ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

D-BSSE
Department of Biosystems
Science and Engineering

Getting started with ImarisXT and IceImarisConnector

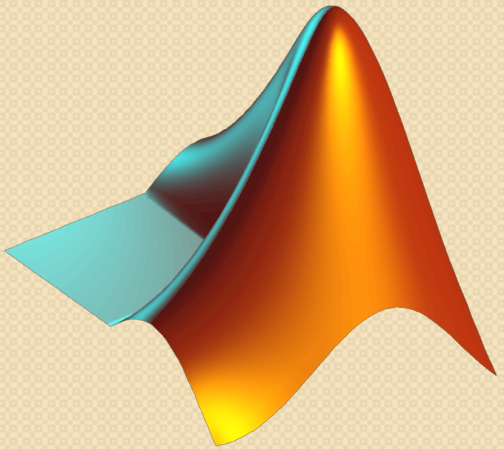
Extract from ImarisOPEN Launch

Belfast, 09.04.2013

```
function myFancyXTension(aImarisApplicationID)
% myFancyXTension is the next great XT function
% ...

% Set up connection between Imaris and MATLAB
conn = IceImarisConnector(aImarisApplicationID);

% Get the currently selected Spots object in the scene
spots = conn.getSurpassSelection('Spots');
if isempty(spots)
  errordlg('Please select a Spots object.');
  return;
end

% Get all Surfaces objects in the scene
surfaces = conn.getAllSurpassChildren(0, 'Surfaces');
if isempty(surfaces)
  errorglg('There are no Surfaces objects in the scene.');
  return;
end

% Get the data volume in row-major order
data = conn.getDataVolumeRM(0, 0);
```

IceImarisConnector is a simple commodity class that eases communication between Imaris and MATLAB using the ImarisXT interface.

