyle specifies the colormap to
plot with. Any of the Matlab color maps may be specified.

Returns: a 1xn Matlab movie frame structure with fields cdata and
colormap.

See also

plot_quiver

Method Details

Access	public
Sealed	false
Static	false

vidobj/plot_quiver

plot_quiver Creates quiver plot of displacement or velocity from vidobj.

Usage: quiver_frame=**plot_quiver**(obj,select_string,qscale,mapstyle,cscale)

Inputs: select_string (string) specifies plotting of displacement
'disp' or velocity 'vel' data. qscale (scalar) specifies the scale
of the quiver arrows in the plot. mapstyle specifies the colormap
to plot with. Any of the Matlab color maps may be specified.
cscale should either be string 'auto' for autoscaling the color
axis or vector [cmin,cmax] to define the min and max range of the
color axis.

Returns: a 1xn Matlab movie frame structure with fields cdata and
colormap.

See also

plot_contour quiver_mod

Method Details

Access	public
Sealed	false
Static	false

vidobj/plot_threshold_contour

plot_threshold_contour Creates contour plot of displacement values above threshold.
Prior to plotting the function mask_threshold must first be used
on the vidobj to create a binary mask.

Usage: threshold_frame=**plot_threshold_contour**(obj,ncont,mapstyle)

Input: ncont (scalar) specifies the number of contours to use in
the plot. mapstyle specifies the color map to plot with. Any of
the Matlab colormaps may be specified.

Returns: threshold_frame, a 1xn Matlab movie frame structure with
fields cdata and colormap.

Method Details

Access	public
Sealed	false
Static	false

vidobj/heatmap_frequency

heatmap_frequency Creates heatmap of contraction frequency from vidobj.
The thresholded binary masks are summed togetherr and normalized
by the number of frames.

Usage: figure_handle=**heatmap_frequency**(obj,ncont,mapstyle)

Inputs: ncont (scalar) specifies the number of contours to use in
the plot. mapstyle specifies the color map to use. Any of the
Matlab color maps may be used.

Returns: figure handle.

See also

heatmap_magnitude

Method Details

Access	public
Sealed	false
Static	false

vidobj/heatmap_magnitude

heatmap_magnitude Creates heatmap of contraction magnitude from vidobj.
The displacement matrices are summed together and normalized by
the number of frames.

Usage: figure_handle=**heatmap_magnitude**(obj,ncont,mapstyle)

Inputs: ncont (scalar) specifies the number of contours to use in
the plot. mapstyle specifies the color map to use. Any of the
Matlab color maps may be used.

Returns: figure handle.

See also

heatmap_frequency

Method Details

Access public

Sealed false

Static false

vidobj/surfplot

surfplot Visualizes displacement in 3D plot using z axis to encode magnitude

Usage: surf_frame=surfplot(obj,mapstyle)

Inputs: mapstyle specifies color map. Any of the Matlab color maps
may be used.

Returns: a 1xn Matlab movie frame structure with fields cdata and
color map.

Method Details

Access public

Sealed false

Static false

vidobj/plot_contraction_volume

plot_contraction_volume Plots the contraction volume against time for the vidobj.

Usage: fig_hand=**plot_contraction_volume**(obj)

Returns: figure handle.

See also

contraction_volume

Method Details

Access public

Sealed false

Static false

vidobj/plot_binary_mask

plot_binary_mask Animation formed by plotting each thresholded binary mask of vidobj.
Thresholded contraction is plotted as white on a black background.

Usage: video=**plot_binary_mask**(obj)

Returns: a 1xn Matlab movie frame structure with fields cdata and
color map.

Method Details

Access	public
Sealed	false
Static	false

vidobj/xy_quiver

xy_quiver Animates quiver plot of displacement components.
Displacement in the x direction is plotted with blue
arrows and displacement in y direction with red arrows.

Usage: video=**xy_quiver**(obj,qscale)

Inputs: qscale (scalar) specifies the quiver arrow scale.

Returns a 1xn Matlab movie frame structure with fields cdata and
color map.

See also

plot_quiver

Method Details

Access	public
Sealed	false
Static	false

vidobj/plot_centroid_areathreshold

plot_centroid_areathreshold Plots centroids for areas greater than area_threshold (scalar).
This helps to check the spread of the larger contracting areas.

Usage: fig_hand=**plot_centroid_areathreshold**(obj,area_threshold)

Inputs: area_threshold (scalar): areas above this threshold are
included in the plot.

Returns: figure handle.

Method Details

Access	public
Sealed	false
Static	false

vidobj/plot_max_disp_each_frame

plot_max_disp_each_frame Plots the maximum displacement in each frame.
Interpretation of this plot is limited since the trace is of the
maximum displacement and not of a single location in the frame.
Using the function plot_disp_location may be more beneficial.

Usage: fig_hand=**plot_max_disp_each_frame**(obj)

Returns: figure handle.

See also

plot_disp_location

Method Details

Access	public
Sealed	false
Static	false

vidobj/plot_disp_location

plot_disp_location Plots displacement for each frame at location specified by index.

Usage: disp=**plot_disp_location**(obj,index)

Input: index can either be string 'max' to plot at the global
maximum, 'min' to plot at the global minimum, or specify index as
vector [x,y] to plot at index location.

Returns 1xn vector of displacement specified by displacement.

Method Details

Access	public
Sealed	false
Static	false

vidobj/percent_area_disp_thresh

percent_area_disp_thresh Obtains percent contracting area above displacement threshold

Usage: area_vect=**percent_area_disp_thresh**(obj,threshold)

Inputs: threshold (scalar), the value to use for the displacement
threshold.

Returns: 1xn vector containing percent areas.

Method Details

Access	public
Sealed	false
Static	false

vidobj/mask_threshold

mask_threshold For each frame, obtains binary mask for contracting areas with displacements greater than threshold.
Calls find_mask in frameobj which sets each binary mask frame.

Usage: **mask_threshold**(obj,threshold)

Inputs: threshold (scalar), the displacement threshold to use.

See also

frameobj/find_mask

Method Details

Access	public
Sealed	false
Static	false

vidobj/contraction_volume

contraction_volume Loops through each frame and calculates the contraction volume.
Calls frameobj method which sets contraction volume in the
frameobj properties.

Usage: **contraction_volume**(obj)

See also

frameobj/calc_contraction_volume

Method Details

Access	public
Sealed	false
Static	false

vidobj/measure_morphology

measure_morphology Calculates morphology properties by passing each frame to calc_morphology, a frameobj method.
The morphology properties are stored in the frameobj "morphology_props" field. The function mask_threshold must be used first to create binary masks which **measure_morphology** can then act on.

Usage: Area_props=**measure_morphology**(obj,type)

Input: type (string) either 'box' or 'convex_hull'. Specifies if regionprops should calculate BoundingBox or ConvexHull for later use. If 'box' is specified then ConvexHull is not calculated. This operation can be computationally expensive.

Returns: cell array containing morphology data.

Method Details

Access public
Sealed false
Static false

vidobj/calc_strain

calc_strain Calculates strain tensor for all video frames.
Passes each frame to frameobj method calc_tensor. The strain values are stored in the frameobj field strain_tensor.

Usage: **calc_strain**(obj,type)

Inputs: type (string) either 'cauchy' or 'green_lagrangian' to use the cauchy or green-lagrangian calculation methods respectively.

See also
frameobj/calc_tensor

Method Details

Access public
Sealed false
Static false

vidobj/transform_strain

transform_strain Transforms the raw strain data using method specified by transform_type.

