Im2mesh Function List and Parameter

Table of Contents

Functions

Functions are sorted according to the workflow of Im2mesh package.

im2mesh

Generate triangular mesh based on grayscale segmented image using MESH2D mesh generator (Darren Engwirda)

```
[ vert, tria, tnum ] = im2mesh( im );   % this use default opt setting
[ vert, tria, tnum ] = im2mesh( im, opt );
```

im2meshBuiltIn

Generate triangular mesh based on grayscale segmented image using matlab built-in function generateMesh

```
[ vert, tria, tnum ] = im2meshBuiltIn( im );   % this use default setting
[ vert, tria, tnum ] = im2meshBuiltIn( im, opt );
```

plotMeshes

Plot meshes

```
plotMeshes( vert, tria, tnum );
```

tricost

Evaluate mesh quality

```
tricost(vert,tria,tnum);
```

im2Bounds

Extract exact polygonal boundaries from grayscale segmented image using getExactBounds.m

```
bounds = im2Bounds( im );
```

getExactBounds

Get the exact boundaries (polygonal) of binary image

```
Bs = getExactBounds( bw );
```

totalNumVertex

Calculate the total number of vertices in all polygonal boundaries

```
num_vert = totalNumVertex( bounds );
```

getCtrlPnts

Get control points in polygon boundaries

```
new_bounds = getCtrlPnts( bounds, tf_avoid_sharp_corner, size_im );
```

plotBounds

Plot polygon boundaries

```
plotBounds( bounds );
plotBounds( bounds, true );     % show starting and control points
```

totalNumCtrlPnt

Calculate the total number of control points in all polygonal boundaries. Each polygon has at least one ccontrol point (i.e., the starting vertex).

```
num_ctrlp = totalNumCtrlPnt( bounds );
```

smoothBounds

Smooth polygon boundaries using 2d Taubin Smoothing (taubinSmooth.m)

```
new_bounds = smoothBounds( bounds, lambda, mu, iters, ...
                           threshold_num_turning, threshold_num_vert );
```

smoothBoundsCCMA

Smooth polygon boundaries using CCMA smoothing algorithm (CCMA.m)

```
new_bounds = smoothBoundsCCMA( bounds, w_ma, w_cc, ...
                           threshold_num_turning, threshold_num_vert );
```

simplifyBounds

Simplify polygon boundaries using Douglas–Peucker algorithm (dpsimplify.m)

```
new_bounds = simplifyBounds( bounds, tolerance, threshold_num_vert )
```

delZeroAreaPoly

Delete polygon with zero area

```
bounds = delZeroAreaPoly( bounds );
```

getPolyNodeEdge

Get nodes and edges of polygonal boundary

```
[ poly_node, poly_edge ] = getPolyNodeEdge( bounds );
```

regroup

Organize cell array poly_node, poly_edge into array nodeU, edgeU & cell array part for MESH2D

```
[ nodeU, edgeU, part ] = regroup( poly_node, poly_edge );
```

poly2mesh

Generate meshes of parts defined by polygons using MESH2D mesh generator (Darren Engwirda)

```
[vert,tria,tnum] = poly2mesh( poly_node, poly_edge, hmax, mesh_kind, grad_limit );
```

poly2meshBuiltIn

generate meshes of parts defined by polygons using matlab built-in function generateMesh.

```
[ vert, tria, tnum ] = im2meshBuiltIn( im, opt );
```

getNodeEle

Get node coordinares and elements from mesh

```
[ nodecoor_list, nodecoor_cell, ele_cell ] = getNodeEle( vert, tria, tnum, ele_order );
```

printInp_multiPart

Print the nodes and elements into Inp file 'test_multi_parts.inp', test in software Abaqus. Each phase corresponds to one part in Abaqus.

```
printInp_multiPart( nodecoor_cell, ele_cell, ele_type, precision_nodecoor );
```

printInp_multiSect

Print the nodes and elements into Inp file 'test_multi_sections.inp', test in software Abaqus. One part with multiple sections. Each phase corresponds to one section in Abaqus.

```
printInp_multiSect( nodecoor_list, ele_cell, ele_type, precision_nodecoor );
```

printBdf

Print the nodes and elements into Inp file 'test.bdf'

```
printBdf( nodecoor_list, ele_cell, precision_nodecoor );
```

printTria

Print nodes and elements into file 'test.node' & 'test.ele'. Only support triangular element with 3 nodes. Precision is number of digits behind decimal point, for node coordinates

```
printTria( vert, tria, tnum, precision_nodecoor );
```

getPixelPercent

calculate the area perccentage of each grayscale in image

```
percent_pixel = getPixelPercent( im );
```

getPolyShapePercent

calculate the area perccentage of each phase in polygonal boundaries

```
percent_polyarea = getPolyShapePercent( bounds );
```

bound2polyshape

Convert polygonal boundaries to a cell array of polyshape object

```
p = bound2polyshape( bounds );
```

