eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

Industria Teknologiaren Ingeniaritzako Gradua

GRADU AMAIERAKO LANA

3D HISTOLOGIA, RGB-TIK 3D-RA: 3D INPRESIO BIDEZKO HISTOLOGIA OBJEKTU BATEN SORKUNTZA

ERANSKINAK

Ikaslea: Mantxola Akizu, Asier

Zuzendaria: Eguiraun Martinez, Harkaitz

Ikasturtea: 2018-2019

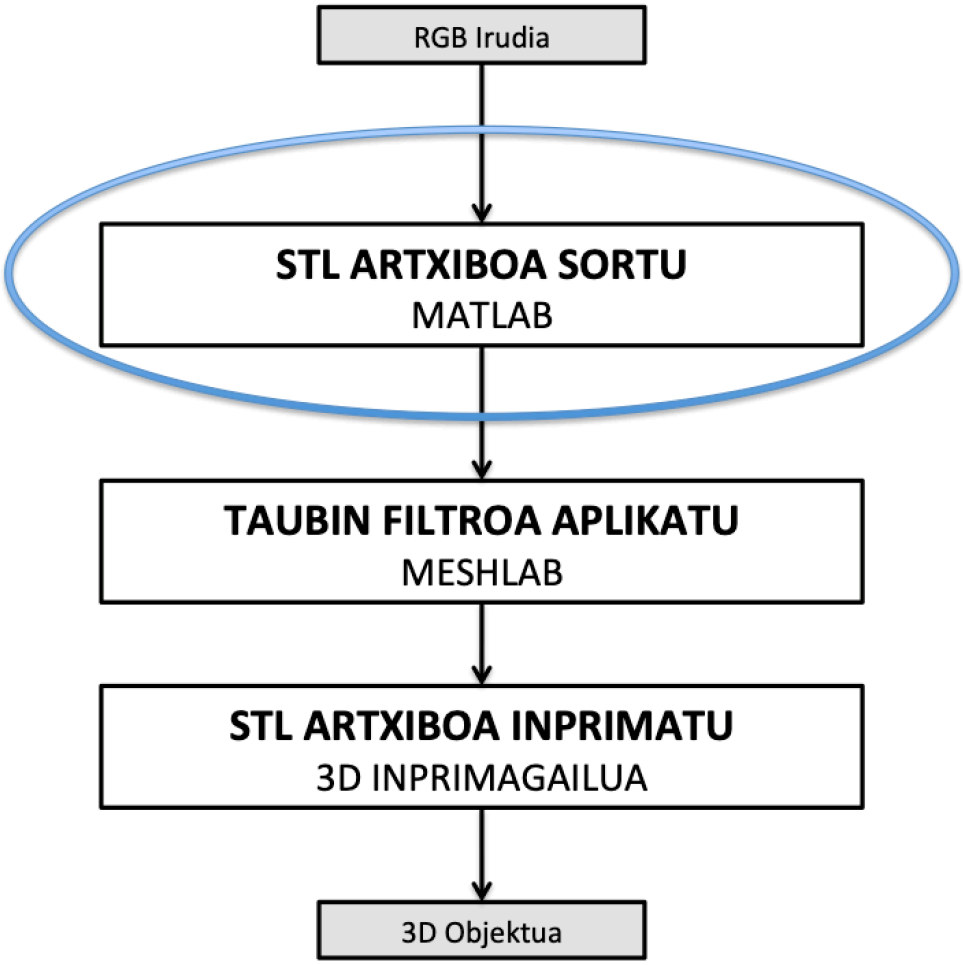
Data: Bilbo, 2019ko Ekainak 24

ERANSKINEN AURKIBIDEA