Usage: strain=**transform_strain**(obj,transform_type)

Inputs: transform_type (string), setting to 'principal' calculates the principal strain from raw strain data. The function calc_strain must be used first to create the raw strain data to transform. This function to be expanded in future releases with more transform options.

Returns: 1xn cell array containing strain structure.

See also

calc_strain

Method Details

Access	public
Sealed	false
Static	false

vidobj/strain_to_cell

strain_to_cell Collects the strain tensor field from each frameobj and returns the strain structure in a 1xn cell array.

Usage: strain=**strain_to_cell**(obj)

Returns: strain, a 1xn cell array containing strain structure

See also

calc_strain

Method Details

Access	public
Sealed	false
Static	false

vidobj/sort_morphology

sort_morphology Sorts the morphology measurement specified by morph_prop from each frame from largest to smallest.

Usage: [sorted_max,frame_index]=**sort_morphology**(obj,morph_prop)

Input: morph_prop (string) specifying the frameobj morphology property to sort: 'Area', 'Perimeter', 'MajorAxisLength', 'MinorAxisLength','Eccentricity', 'Orientation', 'EquivDiameter','Centroid', 'percent_area'.

Returns: sorted_max (1xn array) containing morphology values and frame_index, a 1xn vector containing the index to the frame of the morphology value.

See also

measure_morphology

Method Details

Access	public
Sealed	false
Static	false

vidobj/sort_all_areas

sort_all_areas Sorts all areas in descending order.
Region_tree calls this method to sort areas into data tree.

Usage: sorted=**sort_all_areas**(obj)

Returns: sorted nx3 array with each row containing: area, frame_index, morphology_index.

See also

region_tree

Method Details

Access	public
Sealed	false
Static	false

vidobj/construct_tree

construct_tree Passes vidobj to region_tree which constructs a data tree from the areas. Stores constructed tree in vidobj property tree.

Usage: **construct_tree**(obj,type)

Input: type is a string either 'box' or 'convex_hull'. Specifies method to use to check if child bounded by parent region. 'box' may be significantly faster but may not handle complex shapes well. 'convex_hull' is more accurate but may be significantly slower. Future releases will add more options.

See also

region_tree

Method Details

Access	public
Sealed	false
Static	false

vidobj/get_disp

get_disp Obtains displacement at index for every frame.

Usage: out=**get_disp**(obj,row_index,column_index)

Inputs: row_index (scalar), the row index to query, column_index (scalar), the column index to query.

Returns: 1xn array containing displacement at location specified by row_index and column_index.

Method Details

Access	public
Sealed	false
Static	false

vidobj/get_contraction_volume

get_contraction_volume Gets contraction volume from all frames.

Usage: out=**get_contraction_volume**(obj)

Returns 1xn array containing contraction volume for every frame.

See also

contraction_volume

Method Details

Access	public
Sealed	false
Static	false

vidobj/get_max_velocity

get_max_velocity Gets maximum velocity from each frame.

Usage: out=**get_max_velocity**(obj)

Returns: 1xn array containing maximum velocity magnitude for every frame.

Method Details

Access	public
Sealed	false
Static	false

Functions:

bpm_from_fft

bpm_from_fft Calculates BPM from output of fft_plot.

Usage: [BPM,f_index]=**bpm_from_fft**(P1,f)

Inputs: P1, a 1xn array from the power spectrum of fft_plot. f, a 1xn array containing the frequencies from fft_plot

Returns: vector of sorted BPMs in descending order, BPM, and indices to original frequency, f_index.

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also
fft_plot

calcrefscore

calcrefscore Calculates reference frame score from displacement matrix.

Usage: refscore=calcrefscore(disp_mat,refmethod)

Inputs: disp_mat, nxm displacement array. refmethod (string) specifies the scoring method. 'fro' uses the frobenius norm and 'ent' uses the entropy.

Returns: score value refscore (scalar).

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

DIC_disp

DIC_disp Calculates displacement magnitude from highest numerical DIC filename in directory.

Usage: dispmag=**DIC_disp**(stringtext)

Inputs: stringtext (string) specifies the text to search for to input DIC files.

Returns: dispmag (array) containing displacement values from most recent DIC iteration.

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also
MIJ_DIC vidobj readimagejPIV

export_to_map

export_to_map Transforms and tiles displacement frames.
The output format is suitable for viewing in open source map rendering software Tangram. The displacement is encoded in the surface normals of an RGB plot and height data is translated to the alpha channel.

Usage: [RGBmat,Alpha]=**export_to_map**(vidobj_holder,start_stop,num_col,high_alpha,low_alpha,tile_size,opt)

Input: vidobj_holder is a 1xn cell array which holds vidobjs to plot. The function loops through each vidobj and tiles the specified frames from each. start_stop is a 1xn cell array which defines the range of frames to tile. Each element of the cell array is of the format [start_index, stop_index]. num_col is a scalar which defines the number of columns to divide the specified frames into. num_col must be defined so that the number of rows times the number of columns equals the total number of frames. high_alpha is a scalar which defines the upper limit of the height map. low_alpha is a scalar which defines the lower range of the height map. Using the encoding scheme in Tangram sea level has an alpha value of 0.93. Mt. Diablo in CA, USA has an alpha height of 0.82. tile_size is a scalar which defines the size of the tile block. A standard Tangram tile is 256x256 pixels. The displacement matrix is scaled to fit the tile size. If the displacement frame is less than the tile_size then the block is padded with zeros. opt is a string, either 'maxmin' to plot only the maximum and minimum displacement frames or 'all' to plot the frames specified by start_stop. Returns: RGBmat an nxmx3 matrix containing the tiled displacements with all surface normal values scaled from 0 to 255. Alpha is an nxmx3 matrix containing the height data scaled to the alpha channel.

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

format_frame

export_tree_centroids

export_tree_centroids Exports centroids of all contracting regions to csv. Starting with the leaves, the tree is traversed from each leaf to the root.

Usage: **export_tree_centroids**(leaves,obj,filename)

Input: leaves, 1xn cell array containing leaves of tree with each leaf being a nodeobj. obj, vidobj filename, (string) output filename to save to.

Output:csv file. Each node is exported in form of x, y, t triplets. Each row in file is path from leaf to root of tree. For example a tree with 3 nodes: node1, node2, node3, with centroids (x1,y1), (x2,y2), (x3,y3) respectively a file may have the form:
x1,y1,t1,x2,y2,t2
x3,y3,t1,x2,y2,t2,x3,y3,t3

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

region_tree vidobj

fft_plot

fft_plot Takes FFT of in_vect and plots FFT vs frequency.

Usage: [P1,f]=**fft_plot**(in_vect,Fs)

Input: 1xn array in_vect, Fs (scalar) sampling frequency.

Returns: vectors power spectrum, P1, and frequency, f.

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

bpm_from_fft

format_frame

format_frame Prepares displacement frames for tiling in Tangram mapping software.

Usage: out_frame=**format_frame**(frame,tile_size,scale)

Input: frame is nxm displacement array. tile_size (scalar) sets pixel dimensions of frame. A standard Tangram tile size is 256x256 pixels. scale (scalar) sets scaling factor of frame. If needed, pads the frame with zeros to fill up to tile_size. Applies flipud to flip matrix vertically.

Returns: formatted displacement array.

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

export_to_map

get_max

get_max Finds maximum strain value for type of strain specified by max_type.