Parameters

Parameters and their default values of function im2mesh

```
opt.tf_avoid_sharp_corner = false;
opt.lambda = 0.7;
opt.mu = -0.4;
opt.iters = 10;
opt.threshold_num_turning = 10;
opt.threshold_num_vert_Smo = 10;
opt.tolerance = 0.3;
opt.threshold_num_vert_Sim = 10;
opt.grad_limit = 0.25;
opt.hmax = 500;
opt.mesh_kind = 'delaunay';
opt.select_phase = [];
```

Parameters and their default values of function im2meshBuiltIn

```
opt.tf_avoid_sharp_corner = false;
opt.lambda = 0.7;
opt.mu = -0.4;
opt.iters = 10;
opt.threshold_num_turning = 10;
opt.threshold_num_vert_Smo = 10;
opt.tolerance = 0.3;
opt.threshold_num_vert_Sim = 10;
opt.hgrad = 1.25;
opt.hmax = 500;
opt.hmin = 1;
```

tf_avoid_sharp_corner

Type: boolean.

For getCtrlPnts.

Meaning: Whether to avoid sharp corner when simplifying polygon. If true, two extra control points will be added around one original control point to avoid sharp corner when simplifying polygon.

lambda

Type: Float. Range: 0 < Lambda < 1.

For smoothBounds.

Meaning: How far each node is moved toward the average position of its neighbours during every second iteration.

mu

Type: Float. Range: -1< Mu < 0.

For smoothBounds.

Meaning: How far each node is moved opposite the direction of the average position of its neighbours during every second iteration.

iters

Type: Integer. Range: ≥ 0.

For smoothBounds.

Meaning: Number of iterations in Taubin smoothing. If you don’t need polyline smoothing, set Iterations to 0.

threshold_num_turning

Type: Integer. Range: ≥ 0.

For smoothBounds.

Meaning: Threshold value for the number of turning points in a polyline. Only those polylines with number of turning points greater than this threshold will be smoothed.

threshold_num_vert_Smo

Type: Integer. Range: ≥ 0.

For smoothBounds.

Meaning: Threshold value for the number of vertices in a polyline. Only those polylines with number of vertices greater than this threshold will be smoothed.

tolerance

Type: Float. Range: ≥ 0.

For simplifyBounds.

Meaning: The maximum allowable deviation of a vertex from the simplified curve. It’s for Douglas-Peucker algorithm. If you don’t need to simplify polylines, set tolerance to 0.

threshold_num_vert_Sim

Type: Integer. Range: ≥ 0.

For simplifyBounds.

Meaning: Threshold value for the number of vertices in a polyline. Only those polylines with number of vertices greater than this threshold will be simplified.

grad_limit

Type: Float. Range: > 0. Typical value: 0.2 - 0.5.

For poly2mesh & MESH2D.

Meaning: Gradient-limit, a limit on the gradient of mesh-size function.

hmax

Type: Float. Range: > 0.

For poly2mesh & MESH2D.

Meaning: Maximum mesh edge lengths. This is an approximate upper bound on the mesh edge lengths.

mesh_kind

Value: 'delaunay' or 'delfront'

For poly2mesh & MESH2D.

Meaning: Meshing algorithm used to create mesh-size functions based on an estimate of the "local-feature-size" associated with a polygonal domain. 'delaunay' means Delaunay-refinement. 'delfront' means Frontal-Delaunay.

select_phase

Type: vector

Meaning: Select certain phases in image for meshing. If 'select_phase' is [], all the phases will be chosen.

'select_phase' is an index vector for sorted grayscales (ascending order) in an image. For example, an image with grayscales of 40, 90, 200, 240, 255. If u're interested in 40, 200, and 240, then set 'select_phase' as [1 3 4]. Those phases corresponding to grayscales of 40, 200, and 240 will be chosen to perform meshing.

hgrad

Type: Float. Range: 1 ≤ Mesh Growth Rate ≤ 2. Typical value: 1.2 - 1.5.

For poly2meshBuiltIn & generateMesh

Meaning: Mesh growth rate, the rate at which the mesh transitions between regions of different edge size.

hmin

Type: Float. Range: ≥ 0.

For poly2meshBuiltIn & generateMesh

Meaning: Min mesh edge length, an approximate lower bound on the mesh edge lengths.