IRUDIEN AURKIBIDEA

1. Fluxu Diagrama

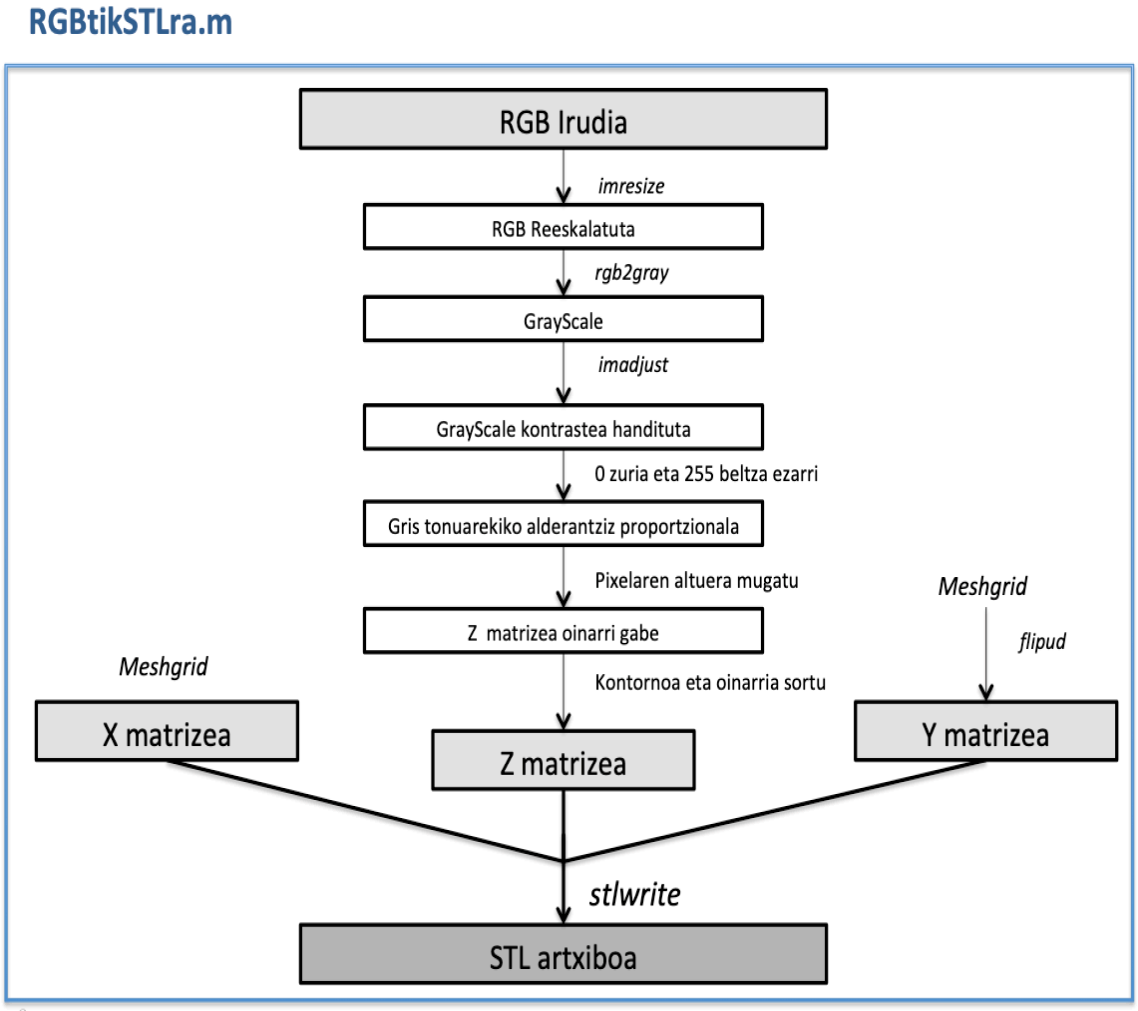
Eranskin hauetan burututako proiektua aurrera eramateko MatLab programan erabili diren script-ak azalduko dira. MatLab programa prozesuaren lehen pausuan erabiltzen da, RGB iruditik abiatuz STL artxiboaren sorkuntzarako, alegia. *1.1 Irudi*-ko fluxogramak jarraian azalduko diren script-ak metodoaren zein puntutan erabili diren erakusten du.