Usage: [max_val,max_frame,row_ind,col_ind]=**get_max**(frames,max_type)

Input: frames is 1xn cell array containing strain structure. max_type is string specifying which maximum to find.

max_type options: 'principal_1' (maximum principal strain), 'principal_2' (maximum of secondary principal strain), 'principal_max_shear' (maximum principal shear),'raw_x' (maximum strain in x/horizontal direction), 'raw_y' (maximum strain in y/vertical direction), 'raw_xy' (maximum raw shear strain). The input frames can be obtained from the functions strain_to_cell or transform_strain which are both methods of the vidobj.

Returns: maximum strain value max_val, for all frames including an index to the frame: max_frame, and the row and column indices row_ind, and col_ind.

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

get_min vidobj/strain_to_cell vidobj/transform_strain

get_min

get_min Finds minimum strain value for type of strain specified by min_type.

Usage: [min_val,min_frame,row_ind,col_ind]=**get_min**(frames,min_type)

Input: frames is a 1xn cell array containing strain structure. min_type is string specifying which minimum to find.

min_type options: 'principal_1' (minimum of primary principal strain), 'principal_2' (minimum principal strain), 'principal_min_shear' (minimum principal shear), 'raw_x' (minimum raw strain in x/horizontal direction, 'raw_y' (minimum raw strain in y/vertical direction), 'raw_xy' (minimum raw shear strain). The input frames can be obtained from the functions strain_to_cell or transform_strain which are both methods of the vidobj.

Returns: minimum strain value min_val, for all frames including an index to the frame: min_frame, and the row and column indices row_ind, and col_ind.

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

get_max vidobj/strain_to_cell vidobj/transform_strain

MIJ_DIC

MIJ_DIC Passes commands to ImageJ for performing digital image correlation

Usage: **MIJ_DIC**(folderpath,vidpath,refframe,currentframe,option_string)

Input: folderpath (string) path of folder to save DIC files to. vidpath (string) path to input video. refframe (scalar) video frame to use as reference frame. currentframe (scalar) current video frame to evaluate. option_string (string) containing setup parameters for ImageJ iterative PIV(Advanced) plugin. Example option_string:
'piv1=128 sw1=256 vs1=64 piv2=64 sw2=128 vs2=32 piv3=48 sw3=96 vs3=24 correlation=0.8'

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

readimagejPIV

play_animation

play_animation code is based almost line for line on Mathwork's example in 'Movie' Help document code for resizing animation frames and playing back with proper axis.

Usage: **play_animation**(Movie_structure,times,fps)

Inputs: Movie_structure is the Matlab movie object, times (scalar) specifies the number of times to play the movie. fps (scalar) specifies the frame rate.

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

plot_node_array

plot_node_array Plots centroids of nodes contained in leaf_array.

Usage: **plot_node_array**(leaf_array,obj)

Inputs: leaf_array is a 1xn cell array of nodeobj containing the leaf nodes. obj is the associated vidobj.

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

nodeobj

plot_power_spectrum

plot_power_spectrum Plots power spectrum of input_image.
Power spectrum plot by taking the 2D FFT, shifting the zero component to the middle and plotting with a log10 transform.

Usage: **plot_power_spectrum**(input_image,scale,caxis_lim)

Inputs: input_image is an nxmx3 array containing the image data. input_image may be obtained from function imread. scale (scalar) defines the pixel scale (for example pixels per micrometer). caxis_lim defines the color axis limits in the form [lower_limit upper_limit].

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

plot_strain

plot_strain Plots strain vs. time at specified row and column index.

Usage: **plot_strain**(frames,type,row_ind,col_ind,step)

When type is 'principal', the function plots the maximum strain, minimum strain, and maximum shear strain. When type is 'raw' the function plots strain in the horizontal (Exx), vertical (Eyy), and shear (Exy) directions.

Inputs: frames a 1xn cell array of strain structures, type (string) either 'principal' or 'raw' to plot principal or raw strains respectively. row_ind (scalar) row index of frame to plot, col_ind (scalar) column index of frame to plot.

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

vidobj/calc_strain vidobj/strain_to_cell vidobj/transform_strain

quiver_mod

quiver_mod Modifies quiver plot with color coding of arrows.
Output is same as quiver but with scaled color data.

Usage: q=**quiver_mod**(x_mat,y_mat,u_mat,v_mat,qscale,mapstyle,mincaxis,maxcaxis)

Input: x_mat,y_mat,u_mat,v_mat,qscale are same as in quiver function. Briefly, these are the matrices of x values, y values, x magnitudes, y magnitudes, and qscale (scalar) which is the quiver arrow scale. mapstyle specifies the color map to use. Any of the Matlab color maps may be specified. mincaxis (scalar) defines the lower value of the colormap. maxcaxis (scalar) defines the maximum value of the color axis.

Returns: quivergroup handle.

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

quiver

readimagejPIV

readimagejPIV Read data files from output of ImageJ PIV plugin.

Usage: [x_mat,y_mat,xdisp_mat, ydisp_mat, disp_mat, u_mat, v_mat, vel_mat]...
=**readimagejPIV**(filename, scale, timestep, xprev (optional), yprev (optional));

Input: filename (string) the input file to read, scale (scalar) defines the pixel scale e.g. px/um, timestep (scalar) defines the time between frames, xprev (optional input, nxm array) contains the previous x-displacement matrix, yprev (optional input, nxm array) contains the previous y-displacement matrix.

Returns: x_mat (nxm array) of x location values, y_mat (nxm array) of y location values, xdisp_mat (nxm array) of x displacements, ydisp_mat (nxm array) of y displacements, disp_mat nxm array of displacement magnitudes, u_mat: nxm array of velocity magnitude in x direction, v_mat: nxm array of velocity magnitude in y direction, vel_mat: nxm array of velocity magnitude

Format of txt files from ImageJ plugin PIV website manual: (https://sites.google.com/site/qingzongtseng/piv/tuto#post): x y ux1 uy1 mag1 ang1 p1 ux2 uy2 mag2 ang2 p2 ux0 uy0 mag0 flag (x, y) is the position of the vector (center of the interrogation window). ux1, uy1 are the x and y component of the vector (displacement) obtained from the 1st correlation peak. mag1 is the magnitude (norm) of the vector. ang1, is the angle between the current vector and the vector interpolated from previous PIV iteration. p1 is the correlation value of the 1st peak. ux2, uy2, mag2, ang2, p2 are the values for the vector obtained from the 2nd correlation peak. ux0, uy0, mag0 are the vector value at (x, y) interpolated from previous PIV iteration. flag is a column used for mark whether this vector value is interpolated (marked as 999) or switched between 1st and 2nd peak (marked as 21), or invalid (-1).

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

scalebar

scalebar Draws a scale bar rectangle over current image.

Useage: scalebar(xstart,ystart,W,H,color)

Input: xstart and ystart are scalars that define the horizontal and vertical coordinates for rectangle origin. W, H are scalars which define the width and height of the rectangle. color defines the scale bar color using any of the default Matlab facecolor options.

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

sort_nat by Douglas M. Schwarz in Matlab File Exchange

Demo scripts:

AR_DIC_Demo

AR_DIC_Demo Demo for Adaptive Reference Digital Image Correlation

This demo runs AR-DIC on the 14 frame example video (or whichever video
is specified by vidpath). Before running download the example video and
create a folder to contain the DIC output files. To setup simply set
folder paths: DIC_folder, folderpath_IJ, and vidpath.
To process the AR-DIC output run the AR_DIC_Demo_process.m script
To plot the processed results run AR_DIC_Demo_plotting.m script

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also
AR_DIC_Demo_process AR_DIC_Demo_plotting MIJ_DIC DIC_disp

AR_DIC_Demo_process

AR_DIC_Demo_process Demo to processes output of AR_DIC_Demo.m

This demo organizes data structures, obtains mechanics measurements, and
builds a data tree from the results. Data can be explored manually by
viewing the vidobj "contract" in the variable viewer. Run the script
AR_DIC_Demo_plotting to demo plotting functions.
To setup simply set DIC_folder to the path of the DIC results.