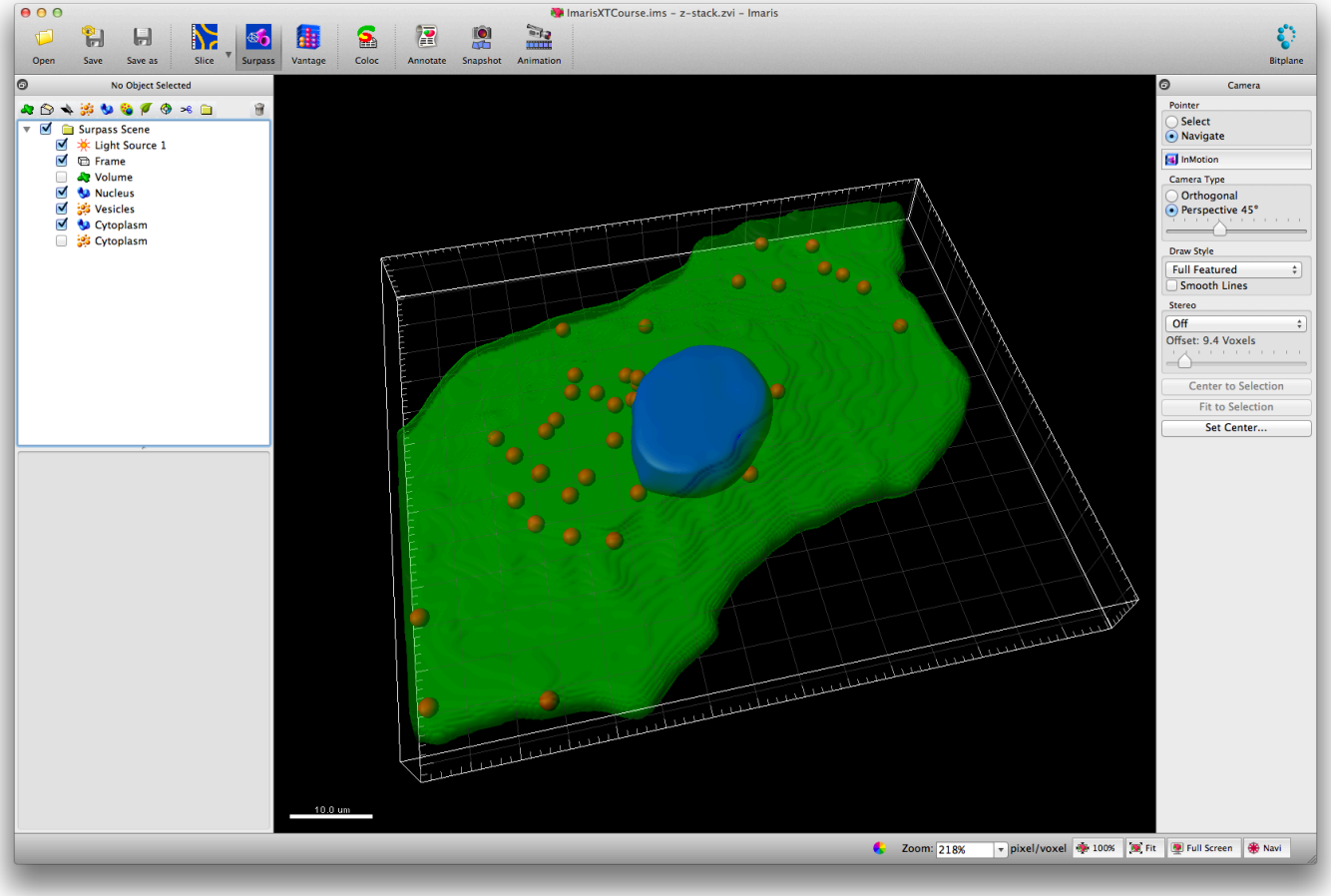
PART II
IceImarisConnector



http://www.scs2.net/next/index.php?id=110

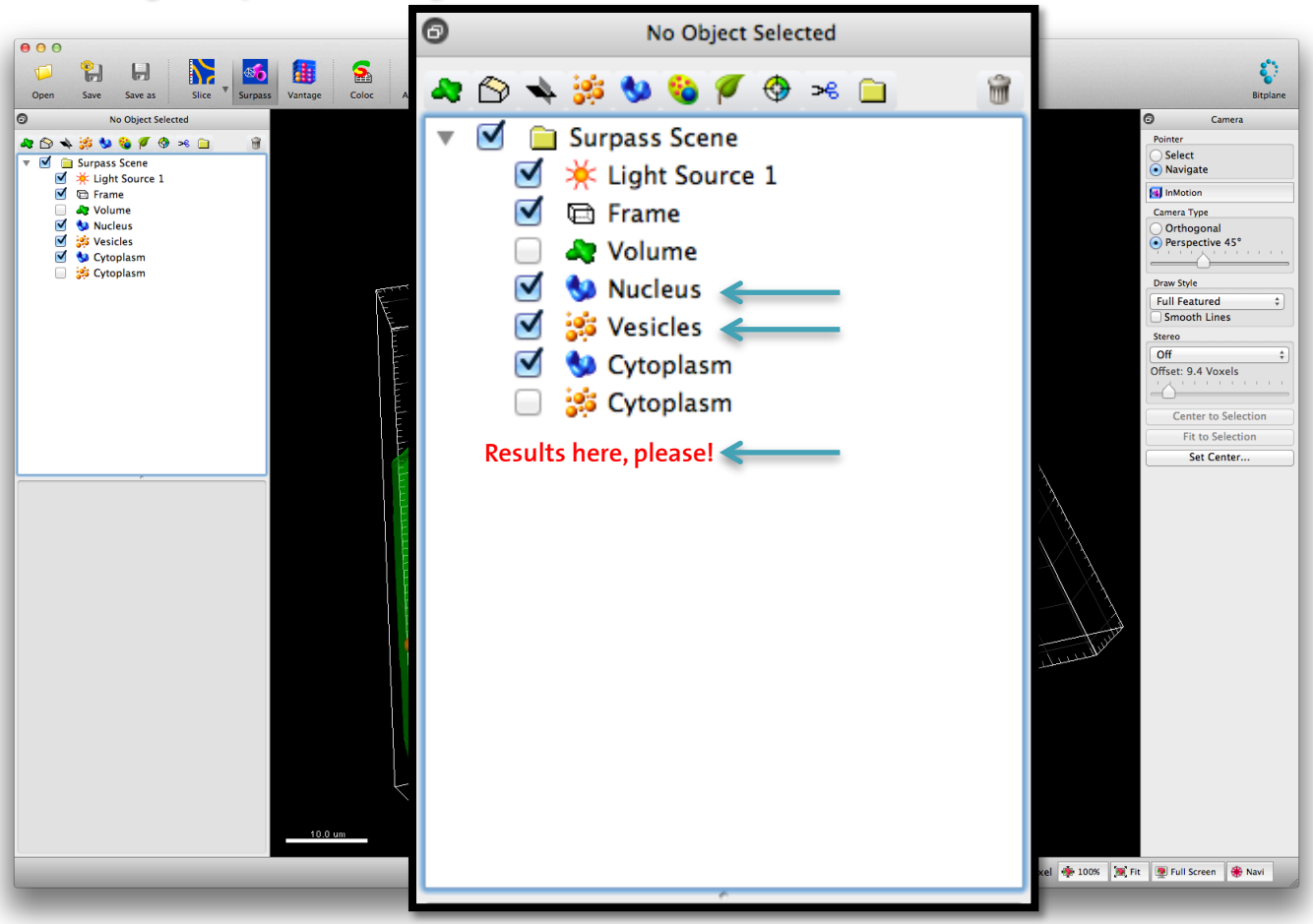
Getting started with ImarisXT / IceImarisConnector

Classify Spots by their distances from a Surface



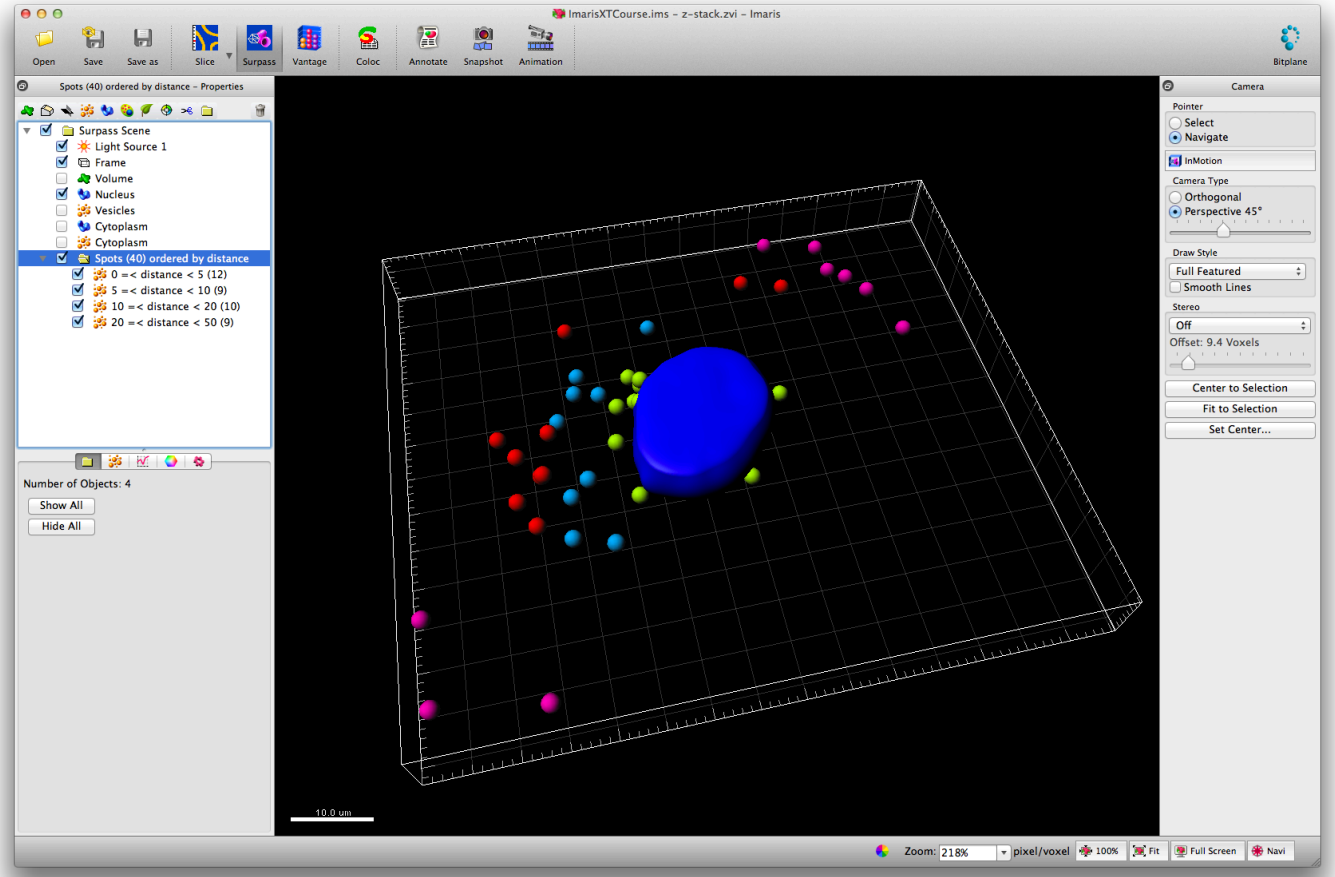
Getting started with ImarisXT / IceImarisConnector

Classify Spots by their distances from a Surface



Getting started with ImarisXT / IceImarisConnector

Classify Spots by their distances from a Surface

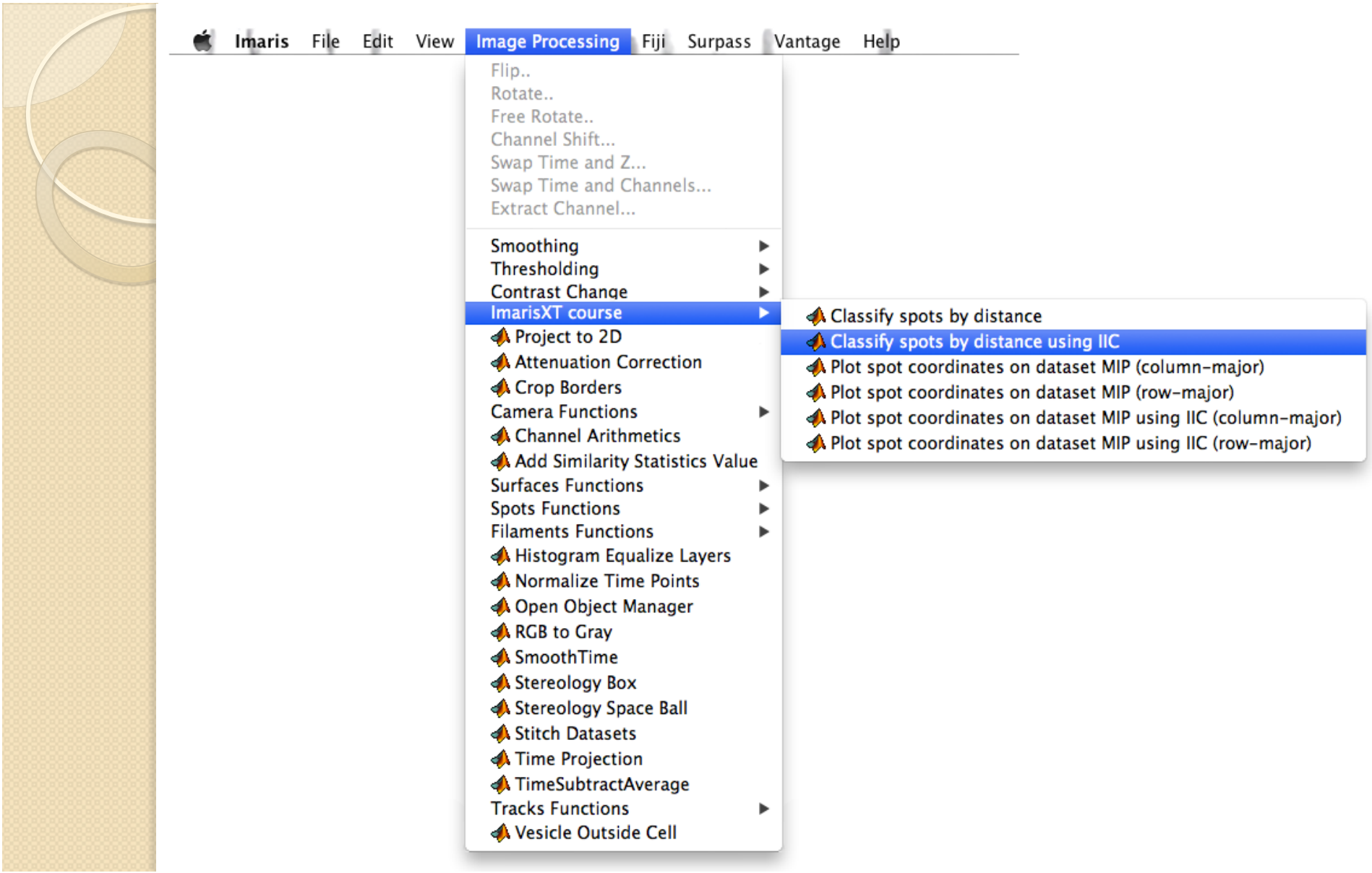


File: spotClassifyByDistanceIIC.m

```
function spotClassifyByDistanceIIC(aImarisApplicationID)
% Classifies the Spots by distance from the center of a surface using IceImarisConnector
%
% The distance limits are specified by the user via input dialogs.
% The classified spots are copied to new Spots objects.
%
%    <CustomTools>
%      <Menu>
%        <Submenu name="ImarisXT course">
%          <Item name="Classify spots by distance using IIC" icon="Matlab">
%            <Command>MatlabXT::spotClassifyByDistanceIIC(%i)</Command>
%          </Item>
%        </Submenu>
%      </Menu>
%      <SurpassTab>
%        <SurpassComponent name="bpSpots">
%          <Item name="Classify spots by distance using IIC" icon="Matlab">
%            <Command>MatlabXT::spotClassifyByDistanceIIC(%i)</Command>
%          </Item>
%        </SurpassComponent>
%      </SurpassTab>
%    </CustomTools>
%
% Copyright (c) 2012 - 2013, Aaron Ponti
```

The Imaris Application ID

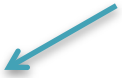
The Imaris Application ID is an identifier that Imaris passes to MATLAB so that communication (via the Ice server) is possible.


Imaris
File
Edit
View
Image Processing
Fiji
Surpass
Vantage
Help
Flip..
Rotate..
Free Rotate..
Channel Shift...
Swap Time and Z...
Swap Time and Channels...
Extract Channel...
Smoothing
Thresholding
Contrast Change
ImarisXT course
Project to 2D
Attenuation Correction
Crop Borders
Camera Functions
Channel Arithmetics
Add Similarity Statistics Value
Surfaces Functions
Spots Functions
Filaments Functions
Histogram Equalize Layers
Normalize Time Points
Open Object Manager
RGB to Gray
SmoothTime
Stereology Box
Stereology Space Ball
Stitch Datasets
Time Projection
TimeSubtractAverage
Tracks Functions
Vesicle Outside Cell
Classify spots by distance
Classify spots by distance using IIC
Plot spot coordinates on dataset MIP (column-major)
Plot spot coordinates on dataset MIP (row-major)
Plot spot coordinates on dataset MIP using IIC (column-major)
Plot spot coordinates on dataset MIP using IIC (row-major)

File: spotClassifyByDistanceIIC.m / continued

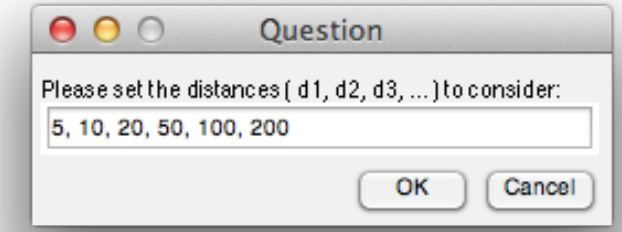
```
% Set up connection between Imaris and MATLAB
conn = IceImarisConnector(aImarisApplicationID);

% Get the currently selected object in the surpass scene
vSpots = conn.getSurpassSelection('Spots');
if isempty(vSpots)
    errordlg('Please select a Spots object!');
    return
end

% Get the Surfaces objects
surfaces = conn.getAllSurpassChildren(false, 'Surfaces');
nSurfaces = numel(surfaces);
if nSurfaces == 0
    errordlg('There are no Surfaces in the scene!');
    return
end
if nSurfaces > 1
    % Ask the user to pick one surface
    vSurface = askUserToPickSurface(surfaces);
    if isempty(vSurface)
        return
    end
else
    vSurface = surfaces{ 1 };
end

% Get the center of mass of the Surfaces object
centerOfMass = vSurface.GetCenterOfMass(0);
```

Simplification…



File: spotClassifyByDistanceIIC.m / continued

```
% Ask the user to specify the distance limits to use
distanceLimits = askUserToSetDistanceLimits();
if isempty(distanceLimits)
    return
end

% Get the spot coordinates, radii and time indices
spotValues = vSpots.Get();
coords = spotValues.mPositionsXYZ;
radii = spotValues.mRadii;
timeIndices = spotValues.mIndicesT;

% Calculate all spot distances from the center of mass
D = sqrt((coords(:, 1) - centerOfMass(1)).^2 + ...
    (coords(:, 2) - centerOfMass(2)).^2 + ...
    (coords(:, 3) - centerOfMass(3)).^2);

% Now classify all spots distances and create new Spots objects for each distance bin
for i = 2 : numel(distanceLimits)
    indx = find(D >= distanceLimits(i - 1) & D < distanceLimits(i));
    if ~isempty(indx)
        name = ['Spots ', num2str(distanceLimits(i - 1)), ...
            ' <= D < ', num2str(distanceLimits(i)), ' [', ...
            num2str(numel(indx)), ']'];
        conn.createAndSetSpots(coords(indx, :), timeIndices(indx), ...
            radii(indx), name, [rand(1, 3) 0]);
    end
end
```



IceImarisConnector

```
% Set up connection between Imaris and MATLAB
conn = IceImarisConnector(aImarisApplicationID);
```

aImarisApplicationID can be:

- nothing
- an Imaris Application ID *(usually from Imaris)*
- an IceImarisConnector reference *(conn)*
- an Imaris Application (ICE) instance.

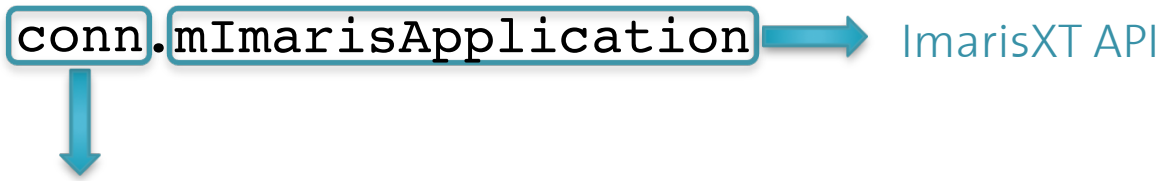
Vanilla ImarisXT

```
% Set up connection between Imaris and MATLAB
if isa(aImarisApplicationID, 'Imaris.IApplicationPrxHelper')
    vImarisApplication = aImarisApplicationID;
else
    % connect to Imaris interface
    javaaddpath ImarisLib.jar
    vImarisLib = ImarisLib;
    if ischar(aImarisApplicationID)
        aImarisApplicationID = round(str2double(aImarisApplicationID));
    end
    vImarisApplication = vImarisLib.GetApplication(aImarisApplicationID);
end
```

The vImariApplication object gives you access to the full ImarisXT API.

IceImarisConnector – Imaris XT: a sinergy

vImarisApplication = vImarisLib.GetApplication(aImarisApplicationID);



ImarisXT API

```
spots = conn.mImarisApplication.GetSurpassSelection();
```

IceImarisConnector API

```
data = conn.getDataVolume(0, 0);
```

IceImarisConnector

```
% Get the currently selected object in the surpass scene
vSpots = conn.getSurpassSelection('Spots');
```

optional filter

All IceImarisConnector methods returning objects take care of autocasting them for you!

Vanilla ImarisXT

```
% Get the currently selected object in the surpass scene
vSpots = vImarisApplication.GetSurpassSelection();
vSpots = vImarisApplication.GetFactory().ToSpots(vSpots);
```

The returned object must be explicitly cast to Spots!

IceImarisConnector

```
% Get the Surfaces objects
surfaces = conn.getAllSurpassChildren(false, 'Surfaces');
```

(do not) recurse into folders

optional filter

Vanilla ImarisXT

```
% Get the Surfaces objects (at the root level!)
surfaces = {}; nSurfaces = 0;
nChildren = vImarisApplication.GetSurpassScene().GetNumberOfChildren();
for i = 0 : (nChildren - 1)
    child = vImarisApplication.GetSurpassScene.GetChild( i );
    if vImarisApplication.GetFactory().IsSurfaces(child)
        nSurfaces = nSurfaces + 1;
        % We must cast the child to a Surfaces object!
        surfaces{nSurfaces} = ...
            vImarisApplication.GetFactory().ToSurfaces(child);
    end
end
```

You should turn this into a recursive function to scan all folders!

IceImarisConnector

```
% Create and add a new Spots object
conn.createAndSetSpots(coords, timeIndices, radii, name, [rand(1, 3) 0], container);
```

transparency

optional

Vanilla ImarisXT

```
% Create and add a new Spots object
newSpots = vImarisApplication.GetFactory().CreateSpots();
newSpots.Set(coords, timeIndices, radii);
newSpots.SetColorRGBA(uint32(randi(255, [1 3]) * [1; 256; 256^2]));
newSpots.SetName(name);
vImarisApplication.GetSurpassScene().AddChild(newSpots, -1);
```

Setting transparency is very tricky with vanilla Imaris XT!

```
% Get the data volume for channel 0 and timepoint 0
stack = conn.getDataVolume(0, 0)
```

This we haven't seen in the original example,
but it is interesting anyway.

Vanilla ImarisXT

```
% Get the iDataSet object
iDataSet = vImarisApplication.GetDataSet();

% Get the data type
switch char(iDataSet.GetType())
    case 'eTypeUInt8',   datatype = 'uint8';
    case 'eTypeUInt16',  datatype = 'uint16';
    case 'eTypeFloat',   datatype = 'single';
    otherwise,           error('Bad value for iDataSet.GetType()');
end

% Allocate memory
stack = zeros([iDataSet.GetSizeX(), iDataSet.GetSizeY(), iDataSet.GetSizeZ()], datatype);

% Get the pixel data
switch char(iDataSet.GetType())
    case 'eTypeUInt8',
        arr = iDataSet.GetDataVolumeAs1DArrayBytes(0, 0);
        stack(:) = typecast(arr, 'uint8');
    case 'eTypeUInt16',
        arr = iDataSet.GetDataVolumeAs1DArrayShorts(0, 0);
        stack(:) = typecast(arr, 'uint16');
    case 'eTypeFloat',
        stack(:) = iDataSet.GetDataVolumeAs1DArrayFloats(0, 0);
    otherwise,
        error('Bad value for type');
end
```

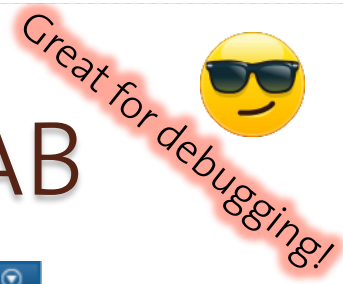
IceImarisConnector takes care of all memory allocations, getting data with the correct XT method and performing type casting as needed.

huge speedup! 😎

typecast needed 😩

typecast needed 😩

Starting Imaris from MATLAB



Command Window

```
>> conn = IceImarisConnector()
IceImarisConnector: not connected to an Imaris instance yet.
>> conn.startImaris()
ans =
     1
>> conn
IceImarisConnector: connected to Imaris.
>>
```

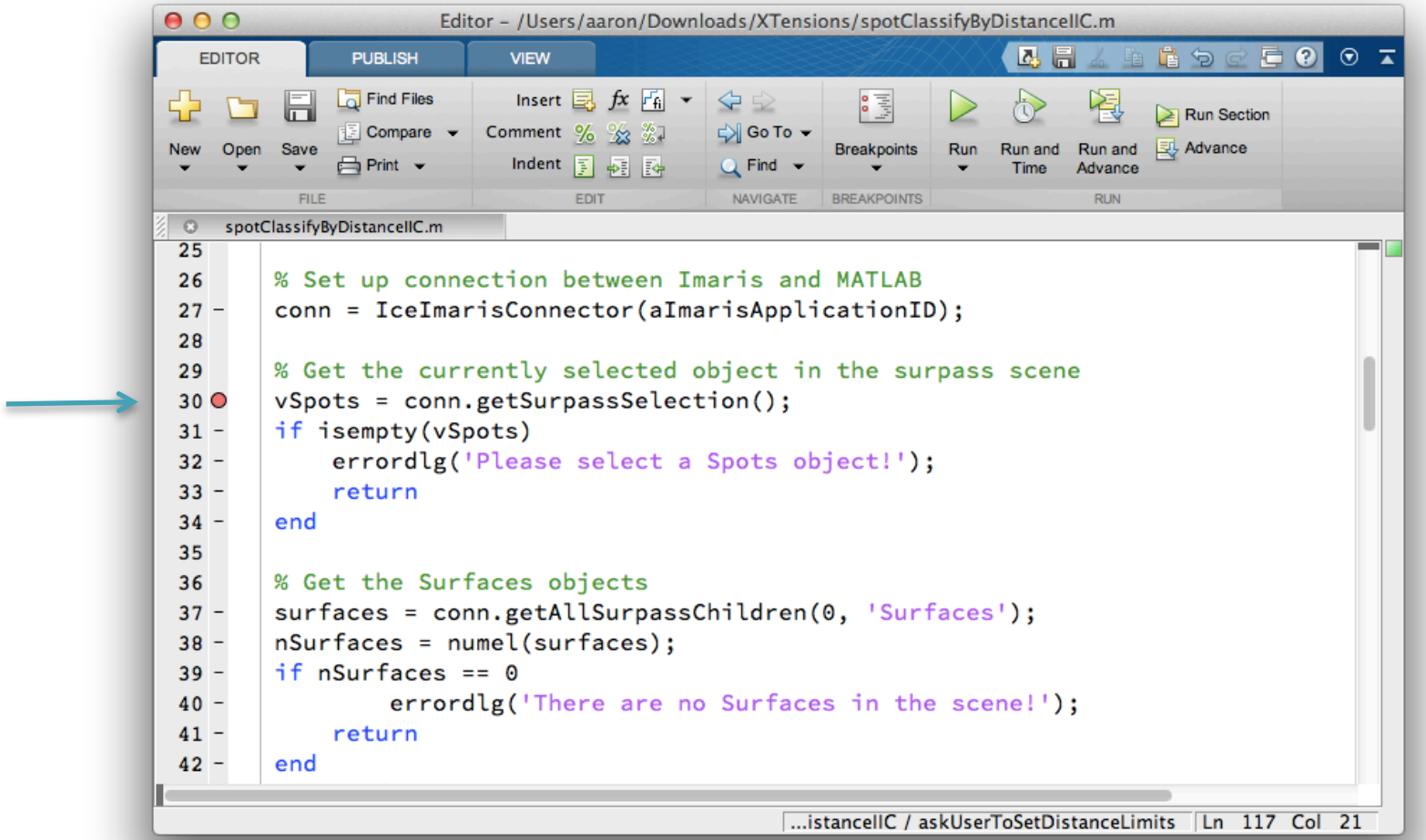
Starts Imaris and connects to it!

Switch to Imaris, load dataset, run spot detection, …

fx

Starting Imaris from MATLAB

Great for debugging!



Starting Imaris from MATLAB

Great for debugging!

```
Command Window
>> conn = IceImarisConnector()
IceImarisConnector: not connected to an Imaris instance yet.
>> conn.startImaris()
ans =
     1
>> conn
IceImarisConnector: connected to Imaris.
>>
```

Switch to Imaris, load dataset, run spot detection, …

```
>> spotClassifyByDistanceIIC(conn)
>>
```

Now we can run/debug the XTension from the MATLAB console passing the IceImarisConnector instance as an argument.

```
>> spots = conn.getSurpassSelection('Spots');
```

When the XTension is finished, we can work interactively with the Imaris instance and the results of our XTension.