1.1 Irudia. *Metodoaren prozesuaren fluxograma. Script-ak non erabiltzen diren ikusi daiteke*

2. RGBtikSTLra.m

Jarraian MatLab programarako sortutako "*RGBtikSTLra.m*" script-a atxikitzen da, zeinek RGB motako irudi batetik abiatuta STL formatuko artxibo bat sortzen duen. *2.1 Irudia*-k RGBtikSTLra.m script-ak ematen dituen pauso guztiak adierazten ditu.



2.1 Irudia. *RGBtikSTLra.m script-aren eskema*

Script barruan egindako aipamen guztiak kursiba motako letrarekin eta kolore berdearekin adieraziko dira, zeinak % ikurra edukiko duten aurretik. RGBtikSTLra.m exekutatzen denean, pauso ezberdinetan lortutako guztia *figure* batzuen bidez modu bisualean erakutsiko da MatLab-en bertan.

```
%% RGBtikSTLra.m

clear all
close all

%% EXEKUTATU AURRETIK, BEHARREZKO DATUAK:
%   - Transformatu nahi dugun irudiaren izena (estentsio eta guzti)
irudiRGB aldagaian gorde
irudiRGB=imread('    ');
%   - Sortu nahi dugun STL artxiboaren izena zehaztu, filename
aldagaian gorde (.stl estentsioarekin)
filename='    ';


%%
% Reeskalaketa prozesua burutuko dugu lehendabizi. Horretarako
kargatu dugun irudiaren tamaina, ilarak eta zutabeak, kalkulatuko
ditugu.
f=size(irudiRGB,1);
c=size(irudiRGB,2);
e=f/c;c=200;f=e*c;f=round(f);
%Zutabe kopurua 200era mugatu eta hilara kopurua horren arabera
ezarri.
irudieskala=imresize(irudiRGB,[f c]);
% Irudiaren tamaina mugatuko dugu inprimatzeko posible den
dimentsio batera.

irudigray=rgb2gray(irudieskala);
% RGB motako irudia grayscale bihurtuko dugu MatLab-en
komandoarekin.
irudigrayad=imadjust(irudigray);
% Komando honekin grayscale-eko kontrastea handitu egiten da.
irudigrayd=double(irudigrayad);
% Egingo ditugun eragiketak direla eta 8bit motako data prezisio
bikoitzeko bihurtu.

figure()
subplot(1,3,1)
imshow(irudiRGB);
title('RGB irudia');
subplot(1,3,2)
imshow(irudigray);
title('GrayScale irudia');
subplot(1,3,3)
imshow(irudigrayad);
title('GrayScale kontrastea handituta');
% Figure honetan RGB, grayscale normala eta kontraste handikoa
erakutsiko ditugu.
```

```
% 3 matrize sortu behar ditugu, X eta Y espazioko koordenatuentzat,
eta Z (0-255)altuera baloreentzat. Horretarako aurretik
kalkulatutako ilara eta zutabe kopurua kontuan hartu.

[X,Y1]=meshgrid(1:c,1:f);
% Irudiaren tamaina kontuan hartuta, bi matrize kalkulatu
koordenatuentzat.
Y=flipud(Y1);
% Irudia egoki ikusi ahal izateko Y matrizea ardatz horizontal
batekiko irauli behar dugu.

z1=irudigrayd;
% Gris eskalako baloreak gordeko dituen matrizea.
z2=zeros(f,c)+255;
% 255 zenbakiz osatutako matrizea sortu.
Z=z2-z1;
% Pixel bakoitzaren altuera alderantziz proportzionala izateko gris
tonuarekiko.

figure()
subplot(2,1,1)
mesh(X,Y,z1)
title('Gris eskala: Beltza= 0 Zuria= 255');
colorbar;
subplot(2,1,2)
surf(X,Y,z1);
colorbar;
% Pixel bakoitzaren altueran zuzenki proportzionala izango balitz.

figure()
subplot(2,1,1)
mesh(X,Y,Z)
title('Gris eskala: Zuria= 0 Beltza= 255');
colorbar;
subplot(2,1,2)
surf(X,Y,Z);
colorbar;
% Figure 2 mesh eta surf irudiak sortu, gris eskalarekiko
alderantzizko proportzionala.

% Z altuera mugatzeko
e1=255/12.7;
% Z ardatzaren maximoa 0,5 inches ezartzeko (12,7 mm)
Ze=Z./e1;
% 0-255 gris eskalatik 0-12,7 tartera pasatzea nahi dugu,
inprimatzea posible izateko.

% Z matrizeari lodiera gehitzeko, 5mm-ko oinarri bat egin nahi
dugu.
L=zeros(f,c)+5;
% Lodieraren matrizea sortzen dugu, 5mm-koa.
```