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also
AR_DIC_Demo AR_DIC_Demo_plotting

AR_DIC_Demo_plotting

AR_DIC_Demo_plotting Demo to plot output of AR_DIC_Demo_process.m

Run this script after AR_DIC_Demo_process.m to plot results.
Run in sections (Ctrl + Enter) to view plots
Basic plots in demo:
Displacement magnitude map
Velocity vector field
Contraction volume vs. time
Contraction frequency heat map
Contraction magnitude heat map
Accumulative spatial contraction map
Smallest independent regions
Raw strain vs. time at maximum strain location
Principal strain vs. time at maximum strain location
Raw strain field
Principal strain field
Displacement vs. time at maximum location, BPM calculation
Power spectrum of reference image
Synthetic topography image to export

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

See also

AR_DIC_Demo AR_DIC_Demo_process

Accumulative SpatioTemporal Contraction (ASTC) Map and Demo

ASTC_map.pde

Accumulative SpatioTemporal Contraction map
This code plots the ASTC map by importing xyt triplets from a csv file.
This csv file may be created from the output of export_tree_centroids in
the AR-DIC Toolbox in Matlab.
Modify the file name in loadTable to open other csv files.

All paths are plotted in gray except those specified in the vector "nums"
which are modified for specific colors.

Map controls:
Left click and drag to rotate
Right click and drag or roll center mouse button to zoom
Click and drag with center mouse button to pan

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

ASTC_demo.pde

Accumulative SpatioTemporal Contraction map demo
This demo plots the ASTC map for a subset of contracting regions.

The centroids of contracting regions are imported from xyt triplets from a csv file.
This csv file may be created from the output of export_tree_centroids in
the AR-DIC Toolbox in Matlab.
Modify the file name in loadTable to open other csv files.

All paths are plotted in gray except those specified in the vector "nums"
which are modified for specific colors.

Map controls:
Left click and drag to rotate
Right click and drag or roll center mouse button to zoom
Click and drag with center mouse button to pan

Adaptive Reference Digital Image Correlation v 1.0 2018
Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln

APPENDIX B: DEMO SCRIPTS WITH EXAMPLE OUTPUT

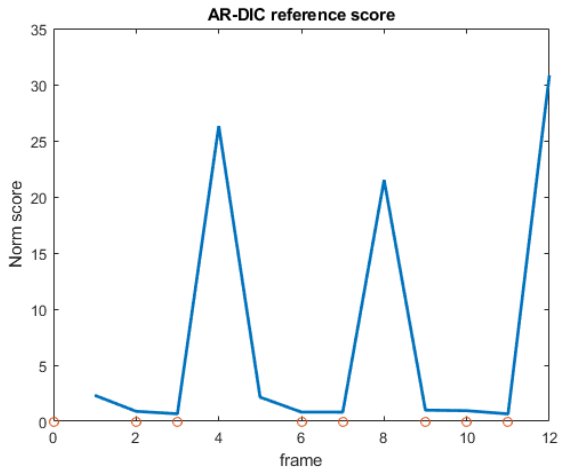
AR_DIC_DEMO

```
% AR_DIC_DEMO Demo for Adaptive Reference Digital Image Correlation
%
% This demo runs AR-DIC on the 14 frame example video (or whichever video
% is specified by vidpath). Before running download the example video and
% create a folder to contain the DIC output files. To setup simply set
% folder paths: DIC_folder, folderpath_IJ, and vidpath.
% To process the AR-DIC output run the AR_DIC_Demo_process.m script
% To plot the processed results run AR_DIC_Demo_plotting.m script
%
% Adaptive Reference Digital Image Correlation v 1.0 2018
% Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln
%
% See also:
% AR_DIC_DEMO_PROCESS
% AR_DIC_DEMO_PLOTTING
% MIJ_DIC
% DIC_disp

clear
clc

%Match path to operating system '\' on Windows and '/' on Unix-based systems
%set folder paths here:
DIC_folder='C:\Users\username\directory\DIC_output';%set DIC output folder
folderpath_IJ='C:\\Users\\username\\directory\\DIC_output\\';%DIC output folder formatted for ImageJ %placing \\
after folder important
vidpath='C:\\Users\\username\\directory\\Demo_CM_contract.avi'; %video to analyze (use double backslashes)

%%%%%%%%%%%%%%%%%%%%%%%%%%User options%%%%%%%%%%%%%%%%%%%%%
startframe=1; %set initial reference frame
endframe=13; %last video frame to analyze
method='fro'; %use frobenius norm method for adaptive reference comparison
norm_threshold=1.5;%frame becomes new reference if score is below norm threshold
stringtext='*PIV3*.txt'; %use only 3rd iteration
option_string='piv1=128 sw1=256 vs1=64 piv2=64 sw2=128 vs2=32 piv3=48 sw3=96 vs3=24 correlation=0.8';
%%%%%%%%%%%%%%%%%%%%%%%%%%End user options%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%Adaptive reference digital image correlation%%%%%%%%%%%%%%%%%%%
%initialize, start MIJI
current_directory=pwd; %get current directory
currentframe=startframe+1;%current frame index
refframe=startframe;%reference frame index
scorekeeper=[];%initialize to hold reference frame score
refstore=startframe; %store start frame as first reference
Miji(false); %to start Fiji without gui: Miji(false);

cd(DIC_folder) %change directory to get DIC output files
while currentframe<endframe+1 %Run AR-DIC while sufficient video frames exist
%DIC portion:
%DIC portion:
MIJ_DIC(folderpath_IJ,vidpath,refframe,currentframe,option_string);%perform DIC in ImageJ

%Adaptive Reference portion:
dispmag=DIC_disp(stringtext); %get DIC output from MIJI and format:
%calculate and store reference score:
```

```
refscore=calcrefscore(dispmag,method);   %calculate reference score of current frame

scorekeeper(end+1)=refscore; %store reference score
% If reference score is below norm threshold then set current frame as reference
if refscore < norm_threshold
   refframe=currentframe; %set current frame as reference
   refstore(end+1)=refframe;%Store reference frame
end
currentframe=currentframe+1; %increment to next video frame
end
MIJ.exit; %Exit ImageJ
cd(current_directory); %return to original directory

figure %plot reference score versus frame
plot(scorekeeper)%plot reference scores
hold on
scatter(refstore-1,zeros(1,length(refstore))); %mark reference frames
title('AR-DIC reference score')
xlabel('frame')
ylabel('Norm score')
hold off
%EOF
```

ImageJ instance ended cleanly

Reference score output



AR_DIC_DEMO_PROCESS

```
% AR_DIC_DEMO_PROCESS Demo to processes output of AR_DIC_Demo.m
%
% This demo organizes data structures, obtains mechanics measurements, and
% builds a data tree from the results. Data can be explored manually by
% viewing the vidobj "contract" in the variable viewer. Run the script
% AR_DIC_Demo_plotting to demo plotting functions.
% To setup simply set DIC_folder to the path of the DIC results.
%
% Adaptive Reference Digital Image Correlation v 1.0 2018
% Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln
%
% See also:
% AR_DIC_DEMO
% AR_DIC_DEMO_PLOTTING

%Match path to operating system '\' on Windows and '/' on Unix-based systems
%Set path here:
DIC_folder='AR_DIC\demo\AR_DIC_demo_output';

%---------------------- User Options: Set parameters----------------------
% Set processing options here. stringtext specifies the file name
% identifier to search for. For example, to process ImageJ output use
% '*PIV3*.txt'. pxscale (scalar) specifies the pixel to physical length
% scale (e.g. px/um). timestep (scalar) specifies the time between image frames.
% threshold (scalar) specifies the threshold above which displacement is considered
% relevant. strain_type (string) specifies the strain calculation method to
% use. Either 'green_lagrangian' or 'cauchy' to use the Green-Lagrangian or
% Cauchy strain calculation methods respectively. tree_opt (string) specifies
% the regioning method to use to build a data tree. Use 'box' for faster
% processing on uncomplicated shapes. Use 'convex_hull' for potentially more
% accurate regioning but at the expense of longer computational time.

stringtext='*PIV3*.txt'; %get only output files from 3rd iteration of plugin
pxscale=1.04; %set pixel micrometer scale px/um (10x objective)
timestep=0.2; %set time between frames (seconds)
threshold=0.14; %define displacement threshold (0.14 from Fukuda et. al)
strain_type='green_lagrangian'; % strain calculation to use. Options: 'green_lagrangian' or 'cauchy'
tree_opt='box';%'box' for faster processing, 'convex_hull' for higher accuracy
%-------------------------End of user Options------------------------------

%--------------------------- Data process---------------------------------
%Primary data processing occurs here:
%initial data processing occurs in vidobj, (displacement, velocity, etc.):
contract=vidobj(stringtext,DIC_folder, pxscale,timestep); %process contracting
data={contract};%combine into cell for iteration
for sets=1:length(data) %iterate for length of datasets
mask_threshold(data{sets},threshold) %set masks based on threshold
contraction_volume(data{sets})%calculate contraction volume
[~]=measure_morphology(data{sets},tree_opt); %calculate morphology parameters
construct_tree(data{sets},tree_opt); %build data tree of regions
[~]=data{sets}.tree.leaf_index; %get indices of all leaves (find smallest independent regions)
calc_strain(data{sets},strain_type);%calculate strains
end
```

```
%----------------------------Explore data---------------------------------
%user may wish to explore data here
leaves_s100=get_node(contract.tree,contract.tree.leaf_index); %get nodes from leaf indices
leaf_path=trace_path(leaves_s100{1},'time_sort'); %trace path from specified leaf to root
[sorted_percent_area,percent_area_frame_index]=sort_morphology(contract,'percent_area');
[sorted_max,frame_index]=sort_morphology(contract,'Area');
%calculate principal strains:
strains_principal_s2=transform_strain(contract,'principal');
strains_s2=strain_to_cell(contract);
%Find maximum, minimum strain
[max_val_s2,max_frame_s2,xind_s2,yind_s2]=get_max(strains_principal_s2,'principal_1');
secondary_max_strain_s2=strains_principal_s2{max_frame_s2}.s2(xind_s2,yind_s2);
[min_val_s2,min_frame_s2,xind_min_s2,yind_min_s2]=get_min(strains_principal_s2,'principal_2');
secondary_min_strain_s2=strains_principal_s2{min_frame_s2}.s1(xind_min_s2,yind_min_s2);

%display variable summary in figure:
test=evalc('contract');
test(1:90)=[];
test=['contract ', test];
figure;
ax_n=text(0,1,test);
ylim([0,2]);
set(ax_n,'Interpreter', 'none')
axis off

%EOF
```

Data set overview

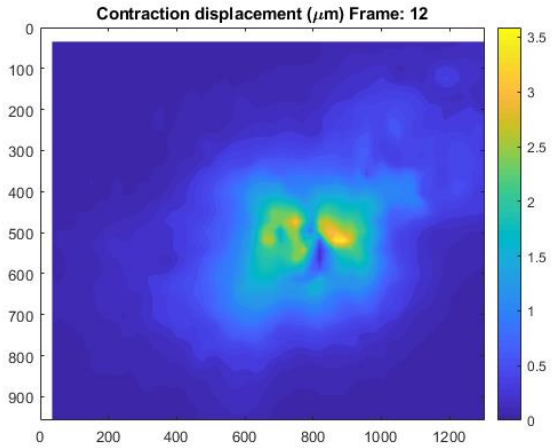
```
contract  properties:

            x_mat: [41×56 double]
            y_mat: [41×56 double]
             xdim: 1.3038e+03
             ydim: 957.6923
        num_frame: 12
            scale: 1.0400
       vect_space: 23.0769
         timestep: 0.2000
     frame_holder: {1×12 cell}
         max_disp: 3.5889
    max_disp_frame: 12
         min_disp: 9.3122e-05
    min_disp_frame: 2
     max_velocity: 17.9445
  max_velocity_frame: 12
             tree: [1×1 region_tree]
```

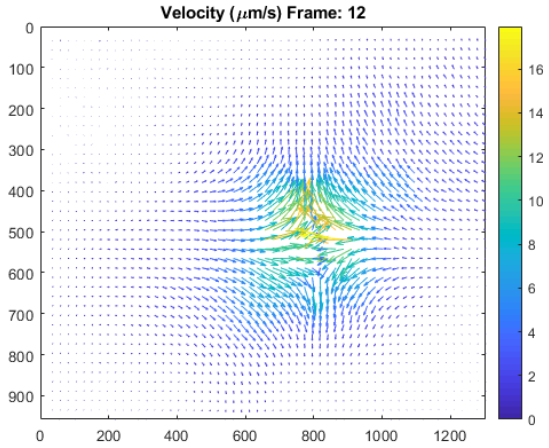
AR_DIC_DEMO_PLOTTING

```
% AR_DIC_DEMO_PLOTTING Demo to plot output of AR_DIC_Demo_process.m
%
% Run this script after AR_DIC_Demo_process.m to plot results.
% Run in sections (Ctrl + Enter) to view plots
% Basic plots in demo:
%   Displacement magnitude map
%   Velocity vector field
%   Contraction volume vs. time
%   Contraction frequency heat map
%   Contraction magnitude heat map
%   Accumulative spatial contraction map
%   Smallest independent regions
%   Raw strain vs. time at maximum strain location
%   Principal strain vs. time at maximum strain location
%   Raw strain field
%   Principal strain field
%   Displacement vs. time at maximum location, BPM calculation
%   Power spectrum of reference image
%   Synthetic topography image to export
%
% Adaptive Reference Digital Image Correlation v 1.0 2018
% Biomaterials and Mechanotransduction Lab University of Nebraska-Lincoln
%
% See also:
% AR_DIC_DEMO
% AR_DIC_DEMO_PROCESS
load('demo_data.mat')%load results from data file. Comment out if using newly
%calculated results.
%Set up plotting options:
ncont=50; %number of contour levels in plots
qscale=4; %quiver plot arrow scale

%set plot defaults, white background, linewidth of 2
[~]=get(0,'Factory');%Commands to set all figure backgrounds to white
set(0,'defaultfigurecolor',[1 1 1]);
set(0,'defaultlinelinewidth',2);
load('cmap_custom.mat')%load custom strain color map
```