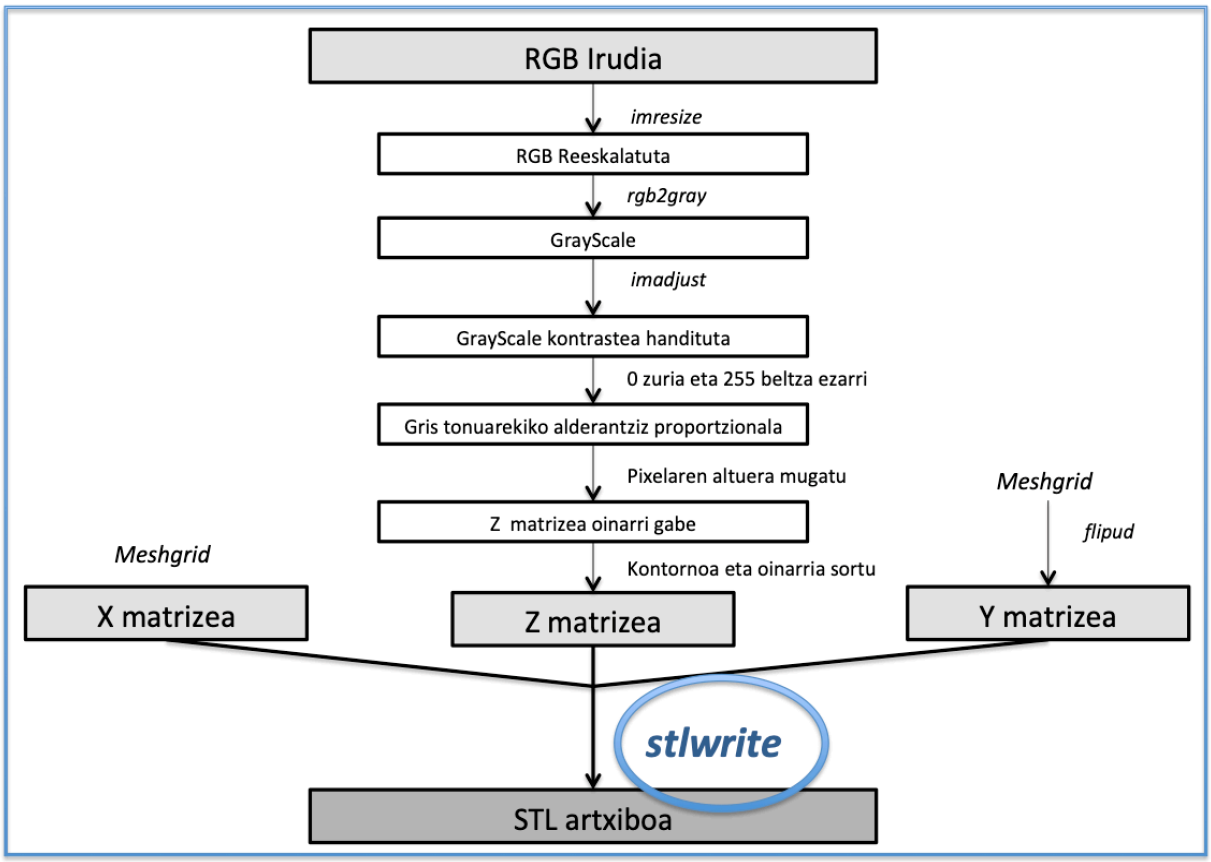
```
Zl=Ze+L;
% Ze matrizeari lodiera gehitzen diogu, base bat eduki dezan
irudiak.

 % Jarraian, inprimatzea posible izateko eta inpresorak artxiboa
ongi interpretatu dezan, 0 mailan dagoen kontorno bat sortuko dugu,
base bat eduki dezan.
Zl(1,:)=0;
Zl(f,:)=0;
Zl(:,1)=0;
Zl(:,c)=0;

figure()
subplot(2,1,1)
mesh(X,Y,Zl)
title('Zl bukaerako matrizea');
colorbar;
subplot(2,1,2)
surf(X,Y,Zl);
colorbar;

% Hasierako RGB iruditik abiatuta, 3 Matrize lortu ditugu. Orain
STL formatuan idatzi behar dugu. Horretarako stlwrite funtzioa
erabiliko da.

stlwrite(filename,X,Y,Zl)

% Funtzio honek zuzenean STL motako artxibo bat sortzen du.
```