Displacement contour, velocity quiver plot

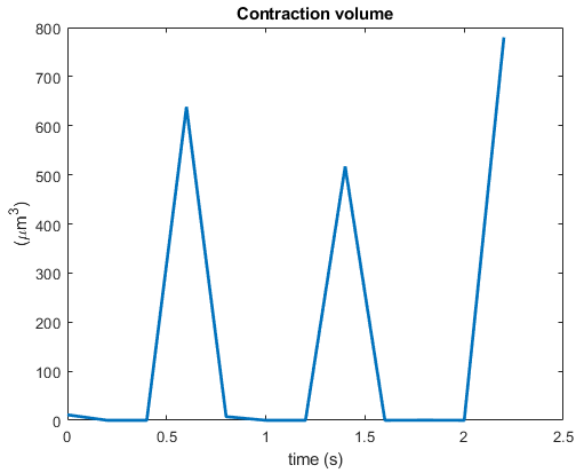
```
%Run in sections to explore plots:
%------------------Displacement, velocity--------------------------------
contour_frame=plot_contour(contract,'disp',ncont,'parula');
quiver_frame=plot_quiver(contract,'vel',qscale,'parula','auto');

%supplementary:
% disp_max=plot_disp_location(contract,'max');%displacement trace
% binary_video=plot_binary_mask(contract);
%------------------------------------------------------------------------
```





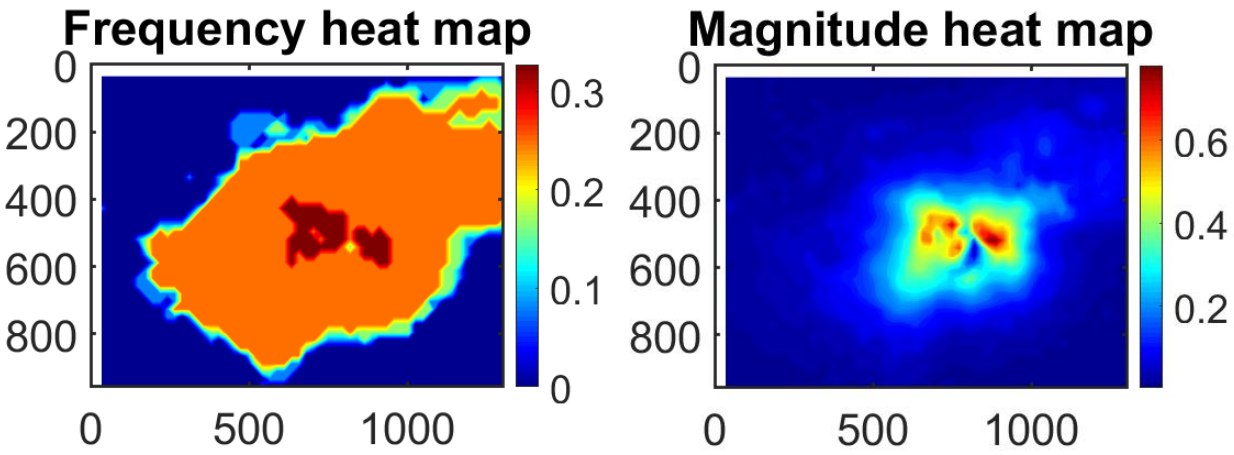
Contraction volume vs. time plot

```
%----------------------Contraction volume---------------------------------
plot_contraction_volume(contract);
```



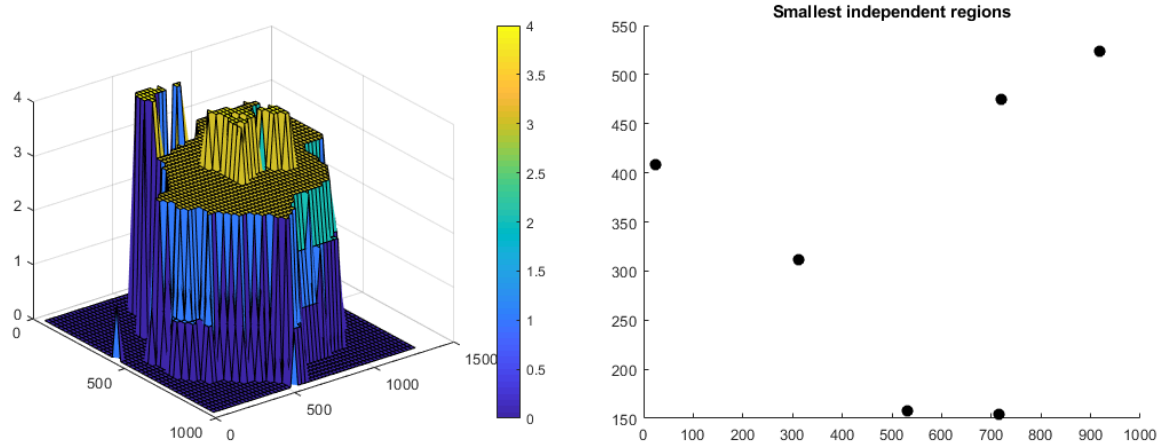
Frequency and magnitude heat maps

```
%----------------------------Heat maps-------------------------------------
figure_handle=heatmap_frequency(contract,ncont,'jet');
magnitude_heatmap=heatmap_magnitude(contract,ncont,'jet');
```



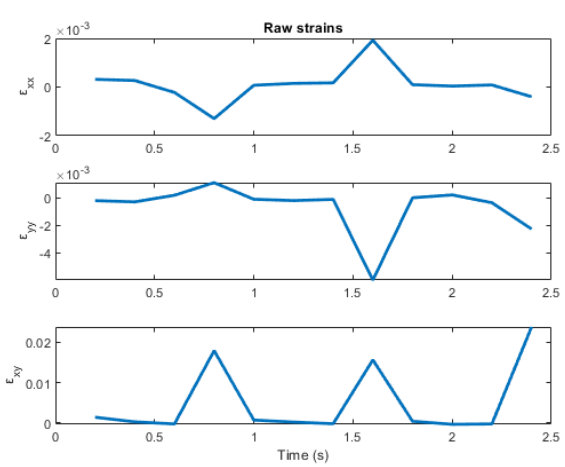
Accumulative spatial contraction map and smallest independent region plot

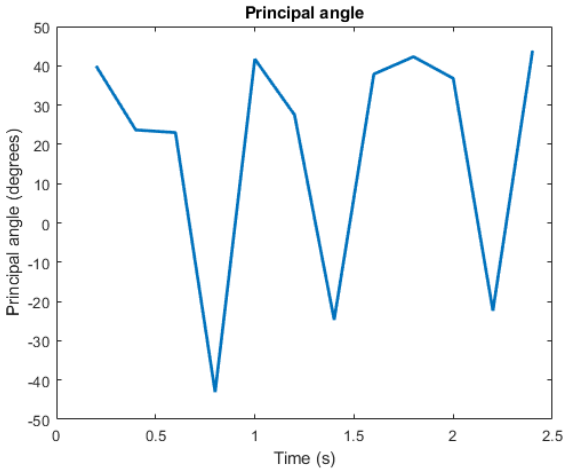
```
%Accumulative spatial contraction (ASC) map, smallest independent regions
ASC_map(contract.tree,'3D',0);%ASC map
plot_node_array(leaves_s100,contract);%smallest independent regions
%-------------------------------------------------------------------------
```

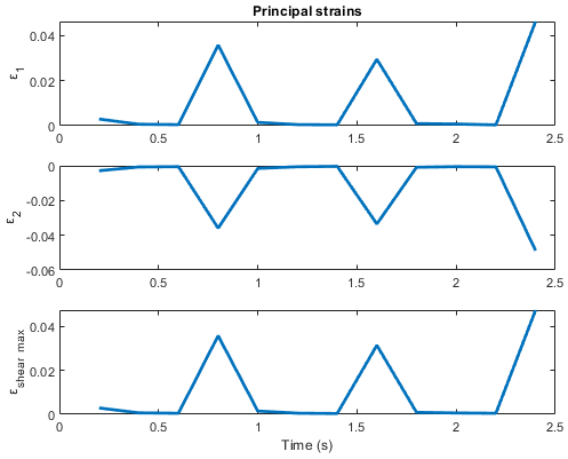


Strain trace plots

```
%-----------------------Strain trace plots--------------------------------
plot_strain(strains_s2,'raw',xind_s2,yind_s2,contract.timestep); %raw strain at maximum location
plot_strain(strains_principal_s2,'principal',xind_s2,yind_s2,contract.timestep);%principal strain at maximum location
```

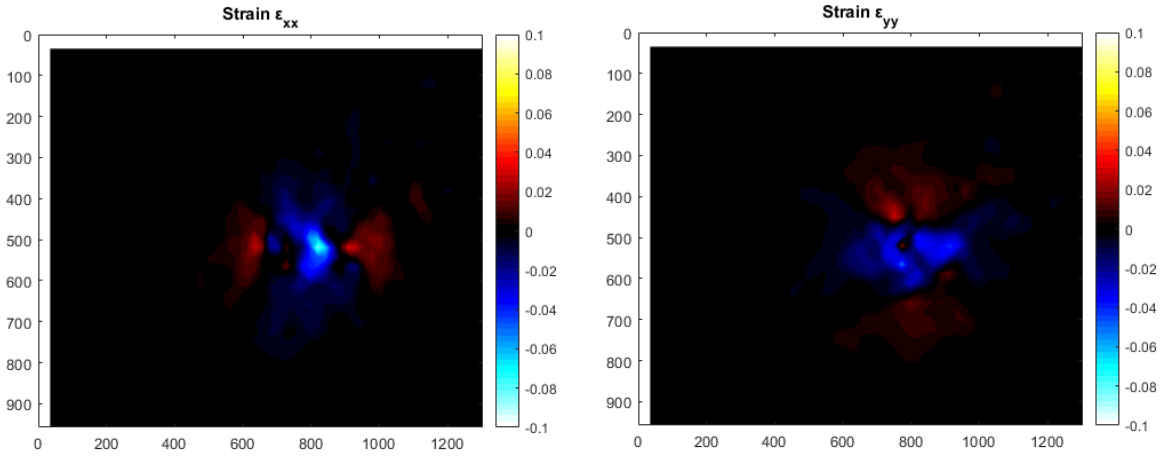


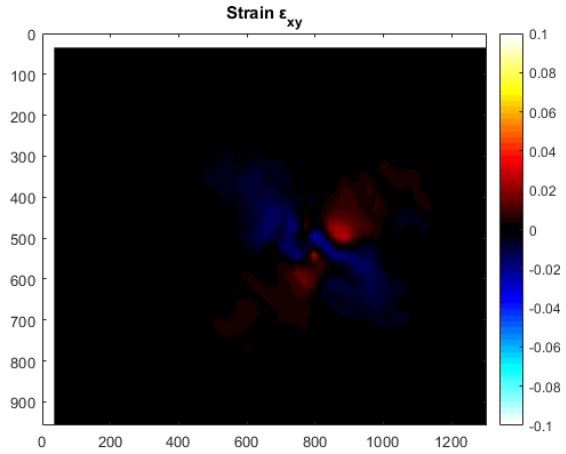




Raw strain maps

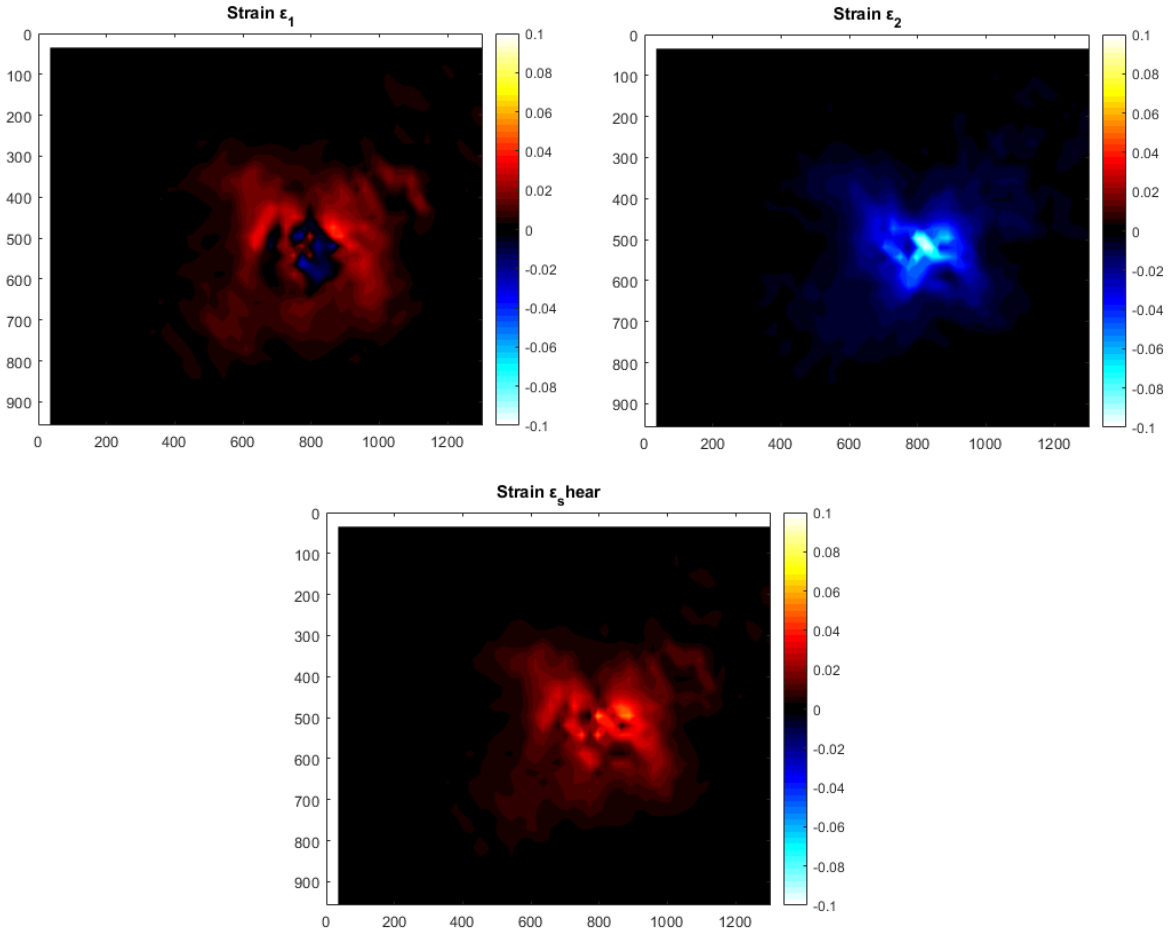
```
%-------------------------Raw Strain maps---------------------------------
% plot strain maps EXX EYY EXY
mapstyle2=cmap_comp_tens;
frame_s=max_frame_s2;
figure
contourf(contract.x_mat,contract.y_mat,strains_s2{1,frame_s}.Exx,ncont,'LineStyle','none');
title(['Strain ',char(949),'_x_x']);
axis ij
axis([0 contract.xdim 0 contract.ydim])
caxis([-0.1,0.1]) %caxis([-max_val_s2,max_val_s2])
colormap(mapstyle2)
colorbar;
figure
contourf(contract.x_mat,contract.y_mat,strains_s2{1,frame_s}.Eyy,ncont,'LineStyle','none');
title(['Strain ',char(949),'_y_y']);
axis ij
axis([0 contract.xdim 0 contract.ydim])
caxis([-0.1,0.1])
colormap(mapstyle2)
colorbar;
figure
contourf(contract.x_mat,contract.y_mat,strains_s2{1,frame_s}.Exy,ncont,'LineStyle','none');
title(['Strain ',char(949),'_x_y']);
axis ij
axis([0 contract.xdim 0 contract.ydim])
caxis([-0.1,0.1])
colormap(mapstyle2)
colorbar;
%-------------------------------------------------------------------------
```