3. Stlwrite.m

Stlwrite funtzioa Sven Holcombek sortua da (Sven Holcombe, 2011) eta eskuragarri dago MathWorks-eko File Exchange interneteko orrialdean. (https://www.mathworks.com/matlabcentral/fileexchange/20922-stlwrite-write-ascii-or-binary-stl-files). Ideia originala Bill McDonald-ek sortutako surf2stl funtzioan oinarritzen da, zenbait hobekuntza, exekuzio arloan zein triangulazioaren optimizazioan aplikatu zaizkiolarik.

Funtzio hau RGBtikSTLra.m script barruan erabili da, azaldutako prozesuan lortutako X, Y eta Z matrizeak dituelarik sarrerako aldagai gisa.

RGBtikSTLra.m



3.1 Irudia. *Stlwrite funtzioa RGBtikSTLra script barnean*

Matrize hauek jada egokituta daude *Stlwrite* funtzioak arazorik izan ez dezan hauek irakurtzean. Funtzioak hiru matrizeetako informazioa triangulatzen du Delaunay motako triangulazioa erabiliz. Honekin STL motako artxiboa lortzen da, non azalera geometria osatzen duten trianguluen bertize eta bektore normalen bidez deskribatzen den. Lortutako STL artxiboa prest dago MeshLab programan inportatu eta filtro ezberdinak aplikatzeko.

```
function stlwrite(filename, varargin)

%STLWRITE   Write STL file from patch or surface data.
%
%   STLWRITE(FILE, FV) writes a stereolithography (STL) file to
FILE for a
%   triangulated patch defined by FV (a structure with fields
'vertices'
%   and 'faces').
%
%   STLWRITE(FILE, FACES, VERTICES) takes faces and vertices
separately,
%   rather than in an FV struct
%
%   STLWRITE(FILE, X, Y, Z) creates an STL file from surface data
in X, Y,
%   and Z. STLWRITE triangulates this gridded data into a
triangulated
%   surface using triangulation options specified below. X, Y and Z
can be
%   two-dimensional arrays with the same size. If X and Y are
vectors with
%   length equal to SIZE(Z,2) and SIZE(Z,1), respectively, they are
passed
%   through MESHGRID to create gridded data. If X or Y are scalar
values,
%   they are used to specify the X and Y spacing between grid
points.
%
%   STLWRITE(...,'PropertyName',VALUE,'PropertyName',VALUE,...)
writes an
%   STL file using the following property values:
%
%   MODE          - File is written using 'binary' (default) or
'ascii'.
%
%   TITLE         - Header text (max 80 chars) written to the STL
file.
%
%   TRIANGULATION - When used with gridded data, TRIANGULATION is
either:
```

```
%                         'delaunay'  - (default) Delaunay
triangulation of X, Y
%                         'f'         - Forward slash division of
grid quads
%                         'b'         - Back slash division of
quadrilaterals
%                         'x'         - Cross division of
quadrilaterals
%                     Note that 'f', 'b', or 't' triangulations now
use an
%                     inbuilt version of FEX entry 28327, "mesh2tri".
%
%   FACECOLOR     - Single colour (1-by-3) or one-colour-per-face
(N-by-3)
%                     vector of RGB colours, for face/vertex input.
RGB range
%                     is 5 bits (0:31), stored in VisCAM/SolidView
format
%
(http://en.wikipedia.org/wiki/STL_(file_format)#Color_in_binary_STL
)
%
%   Example 1:
%     % Write binary STL from face/vertex data
%     tmpvol = false(20,20,20);      % Empty voxel volume
%     tmpvol(8:12,8:12,5:15) = 1;    % Turn some voxels on
%     fv = isosurface(~tmpvol, 0.5); % Make patch w. faces "out"
%     stlwrite('test.stl',fv)        % Save to binary .stl
%
%   Example 2:
%     % Write ascii STL from gridded data
%     [X,Y] = deal(1:40);             % Create grid reference
%     Z = peaks(40);                  % Create grid height
%     stlwrite('test.stl',X,Y,Z,'mode','ascii')
%
%   Example 3:
%     % Write binary STL with coloured faces
%     cVals = fv.vertices(fv.faces(:,1),3); % Colour by Z height.
%     cLims = [min(cVals) max(cVals)];      % Transform height
values
%     nCols = 255;  cMap = jet(nCols);      % onto an 8-bit colour
map
%     fColsDbl =
interp1(linspace(cLims(1),cLims(2),nCols),cMap,cVals);
%     fCols8bit = fColsDbl*255; % Pass cols in 8bit (0-255) RGB
triplets
%     stlwrite('testCol.stl',fv,'FaceColor',fCols8bit)
%   Original idea adapted from surf2stl by Bill McDonald. Huge
speed
%   improvements implemented by Oliver Woodford. Non-Delaunay
triangulation
```

```
%   of quadrilateral surface courtesy of Kevin Moerman. FaceColor
%   implementation by Grant Lohsen.
%
%   Author: Sven Holcombe, 11-24-11
% Check valid filename path
path = fileparts(filename);
if ~isempty(path) && ~exist(path,'dir')
    error('Directory "%s" does not exist.',path);
end
% Get faces, vertices, and user-defined options for writing
[faces, vertices, options] = parseInputs(varargin{:});
asciiMode = strcmp( options.mode ,'ascii');
% Create the facets
facets = single(vertices');
facets = reshape(facets(:,faces'), 3, 3, []);
% Compute their normals
V1 = squeeze(facets(:,2,:) - facets(:,1,:));
V2 = squeeze(facets(:,3,:) - facets(:,1,:));
normals = V1([2 3 1],:) .* V2([3 1 2],:) - V2([2 3 1],:) .* V1([3 1
2],:);
clear V1 V2
normals = bsxfun(@times, normals, 1 ./ sqrt(sum(normals .* normals,
1)));
facets = cat(2, reshape(normals, 3, 1, []), facets);
clear normals
% Open the file for writing
permissions = {'w','wb+'};
fid = fopen(filename, permissions{asciiMode+1});
if (fid == -1)
    error('stlwrite:cannotWriteFile', 'Unable to write to %s',
filename);
end
% Write the file contents
if asciiMode
    % Write HEADER
    fprintf(fid,'solid %s\r\n',options.title);
    % Write DATA
    fprintf(fid,[...
        'facet normal %.7E %.7E %.7E\r\n' ...
        'outer loop\r\n' ...
        'vertex %.7E %.7E %.7E\r\n' ...
        'vertex %.7E %.7E %.7E\r\n' ...
        'vertex %.7E %.7E %.7E\r\n' ...
        'endloop\r\n' ...
        'endfacet\r\n'], facets);
    % Write FOOTER
    fprintf(fid,'endsolid %s\r\n',options.title);

else % BINARY
    % Write HEADER
    fprintf(fid, '%-80s', options.title);             % Title
```

```
    fwrite(fid, size(facets, 3), 'uint32');           % Number of
facets
    % Write DATA
    % Add one uint16(0) to the end of each facet using a
typecasting trick
    facets = reshape(typecast(facets(:), 'uint16'), 12*2, []);
    % Set the last bit to 0 (default) or supplied RGB
    facets(end+1,:) = options.facecolor;
    fwrite(fid, facets, 'uint16');
end
% Close the file
fclose(fid);
fprintf('Wrote %d faces\n',size(faces, 2));
%% Input handling subfunctions
function [faces, vertices, options] = parseInputs(varargin)
% Determine input type
if isstruct(varargin{1}) % stlwrite('file', FVstruct, ...)
    if ~all(isfield(varargin{1},{'vertices','faces'}))
        error( 'Variable p must be a faces/vertices structure' );
    end
    faces = varargin{1}.faces;
    vertices = varargin{1}.vertices;
    options = parseOptions(varargin{2:end});

elseif isnumeric(varargin{1})
    firstNumInput = cellfun(@isnumeric,varargin);
    firstNumInput(find(~firstNumInput,1):end) = 0; % Only consider
numerical input PRIOR to the first non-numeric
    numericInputCnt = nnz(firstNumInput);

    options = parseOptions(varargin{numericInputCnt+1:end});
    switch numericInputCnt
        case 3 % stlwrite('file', X, Y, Z, ...)
            % Extract the matrix Z
            Z = varargin{3};

            % Convert scalar XY to vectors
            ZsizeXY = fliplr(size(Z));
            for i = 1:2
                if isscalar(varargin{i})
                    varargin{i} = (0:ZsizeXY(i)-1) * varargin{i};
                end
            end

            % Extract X and Y
            if isequal(size(Z), size(varargin{1}),
size(varargin{2}))
                % X,Y,Z were all provided as matrices
                [X,Y] = varargin{1:2};
            elseif numel(varargin{1})==ZsizeXY(1) &&
numel(varargin{2})==ZsizeXY(2)
```

```
                % Convert vector XY to meshgrid
                [X,Y] = meshgrid(varargin{1}, varargin{2});
            else
                error('stlwrite:badinput', 'Unable to resolve X and
Y variables');
            end

            % Convert to faces/vertices
            if strcmp(options.triangulation,'delaunay')
                faces = delaunay(X,Y);
                vertices = [X(:) Y(:) Z(:)];
            else
                if ~exist('mesh2tri','file')
                    error('stlwrite:missing', '"mesh2tri" is
required to convert X,Y,Z matrices to STL. It can be downloaded
from:\n%s\n',...

'http://www.mathworks.com/matlabcentral/fileexchange/28327')
                end
                [faces, vertices] = mesh2tri(X, Y, Z,
options.triangulation);
            end

        case 2 % stlwrite('file', FACES, VERTICES, ...)
            faces = varargin{1};
            vertices = varargin{2};

        otherwise
            error('stlwrite:badinput', 'Unable to resolve input
types.');
    end
end
if size(faces,2)~=3
    errorMsg = {
        sprintf('The FACES input array should hold triangular faces
(N x 3), but was detected as N x %d.',size(faces,2))
        'The STL format is for triangulated surfaces (i.e.,
surfaces made from 3-sided triangles).'
        'The Geom3d package
(https://www.mathworks.com/matlabcentral/fileexchange/24484-geom3d)
contains'
        'a "triangulateFaces" function which can be used convert
your faces into triangles.'
        };
    error('stlwrite:nonTriangles', '%s\n',errorMsg{:})
end
if ~isempty(options.facecolor) % Handle colour preparation
    facecolor = uint16(options.facecolor);
    %Set the Valid Color bit (bit 15)
    c0 = bitshift(ones(size(faces,1),1,'uint16'),15);
    %Red color (10:15), Blue color (5:9), Green color (0:4)
```

```
    c0 = bitor(bitshift(bitand(2^6-1, facecolor(:,1)),10),c0);
    c0 = bitor(bitshift(bitand(2^11-1, facecolor(:,2)),5),c0);
    c0 = bitor(bitand(2^6-1, facecolor(:,3)),c0);
    options.facecolor = c0;
else
    options.facecolor = 0;
end
function options = parseOptions(varargin)
IP = inputParser;
IP.addParamValue('mode', 'binary', @ischar)
IP.addParamValue('title', sprintf('Created by stlwrite.m
%s',datestr(now)), @ischar);
IP.addParamValue('triangulation', 'delaunay', @ischar);
IP.addParamValue('facecolor',[], @isnumeric)
IP.addParamValue('facecolour',[], @isnumeric)
IP.parse(varargin{:});
options = IP.Results;
if ~isempty(options.facecolour)
    options.facecolor = options.facecolour;
end
function [F,V]=mesh2tri(X,Y,Z,tri_type)
% function [F,V]=mesh2tri(X,Y,Z,tri_type)
%
% Available from
http://www.mathworks.com/matlabcentral/fileexchange/28327
% Included here for convenience. Many thanks to Kevin Mattheus
Moerman
% kevinmoerman@hotmail.com
% 15/07/2010
%-------------------------------------------------------------------
------
[J,I]=meshgrid(1:1:size(X,2)-1,1:1:size(X,1)-1);
switch tri_type
    case 'f'%Forward slash
        TRI_I=[I(:),I(:)+1,I(:)+1;  I(:),I(:),I(:)+1];
        TRI_J=[J(:),J(:)+1,J(:);   J(:),J(:)+1,J(:)+1];
        F = sub2ind(size(X),TRI_I,TRI_J);
    case 'b'%Back slash
        TRI_I=[I(:),I(:)+1,I(:);  I(:)+1,I(:)+1,I(:)];
        TRI_J=[J(:)+1,J(:),J(:);   J(:)+1,J(:),J(:)+1];
        F = sub2ind(size(X),TRI_I,TRI_J);
    case 'x'%Cross
        TRI_I=[I(:)+1,I(:);  I(:)+1,I(:)+1;  I(:),I(:)+1;
I(:),I(:)];
        TRI_J=[J(:),J(:);    J(:)+1,J(:);    J(:)+1,J(:)+1;
J(:),J(:)+1];
        IND=((numel(X)+1):numel(X)+prod(size(X)-1))';
        F = sub2ind(size(X),TRI_I,TRI_J);
        F(:,3)=repmat(IND,[4,1]);
        Fe_I=[I(:),I(:)+1,I(:)+1,I(:)];
Fe_J=[J(:),J(:),J(:)+1,J(:)+1];
```

```
        Fe = sub2ind(size(X),Fe_I,Fe_J);
        Xe=mean(X(Fe),2); Ye=mean(Y(Fe),2);  Ze=mean(Z(Fe),2);
        X=[X(:);Xe(:)]; Y=[Y(:);Ye(:)]; Z=[Z(:);Ze(:)];
end
V=[X(:),Y(:),Z(:)];
```