Principal strain maps

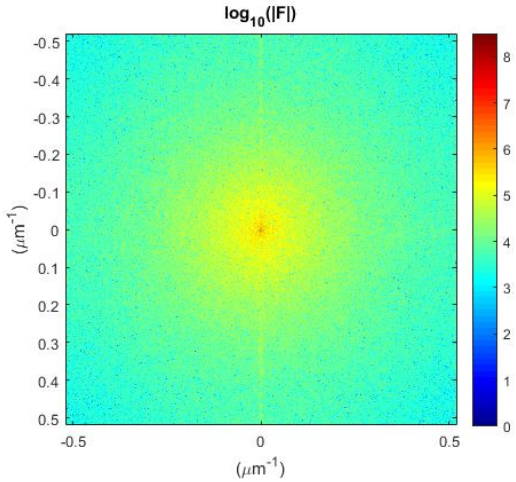
```
%--------------------Principal Strain maps--------------------------------
%strain map S1, S2
mapstyle2=cmap_comp_tens;% mapstyle2='hot';
frame_s=contract.max_disp_frame;
figure
contourf(contract.x_mat,contract.y_mat,strains_principal_s2{1,frame_s}.s1,ncont,'LineStyle','none');
title(['Strain ',char(949),'_1']);
axis ij
axis([0 contract.xdim 0 contract.ydim])
caxis([-0.1,0.1]) %caxis([-max_val_s2,max_val_s2])
colormap(mapstyle2)
colorbar;
figure
contourf(contract.x_mat,contract.y_mat,strains_principal_s2{1,frame_s}.s2,ncont,'LineStyle','none');
title(['Strain ',char(949),'_2']);
axis ij
axis([0 contract.xdim 0 contract.ydim])
caxis([-0.1,0.1])
colormap(mapstyle2)
colorbar;
figure
contourf(contract.x_mat,contract.y_mat,strains_principal_s2{1,frame_s}.max_eng_shear,ncont,'LineStyle','none');
title(['Strain ',char(949),'_shear']);
axis ij
axis([0 contract.xdim 0 contract.ydim])
caxis([-0.1,0.1])
colormap(mapstyle2)
colorbar;
```




```
%supplementary:
% figure
% for nn=1:contract.num_frame
% mapstyle2=cmap_comp_tens;% mapstyle2='hot';
% frame_s=nn;
% contourf(contract.x_mat,contract.y_mat,strains_s2{1,frame_s}.Exx,ncont,'LineStyle','none');
% title(['Strain ',char(949),'_x_x']);
% axis ij
% axis([0 contract.xdim 0 contract.ydim])
% caxis([-0.1,0.1]) %caxis([-max_val_s2,max_val_s2])
% colormap(mapstyle2)
% colorbar;
% strain_frame(nn)=getframe(gcf);
% end
%--------------------------------------------------------------------------
```

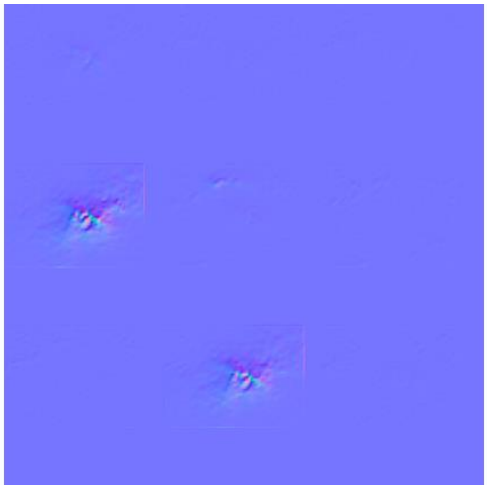
Reference power spectrum

```
%--------------------Plot reference power spectrum-----------
val_image=imread('demo_frame.tif'); %reference frame
caxis_lim=[0,8.5];
plot_power_spectrum(val_image,pxscale,caxis_lim);
%------------------------------------------------------------------------
```



Synthetic topography

```
%------------------intuitive topography map in Tangram--------
%input parameters
save_images=0;%to save images set save_images to 1.
bits=16; %image bitdepth to save
high_alpha=0.82;  %.82 (Mt. Diablo elevation)
low_alpha=.93;   %.93  (Sea level)
tile_size=256; %standard tile size of Tangram tile (256 x 256 px)
opt='all'; %'all' or 'maxmin'
row=6; %number of rows
vidobj_holder={contract};
frame_holder={[1,9]}; %frames to plot
[RGBmat,Alpha]=export_to_map(vidobj_holder,frame_holder,row,high_alpha,low_alpha,tile_size,opt);
if save_images==1
   imwrite(RGBmat,'testrgb.png','png','Alpha',Alpha,'BitDepth',bits);
   imwrite(rgb2gray(RGBmat),'testgray.png','png','BitDepth',bits);
end
imshow(RGBmat)
%-------------------------------------------------------------------------
```



ASTC file export

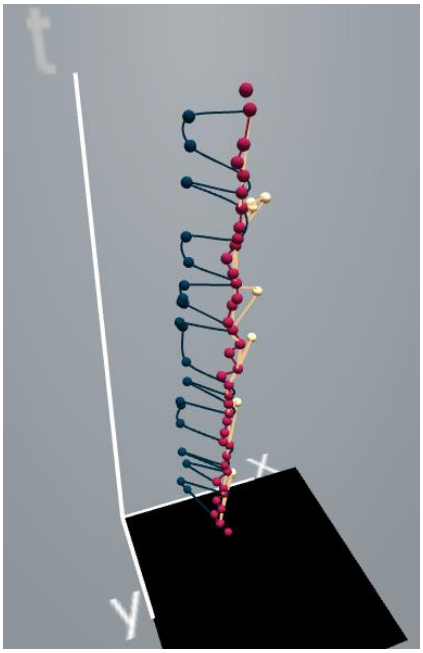
```
%Export tree centroids to csv file for ASTC map
 export_tree_centroids(leaves_s100,contract,'demo_ASTC_output');
%Example how to view saved displacement magnitude animation
%uncomment and run line:
% play_animation(contour_frame,1,3); %play animation 1 time at 3 fps
%Example how to view individual frame of saved velocity quiver animation
%uncomment and run line:
% play_animation(quiver_frame(4),1,1); %view frame 4 of quiver video
%----------------------------To save video-------------------------
% mov=binary_video;
% movie2avi(mov, 'export_video')
%-------------------------------------------------------------------------
%EOF
```

ASTC_demo

Simply open the ASTC_demo.pde file in Processing and run the script.
Use the mouse controls to examine the example map:

Left click and drag to rotate,
Right click and drag or roll center mouse button to zoom
Click and drag with center mouse button to pan

Expected output:



Synthetic Topography Demo

After configuring the local HTTP server by setting the folder path as described in the demo section, start the server and open a web browser. Navigate to http://localhost:8000/

Expected view with Python server in foreground: