Uvas

Generated by Doxygen 1.8.5

Thu Jan 2 2014 20:10:10
Contents

1 Hierarchical Index ........................................ 1
  1.1 Class Hierarchy ...................................... 1

2 Class Index .............................................. 3
  2.1 Class List ........................................... 3

3 Class Documentation ...................................... 5
  3.1 Action Class Reference ............................... 5
    3.1.1 Detailed Description ................................ 5
    3.1.2 Constructor & Destructor Documentation ............... 6
      3.1.2.1 Action ..................................... 6
    3.1.3 Member Function Documentation ......................... 6
      3.1.3.1 getName .................................... 6
      3.1.3.2 getPriority ................................ 6
      3.1.3.3 run ........................................ 6
  3.2 ActionCompare Struct Reference ....................... 6
    3.2.1 Detailed Description ................................ 6
  3.3 ActionParams Struct Reference .......................... 7
    3.3.1 Detailed Description ................................ 7
    3.3.2 Member Data Documentation ............................ 7
      3.3.2.1 object ..................................... 7
  3.4 AddObjectDialog Class Reference ......................... 7
    3.4.1 Detailed Description ................................ 7
  3.5 ApplicationData Class Reference ......................... 8
    3.5.1 Detailed Description ................................ 8
    3.5.2 Member Function Documentation ......................... 9
      3.5.2.1 addCamera2D ................................ 9
      3.5.2.2 addImage2D ................................ 9
      3.5.2.3 addObject .................................. 9
      3.5.2.4 addObject .................................. 9
      3.5.2.5 copyFrom ................................... 10
      3.5.2.6 getCamera2D ................................ 10
# 3.17.2.1 run

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td>29</td>
</tr>
</tbody>
</table>

## 3.18 HoughSolution Struct Reference

### 3.18.1 Detailed Description

Page 29

## 3.19 Hypothesis::Image2D Class Reference

### 3.19.1 Detailed Description

Page 30

### 3.19.2 Member Function Documentation

- addObject: 30
- clear: 31
- findObject: 31
- getDistances: 31
- getIndex: 31
- getMask: 31
- getMaskAroundObject: 32
- getObject: 32
- getObjectCount: 33
- removeObject: 33
- removeObject: 33
- setDistances: 33
- setObject: 34
- sortObjectsVectorWithOrder: 34

## 3.20 IndexPair Struct Reference

### 3.20.1 Detailed Description

Page 34

## 3.21 Hypothesis::Node::Link Struct Reference

### 3.21.1 Detailed Description

Page 35

## 3.22 MainWindow Class Reference

### 3.22.1 Detailed Description

Page 35

### 3.23 ManualStageWidget Class Reference

### 3.23.2 Constructor & Destructor Documentation

- ManualStageWidget: 37

### 3.23.3 Member Function Documentation

- addObject: 37
- getViewport2D: 38
- keyPressEvent: 38

## 3.24 Hypothesis::Node Class Reference

### 3.24.1 Detailed Description

Page 39

### 3.24.2 Member Function Documentation

- clearLinks: 39
- findNode: 39
- getLinkCount: 40
3.24.2.4  ID ................................................................. 40
3.24.2.5  insertLink ................................................. 40
3.24.2.6  insertLink ................................................. 40
3.24.2.7  removeLink ................................................ 40
3.24.2.8  removeLink ................................................ 41
3.24.2.9  setCost ..................................................... 41
3.24.2.10 setCost ..................................................... 41
3.24.2.11 setID ......................................................... 41
3.24.2.12 sortLinksVectorWithOrder ................................ 41
3.24.2.13 type ........................................................ 42

3.25 Hypothesis::Object2D Class Reference ................................................. 42
   3.25.1 Detailed Description ........................................... 43
   3.25.2 Member Function Documentation ...................................... 43
       3.25.2.1  addEllipse ............................................. 43
       3.25.2.2  addEllipse ............................................. 43
       3.25.2.3  addEllipse ............................................. 43
       3.25.2.4  calcDistance .......................................... 44
       3.25.2.5  calcHiddenIndex ...................................... 44
       3.25.2.6  calcInFrontOf ......................................... 44
       3.25.2.7  calcMinMax ........................................... 44
       3.25.2.8  calcOverlappingBack .................................. 44
       3.25.2.9  calcOverlappingFront .................................. 44
       3.25.2.10 checkOverlapping ...................................... 44
       3.25.2.11 clear ................................................... 45
       3.25.2.12 copyFrom .............................................. 45
       3.25.2.13 deleteEllipse ......................................... 45
       3.25.2.14 fillEllipse ............................................ 45
       3.25.2.15 getColor ................................................ 45
       3.25.2.16 getDistanceToCamera ................................ 45
       3.25.2.17 getEllipse ............................................. 46
       3.25.2.18 getEllipseCount ...................................... 46
       3.25.2.19 getHiddenIndex ....................................... 46
       3.25.2.20 getIndex ................................................ 46
       3.25.2.21 getNodeAsObject2D ................................... 46
       3.25.2.22 getOverlappingListBack ................................ 46
       3.25.2.23 getOverlappingListFront ................................ 46
       3.25.2.24 overlappingCalculation ................................ 47
       3.25.2.25 setColor ............................................... 48
       3.25.2.26 setColor ............................................... 48
       3.25.2.27 setDistanceToCamera ................................ 48
3.29.2.1 calcIndex ................................................................. 56
3.30 Hypothesis::Scene3D Class Reference .................................. 56
  3.30.1 Detailed Description .................................................. 57
  3.30.2 Member Function Documentation .................................... 57
    3.30.2.1 addObject ........................................................ 57
    3.30.2.2 clear .............................................................. 57
    3.30.2.3 findObject ....................................................... 57
    3.30.2.4 getObject ......................................................... 58
    3.30.2.5 getObjectCount .................................................. 58
    3.30.2.6 globalMakeLinks ................................................ 58
    3.30.2.7 makeLinks ........................................................ 58
    3.30.2.8 makeLinks ........................................................ 58
    3.30.2.9 removeObject ..................................................... 59
    3.30.2.10 removeObject ..................................................... 59
    3.30.2.11 setObject ....................................................... 59
    3.30.2.12 sortObjectsVectorWithOrder ................................ 59
3.31 SceneTreeOptimizer Class Reference ................................... 60
  3.31.1 Detailed Description ................................................ 60
  3.31.2 Member Function Documentation .................................... 61
    3.31.2.1 addDependency .................................................. 61
    3.31.2.2 exploreTreeManual ............................................ 61
    3.31.2.3 optimize ........................................................ 61
3.32 sorter<T> Class Template Reference .................................. 61
3.33 Hypothesis::Sphere Class Reference .................................. 61
  3.33.1 Detailed Description ................................................ 62
  3.33.2 Member Function Documentation .................................... 62
    3.33.2.1 copyFrom ....................................................... 62
    3.33.2.2 getAbsCoords ................................................... 62
    3.33.2.3 getCoords ...................................................... 62
    3.33.2.4 getID ............................................................. 62
    3.33.2.5 getParent ........................................................ 63
    3.33.2.6 getPosition ..................................................... 63
    3.33.2.7 getRadius ....................................................... 63
    3.33.2.8 setAbsCoords ................................................... 63
    3.33.2.9 setCoords ....................................................... 63
    3.33.2.10 setCoords ...................................................... 63
    3.33.2.11 setID ........................................................... 64
    3.33.2.12 setParent ...................................................... 64
    3.33.2.13 setPosition .................................................... 64
    3.33.2.14 setPosition .................................................... 64
3.34 SphereGenerator Class Reference .................................................. 64
  3.34.1 Detailed Description ......................................................... 65
  3.34.2 Constructor & Destructor Documentation .......................... 65
  3.34.2.1 SphereGenerator ......................................................... 65
  3.34.3 Member Function Documentation ......................................... 65
    3.34.3.1 addHolePoint ......................................................... 65
    3.34.3.2 hough ............................................................... 65
3.35 StateOptim Struct Reference .................................................... 66
  3.35.1 Detailed Description ......................................................... 66
3.36 Supervisor Class Reference ..................................................... 66
  3.36.1 Detailed Description ......................................................... 67
  3.36.2 Constructor & Destructor Documentation .......................... 67
  3.36.2.1 Supervisor ............................................................ 67
  3.36.3 Member Function Documentation ......................................... 67
    3.36.3.1 addAction ............................................................ 67
    3.36.3.2 addObject .......................................................... 67
    3.36.3.3 getApplicationData ............................................ 67
    3.36.3.4 getBitmapManager ............................................ 68
    3.36.3.5 getFeatureMatcher ........................................... 68
    3.36.3.6 getParameters ................................................... 68
    3.36.3.7 getScene ............................................................. 68
    3.36.3.8 hasFinished ....................................................... 68
    3.36.3.9 intermediateCallback ........................................ 68
    3.36.3.10 onStart ............................................................. 69
    3.36.3.11 setApplicationData .......................................... 69
3.37 TreeNode Class Reference ........................................................ 69
  3.37.1 Detailed Description ......................................................... 69
3.38 Viewport2D Class Reference ..................................................... 69
  3.38.1 Detailed Description ......................................................... 71
  3.38.2 Member Function Documentation ......................................... 71
    3.38.2.1 getParent ............................................................ 71
    3.38.2.2 getSourceImage .................................................. 71
    3.38.2.3 loadCalibration ................................................ 71
    3.38.2.4 loadCalibration ................................................ 71
    3.38.2.5 loadImage .......................................................... 71
    3.38.2.6 loadImage .......................................................... 72
    3.38.2.7 setBitmapManager ............................................ 72
    3.38.2.8 setFeatureMatcher .......................................... 72
    3.38.2.9 setImage .......................................................... 72
### CONTENTS

3.38.2.10 setPointMode ........................................ 72  
3.38.2.11 setSphereMode ....................................... 72  
3.38.2.12 setSpineMode ......................................... 72  
3.38.2.13 showImage ............................................ 72  

3.39 Viewport3D Class Reference ................................. 73  
  3.39.1 Detailed Description .................................. 73  
  3.39.2 Member Function Documentation ....................... 74  
    3.39.2.1 setAutoCenter ................................ 74  
    3.39.2.2 updateValues ................................ 74  

Index .......................................................... 75
## 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>5</td>
</tr>
<tr>
<td>CreateBottomGrapeAction</td>
<td>15</td>
</tr>
<tr>
<td>CreateRosetteAction</td>
<td>16</td>
</tr>
<tr>
<td>HoughAction</td>
<td>28</td>
</tr>
<tr>
<td>OptimizeObjectAction</td>
<td>53</td>
</tr>
<tr>
<td>ActionCompare</td>
<td>6</td>
</tr>
<tr>
<td>ActionParams</td>
<td>7</td>
</tr>
<tr>
<td>ApplicationData</td>
<td>8</td>
</tr>
<tr>
<td>BitmapManager</td>
<td>12</td>
</tr>
<tr>
<td>Dependency</td>
<td>16</td>
</tr>
<tr>
<td>Hypothesis::Ellipse</td>
<td>19</td>
</tr>
<tr>
<td>FeatureMatcher</td>
<td>20</td>
</tr>
<tr>
<td>FeatureTreeNode</td>
<td>25</td>
</tr>
<tr>
<td>HoughSolution</td>
<td>29</td>
</tr>
<tr>
<td>Hypothesis::Image2D</td>
<td>30</td>
</tr>
<tr>
<td>IndexPair</td>
<td>34</td>
</tr>
<tr>
<td>Hypothesis::Node::Link</td>
<td>35</td>
</tr>
<tr>
<td>Hypothesis::Node</td>
<td>38</td>
</tr>
<tr>
<td>Hypothesis::Object2D</td>
<td>42</td>
</tr>
<tr>
<td>Hypothesis::Object3D</td>
<td>49</td>
</tr>
<tr>
<td>QDialog</td>
<td>7</td>
</tr>
<tr>
<td>AddObjectDialog</td>
<td>14</td>
</tr>
<tr>
<td>CompareDialog</td>
<td></td>
</tr>
<tr>
<td>QFrame</td>
<td></td>
</tr>
<tr>
<td>DrawableLayout</td>
<td>17</td>
</tr>
<tr>
<td>QGLWidget</td>
<td></td>
</tr>
<tr>
<td>GLWidget</td>
<td>26</td>
</tr>
<tr>
<td>QMainWindow</td>
<td></td>
</tr>
<tr>
<td>MainWindow</td>
<td>35</td>
</tr>
<tr>
<td>QWidget</td>
<td></td>
</tr>
<tr>
<td>ManualStageWidget</td>
<td>36</td>
</tr>
<tr>
<td>Viewport2D</td>
<td>69</td>
</tr>
<tr>
<td>Viewport3D</td>
<td>73</td>
</tr>
<tr>
<td>Hypothesis::Scene3D</td>
<td>56</td>
</tr>
<tr>
<td>SceneTreeOptimizer</td>
<td>60</td>
</tr>
<tr>
<td>sorter&lt;T&gt;</td>
<td>61</td>
</tr>
<tr>
<td>Hypothesis::Sphere</td>
<td>61</td>
</tr>
</tbody>
</table>
Hierarchical Index

SphereGenerator ............................................. 64
StateOptim .................................................. 66
 Supervisor .................................................. 66
GrapeSupervisor ............................................. 27

TNLP
  OptSphereNLP ............................................. 54
  OptSpherePoints ......................................... 55
  TreeNode .................................................. 69
Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

- **Action**
  Class that represents an action performed by a Supervisor ........................................ 5
- **ActionCompare**
  Struct used to order the actions in the supervisor’s action queue ................................ 6
- **ActionParams**
  Contains the required parameters for creating an action .................................................. 7
- **AddObjectDialog**
  Dialog for creating a new 3D Object .................................................................................. 7
- **ApplicationData**
  Class that encapsulates all the scene and images data ....................................................... 8
- **BitmapManager**
  Class that manages different bitmaps derived from the source images ............................... 12
- **CompareDialog**
  Dialog for comparing two images ....................................................................................... 14
- **CreateBottomGrapeAction**
  This action creates a 3D object corresponding to the bottom grape of the images ............... 15
- **CreateRosetteAction**
  .......................................................................................................................... 16
- **Dependency**
  Struct that represents a possible dependency between a objects towards another ................ 16
- **DrawableLayout**
  Drawable area where 2D images are displayed .................................................................... 17
- **Hypothesis::Ellipse**
  This class represents a projected ellipse. It’s part of an Object2D ....................................... 19
- **FeatureMatcher**
  Class that offers various methods to extract 3D points from two 2D stereoscopic images ...... 20
- **FeatureTreeNode**
  Node of the dynamic programming tree for matching ....................................................... 25
- **GLWidget**
  Widget where the 3D scene is represented ............................................................................ 26
- **GrapeSupervisor**
  Grape reconstruction supervisor .......................................................................................... 27
- **HoughAction**
  This Action creates 3D objects through the Hough method ............................................. 28
- **HoughSolution**
  Struct that contains the info necessary to represent a solution of the Hough method ........... 29
- **Hypothesis::Image2D**
  This class represents a scene of Object2D which are organized as a graph .......................... 30
IndexPair
Pair of pixels on a row and the cost of matching them .......................... 34
Hypothesis::Node::Link
A link (or edge) consisting of a destination node and a cost .................. 35
MainWindow
Main window. Contains the graphical interface ................................. 35
ManualStageWidget
Widget that contains the interface for editing the scene and its projections ........ 36
Hypothesis::Node
This class holds all the necessary data and methods to use it as a Node of a graph .... 38
Hypothesis::Object2D
This class represents a 2D object on an image. It is modelled as a set of ellipses and its tangents 42
Hypothesis::Object3D
This class represents a 3D object on a scene. It is modelled as a polysphere ........ 49
OptimizeObjectAction
This Action performs the parameter optimization on a 3D object and its neighbors ........ 53
OptSphereNLP
Class that represents a NLP problem for the optimization of the parameters of a sphere through its projections ................................................................. 54
OptSpherePoints
Class that represents a NLP problem for the optimization of the parameters of a sphere given some of its 3D points ................................................................. 55
Hypothesis::Scene3D
This class represents a scene of Object3D’s organized as a graph ............... 56
SceneTreeOptimizer
Optimizes the parameters of the objects in a 3D scene ............................ 60
caster<T>
........................................................................................................ 61
Hypothesis::Sphere
A sphere that is part of an Object3D .................................................... 61
SphereGenerator
Class that generates 3D hypotheses through the Hough method .................. 64
StateOptim
State of the SceneTreeOptimizer in a specific node ............................... 66
Supervisor
Base class which defines a common interface for different stereoscopic edge-detection applications ...................................................................................... 66
TreeNode
A node of the tree in the SceneTreeOptimizer ....................................... 69
Viewport2D
Widget that represents a 2D viewport ..................................................... 69
Viewport3D
Widget that represents a 3D viewport ..................................................... 73
Chapter 3

Class Documentation

3.1 Action Class Reference

Class that represents an action performed by a Supervisor.

#include <action.h>

Inheritance diagram for Action:

```
+--------+        +----------+       +----------+       +----------+
| Action | -|--| CreateBottomGrapeAction | +|--| CreateRosetteAction | +|--| HoughAction | +|--| OptimizeObjectAction |
```

Public Member Functions

- `Action (const char *name, ActionParams &params, float priority)`
- virtual `bool run ()=0`
- `const string & getName () const`
  
  Returns the identifying name of this Action.
- `float getPriority () const`

Protected Attributes

- string `m_name`
- `ActionParams m_params`
- float `m_priority`

3.1.1 Detailed Description

Class that represents an action performed by a Supervisor.

An Action will usually perform a single task. It may also spawn more actions when it is finished.

Derived classes must implement the method `run()` which will be called when this Action is in top of the Supervisor's action queue. This method is where the Action must perform its operations.

Each Action must be assigned a priority, from 0.0 to 1.0, where the higher the number, the more prioritary the Action will be.

Also, in order to identify each Action's type, when creating a subclass the programmer must enter a unique identifying name as a string in the constructor of the subclass. (see optimizeobjectaction.h for reference)
3.1.2 Constructor & Destructor Documentation

3.1.2.1 Action::Action ( const char * name, ActionParams & params, float priority ) [inline]

Constructor
Parameters

<table>
<thead>
<tr>
<th>name</th>
<th>Unique identifying name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>params</td>
<td>The parameters for the action.</td>
</tr>
<tr>
<td>priority</td>
<td>Priority of this action in the supervisor queue (from 0.0 to 1.0, less to higher priority).</td>
</tr>
</tbody>
</table>

3.1.3 Member Function Documentation

3.1.3.1 const string& Action::getName ( ) const [inline]

Returns the identifying name of this Action.
Useful when the user has an Action* pointer but wants to know the specific subclass.

3.1.3.2 float Action::getPriority ( ) const [inline]

Returns this Action's priority.

3.1.3.3 virtual bool Action::run ( ) [pure virtual]

This method must contain the operations specific to each Action.
Returns
false if there was an error, true otherwise.

Implemented in CreateRosetteAction, CreateBottomGrapeAction, HoughAction, and OptimizeObjectAction.

The documentation for this class was generated from the following file:

- oncovation/Interfaz/Interfaz/action.h

3.2 ActionCompare Struct Reference

Struct used to order the actions in the supervisor's action queue.
#include <action.h>

Public Member Functions

- bool operator() (const Action *left, const Action *right) const

3.2.1 Detailed Description

Struct used to order the actions in the supervisor's action queue.
The documentation for this struct was generated from the following file:

- oncovation/Interfaz/Interfaz/action.h
3.3 ActionParams Struct Reference

Contains the required parameters for creating an action.

```
#include <action.h>
```

Public Attributes

- **Supervisor * supervisor**
  A pointer to the Supervisor.
- **ApplicationData * data**
  The ApplicationData the action will perform its operations on.
- **Object3D * object**
  An situational parameter specifying which object will the action perform its operations on.

3.3.1 Detailed Description

Contains the required parameters for creating an action.

3.3.2 Member Data Documentation

3.3.2.1 Object3D * ActionParams::object

An situational parameter specifying which object will the action perform its operations on.
Some actions may ignore this parameter. If that's the case, just set it to NULL.

The documentation for this struct was generated from the following file:

- oncovision/Interfaz/Interfaz/action.h

3.4 AddObjectDialog Class Reference

Dialog for creating a new 3D Object.

```
#include <addobjectdialog.h>
```

Inheritance diagram for AddObjectDialog:

```
QDialog
AddObjectDialog
```

Public Member Functions

- **AddObjectDialog (QWidget *parent=0)**
- double getXPos ()
  Returns the x position set by the user.
- double getYPos ()
  Returns the y position set by the user.
- double getZPos ()

Generated on Thu Jan 2 2014 20:10:10 for Uvas by Doxygen
3.4.1 Detailed Description

Dialog for creating a new 3D Object.

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/addobjectdialog.h
- oncovision/Interfaz/Interfaz/addobjectdialog.cpp

3.5 ApplicationData Class Reference

Class that encapsulates all the scene and images data.

```cpp
#include <applicationdata.h>
```

Public Member Functions

- bool addObject (Object3D *object)
- Object3D *addObject (double coords[4])
- bool removeObject (int index)
- Object3D *getObject (int index) const
- Object3D *getLastObject () const
- int getObjectCount () const
- Scene3D *getScene3D ()

  Returns a pointer to the Scene3D.

- Camera *addCamera2D ()
- void deleteCamera2D ()

  Deletes THE LAST ADDED Camera.

- Camera *getCamera2D (int index) const
- int getCameras2DCount () const

  Returns the number of Cameras in this application data.

- Image2D *addImage2D ()
- void deleteImage2D ()

  Deletes THE LAST ADDED Image2D.

- Image2D *getImage2D (int index) const
- int getImages2DCount () const

  Returns the number of Images2D in this application data.

- void project (int camera)

  Projects all the objects of the Scene2D into a specific Image2D.

- void updateSingleSphere (int camera, int ID, double coords[4])

  Updates the coordinates of a single 3D object and its projection in a specific image.

- double getMinValue ()

  Calculates the minimum error index of all the 2D Objects.

- void copyFrom (const ApplicationData *data)
- void makeGroups (int position, double offset=-1.0)

  Links a object to the others which are within a given distance in pixels.

- bool checkSanity () const
- void clearDeb ()
3.5 ApplicationData Class Reference

Public Attributes

- double optimValue
- double optimMinimumValue
- double distanceThreshold

3.5.1 Detailed Description

Class that encapsulates all the scene and images data.

This class is used extensively in many parts of the code. It contains all the data necessary to represent the problem of 2D-to-3D sphere-based reconstruction.

Typically it should contain a Scene3D and two Images2D (with their associated Cameras) for this application uses stereoscopic images. Images2D and Cameras are associated by index. Therefore the first Image2D is associated with the first Camera, and so on.

It supports, however, an arbitrary number of Images2D, which is useful if the input comes from more than two cameras. Note that this is done for potential uses beyond this case and adding more than two Image2D is not supported by other parts of the supervisor.

3.5.2 Member Function Documentation

3.5.2.1 Camera * ApplicationData::addCamera2D ( )

Creates a new Camera and adds it to the ApplicationData.

Returns

A pointer to the Camera just added.

3.5.2.2 Image2D * ApplicationData::addImage2D ( )

Creates a new Image2D and adds it to the ApplicationData.

Returns

A pointer to the Image2D just added.

3.5.2.3 bool ApplicationData::addObject ( Object3D * object )

Adds a 3D object to the Scene3D.

Parameters

| object | The Object3D to be added. |

Returns

false if the maximum object count is reached, true otherwise.

3.5.2.4 Object3D * ApplicationData::addObject ( double coords[4] )

Creates a 3D object with a single sphere in its center and adds it to the Scene3D.
Parameters

| coords | Coordinates of the object and the radius of the sphere [x, y, z, r]. |

Returns

A pointer to the created object.

3.5.2.5 void ApplicationData::copyFrom (const ApplicationData *data)

Copies all the data from another ApplicationData

Parameters

| data | The ApplicationData instance to copy the data from. |

Warning

Deletes the previous data!

3.5.2.6 Camera * ApplicationData::getCamera2D (int index) const

Returns a Camera.

Parameters

| index | Index of the Camera. |

3.5.2.7 Image2D * ApplicationData::getImage2D (int index) const

Returns a Image2D.

Parameters

| index | Index of the Camera. |

3.5.2.8 Object3D * ApplicationData::getLastObject ( ) const

Returns the last object added to the Scene3D.

Returns

A pointer to the last Object3D or NULL if there is none (empty scene).

3.5.2.9 double ApplicationData::getMinValue ( )

Calculates the minimum error index of all the 2D Objects.

The error index of a 2D object is the maximum between the error index of its projections on the different Images2D.

3.5.2.10 Object3D * ApplicationData::getObject (int index) const

Returns an object from the Scene3D.
3.5 ApplicationData Class Reference

Parameters

| index | Position of the object in the scene. |

Returns

A pointer to the Object3D or NULL if index not valid.

3.5.2.11 int ApplicationData::getObjectCount ( ) const

Returns the number of objects in the Scene3D.

Returns

The object count.

3.5.2.12 void ApplicationData::makeGroups ( int position, double offset = -1.0 )

Links a object to the others which are within a given distance in pixels.

Used to restrict which objects affect others when optimizing their position. With this, a object which is far from other won’t be taken into account when optimizing, thus making less calculations.

Note that this method works in 2D. The selected object is projected. Its projection is then scaled by “offset” pixels in the large axis, and every object whose center its inside the scaled projection will be linked to the selected object.

Parameters

| position | Position of the object. Number of pixels the projection will be scaled with. If it’s less than 0, 1.5+maxAxis is used as parameter. |

3.5.2.13 void ApplicationData::project ( int camera )

Projects all the objects of the Scene2D into a specific Image2D.

This will delete all the objects(ellipses) in the selected Image2D and will create one for each object(sphere) in the Scene3D.

Usually, pass 0 as argument to project onto the left camera or 1 for the right one.

Parameters

| camera | Index of the Image2D to project the scene on. |

3.5.2.14 bool ApplicationData::removeObject ( int index )

Removes a object from the Scene3D.

Parameters

| index | Position of the object to be removed. |

Returns

true if the object is effectively removed, false otherwise (index out of bounds).
3.5.2.15  void ApplicationData::updateSingleSphere ( int camera, int ID, double coords[4] )

Updates the coordinates of a single 3D object and its projection in a specific image. This is used for optimization purposes. Sometimes we only need to update a single object in a single camera, instead of projecting all objects with project(). That's when we use updateSingleSphere().

Parameters

| camera | Index of the Image2D whose ellipse will be updated. |
| camera | Identifier of the object. |
| coords | New coordinates of the 3D object [x, y, z, r]. |

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/applicationdata.h
- oncovision/Interfaz/Interfaz/applicationdata.cpp

### 3.6 BitmapManager Class Reference

Class that manages different bitmaps derived from the source images.

```cpp
#include <bitmapmanager.h>
```

**Public Member Functions**

- **BitmapManager** (const ApplicationData &data)
- bool calculateOverlap (const Object3D &object, vector<float> *pctg=NULL)
- bool isInBackground (const Object3D &object)
- void fillObject (const Object3D &object)
- void applyMask (int index, Mat &mask)
- void applyRectangularMask (int index, int w1, int w2, int h1, int h2)
- void segmentate (int index, Mat &src)
- Mat *getBitmap (int index)
- void setBitmap (int index, Mat &bitmap)

**3.6.1 Detailed Description**

Class that manages different bitmaps derived from the source images. It holds one bitmap for each camera/source image. These bitmaps are used as a canvas where the each pixel may be of the background, foreground or object.

**3.6.2 Member Function Documentation**

3.6.2.1 void BitmapManager::applyRectangularMask ( int index, int w1, int w2, int h1, int h2 )

Applies a rectangular mask to a specific bitmap. Only the area inside the rectangle will be set as valid.
3.6.2.2 bool BitmapManager::calculateOverlap ( const Object3D & object, vector<float> * pctg = NULL )

Determines if the object is fully overlaped on atleast one of the cameras.
Parameters

<table>
<thead>
<tr>
<th>out</th>
<th>object</th>
<th>Object to evaluate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pctg</td>
<td>An optional vector which will contain the percentage of overlapping on each camera.</td>
</tr>
</tbody>
</table>

3.6.2.3 void BitmapManager::fillObject ( const Object3D & object )

Fills all the bitmaps with the projection of the object.

Parameters

| object | Object to project |

3.6.2.4 bool BitmapManager::isInBackground ( const Object3D & object )

Determines if the object is touching the background in any image

Parameters

| object | Object to evaluate |

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/bitmapmanager.h
- oncovision/Interfaz/Interfaz/bitmapmanager.cpp

3.7 CompareDialog Class Reference

Dialog for comparing two images.

#include <comparedialog.h>

Inheritance diagram for CompareDialog:

```
QDialog
  CompareDialog
```

Public Member Functions

- **CompareDialog** (QComboBox *images, QWidget *parent=0)
- **QString getImage1 ()**
  Returns the name of the first image.
- **QString getImage2 ()**
  Returns the name of the second image.
- **QString getMethod ()**
  Returns the name of the method to use.
3.8 CreateBottomGrapeAction Class Reference

3.7.1 Detailed Description

Dialog for comparing two images.
The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/comparedialog.h
- oncovision/Interfaz/Interfaz/comparedialog.cpp

3.8 CreateBottomGrapeAction Class Reference

This action creates a 3D object corresponding to the bottom grape of the images.

```cpp
#include <createbottomgrapeaction.h>
```

Inheritance diagram for CreateBottomGrapeAction:

```
Action

CreateBottomGrapeAction
```

Public Member Functions

- `CreateBottomGrapeAction (ActionParams &params, float priority)`
- `virtual bool run ()`

Additional Inherited Members

3.8.1 Detailed Description

This action creates a 3D object corresponding to the bottom grape of the images.
It makes use of the FeatureMatcher's `matchLowestGrape()` method to extract the most 3D points of the border of the grape as possible. Then uses the OptSpherePoints optimizer to find the Object3D that better fits those points.

3.8.2 Member Function Documentation

3.8.2.1 `virtual bool CreateBottomGrapeAction::run ( ) [inline],[virtual]`

This method must contain the operations specific to each Action.

Returns

false if there was an error, true otherwise.

Implements Action.

The documentation for this class was generated from the following file:

- oncovision/Interfaz/Interfaz/createbottomgrapeaction.h
3.9 CreateRosetteAction Struct Reference

Inheritance diagram for CreateRosetteAction:

Action

CreateRosetteAction

Public Member Functions

- CreateRosetteAction (ActionParams &params, float priority)
- virtual bool run ()

Additional Inherited Members

3.9.1 Member Function Documentation

3.9.1.1 virtual bool CreateRosetteAction::run () [inline],[virtual]

This method must contain the operations specific to each Action.

Returns

false if there was an error, true otherwise.

Implements Action.

The documentation for this struct was generated from the following file:

- oncovision/Interfaz/Interfaz/createrosetteaction.h

3.10 Dependency Struct Reference

Struct that represents a possible dependency between a objects towards another.

#include <optimizer.h>

Public Attributes

- int idFrom
  
  Id of the first object.
- int idTo
  
  Id of the second object.
- double valFrom
  
  Error value of the first object.
- double valTo
  
  Error value of the second object.
- int depth
  
  Depth of the dependency on the exploration tree.
3.10.1 Detailed Description

Struct that represents a possible dependency between objects towards another.
Used for optimization purposes in the SceneTreeOptimizer class.
If a dependency from a object to another exists too from the second object to the first, and the values and depth coincide, it means there is no need to process that branch of the tree.
The documentation for this struct was generated from the following file:

- oncovision/Interfaz/Interfaz/optimizer.h

3.11 DrawableLayout Class Reference

Drawable area where 2D images are displayed.

#include <drawablelayout.h>

Inheritance diagram for DrawableLayout:

```
QFrame

DrawableLayout
```

Public Member Functions

- **DrawableLayout** (Viewport2D *viewport, QWidget *parent=0)
- void **setImage** (QImage &image)
  
Sets the image to display.
- void **setState** (int state)
  
Sets the current state.
- void **paintEvent** (QPaintEvent *event)
  
Where the painting is implemented.
- void **mousePressEvent** (QMouseEvent *event)
  
Where the mouse press event is processed.
- void **mouseMoveEvent** (QMouseEvent *event)
  
Where the mouse movement is processed.
- void **calculateSphere** (double sphereCoords[4])
  
Calculates the sphere from a ellipse and a point.
- void **calculateSpine** (Vec3f &spine1, Vec3f &spine2)
  
Calculates 3D spine from 4 2D points.
- void **calculatePoint** (Vec3f &point3d)
  
Calculates 3D point from two 2D points.

Public Attributes

- QPoint **m_line1**
  
  First point of the line.
- QPoint **m_line2**
  
  Second point of the line.
- QPoint **m_spine1**
First point of the spine.
- QPoint m_spine2

Second point of the spine.
- QPoint m_3dPoint

Point used to calculate a single 3D point.
- double m_ellipse [5]

Ellipse parameters.

3.11.1 Detailed Description

Drawable area where 2D images are displayed.
The user can also interact with this area to define new spheres, spines or 3D points.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 DrawableLayout::DrawableLayout ( Viewport2D * viewport, QWidget * parent = 0 )

Constructor

Parameters

| viewport | The viewport that owns this DrawableLayout. |

3.11.3 Member Function Documentation

3.11.3.1 void DrawableLayout::calculatePoint ( Vec3f & point3d )

Calculates 3D point from two 2D points.
This method has no arguments because takes the parameters from the available DrawableLayouts.

Parameters

| out | point3d | Resulting 3D point. |

3.11.3.2 void DrawableLayout::calculateSphere ( double sphereCoords[4] )

Calculates the sphere from an ellipse and a point.
This method has no arguments because takes the parameters from the available DrawableLayouts.

Parameters

| out | sphereCoords | Coordinates of the sphere. |

3.11.3.3 void DrawableLayout::calculateSpine ( Vec3f & spine1, Vec3f & spine2 )

Calculates 3D spine from 4 2D points.
This method has no arguments because takes the parameters from the available DrawableLayouts.
### Parameters

<table>
<thead>
<tr>
<th>out</th>
<th>spine1</th>
<th>First point of the 3D spine.</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>spine2</td>
<td>Second point of the 3D spine.</td>
</tr>
</tbody>
</table>

#### 3.11.3.4 void DrawableLayout::setState ( int state ) [inline]

Sets the current state.

-2: Disabled

-1: None

0 to 6: Sphere Mode

7 to 11: Spine Mode

12 to 14: Point Mode

**Parameters**

<table>
<thead>
<tr>
<th>state</th>
<th>New state.</th>
</tr>
</thead>
</table>

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/drawablelayout.h
- oncovision/Interfaz/Interfaz/drawablelayout.cpp

### 3.12 Hypothesis::Ellipse Class Reference

This class represents a projected ellipse. It's part of an Object2D.

#include <ellipse.h>

#### Public Member Functions

- **Ellipse** (double params[5])
- **Ellipse** (double x, double y, double a, double b, double angle)
- void getParameters (double params[5])
- void setParameters (double params[5])
- void setParameters (double x, double y, double a, double b, double angle)
- const double ∗ getPoints ()
- void calcPoints ()

#### Static Public Attributes

- static const unsigned int NPOINTS = 100
  
  Resolution of the ellipse in points.

### 3.12.1 Detailed Description

This class represents a projected ellipse. It's part of an Object2D.
3.12.2 Member Function Documentation

3.12.2.1 void Hypothesis::Ellipse::calcPoints ( )

Calculates the points of the ellipse from its parameters.

3.12.2.2 void Hypothesis::Ellipse::getParameters ( double params[5] )

Returns the parameters of the ellipse.
Parameters

<table>
<thead>
<tr>
<th>out</th>
<th>params</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parameters of the ellipse (cx, cy, ra, rb, angle).</td>
</tr>
</tbody>
</table>

3.12.2.3 const double * Hypothesis::Ellipse::getPoints ( )

Returns a pointer to the calculated points of the ellipse.
Returns
A const pointer so the points are read-only. The size of the vector will always be 2*NPOINTS.

3.12.2.4 void Hypothesis::Ellipse::setParameters ( double params[5] )

Sets the parameters of the ellipse.
Parameters

<table>
<thead>
<tr>
<th>params</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters of the ellipse (cx, cy, ra, rb, angle).</td>
</tr>
</tbody>
</table>

3.12.2.5 void Hypothesis::Ellipse::setParameters ( double x, double y, double a, double b, double angle )

Sets the parameters of the ellipse.
Parameters

<table>
<thead>
<tr>
<th>x</th>
<th>X coordinate of the center of the ellipse.</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>Y coordinate of the center of the ellipse.</td>
</tr>
<tr>
<td>a</td>
<td>Major axis.</td>
</tr>
<tr>
<td>b</td>
<td>Minor axis.</td>
</tr>
<tr>
<td>angle</td>
<td>Angle of the ellipse (in radians).</td>
</tr>
</tbody>
</table>

The documentation for this class was generated from the following files:

- oncovation/Interfaz/Interfaz/hypothesis/ellipse.h
- oncovation/Interfaz/Interfaz/hypothesis/ellipse.cpp

3.13 FeatureMatcher Class Reference

Class that offers various methods to extract 3D points from two 2D stereoscopic images.

#include <FeatureMatcher.h>
Public Member Functions

- void setLeftM (Mat &M)
  
  Sets the M parameter matrix from the camera calibration.

- void setSrcLeft (Mat &srcLeft)
  
  Sets the left source image.

- void setSrcRight (Mat &srcRight)
  
  Sets the right source image.

- void setMaskLeft (Mat &maskLeft)
  
  Sets the left mask image.

- void setMaskRight (Mat &maskRight)
  
  Sets the right mask image.

- void setHolesLeft (Mat &holesLeft)
  
  Sets the left holes image.

- void setHolesRight (Mat &holesRight)
  
  Sets the right holes image.

- void calculateGradient ()
  
  Calculates the sin, cos and magnitude of the gradient.

- void showMagnitudeHistograms (int row)
  
  Shows the magnitude histograms of the left and right images for a single row.

- void showAngleHistograms (int row)
  
  Shows the sinus histograms of the left and right images for a single row.

- void showBorderHistograms (int row, int radius)
  
  Shows the border histograms of the left and right image for a single row with a given radius.

- void showDisparityHistogram ()
  
  Shows the disparity histogram of the results of the 3D point generating methods.

- void writeData (int row)
  
  Writes the data for a single row to a text file.

- void optimizeMatches (int row)
  
  Matches the horizontal edge borders for a single row.

- void optimizeHoleMatches (int row)
  
  Matches the hole areas for a single row.

- void optimizeMatchesColor (int row)
  
  Matches the hole areas for a single row.

- bool isSolution (FeatureTreeNode &node, int endLeft, int endRight)
  
- bool isFactible (FeatureTreeNode &node, int endLeft, int endRight)
  
- float getUpperBound (FeatureTreeNode &node, int endLeft, int endRight)
  
- bool getRowLimits (int row, int &startLeft, int &endLeft, int &startRight, int &endRight)
  
- bool getRowLimitsFromCenter (int row, int colL, int colR, int &startLeft, int &endLeft, int &startRight, int &endRight)
  
  Returns the limits of a row starting from the center of the row.

- double getMaxMagnitude (int row, int id)
  
- double getMaxHorMagnitude (int row, int id)
  
- void unProjectPoint (const Point2f &pointL, const Point2f &pointR, Point3f &point3d)
  
- void filterPoints (int threshold)
  
  Removes the points whose disparity is less than the given threshold.

- int calculateOtsuThreshold ()
  
  Calculates the Otsu threshold for the disparity histogram of the solutions.

- void reset ()
  
  Cleans all data.

- void matchLowestPoint ()
  
- void matchLowestGrape (int maxDispDif=10)
  
- int getRows ()
  
  Returns the number of rows in the image (height)
Public Attributes

- **int** `maxDisp`  
  Maximum valid disparity for the generated points.

- **int** `minDisp`  
  Minimum valid disparity for the generated points.

- **vector** `<` `Point3f` `>` `m_points`  
  Vector where the edge points generated by the different methods are stored.

- **vector** `<` `Point3f` `>` `m_holePoints`  
  Vector where the hole points generated by `optimizeHoleMatches()` method are stored.

### 3.13.1 Detailed Description

Class that offers various methods to extract 3D points from two 2D stereoscopic images.  
In order to work correctly, the class has to be carefully initialized by providing the camera calibration, the source images, the background masks and the hole images (segmented images where the black areas from inbetween grapes are highlighted). When these images are provided, `calculateGradient()` must be called to finish the initialization.

### 3.13.2 Member Function Documentation

#### 3.13.2.1 `void FeatureMatcher::calculateGradient ( )`
Calculates the sin, cos and magnitude of the gradient.  
This method must be called after setting the needed images, but before using any other methods. Gradient X Gradient Y

#### 3.13.2.2 `void FeatureMatcher::filterPoints ( int threshold )`
Removes the points whose disparity is less than the given threshold.  
This effectively removes the points which are beyond a specific Z value.

#### 3.13.2.3 `double FeatureMatcher::getMaxHorMagnitude ( int row, int id )`
Returns the maximum value of a row of the horizontal gradient magnitude matrix.  
**Parameters**

<table>
<thead>
<tr>
<th>row</th>
<th>Row to evaluate.</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>0 for left matrix, 1 for right matrix.</td>
</tr>
</tbody>
</table>

#### 3.13.2.4 `double FeatureMatcher::getMaxMagnitude ( int row, int id )`
Returns the maximum value of a row of the gradient magnitude matrix.  
**Parameters**

<table>
<thead>
<tr>
<th>row</th>
<th>Row to evaluate.</th>
</tr>
</thead>
</table>
3.13.2.5 bool FeatureMatcher::getRowLimits ( int row, int & startLeft, int & endLeft, int & startRight, int & endRight )

Returns the limits of a row in both images.

Parameters

<table>
<thead>
<tr>
<th>row</th>
<th>The row to evaluate.</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>startLeft</td>
</tr>
<tr>
<td>out</td>
<td>endLeft</td>
</tr>
<tr>
<td>out</td>
<td>startRight</td>
</tr>
<tr>
<td>out</td>
<td>endRight</td>
</tr>
</tbody>
</table>

Returns

false if there are no pixels in the given row, true otherwise.

3.13.2.6 bool FeatureMatcher::getRowLimitsFromCenter ( int row, int colL, int colR, int & startLeft, int & endLeft, int & startRight, int & endRight )

Returns the limits of a row starting from the center of the row.

Because this method starts from the center, it may no reach the absolute limit of the row, but stop when it finds a non-object pixel, even though there are more object pixels further away.

Useful for looking for points of a specific border grape.

Parameters

<table>
<thead>
<tr>
<th>row</th>
<th>The row to evaluate.</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>startLeft</td>
</tr>
<tr>
<td>out</td>
<td>endLeft</td>
</tr>
<tr>
<td>out</td>
<td>startRight</td>
</tr>
<tr>
<td>out</td>
<td>endRight</td>
</tr>
</tbody>
</table>

Returns

false if there are no pixels in the given row, true otherwise.

3.13.2.7 float FeatureMatcher::getUpperBound ( FeatureTreeNode & node, int endLeft, int endRight )

Gets an estimated upper bound of the cost of a given node.

Parameters

<table>
<thead>
<tr>
<th>node</th>
<th>The node to evaluate</th>
</tr>
</thead>
<tbody>
<tr>
<td>endLeft</td>
<td>The last valid pixel of the left image.</td>
</tr>
<tr>
<td>endRight</td>
<td>The last valid pixel of the right image.</td>
</tr>
</tbody>
</table>

3.13.2.8 bool FeatureMatcher::isFactible ( FeatureTreeNode & node, int endLeft, int endRight )

Determines if a node of the dynamic programming tree is factible
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node to evaluate</td>
</tr>
<tr>
<td>endLeft</td>
<td>The last valid pixel of the left image.</td>
</tr>
<tr>
<td>endRight</td>
<td>The last valid pixel of the right image.</td>
</tr>
</tbody>
</table>

3.13.2.9 bool FeatureMatcher::isSolution ( FeatureTreeNode & node, int endLeft, int endRight )

Determines if a node of the dynamic programming tree is a solution.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node to evaluate</td>
</tr>
<tr>
<td>endLeft</td>
<td>The last valid pixel of the left image.</td>
</tr>
<tr>
<td>endRight</td>
<td>The last valid pixel of the right image.</td>
</tr>
</tbody>
</table>

3.13.2.10 void FeatureMatcher::matchLowestGrape ( int maxDispDif = 10 )

Finds the most possible points which correspond to the lowest grape.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxDispDif</td>
<td>Maximum disparity difference.</td>
</tr>
</tbody>
</table>

3.13.2.11 void FeatureMatcher::matchLowestPoint ( )

Finds the lowest point of the grapes

Warning

DEPRECATED

3.13.2.12 void FeatureMatcher::optimizeMatches ( int row )

Matches the horizontal edge borders for a single row.

This method uses dynamic programming to try and match the borders of the left image with the ones on the right image, so 3D points can be created.

3.13.2.13 void FeatureMatcher::optimizeMatchesColor ( int row )

Warning

DEPRECATED

3.13.2.14 void FeatureMatcher::showBorderHistograms ( int row, int radius )

Shows the border histograms of the left and right image for a single row with a given radius.

These histograms show how many border pixels are near each pixel of the row.

3.13.2.15 void FeatureMatcher::unProjectPoint ( const Point2f & pointL, const Point2f & pointR, Point3f & point3d )

Returns a 3D point given two 2D points
### Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>pointL</strong></td>
<td>Left point.</td>
</tr>
<tr>
<td><strong>pointR</strong></td>
<td>Right point.</td>
</tr>
<tr>
<td><strong>out</strong></td>
<td><strong>point3D</strong></td>
</tr>
</tbody>
</table>

#### 3.13.2.16 void FeatureMatcher::writeData ( int row )

Writes the data for a single row to a text file.
The data will be written to a file called ‘data.txt’ on the current working directory.
The documentation for this class was generated from the following files:

- oncovation/Interfaz/Interfaz/FeatureMatcher.h
- oncovation/Interfaz/Interfaz/FeatureMatcher.cpp

### 3.14 FeatureTreeNode Struct Reference

Node of the dynamic programming tree for matching.

```cpp
#include <featuretreenode.h>
```

#### Public Member Functions

- **FeatureTreeNode** (int iL, int iR, float c)

#### Public Attributes

- int **indexL**
  
  *Index in the left image.*
- int **indexR**
  
  *Index in the right image.*
- float **cost**
  
  *Cost of the solution up to this point.*
- float **lastZ**
  
  *Last Z calculated.*
- vector<IndexPair> **matches**
  
  *Matches found.*

#### 3.14.1 Detailed Description

Node of the dynamic programming tree for matching.

Used in the **FeatureMatcher** class.

The documentation for this struct was generated from the following files:

- oncovation/Interfaz/Interfaz/featuretreenode.h
- oncovation/Interfaz/Interfaz/featuretreenode.cpp
3.15  GLWidget Class Reference

Widget where the 3D scene is represented.

```
#include <glwidget.h>
```

Inheritance diagram for GLWidget:

```
QGLWidget

GLWidget
```

### Public Slots

- `void resetCamera ()`
  
  Restores the camera to the original position.

- `void centerCamera ()`
  
  Centers the camera on the selected object.

### Signals

- `void selectionChanged ()`
  
  This signal is emitted when the selected object changes.

- `void selectionChanged (int index)`
  
  This signal is emitted when the selected object changes.

### Public Member Functions

- `GLWidget (GrapeSupervisor *app, ApplicationData *data, QWidget *parent=0)`
- `void initializeGL ()`
  
  Initializes the OpenGL context.

- `void resizeGL (int w, int h)`
  
  Resizes the OpenGL area.

- `void paintGL ()`
  
  Draws the 3D scene.

- `void mousePressEvent (QMouseEvent *event)`
  
  Manages the mouse button press event.

- `void mouseMoveEvent (QMouseEvent *event)`
  
  Manages the mouse move event.

- `void mouseReleaseEvent (QMouseEvent *event)`
  
  Manages the mouse button release event.

- `void wheelEvent (QWheelEvent *event)`
  
  Manages the mouse wheel event.

- `int getSelectedIndex ()`
  
  Returns the index of the currently selected object.

- `void setSelectedIndex (int index)`

### 3.15.1 Detailed Description

Widget where the 3D scene is represented.
3.15.2 Member Function Documentation

3.15.2.1 void GLWidget::setSelectedIndex ( int index )

Selects an object.

Parameters

| index | Index of the object to select. |

The documentation for this class was generated from the following files:

- oncovation/Interfaz/Interfaz/glwidget.h
- oncovation/Interfaz/Interfaz/GeneratedFiles/Debug/moc_glwidget.cpp
- oncovation/Interfaz/Interfaz/GeneratedFiles/Release/moc_glwidget.cpp
- oncovation/Interfaz/Interfaz/glwidget.cpp

3.16 GrapeSupervisor Class Reference

Grape reconstruction supervisor.

#include <grapesupervisor.h>

Inheritance diagram for GrapeSupervisor:

```
Supervisor

GrapeSupervisor
```

Public Member Functions

- GrapeSupervisor (const ApplicationData &data)
- virtual void onStart ()
  The initial actions are added here.
- virtual bool hasFinished ()
- void getSpine (Vec3f &spine1, Vec3f &spine2)
- void setSpine (const Vec3f &spine1, const Vec3f &spine2)

Additional Inherited Members

3.16.1 Detailed Description

Grape reconstruction supervisor.

This class makes use of the Action system defined by the Supervisor class.

3.16.2 Constructor & Destructor Documentation

3.16.2.1 GrapeSupervisor::GrapeSupervisor ( const ApplicationData & data )

Constructor.
Parameters

| data | The supervisor will store a copy from this data. |

### 3.16.3 Member Function Documentation

#### 3.16.3.1 void GrapeSupervisor::getSpine ( Vec3f & spine1, Vec3f & spine2 ) [inline]

Sets the current spine of the grape.

Parameters

| out spine1 | Top point of the spine. |
| out spine2 | Bottom point of the spine. |

Warning

The spine is currently unused, but might be useful in the future.

#### 3.16.3.2 bool GrapeSupervisor::hasFinished ( ) [virtual]

Returns always false. This means the supervisor will finish when there are no actions left.

Implements Supervisor.

#### 3.16.3.3 void GrapeSupervisor::setSpine ( const Vec3f & spine1, const Vec3f & spine2 ) [inline]

Sets the current spine of the grape.

Parameters

| spine1 | Top point of the spine. |
| spine2 | Bottom point of the spine. |

Warning

The spine is currently unused, but might be useful in the future.

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/grapesupervisor.h
- oncovision/Interfaz/Interfaz/grapesupervisor.cpp

### 3.17 HoughAction Class Reference

This Action creates 3D objects through the Hough method.

```cpp
#include <houghaction.h>
```

Inheritance diagram for HoughAction:
Public Member Functions

- **HoughAction** (ActionParams &params, float priority)
- virtual bool run ()
- void setLimits (double xMin, double xMax, double xStep, double yMin, double yMax, double yStep, double zMin, double zMax, double zStep, double rMin, double rMax, double rStep)
  
  Sets limits and resolution of the Hough domain.

Additional Inherited Members

3.17.1 Detailed Description

This Action creates 3D objects through the Hough method.

Using the FeatureMatcher, this action obtains the 3D points of the horizontal edges of the images. These points are then passed to the SphereGenerator which performs Hough on them to extract 3D hypotheses. The action finally creates Object3D from the valid hypotheses.

3.17.2 Member Function Documentation

3.17.2.1 bool HoughAction::run ( ) [virtual]

This method must contain the operations specific to each Action.

Returns

false if there was an error, true otherwise.

Implements Action.

The documentation for this class was generated from the following files:

- oncovation/Interfaz/Interfaz/houghaction.h
- oncovation/Interfaz/Interfaz/houghaction.cpp

3.18 HoughSolution Struct Reference

Struct that contains the info necessary to represent a solution of the Hough method.

#include <spheregenerator.h>

Public Member Functions

- **HoughSolution** (Vec4d sol, int v)

Public Attributes

- Vec4d **solution**
  
  Parameters of the solution sphere.
- int **votes**
  
  Votes obtained in the Hough method.
- bool **valid**
  
  If its a valid solution.
3.18.1 Detailed Description

Struct that contains the info necessary to represent a solution of the Hough method.

The documentation for this struct was generated from the following file:

- oncovision/Interfaz/Interfaz/sphereregenerator.h

3.19 Hypothesis::Image2D Class Reference

This class represents a scene of Object2D which are organized as a graph.

`#include <image2d.h>`

Public Member Functions

- `double getIndex ()`
- `Mat getDistances ()`
- `void setDistances (const Mat &distances)`
- `Mat getMask ()`
- `Mat getMaskAroundObject (Object2D *object, double offset=-1.0)`
- `bool addObject (Object2D *object)`
- `int findObject (Object2D *object)`
- `int findObject (int ID)`
- `Object2D *getObject (int position)`
- `int getObjectCount ()`
- `bool setObject (int position, Object2D *object)`
- `bool removeObject (int position)`
- `bool removeObject (Object2D *object)`
- `void clear ()`
- `bool sortObjectsVectorWithOrder (int *indexOrder, int indexCount)`
- `void makeLinks (int position, double distanceThreshold)`
- `void makeLinks (Object2D *object, double distanceThreshold)`
- `void globalMakeLinks (double distanceThreshold)`
- `double updateGlobalDistance (Mat &distances, Mat &mask)`
- `double calcDistance (vector<Point2f> &ellipse, Mat &BW, Mat mask, double *confidence=NULL)`
- `void copyFrom (Image2D *img)`

Public Attributes

- `Mat m_points`

3.19.1 Detailed Description

This class represents a scene of Object2D which are organized as a graph.

3.19.2 Member Function Documentation

3.19.2.1 `bool Hypothesis::Image2D::addObject ( Object2D * object )`

Adds an object to the scene.
Parameters

| object | The object to be added. |

Returns

false if the maximum count object is reached, true otherwise.

3.19.2.2 \textbf{void Hypothesis::Image2D::clear ( )}
Removes all objects from the scene

Warning

the objects will be deleted.

3.19.2.3 \textbf{int Hypothesis::Image2D::findObject ( Object2D * object )}
Finds an object in the scene vector.

Parameters

| object | The object to be found. |

Returns

The position of the object in the vector, -1 if it's not in the scene.

3.19.2.4 \textbf{Mat Hypothesis::Image2D::getDistances ( )}
Returns the distance matrix for this image.

The distance matrix is an image whose pixels have their values set to the distance to the nearest edge. I.e, edge pixels have a distance of 0.

Returns

Current distance matrix.

3.19.2.5 \textbf{double Hypothesis::Image2D::getIndex ( )}
Calculates the total index of this image.

Returns

The total error index of this image.

3.19.2.6 \textbf{Mat Hypothesis::Image2D::getMask ( )} [inline]

Returns the distance mask.

The distance mask determines which pixels are taken into account to determine the image's error index.
3.19.2.7 Mat Hypothesis::Image2D::getMaskAroundObject ( Object2D * object, double offset = -1.0 )

Returns the a mask around a given object.

3.19.2.8 Object2D * Hypothesis::Image2D::getObject ( int position )

Gets an object from the scene.
Parameters

| position | Position of the object on the scene vector. |

Returns

A pointer to the object, 0 if the position is out of bounds.

3.19.2.9 int Hypothesis::Image2D::getObjectCount()

Returns the number of objects in the scene.

3.19.2.10 bool Hypothesis::Image2D::removeObject(int position)

Removes an object from the scene

Warning

the object will be deleted.

Parameters

| position | Position of the object to be removed. |

Returns

false if the position is out of bounds, true otherwise

3.19.2.11 bool Hypothesis::Image2D::removeObject(Object2D *object)

Removes an object from the scene

Warning

the object will be deleted.

Parameters

| object | Object to be removed |

Returns

false if the object is not in the scene, true otherwise

3.19.2.12 void Hypothesis::Image2D::setDistances(const Mat &distances)

Sets the distance matrix for this image.

The distance matrix is an image whose pixels have their values set to the distance to the nearest edge. I.e, edge pixels have a distance of 0.
Parameters

| distances | Distance matrix calculated beforehand. |

3.19.2.13 bool Hypothesis::Image2D::setObject ( int position, Object2D * object )

Replaces an object from the scene with a new one.

Warning

The old one will be deleted.

Parameters

| position | Position of the object to be replaced. |
| object   | The new object. |

Returns

false if the position is out of bounds, true otherwise

3.19.2.14 bool Hypothesis::Image2D::sortObjectsVectorWithOrder ( int * indexOrder, int indexCount )

Sorts the objects according to an index vector.

Parameters

| indexOrder | A vector of integers with the new object order. |
| indexCount | The length of the indexOrder vector. |

Returns

false if the indexOrder vector doesn’t match the number of objects in the scene or if the scene is empty, true otherwise.

The documentation for this class was generated from the following files:

- oncovation/Interfaz/Interfaz/hypothesis/image2d.h
- oncovation/Interfaz/Interfaz/hypothesis/image2d.cpp

3.20 IndexPair Struct Reference

Pair of pixels on a row and the cost of matching them.

#include <featuretreenode.h>

Public Member Functions

- IndexPair (int _l, int _r, float _o)

Public Attributes

- int l
  
  Left pixel.
3.21 Hypothesis::Node::Link Struct Reference

- int r
  
  Right pixel.
- float c
  
  Cost.

3.20.1 Detailed Description

Pair of pixels on a row and the cost of matching them.

The documentation for this struct was generated from the following file:

- oncovision/Interfaz/Interfaz/featuretreenode.h

3.21 Hypothesis::Node::Link Struct Reference

A link (or edge) consisting of a destination node and a cost.

```cpp
#include <node.h>
```

Public Member Functions

- Link (const Link &link)
- Link (Node *node_, double cost_)
- Link & operator= (const Link &link)

Public Attributes

- Node * node
- double cost

3.21.1 Detailed Description

A link (or edge) consisting of a destination node and a cost.

The documentation for this struct was generated from the following files:

- oncovision/Interfaz/Interfaz/hypothesis/node.h
- oncovision/Interfaz/Interfaz/hypothesis/node.cpp

3.22 MainWindow Class Reference

Main window. Contains the graphical interface.

```cpp
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:

```
QMainWindow
    
MainWindow
```

Generated on Thu Jan 2 2014 20:10:10 for Uvas by Doxygen
Public Slots

- void run ()
  
  Starts the object detection process.

- void save ()
  
  Opens the Save Data dialog.

- void load ()
  
  Opens the Load Data dialog.

Public Member Functions

- MainWindow (QWidget ∗parent=0, Qt::WFlags flags=0)
  
  Initializes all the actions and connects them to the slots.

- void createToolBar ()
  
  Initializes the toolbar.

- void saveData (const QString &fileName)
  
  Saves the data to a file in XML format.

- void loadData (const QString &fileName)
  
  Loads the data from a XML file.

3.22.1 Detailed Description

Main window. Contains the graphical interface.

It acts as a container of the interface and owns the main instance of the Grape Application.

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/mainwindow.h
- oncovision/Interfaz/Interfaz/mainwindow.cpp

3.23 ManualStageWidget Class Reference

Widget that contains the interface for editing the scene and its projections.

#include <manualstagewidget.h>

Inheritance diagram for ManualStageWidget:

```
QWidget

ManualStageWidget
```

Public Slots

- void addCamera ()
  
  Adds a new camera to the interface.

- void deleteCamera ()
  
  Deletes the last camera from the interface.

- void addObject ()
Shows a dialog for adding an object.

- void deleteSelectedObject ()
  Deletes the selected object.
- void copySelectedObject ()
  Makes a copy of the selected object with the same parameters.

Public Member Functions

- ManualStageWidget (ApplicationData ∗data, GrapeSupervisor ∗app, QWidget ∗parent=0)
  Constructor.
- ~ManualStageWidget ()
  Destructor.
- void keyPressEvent (QKeyEvent ∗ev)
  Function that handles the key events.
- ApplicationData ∗getApplicationData ()
  Return the data the user is working with.
- Viewport2D ∗getViewport2D (int index)
- Viewport3D ∗getViewport3D ()
  Return the 3D viewport.
- void addObject (double coords[4], bool optimize=false)
- GrapeSupervisor ∗getSupervisor ()
  Returns a pointer to the supervisor.

3.23.1 Detailed Description

Widget that contains the interface for editing the scene and its projections.

The interface is organized in a hierarchical fashion. The ManualStageWidget is divided into a bottom 3D viewport and an arbitrary number of 2D viewports at the top, corresponding to each one of the cameras we are working with. This means that usually there will be two 2D viewports (left and right cameras).

This widget also includes a sidebar that provides object management functionality (add, copy and delete object).

3.23.2 Constructor & Destructor Documentation

3.23.2.1 ManualStageWidget::ManualStageWidget ( ApplicationData ∗data, GrapeSupervisor ∗app, QWidget ∗parent = 0 )
Constructor.

Parameters

| data | A pointer to the application data. |
| app  | A pointer to the grape supervisor. |

3.23.3 Member Function Documentation

3.23.3.1 void ManualStageWidget::addObject ( double coords[4], bool optimize = false )

Adds a new object to the 3D scene.
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>coords</td>
<td>Parameters of the new object [x, y, z, r]</td>
</tr>
<tr>
<td>optimize</td>
<td>The object’s parameters will be optimized if set to true.</td>
</tr>
</tbody>
</table>

#### 3.23.3.2 Viewport2D

Return a specific viewport

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>Index of the viewport to return</td>
</tr>
</tbody>
</table>

#### 3.23.3.3 void ManualStageWidget::keyPressEvent (QKeyEvent *ev)

Function that handles the key events.

In its current state only captures the “Delete” key to delete the currently selected object. More shortcuts may be added later.

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/manualstagewidget.h
- oncovision/Interfaz/Interfaz/manualstagewidget.cpp

### 3.24 Hypothesis::Node Class Reference

This class holds all the necessary data and methods to use it as a Node of a graph.

```cpp
#include <node.h>
```

Inheritance diagram for Hypothesis::Node:

```
Hypothesis::Node
   ▼
  /   
Hypothesis::Object2D Hypothesis::Object3D
```

**Classes**

- struct Link
  
  A link (or edge) consisting of a destination node and a cost.

**Public Member Functions**

- void operator= (const Node &der)
- Node (const Node &der)
- int ID ()
- void setID (int id)
- int type ()
- int getLinkCount ()
- bool insertLink (const Link &link)
- bool insertLink (Node *node, double cost)
3.24 Hypothesis::Node Class Reference

- int findNode (Node *node)
- bool removeLink (const Link &link)
- bool removeLink (Node *node)
- void clearLinks ()
- bool setCost (int position, double cost)
- bool setCost (Node *node, double cost)
- bool sortLinksVectorWithOrder (int *indexOrder, int indexCount)

Public Attributes

- vector<Link> m_links
  Links to the neighbors.
- double m_confidence
  Confidence.

Static Public Attributes

- static const unsigned int MAX_LINKS = 13
  Maximum link count.
- static const int TYPE_NODE = 0
  Node types.
- static const int TYPE_OBJECT2D = 1
- static const int TYPE_OBJECT3D = 2

Protected Attributes

- int m_id
  Identifier.
- int m_type
  Type of the node (Object3D or Object2D)

3.24.1 Detailed Description

This class holds all the necessary data and methods to use it as a Node of a graph.

3.24.2 Member Function Documentation

3.24.2.1 void Hypothesis::Node::clearLinks ( )

Removes all links from the link list.

3.24.2.2 int Hypothesis::Node::findNode ( Node * node )

Finds a node on the link list.

Parameters
node | The node to find.

Returns

The position of the link on the list, -1 if there's no link to that node.

3.24.2.3 int Hypothesis::Node::getLinkCount ( )

Gets the number of the links to neighbors.

Returns

The size of the link list.

3.24.2.4 int Hypothesis::Node::ID ( )

Returns the node identifier.

Returns

An integer identifying the node.

3.24.2.5 bool Hypothesis::Node::insertLink ( const Link & link )

Inserts a link to another node.

Parameters

| link | The link to insert |

Returns

false if the link max has been reached, true otherwise.

3.24.2.6 bool Hypothesis::Node::insertLink ( Node * node, double cost )

Inserts a link to another node.

Parameters

| node | The node the link will point to. |
| cost | The cost of the link. |

Returns

false if the link max has been reached, true otherwise.

3.24.2.7 bool Hypothesis::Node::removeLink ( const Link & link )

Removes a link from the link list.
3.24 Hypothesis::Node Class Reference

Parameters

| link | The link to remove (only uses the Node field, so the cost doesn’t matter). |

Returns

false if the link doesn’t exist on the list, true otherwise.

3.24.2.8 bool Hypothesis::Node::removeLink ( Node * node )

Removes a link from the link list.

Parameters

| node | The node of the link to be removed. |

Returns

false if the link doesn’t exist on the list, true otherwise.

3.24.2.9 bool Hypothesis::Node::setCost ( int position, double cost )

Sets the cost of a link.

Parameters

| position | The position of the link on the list. |

Returns

false if the link doesn’t exist on the list, true otherwise.

3.24.2.10 bool Hypothesis::Node::setCost ( Node * node, double cost )

Sets the cost of a link.

Parameters

| node | The node the link points to. |

Returns

false if the link doesn’t exist on the list, true otherwise.

3.24.2.11 void Hypothesis::Node::setID ( int id )

Sets the node identifier.

Parameters

| id | The new identifier |

3.24.2.12 bool Hypothesis::Node::sortLinksVectorWithOrder ( int * indexOrder, int indexCount )

Sorts the links according to an index vector.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>indexOrder</td>
<td>A vector of integers with the new link order.</td>
</tr>
<tr>
<td>indexCount</td>
<td>The length of the indexOrder vector.</td>
</tr>
</tbody>
</table>

Returns

false if the indexOrder vector doesn't match the number of links or if there are no links to order on the node, true otherwise.

3.24.2.13 int Hypothesis::Node::type()

Returns the type of the node.

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/hypothesis/node.h
- oncovision/Interfaz/Interfaz/hypothesis/node.cpp

3.25 Hypothesis::Object2D Class Reference

This class represents a 2D object on an image. It is modelled as a set of ellipses and its tangents.

#include <object2d.h>

Inheritance diagram for Hypothesis::Object2D:

```
+-------------------+-------------------+
| Hypothesis::Node  | Hypothesis::Object2D |
|                   |                   |
```

Public Member Functions

- void getColor (double color[3]) const
- void setColor (double color[3])
- void setColor (double r, double g, double b)
- double getIndex () const
- void setIndex (double val)
- double getHiddenIndex () const
- void setHiddenIndex (double val)
- void calcHiddenIndex ()
- double getDistanceToCamera () const
- void setDistanceToCamera (double value)
- void addEllipse (Ellipse ellipse)
- void addEllipse (double params[5])
- void addEllipse (double x, double y, double a, double b, double angle)
- Ellipse * getEllipse (int index) const
- int getEllipseCount () const
- void deleteEllipse (int index)
- void clear ()
- void calcOverlappingFront ()
- void calcOverlappingBack ()
• vector<Object2D*> getOverlappingListFront()
• vector<Object2D*> getOverlappingListBack()
• void calcInFrontOf()
• double calcDistance(Object2D* object)
• void updateLinks()
• bool checkOverlapping(Object2D** front, int count, Mat &resImage, double &minX, double &minY)
• void calcMinMax(const double* points, int count, double &minX, double &minY, double &maxX, double &maxY)
• void fillEllipse(Mat &image, double params[5], double minX, double minY)
• void fillEllipse(Mat &image, Object2D* obj, double minX, double minY)
• vector<Point2f> overlappingCalculation(Object2D** front, int count)
• void copyFrom(Object2D* obj)

Static Public Member Functions

• static Object2D* getNodeAsObject2D(Node* node)

Additional Inherited Members

3.25.1 Detailed Description

This class represents a 2D object on a image. It is modelled as a set of ellipses and its tangents.
WARNING: Right now it’s assumed in a lot of cases that the Object has one and only one ellipse. This has to be changed sometime to support multiple ellipses and its tangents.

3.25.2 Member Function Documentation

3.25.2.1 void Hypothesis::Object2D::addEllipse( Ellipse* ellipse )

Adds a new ellipse to the object.
Parameters

| ellipse | A pointer to the ellipse that must be added. From now on this instance will own the ellipse. |

3.25.2.2 void Hypothesis::Object2D::addEllipse( double params[5] )

Adds a new ellipse to the object.
Parameters

| params | A 5-element vector with the parameters of the ellipse to be added (cx, cy, ra, rb, angle). |

3.25.2.3 void Hypothesis::Object2D::addEllipse( double x, double y, double a, double b, double angle )

Adds a new ellipse to the object.
Parameters

<p>| x | X coordinate of the center of the ellipse. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>Y coordinate of the center of the ellipse.</td>
</tr>
<tr>
<td>$a$</td>
<td>Major axis.</td>
</tr>
<tr>
<td>$b$</td>
<td>Minor axis.</td>
</tr>
<tr>
<td>angle</td>
<td>Angle of the ellipse (in radians).</td>
</tr>
</tbody>
</table>

3.25.2.4  double Hypothesis::Object2D::calcDistance ( Object2D * object )

Calculates the distance to another object.

3.25.2.5  void Hypothesis::Object2D::calcHiddenIndex ( )

Calculate this object’s hidden index.

3.25.2.6  void Hypothesis::Object2D::calcInFrontOf ( )

Sorts the neighbours of this object given its distance to camera and determines which one is this object in front of (i.e. which neighbour is immediately behind him).

3.25.2.7  void Hypothesis::Object2D::calcMinMax ( const double * points, int count, double & minX, double & minY, double & maxX, double & maxY )

Helper function that calculates the bounding rectangle of the given points (generally ellipse points).

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>points</td>
<td>Array of points.</td>
</tr>
<tr>
<td>count</td>
<td>Number of points.</td>
</tr>
<tr>
<td>minX[out]</td>
<td>X coordinate of the bottom-left of the rect.</td>
</tr>
<tr>
<td>minY[out]</td>
<td>Y coordinate of the bottom-left of the rect.</td>
</tr>
<tr>
<td>maxX[out]</td>
<td>X coordinate of the top-right of the rect.</td>
</tr>
<tr>
<td>maxY[out]</td>
<td>Y coordinate of the top-right of the rect.</td>
</tr>
</tbody>
</table>

3.25.2.8  void Hypothesis::Object2D::calcOverlappingBack ( )

Calculates, from the objects that are in behind this object, which ones overlap with this object and add them to the overlappedBack vector.

3.25.2.9  void Hypothesis::Object2D::calcOverlappingFront ( )

Calculates, from the objects that are in front of this object, which ones overlap with this object and add them to the overlappedFront vector.

3.25.2.10 bool Hypothesis::Object2D::checkOverlapping ( Object2D ** front, int count, Mat & resImage, double & minX, double & minY )

Given all the objects in front of this one, checks if there is overlapping and generates the overlapped binary image. If there is no overlap, the generated image will correspond to this object’s silhouette.
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>front</code></td>
<td>Array of the objects to check the overlapping against.</td>
</tr>
<tr>
<td><code>count</code></td>
<td>Number of objects.</td>
</tr>
<tr>
<td><code>minX[out]</code></td>
<td>X coordinate of the starting point of the image.</td>
</tr>
<tr>
<td><code>minY[out]</code></td>
<td>Y coordinate of the starting point of the image.</td>
</tr>
</tbody>
</table>

#### 3.25.2.11 void Hypothesis::Object2D::clear ( )

Deletes all the ellipses.

#### 3.25.2.12 void Hypothesis::Object2D::copyFrom ( Object2D * obj )

Copies the object's properties from a different one.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>obj</code></td>
<td>The object to copy the properties from.</td>
</tr>
</tbody>
</table>

#### 3.25.2.13 void Hypothesis::Object2D::deleteEllipse ( int index )

Deletes an ellipse given its index.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>index</code></td>
<td>The position of the ellipse in the ellipse vector.</td>
</tr>
</tbody>
</table>

#### 3.25.2.14 void Hypothesis::Object2D::fillEllipse ( Mat & image, double params[5], double minX, double minY )

Helper function that draws a filled ellipse on a given image.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>image</code></td>
<td>Image where the ellipse will be drawn.</td>
</tr>
<tr>
<td><code>params</code></td>
<td>A 5-element vector with the parameters of the ellipse to be drawn (cx, cy, ra, rb, angle).</td>
</tr>
<tr>
<td><code>minX[out]</code></td>
<td>X coordinate of the bottom-left of the image.</td>
</tr>
<tr>
<td><code>minY[out]</code></td>
<td>Y coordinate of the bottom-left of the image.</td>
</tr>
</tbody>
</table>

#### 3.25.2.15 void Hypothesis::Object2D::getColor ( double color[3] ) const

Returns the object color (RGB, ranging from 0.0 to 1.0).

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>out</code></td>
<td>Vector of 3 elements where the color will be returned.</td>
</tr>
</tbody>
</table>

#### 3.25.2.16 double Hypothesis::Object2D::getDistanceToCamera ( ) const

Retruns the distance to camera.

**Returns**

- This object's distance to camera.
3.25.2.17 Ellipse * Hypothesis::Object2D::getEllipse ( int index ) const

Returns an ellipse given its index.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>The position of the ellipse in the ellipse vector.</td>
</tr>
</tbody>
</table>

Returns

A pointer to the desired ellipse.

3.25.2.18 int Hypothesis::Object2D::getEllipseCount ( ) const

Returns the number of the ellipses of this object.

Returns

The size of the ellipse vector.

3.25.2.19 double Hypothesis::Object2D::getHiddenIndex ( ) const

Returns the hidden index.

Returns

This object's hidden index.

3.25.2.20 double Hypothesis::Object2D::getIndex ( ) const

Returns the error index.

Returns

This object's index.

3.25.2.21 Object2D * Hypothesis::Object2D::getNodeAsObject2D ( Node * node ) [static]

Returns a node as an Object2D pointer.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>The node to be cast as Object2D</td>
</tr>
</tbody>
</table>

Returns

The given node casted as Object2D, or 0 if the node is not a valid Object2D.

3.25.2.22 vector<Object2D *> Hypothesis::Object2D::getOverlappingListBack ( )

Retruns the pointers to the objects that overlap and are behind this object.

3.25.2.23 vector<Object2D *> Hypothesis::Object2D::getOverlappingListFront ( )

Retruns the pointers to the objects that overlap and are in front of this object.
3.25.2.24 \texttt{vector\textless\ Point2f \textgreater\ \texttt{Hypothesis::Object2D::overlappingCalculation ( Object2D * front, int count )}}

Calculates the boundaries of the overlapping against all the front objects.
Parameters

| front | Array of the objects to check the overlapping against. |
| count | Number of objects. |

Returns

Vector of points.

3.25.2.25 void Hypothesis::Object2D::setColor ( double color[3] )

Sets the object color (RGB, ranging from 0.0 to 1.0).
Parameters

| position | Vector of 3 elements holding the new color. |

3.25.2.26 void Hypothesis::Object2D::setColor ( double r, double g, double b )

Sets the object color (RGB, ranging from 0.0 to 1.0).
Parameters

| r | The r component of the new color. |
| g | The g component of the new color. |
| b | The b component of the new color. |

3.25.2.27 void Hypothesis::Object2D::setDistanceToCamera ( double value )

Sets this object's distance to camera.
Parameters

| value | The new distance. |

3.25.2.28 void Hypothesis::Object2D::setHiddenIndex ( double val )

Sets this object's hidden index.
Parameters

| val | The new hidden index. |

3.25.2.29 void Hypothesis::Object2D::setIndex ( double val )

Sets this object's index.
Parameters

| val | The new error index. |

3.25.2.30 void Hypothesis::Object2D::updateLinks ( )

Sorts the links and calculates the overlapping.
3.26 Hypothesis::Object3D Class Reference

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/hypothesis/object2d.h
- oncovision/Interfaz/Interfaz/hypothesis/object2d.cpp

3.26 Hypothesis::Object3D Class Reference

This class represents a 3D object on a scene. It is modelled as a polysphere.

#include <object3d.h>

Inheritance diagram for Hypothesis::Object3D:

```
Hypothesis::Object3D
Hypothesis::Node
```

Public Member Functions

- Object3D (int id=0)
- Object3D (double position[3], int id=0)
- Object3D (double x, double y, double z, int id=0)
- double getIndex () const
- void setIndex (double val)
- void calcIndex ()
- void getPosition (double position[3]) const
- void setPosition (double coords[3])
- void setPosition (double x, double y, double z)
- void getColor (double color[3]) const
- void setColor (double color[3])
- void setColor (double r, double g, double b)
- vector<Sphere*> getSpheres ()
- int getSphereCount () const
- void addSphere (double coords[4])
- void addSphere (double position[3], double r)
- void addSphere (double x, double y, double z, double r=1.0)
- Sphere* getSphere (int index) const
- void deleteSphere (int index)
- void clear ()
- double calcDistance (Object3D *object)
- void calcDistanceToNeighbours ()
- void updateLinks ()
- double calcDistanceToCamera (Camera *cam)
- void copyFrom (Object3D *object)

Static PublicMember Functions

- static Object3D* getNodeAsObject3D (Node *node)
Additional Inherited Members

3.26.1 Detailed Description

This class represents a 3D object on a scene. It is modelled as a polysphere.

3.26.2 Member Function Documentation

3.26.2.1 void Hypothesis::Object3D::addSphere ( double coords[4] )

Adds a new sphere to the object.
Parameters

| coords | Vector of 4 elements holding the new sphere's coordinates (x, y, z, radius). |

3.26.2.2 void Hypothesis::Object3D::addSphere ( double position[3], double r )

Adds a new sphere to the object.
Parameters

| position | Vector of 3 elements holding the new sphere's position (x, y, z). |
| r | Radius of the new sphere. |

3.26.2.3 void Hypothesis::Object3D::addSphere ( double x, double y, double z, double r = 1.0 )

Adds a new sphere to the object.
Parameters

| x | The x coordinate of the new sphere's position. |
| y | The y coordinate of the new sphere's position. |
| z | The z coordinate of the new sphere's position. |
| r | Radius of the new sphere. |

3.26.2.4 double Hypothesis::Object3D::calcDistance ( Object3D * object )

Calculates the distance to another 3D object.
Parameters

| object | The 3D object we want to calculate the distance to. |

Returns
The distance calculated.

3.26.2.5 double Hypothesis::Object3D::calcDistanceToCamera ( Camera * cam )

Calculates this object's distance to a camera.
Parameters

| cam | The camera we want to measure the distance to. |

3.26.2.6 void Hypothesis::Object3D::calcDistanceToNeighbours()

Calculates the distances to all its neighbours and sets them as cost of the links.

3.26.2.7 void Hypothesis::Object3D::calcIndex()

Calculates the new error index.
This index is based on a weighted distance to its nearest object.

3.26.2.8 void Hypothesis::Object3D::clear()

Deletes all the spheres of the object.

3.26.2.9 void Hypothesis::Object3D::copyFrom(Object3D *object)

Copies the object’s properties from a different one.
Parameters

| object | The object to copy the properties from. |

3.26.2.10 void Hypothesis::Object3D::deleteSphere(int index)

Deletes a sphere of the object.
Parameters

| index | Position of the sphere on the spheres vector. |

3.26.2.11 double Hypothesis::Object3D::getColor(double color[3]) const

Returns the object color (RGB, ranging from 0.0 to 1.0).
Parameters

| out | position | Vector of 3 elements where the color will be returned. |

3.26.2.12 double Hypothesis::Object3D::getIndex() const

Returns the error index.

Returns

This object’s error index.

3.26.2.13 Object3D *Hypothesis::Object3D::getNodeAsObject3D(Node *node)[static]

Returns a node as a Object3D* pointer.
Parameters

| node | The node to be cast as Object3D |

Returns

The given node casted as Object3D, or 0 if the node is not a valid Object3D.


Returns the object position (x, y, z).

Parameters

| out | position | Vector of 3 elements where the position will be returned. |

3.26.2.15 Sphere * Hypothesis::Object3D::getSphere ( int index ) const

Returns a sphere of the object.

Parameters

| index | Position of the sphere on the spheres vector. |

Returns

A pointer to the sphere, or 0 if the index parameter is out of bounds.

3.26.2.16 int Hypothesis::Object3D::getSphereCount ( ) const

Returns the number of the spheres that form the object.

3.26.2.17 vector<Sphere *> Hypothesis::Object3D::getSpheres ( )

Returns a vector with the pointers to the spheres of the object.

Returns

A std::vector of Sphere pointers.

3.26.2.18 void Hypothesis::Object3D::setColor ( double color[3] )

Sets the object color (RGB, ranging from 0.0 to 1.0).

Parameters

| position | Vector of 3 elements holding the new color. |

3.26.2.19 void Hypothesis::Object3D::setColor ( double r, double g, double b )

Sets the object color (RGB, ranging from 0.0 to 1.0).
### 3.27 OptimizeObjectAction Struct Reference

This Action performs the parameter optimization on a 3D object and its neighbors.

```cpp
#include <optimizeobjectaction.h>
```

Inheritance diagram for OptimizeObjectAction:

```
  Action
    OptimizeObjectAction
```

---

### Parameters

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>The r component of the new color.</td>
</tr>
<tr>
<td>g</td>
<td>The g component of the new color.</td>
</tr>
<tr>
<td>b</td>
<td>The b component of the new color.</td>
</tr>
</tbody>
</table>

---

### 3.26.2.20 void Hypothesis::Object3D::setIndex ( double val )

Sets this object's index.

**Parameters**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>val</td>
<td>The new error index.</td>
</tr>
</tbody>
</table>

---

### 3.26.2.21 void Hypothesis::Object3D::setPosition ( double coords[3] )

Sets the object position.

**Parameters**

| position | Vector of 3 elements holding the new position. |

---

### 3.26.2.22 void Hypothesis::Object3D::setPosition ( double x, double y, double z )

Sets the object position.

**Parameters**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The x coordinate of the new position.</td>
</tr>
<tr>
<td>y</td>
<td>The y coordinate of the new position.</td>
</tr>
<tr>
<td>z</td>
<td>The z coordinate of the new position.</td>
</tr>
</tbody>
</table>

---

### 3.26.2.23 void Hypothesis::Object3D::updateLinks ( )

Updates the link information.

---

The documentation for this class was generated from the following files:

- `oncovision/Interfaz/Interfaz/hypothesis/object3d.h`
- `oncovision/Interfaz/Interfaz/hypothesis/object3d.cpp`
Public Member Functions

- **OptimizeObjectAction** *(ActionParams &params, float priority)*
- `virtual bool run ()`

Additional Inherited Members

3.27.1 Detailed Description

This **Action** performs the parameter optimization on a 3D object and its neighbors. The Object3D to optimize around must be specified through the **ActionParams**' "object" parameter. This action will create a copy of the original data with only the specified object and its neighbors. The optimization will be performed on this copy and then the resulting parameters will be copied back to the original data.

3.27.2 Member Function Documentation

3.27.2.1 `virtual bool OptimizeObjectAction::run ()` *

This method must contain the operations specific to each **Action**.

Returns
false if there was an error, true otherwise.

Implements **Action**.

The documentation for this struct was generated from the following file:

- oncovision/Interfaz/Interfaz/optimizeobjectaction.h

3.28 **OptSphereNLP Class Reference**

Class that represents a NLP problem for the optimization of the parameters of a sphere through its projections.

```cpp
#include <optspobrenl.h>
```

Inheritance diagram for OptSphereNLP:

```
  TOptSphereNLP
     |
     -- TNLP
```

Public Member Functions

- **OptSphereNLP** *(ApplicationData *data, Mat *masks)*
- `virtual bool get_nlp_info (Index &n, Index &m, Index &nnz_jac_g, Index &nnz_h_lag, IndexStyleEnum &index_style)`
- `virtual bool get_bounds_info (Index n, Number *x_l, Number *x_u, Index m, Number *g_l, Number *g_u)`
- `virtual bool get_starting_point (Index n, bool init_x, Number *x, bool init_z, Number *z_L, Number *z_U, Index m, bool init_lambda, Number *lambda)`
- `virtual bool eval_f (Index n, const Number *x, bool new_x, Number &obj_value)`
• virtual bool `eval_grad_f` (Index n, const Number *x, bool new_x, Number *grad_f)
• virtual bool `eval_g` (Index n, const Number *x, bool new_x, Index m, Number *g)
• virtual bool `eval_jac_g` (Index n, const Number *x, bool new_x, Index m, Index nele_jac, Index *iRow, Index *jCol, Number *values)
• virtual void finalize_solution (SolverReturn status, Index n, const Number *x, const Number *z_L, const Number *z_U, Index m, const Number *g, const Number *lambda, Number obj_value, const IpoptData *ip_data, IpoptCalculatedQuantities *ip_cq)
  - Storage of the solution and error control.
• virtual bool `intermediate_callback` (AlgorithmMode mode, Index iter, Number obj_value, Number inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du, Number alpha_pr, Index ls_trials, const IpoptData *ip_data, IpoptCalculatedQuantities *ip_cq)
  - In this callback we evaluate if the a solution has been reached.
• double `calcIndex` (double x, double y, double z, double r)
  - Calculates the error index of a sphere given its parameters.

Public Attributes

• double `m_bestObjValue`

3.28.1 Detailed Description

Class that represents a NLP problem for the optimization of the parameters of a sphere through its projections.

The optimizer will iteratively modify the parameters of the sphere and check the error value of its projections on the different cameras.

It implements the interface required so the problem can be solved by the Ipopt library. For a detailed description of the use of the methods by Ipopt, you can find its documentation at [http://www.coin-or.org/-Ipopt/documentation/node23.html](http://www.coin-or.org/-Ipopt/documentation/node23.html)

3.28.2 Member Function Documentation

3.28.2.1 double OptSphereNLP::calcIndex ( double x, double y, double z, double r )

Calculates the error index of a sphere given its parameters.

In this case, the error is calculated using the distance the projected ellipses to the edge pixels.

The documentation for this class was generated from the following files:

• oncovision/Interfaz/Interfaz/optspherenlp.h
• oncovision/Interfaz/Interfaz/optspherenlp.cpp

3.29 OptSpherePoints Class Reference

Class that represents a NLP problem for the optimization of the parameters of a sphere given some of its 3D points.

#include <optspherenlp.h>

Inheritance diagram for OptSpherePoints:

```
        TNLP
         |
         v
OptSpherePoints
```
Public Member Functions

- **OptSpherePoints** (double xMin, double xMax, double xStep, double yMin, double yMax, double yStep, double zMin, double zMax, double zStep, double rMin, double rMax, double rStep)
- virtual bool *get_nlp_info* (Index &n, Index &m, Index &nnz_jac_g, Index &nnz_h_lag, IndexStyleEnum &index_style)
- virtual bool *get_bounds_info* (Index n, Number ∗x_l, Number ∗x_u, Index m, Number ∗g_l, Number ∗g_u)
- virtual bool *get_starting_point* (Index n, bool init_x, Number ∗x, bool init_z, Number ∗z_L, Number ∗z_U, Index m, bool init_lambda, Number ∗λ)
- virtual bool *eval_f* (Index n, const Number ∗x, bool new_x, Number &obj_value)
- virtual bool *eval_grad_f* (Index n, const Number ∗x, bool new_x, Number ∗grad_f)
- virtual bool *eval_g* (Index n, const Number ∗x, bool new_x, Index m, Number ∗g)
- virtual bool *eval_jac_g* (Index n, const Number ∗x, bool new_x, Index m, Index nele_jac, Index ∗iRow, Index ∗jCol, Number ∗values)
- virtual void finalize_solution (SolverReturn status, Index n, const Number ∗x, const Number ∗z_L, const Number ∗z_U, Index m, const Number ∗g, const Number ∗λ, Number obj_value, const IpoptData ∗ip_data, IpoptCalculatedQuantities ∗ip_cq)

Storage of the solution and error control.

- double *calcIndex* (double x, double y, double z, double r)

Calculates the error index of a sphere given its parameters.

- bool *addPoint* (const Point3f &point)

Public Attributes

- double m_solution [4]
  The final solution.

3.29.1 Detailed Description

Class that represents a NLP problem for the optimization of the parameters of a sphere given some of its 3D points.

The optimizer will try to adjust the sphere to those 3D points.

It implements the interface required so the problem can be solved by the Ipopt library.

For a detailed description of the use of the methods by Ipopt, you can find its documentation at [http://www.coin-or.org/Ipopt/documentation/node23.html](http://www.coin-or.org/Ipopt/documentation/node23.html)

3.29.2 Member Function Documentation

3.29.2.1 double OptSpherePoints::calcIndex ( double x, double y, double z, double r )

Calculates the error index of a sphere given its parameters.

In this case, the error is calculated using the distance from the 3D points to the sphere.

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/optspherepoints.h
- oncovision/Interfaz/Interfaz/optspherepoints.cpp

3.30 Hypothesis::Scene3D Class Reference

This class represents a scene of **Object3D**'s organized as a graph.

#include <scene3d.h>
Public Member Functions

- bool addObject (Object3D *object, bool autoID=true)
- int findObject (Object3D *object)
- int findObject (int ID)
- Object3D *getObject (int position)
- Object3D *getObject (int position) const
- int getObjectCount () const
- bool setObject (int position, Object3D *object)
- bool removeObject (int position)
- bool removeObject (Object3D *object)
- void clear ()
- bool sortObjectsVectorWithOrder (int *indexOrder, int indexCount)
- void makeLinks (int position, double distanceThreshold)
- void makeLinks (Object3D *object, double distanceThreshold)
- void globalMakeLinks (double distanceThreshold)

3.30.1 Detailed Description

This class represents a scene of Object3D's organized as a graph.

3.30.2 Member Function Documentation

3.30.2.1 bool Hypothesis::Scene3D::addObject ( Object3D * object, bool autoID = true )

Adds an object to the scene.

Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>The object to be added.</td>
</tr>
<tr>
<td>autoID</td>
<td>If true a new ID is automatically assigned to the object.</td>
</tr>
</tbody>
</table>

Returns

false if the maximum object count is reached, true otherwise.

3.30.2.2 void Hypothesis::Scene3D::clear ( )

Removes all objects from the scene

Warning

the objects will be deleted.

3.30.2.3 int Hypothesis::Scene3D::findObject ( Object3D * object )

Finds an object in the scene vector.

Parameters
### Class Documentation

<table>
<thead>
<tr>
<th>object</th>
<th>The object to be found.</th>
</tr>
</thead>
</table>

Returns

The position of the object in the vector, -1 if it's not in the scene.

### 3.30.2.4 Object3D * Hypothesis::Scene3D::getObject ( int position )

Gets an object from the scene.

Parameters

<table>
<thead>
<tr>
<th>position</th>
<th>Position of the object on the scene vector.</th>
</tr>
</thead>
</table>

Returns

A pointer to the object, 0 if the position is out of bounds.

### 3.30.2.5 int Hypothesis::Scene3D::getObjectCount ( ) const

Returns the number of objects in the scene.

### 3.30.2.6 void Hypothesis::Scene3D::globalMakeLinks ( double distanceThreshold )

Creates the links between all the objects which are within a given distance threshold.

Parameters

<table>
<thead>
<tr>
<th>distance-Threshold</th>
<th>All objects within this parameter will be linked.</th>
</tr>
</thead>
</table>

### 3.30.2.7 void Hypothesis::Scene3D::makeLinks ( int position, double distanceThreshold )

Creates the links between the object at a given position and all the ones inside the distance threshold.

Parameters

<table>
<thead>
<tr>
<th>position</th>
<th>Position of the object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>distance-Threshold</td>
<td>All objects nearer than this parameter will be linked to the selected object.</td>
</tr>
</tbody>
</table>

### 3.30.2.8 void Hypothesis::Scene3D::makeLinks ( Object3D * object, double distanceThreshold )

Creates the links between the object at a given position and all the ones inside the distance threshold.

Parameters

<table>
<thead>
<tr>
<th>object</th>
<th>Object3D to make the links from (must belong to this Scene3D).</th>
</tr>
</thead>
<tbody>
<tr>
<td>distance-Threshold</td>
<td>All objects nearer than this parameter will be linked to the selected object.</td>
</tr>
</tbody>
</table>
3.30.2.9 bool Hypothesis::Scene3D::removeObject ( int position )

Removes an object from the scene

Warning
the object will be deleted.

Parameters

| position | Position of the object to be removed. |

Returns
false if the position is out of bounds, true otherwise

3.30.2.10 bool Hypothesis::Scene3D::removeObject ( Object3D * object )

Removes an object from the scene

Warning
the object will be deleted.

Parameters

| object | Object to be removed |

Returns
false if the object is not in the scene, true otherwise

3.30.2.11 bool Hypothesis::Scene3D::setObject ( int position, Object3D * object )

Replaces an object from the scene with a new one.

Warning
The old one will be deleted.

Parameters

| position | Position of the object to be replaced. |
| object   | The new object. |

Returns
false if the position is out of bounds, true otherwise

3.30.2.12 bool Hypothesis::Scene3D::sortObjectsVectorWithOrder ( int * indexOrder, int indexCount )

Sorts the objects according to an index vector.
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>indexOrder</code></td>
<td>A vector of integers with the new object order.</td>
</tr>
<tr>
<td><code>indexCount</code></td>
<td>The length of the <code>indexOrder</code> vector.</td>
</tr>
</tbody>
</table>

### Returns

false if the `indexOrder` vector doesn’t match the number of objects in the scene or if the scene is empty, true otherwise.

The documentation for this class was generated from the following files:

- `oncovision/Interfaz/Interfaz/hypothesis/scene3d.h`
- `oncovision/Interfaz/Interfaz/hypothesis/scene3d.cpp`

### 3.31 SceneTreeOptimizer Class Reference

Optimizes the parameters of the objects in a 3D scene.

```cpp
#include <optimizer.h>
```

#### Public Member Functions

- **`SceneTreeOptimizer`** *(ApplicationData *data, int mainIndex)*
  - void `exploreTreeGreedy` (sequential_tree< TreeNode * > *initNode)
    
    Makes a greedy exploration of the tree.
  - void `exploreTreeManual` (const vector< int > &objectOrder)
  - void `exploreChildren` (sequential_tree< TreeNode * > *initNode)
    
    Explores the children of a given node.
  - void `exploreTreeTest` (sequential_tree< TreeNode * > *initNode)
    
    Makes an iteration of exploring all the tree.
  - void `exploreTreeLauncher` (sequential_tree< TreeNode * > *initNode, int depthEnd)
    
    Launches a full exploration.
  - void `exploreTreeLevels` (sequential_tree< TreeNode * > *initNode, int levels)
  - double `optimize` (ApplicationData *data)
    
    Optimizes a given scene.
  - void `addDependency` (sequential_tree< TreeNode * > *node)
  - bool `checkDependency` (sequential_tree< TreeNode * > *node)
  - void `run` ()
    
    Go-to method. Performs a greedy exploration and then a full exploration.

#### Public Attributes

- sequential_tree< TreeNode * > * `m_tree`
- int `insertedNodes`

### 3.31.1 Detailed Description

Optimizes the parameters of the objects in a 3D scene.

To correctly optimize them they must be ordered based on their distance to the camera. Because that order is unknown beforehand, different orders are tested in a tree.
3.31.2 Member Function Documentation

3.31.2.1 void SceneTreeOptimizer::addDependency ( sequential_tree< TreeNode * > * node )

Adds a possible dependency param node Current node.

3.31.2.2 void SceneTreeOptimizer::exploreTreeManual ( const vector<int> & objectOrder )

Makes a manual exploration of the tree.
Parameters

| objectOrder | Vector with the indexes of the objects to explore in each step. |

3.31.2.3 double SceneTreeOptimizer::optimize ( ApplicationData * data )

Optimizes a given scene.
Used each iteration when exploring the tree to optimize the scene of a specific node.
Returns

The error value after optimizing.

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/optimizer.h
- oncovision/Interfaz/Interfaz/optimizer.cpp

3.32 sorter< T > Class Template Reference

Public Member Functions

- sorter (const std::vector<T> &v)
- bool operator() (int a, int b)

The documentation for this class was generated from the following file:

- oncovision/Interfaz/Interfaz/optimizer.cpp

3.33 Hypothesis::Sphere Class Reference

A sphere that is part of an Object3D.

#include <sphere.h>

Public Member Functions

- Sphere (int id=0)
- Sphere (double coords[4], int id=0)
- Sphere (double position[3], double radius, int id=0)
- Sphere (double x, double y, double z, double r=1.0, int id=0)
- Object3D = getParent ()
• void setParent (Object3D *parent)
• void getAbsCoords (double coords[4]) const
• void setAbsCoords (double x, double y, double z, double r)
• void getCoords (double coords[4]) const
• void setCoords (double coords[4])
• void getPosition (double position[3]) const
• void setPosition (double x, double y, double z)
• double getRadius () const
• void setRadius (double r)
• int getID ()
• void setID (int id)
• void copyFrom (Sphere *sphere)

### 3.33.1 Detailed Description

A sphere that is part of an Object3D.

### 3.33.2 Member Function Documentation

#### 3.33.2.1 void Hypothesis::Sphere::copyFrom ( Sphere * sphere )

Copies the sphere's properties from a different one.

**Parameters**

| sphere | The sphere to copy the properties from. |

#### 3.33.2.2 void Hypothesis::Sphere::getAbsCoords ( double coords[4] ) const

Returns the absolute coordinates of the sphere.

**Parameters**

| out | coords | Absolute coordinates and radius of the sphere [x, y, z, r]. |

#### 3.33.2.3 void Hypothesis::Sphere::getCoords ( double coords[4] ) const

Returns the coordinates of the sphere relative to its parent object center.

**Parameters**

| out | coords | Coordinates and radius of the sphere [x, y, z, r]. |

#### 3.33.2.4 int Hypothesis::Sphere::getID ( )

Returns the identifier of this sphere.

Each sphere has an unique identifier inside an specific object.
3.33.2.5 **Object3D::Hypothesis::Sphere::getParent ( )** [inline]

Returns the object this sphere belongs to.

Returns

The parent object.

3.33.2.6 **void Hypothesis::Sphere::getPosition ( double position[3] ) const**

Returns the position of the sphere relative to its parent object center.

Parameters

| out | position | Coordinates of the sphere [x, y, z]. |

3.33.2.7 **double Hypothesis::Sphere::getRadius ( ) const**

Returns the radius of the sphere.

3.33.2.8 **void Hypothesis::Sphere::setAbsCoords ( double x, double y, double z, double r )**

Sets the absolute coordinates of the sphere.

Parameters

| x   | X coordinate. |
| y   | Y coordinate. |
| z   | Z coordinate. |
| r   | Radius.       |

Warning

Modifies the position of the parent object.

3.33.2.9 **void Hypothesis::Sphere::setCoords ( double coords[4] )**

Sets the coordinates of the sphere relative to its parent object center.

Parameters

| in | coords | Coordinates and radius of the sphere [x, y, z, r]. |

3.33.2.10 **void Hypothesis::Sphere::setCoords ( double x, double y, double z, double r )**

Sets the coordinates of the sphere relative to its parent object center.

Parameters

| x   | X coordinate. |
| y   | Y coordinate. |
3.33.2.11  void Hypothesis::Sphere::setID ( int id )

Sets the identifier of the sphere.

Parameters

| id | The new identifier. |

3.33.2.12  void Hypothesis::Sphere::setParent ( Object3D ∗ parent ) [inline]

Sets to which object this sphere belongs to.

Parameters

| parent | The new parent. |

3.33.2.13  void Hypothesis::Sphere::setPosition ( double coords[3] )

Sets the position of the sphere relative to its parent object center.

Parameters

| in | coords | Coordinates of the sphere [x, y, z]. |

3.33.2.14  void Hypothesis::Sphere::setPosition ( double x, double y, double z )

Sets the position of the sphere relative to its parent object center.

Parameters

| x | X coordinate. |
| y | Y coordinate. |
| z | Z coordinate. |

3.33.2.15  void Hypothesis::Sphere::setRadius ( double r )

Sets the radius of the sphere.

Parameters

| r | Radius. |

The documentation for this class was generated from the following files:

- oncoshion/Interfaz/Interfaz/hypothesis/sphere.h
- oncoshion/Interfaz/Interfaz/hypothesis/sphere.cpp

### 3.34 SphereGenerator Class Reference

Class that generates 3D hypotheses through the Hough method.

```cpp
#include <sphereregenger.h>
```
Public Member Functions

- **SphereGenerator** (double xMin, double xMax, double xStep, double yMin, double yMax, double yStep, double zMin, double zMax, double zStep, double rMin, double rMax, double rStep)
  
  Constructor.

- **bool addPoint** (const Point3f &point)
  
  Adds a point to be used by the Hough method.

- **bool addHolePoint** (const Point3f &point)
  
  Adds a hole point.

- **void hough** ()
  
  Performs the Hough method.

Public Attributes

- vector< HoughSolution > m_solutions
  
  Vector of solutions.

3.34.1 Detailed Description

Class that generates 3D hypotheses through the Hough method.

Some of the solutions may be not valid because they don’t are compliant to some of the restrictions, so it’s important to check for the valid flag before using a solution.

3.34.2 Constructor & Destructor Documentation

3.34.2.1 SphereGenerator::SphereGenerator ( double xMin, double xMax, double xStep, double yMin, double yMax, double yStep, double zMin, double zMax, double zStep, double rMin, double rMax, double rStep )

Constructor.

The domain for the Hough method is passed as parameter.

3.34.3 Member Function Documentation

3.34.3.1 bool SphereGenerator::addHolePoint ( const Point3f & point )

Adds a hole point.

Hole points are used to filter non-valid solutions. If a solution has a hole point near its center it’ll be flagged as non-valid.

3.34.3.2 void SphereGenerator::hough ()

Performs the Hough method.

The solutions will be stored in the m_solutions vector

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/spheregenerator.h
- oncovision/Interfaz/Interfaz/spheregenerator.cpp
Class Documentation

3.35 StateOptim Struct Reference

State of the SceneTreeOptimizer in a specific node.

```cpp
#include <treenode.h>
```

### Public Attributes

- `int depth`
- `int index`
- `double value`
- `double sphereIndex`
- `vector< double > distances`
- `vector< int > order`

3.35.1 Detailed Description

State of the SceneTreeOptimizer in a specific node.

The documentation for this struct was generated from the following files:

- `oncovision/Interfaz/Interfaz/treenode.h`
- `oncovision/Interfaz/Interfaz/treenode.cpp`

3.36 Supervisor Class Reference

Base class which defines a common interface for different stereoscopic edge-detection applications.

```cpp
#include <supervisor.h>
```

Inheritance diagram for Supervisor:

```
Supervisor
    GrapeSupervisor
```

### Public Member Functions

- `Supervisor (const ApplicationData &data)`
- `virtual void onStart ()=0`
- `virtual void intermediateCallback ()`
  
  *This method is called after each Action is executed.*
- `virtual bool hasFinished ()=0`
  
  *The ending condition. This method is called after each Action is executed.*
- `void run ()`
  
  *Starts the supervisor.*
- `void addAction (Action *action)`
- `void addObject (Object3D *object)`
- `Scene3D *getScene ()`
- `ApplicationData *getApplicationData ()`
- `void setApplicationData (const ApplicationData &data)`
3.36 Supervisor Class Reference

- **ActionParams getParameters ()**
  Creates an ActionParams struct with the essential fields filled correctly.
- **BitmapManager & getBitmapManager ()**
- **FeatureMatcher & getFeatureMatcher ()**

Protected Attributes

- **ApplicationData m_data**

3.36.1 Detailed Description

Base class which defines a common interface for different stereoscopic edge-detection applications.

This class implements an Action system. The user must add Actions to the system and they will be performed ordered by their priority.

All executed actions are stored in a stack so they can be accessed. This lets the user check the past actions to recognize patterns in the behaviour of the supervisor, or just use it for debugging purposes.

Note that the Supervisor stores his own copy of the data, which means that when it finishes you have to copy it back to your own data if you want to access the results.

3.36.2 Constructor & Destructor Documentation

3.36.2.1 Supervisor::Supervisor ( const ApplicationData & data )

Constructor.

Parameters

| data | The supervisor will store a copy from this data. |

3.36.3 Member Function Documentation

3.36.3.1 void Supervisor::addAction ( Action * action )

Adds an Action to the system.

Parameters

| action | The Action to be added. The supervisor takes ownership of the pointer. |

3.36.3.2 void Supervisor::addObject ( Object3D * object )

Adds an object to the scene.

Parameters

| object | The Object3D to be added. The scene takes ownership of the pointer. |

3.36.3.3 ApplicationData * Supervisor::getApplicationData ( )

Returns a pointer to the data the app is working on.
Returns

A pointer to the ApplicationData.

3.36.3.4  BitmapManager& Supervisor::getBitmapManager ( ) [inline]

Returns the bitmap manager of the supervisor.

Returns

A reference to the BitmapManager.

3.36.3.5  FeatureMatcher& Supervisor::getFeatureMatcher ( ) [inline]

Returns the feature matcher of the supervisor.

Returns

A reference to the FeatureMatcher.

3.36.3.6  ActionParams Supervisor::getParameters ( )

Creates an ActionParams struct with the essential fields filled correctly.
When creating a new Action, this is used to get the correct parameters needed in the constructor.
Example:

```
ActionParams params = getParameters();
addAction(new AnAction(params, 1.0));
```

3.36.3.7  Scene3D Supervisor::getScene ( )

Returns the scene the supervisor is working on.

Returns

A pointer to the Scene3D.

3.36.3.8  virtual bool Supervisor::hasFinished ( ) [pure virtual]

The ending condition. This method is called after each Action is executed.
If it returns true the supervisor will stop.

Returns

true if the supervisor has finished, false otherwise.

Implemented in GrapeSupervisor.

3.36.3.9  virtual void Supervisor::intermediateCallback ( ) [inline],[virtual]

This method is called after each Action is executed.
It’s default implementation is void. It can be overloaded for debugging purposes.
3.36.3.10 virtual void Supervisor::onStart ( ) [pure virtual]

This method is where all the actions must be added to the system with a specific priority.
Implemented in GrapeSupervisor.

3.36.3.11 void Supervisor::setApplicationData ( const ApplicationData & data )

Sets the data the app will work on.
Parameters

| data | The app will store a copy from this data. |

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/supervisor.h
- oncovision/Interfaz/Interfaz/supervisor.cpp

3.37 TreeNode Class Reference

A node of the tree in the SceneTreeOptimizer.
#include <treenode.h>

Public Attributes

- ApplicationData data
- StateOptim stateOptim

3.37.1 Detailed Description

A node of the tree in the SceneTreeOptimizer.
The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/treenode.h
- oncovision/Interfaz/Interfaz/treenode.cpp

3.38 Viewport2D Class Reference

Widget that represents a 2D viewport.
#include <viewport2d.h>

Inheritance diagram for Viewport2D:

```
QWidget

Viewport2D
```
Public Slots

- **void zoomOut ()**
  
  Zooms out.

- **void zoomIn ()**
  
  Zooms in.

- **void loadImage ()**
  
  Opens the load image dialog and loads the selected image.

- **void loadCalibration ()**
  
  Opens the load calibration dialog and loads the selected calibration file.

- **void saveImage ()**
  
  Opens the save image dialog and saves the currently displayed image to the hard drive.

- **void render ()**
  
  Makes a render of the current scene.

- **void showImage (const QString &name)**
  
  Opens the compare image dialog and creates a new image comparing two existing ones.

- **void generateEdges ()**
  
  Shows the detected edges of the source image.

- **void project ()**
  
  Projects the 3D objects on the viewport as 2D ellipses.

- **void setSphereMode ()**
  
  Sets the viewport into sphere mode.

- **void setSpineMode ()**
  
  Sets the viewport into spine mode.

- **void setPointMode ()**
  
  Sets the viewport into point mode.

- **void resetMode ()**
  
  Clears the viewport from any mode.

Public Member Functions

- **Viewport2D (ApplicationData ∗data, ManualStageWidget ∗parent=0)**

- **Mat getSourceImage ()**
  
  Returns the source image.

- **ManualStageWidget ∗getParent ()**
  
  Returns the parent widget.

- **DrawableLayout ∗getDrawableLayout ()**
  
  Returns this viewport’s DrawableLayout.

- **Camera ∗getCamera ()**
  
  Returns this viewport’s assigned Camera.

- **QImage ∗getCurrentImage ()**
  
  Returns the currently displayed image.

- **void setImage (QImage ∗image)**

- **int getID ()**
  
  Returns the ID of this viewport.

- **void setID (int id)**
  
  Sets the ID of this viewport.

- **void updateCalibrationText ()**
  
  Updates the bottom text with the calibration’s filename.

- **float getScaleFactor ()**
Returns the scale factor (zoom) of the viewport.

- void setBitmapManager (BitmapManager *bm)
- void setFeatureMatcher (FeatureMatcher *fm)
- void loadImage (const QString &fileName)
 Loads a new image and sets it as the source image.
- void loadCalibration (const QString &fileName)

### 3.38.1 Detailed Description

Widget that represents a 2D viewport.

A 2D viewport consists of the necessary toolbars for camera and image management, and a 2D area (Drawable-Layout class) where the user can edit and project the objects.

Each viewport represents a camera, therefore this application will usually have two 2D viewports (left and right cameras).

### 3.38.2 Member Function Documentation

#### 3.38.2.1 ManualStageWidget * Viewport2D::getParent () [inline]

Returns the parent widget.

This can be used to access the other Viewport2D from this one.

#### 3.38.2.2 Mat Viewport2D::getSourceImage () [inline]

Returns the source image.

The source image refers to, in this case, the image originally loaded by the user.

#### 3.38.2.3 void Viewport2D::loadCalibration ( const QString & fileName )

Loads a the calibration file for this viewport’s camera

**Parameters**

<table>
<thead>
<tr>
<th>fileName</th>
<th>Path to the calibration file.</th>
</tr>
</thead>
</table>

#### 3.38.2.4 void Viewport2D::loadCalibration () [slot]

Opens the load calibration dialog and loads the selected calibration file.
Internally calls the "loadCalibration(const QString &fileName)" method.

#### 3.38.2.5 void Viewport2D::loadImage ( const QString & fileName )

Loads a new image and sets it as the source image.
Besides loading the image, this method also processes it, generating the distance matrix, its mask, etc...

**Parameters**
### 3.38.2.6 void Viewport2D::loadImage() [slot]

Opens the load image dialog and loads the selected image. Internally calls the "loadImage(const QString &fileName)" method.

### 3.38.2.7 void Viewport2D::setBitmapManager ( BitmapManager * bm ) [inline]

Sets the BitmapManager

**Parameters**

| bm     | A pointer to a BitmapManager |

### 3.38.2.8 void Viewport2D::setFeatureMatcher ( FeatureMatcher * fm ) [inline]

Sets the FeatureMatcher

**Parameters**

| bm     | A pointer to a FeatureMatcher |

### 3.38.2.9 void Viewport2D::setImage ( QImage * image )

Sets the currently displayed image.

**Parameters**

| image | The image to display in the viewport. |

### 3.38.2.10 void Viewport2D::setPointMode() [inline],[slot]

Sets the viewport into point mode.

In this mode the user defines a 3D point by defining a point in one viewport and one in the other.

### 3.38.2.11 void Viewport2D::setSphereMode() [inline],[slot]

Sets the viewport into sphere mode.

In this mode the user can create new 3D spheres by defining 5 points in one viewport and 1 in the other.

### 3.38.2.12 void Viewport2D::setSpineMode() [inline],[slot]

Sets the viewport into spine mode.

In this mode the user defines a 3D spine (line) by defining 2 points in one viewport and 2 in the other.

### 3.38.2.13 void Viewport2D::showImage ( const QString & name ) [slot]

Changes the currently displayed image
Parameters

| name | A name corresponding to an image type. Valid types are “Source”, “Render”, “Compared”, “Edges” and “Projected”. |

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/viewport2d.h
- oncovision/Interfaz/Interfaz/viewport2d.cpp

3.39 Viewport3D Class Reference

Widget that represents a 3D viewport.

```cpp
#include <viewport3d.h>
```

Inheritance diagram for Viewport3D:

```
QWidget

Viewport3D
```

Public Slots

- void `updateViewport()`  
  
  Updates the viewport with the changes made to the 3D scene.
- void `updateValues(double ignore)`
- void `updateSpinners()`  
  
  Updates the values of the object properties’ spinners.
- void `setAutoCenter(bool val)`  
  
  Sets AutoCenter.

Signals

- void `spinnersUpdated()`  
  
  This signal is emitted when a spinner changes.

Public Member Functions

- `Viewport3D(ApplicationData *data, ManualStageWidget *parent=0)`
- `GLWidget *getGLWidget()`  
  
  Returns a pointer to the GLWidget.

3.39.1 Detailed Description

Widget that represents a 3D viewport.

A 3D viewport consists of the necessary interface elements to manage the 3D objects.

Among these elements is the GLWidget, the 3D area where the user can navigate the 3D space and select the various objects.
3.39.2 Member Function Documentation

3.39.2.1 void Viewport3D::setAutoCenter ( bool val ) [inline],[slot]

Sets AutoCenter.
When AutoCenter is set to true the camera will follow automatically the selected object.

3.39.2.2 void Viewport3D::updateValues ( double ignore ) [slot]

Updates the objects given the spinners' values.

Parameters

| ignore | Ignore this parameter. |

The documentation for this class was generated from the following files:

- oncovision/Interfaz/Interfaz/viewport3d.h
- oncovision/Interfaz/Interfaz/GeneratedFiles/Debug/moc_viewport3d.cpp
- oncovision/Interfaz/Interfaz/GeneratedFiles/Release/moc_viewport3d.cpp
- oncovision/Interfaz/Interfaz/viewport3d.cpp
Index

Action, 5
  Action, 6
    getName, 6
    getPriority, 6
    run, 6
ActionCompare, 6
ActionParams, 7
  object, 7
addAction
  Supervisor, 67
addCamera2D
  ApplicationData, 9
addDependency
  SceneTreeOptimizer, 61
addEllipse
  Hypothesis::Object2D, 43
addHolePoint
  SphereGenerator, 65
addImage2D
  ApplicationData, 9
addObject
  ApplicationData, 9
  Hypothesis::Image2D, 30
  Hypothesis::Scene3D, 57
  ManualStageWidget, 37
  Supervisor, 67
AddObjectDialog, 7
addSphere
  Hypothesis::Object3D, 50
ApplicationData, 8
  addCamera2D, 9
  addImage2D, 9
  addObject, 9
  copyFrom, 10
  getCamera2D, 10
  getImage2D, 10
  getLastObject, 10
  getMinValue, 10
  getObject, 10
  getObjectCount, 11
  makeGroups, 11
  project, 11
  removeObject, 11
  updateSingleSphere, 11
applyRectangularMask
  BitmapManager, 12
BitmapManager, 12
  applyRectangularMask, 12
  calculateOverlap, 12
fillObject, 14
isInBackground, 14
calcDistance
  Hypothesis::Object2D, 44
  Hypothesis::Object3D, 50
calcDistanceToCamera
  Hypothesis::Object3D, 50
calcDistanceToNeighbours
  Hypothesis::Object3D, 51
calcHiddenIndex
  Hypothesis::Object2D, 44
calcInFrontOf
  Hypothesis::Object2D, 44
calcIndex
  Hypothesis::Object3D, 51
  OptSphereNLP, 55
  OptSpherePoints, 56
calcMinMax
  Hypothesis::Object2D, 44
calcOverlappingBack
  Hypothesis::Object2D, 44
calcOverlappingFront
  Hypothesis::Object2D, 44
calcPoints
  Hypothesis::Ellipse, 20
calculateGradient
  FeatureMatcher, 22
calculateOverlap
  BitmapManager, 12
calculatePoint
  DrawableLayout, 18
calculateSphere
  DrawableLayout, 18
calculateSpine
  DrawableLayout, 18
checkOverlapping
  Hypothesis::Object2D, 44
clear
  Hypothesis::Image2D, 31
  Hypothesis::Object2D, 45
  Hypothesis::Object3D, 51
  Hypothesis::Scene3D, 57
clearLinks
  Hypothesis::Node, 39
CompareDialog, 14
copyFrom
  ApplicationData, 10
  Hypothesis::Object2D, 45
  Hypothesis::Object3D, 51
getOverlappingListFront
Hypothesis::Object2D, 46
getParameters
Hypothesis::Ellipse, 20
Supervisor, 68
getParent
Hypothesis::Sphere, 62
Viewport2D, 71
getPoints
Hypothesis::Ellipse, 20
getPosition
Hypothesis::Object3D, 52
Hypothesis::Sphere, 63
getPriority
Action, 6
getRadius
Hypothesis::Sphere, 63
getRowLimits
FeatureMatcher, 23
ggetRowLimitsFromCenter
FeatureMatcher, 23
getScene
Supervisor, 68
getSourceImage
Viewport2D, 71
getSphere
Hypothesis::Object3D, 52
getSphereCount
Hypothesis::Object3D, 52
getSpheres
Hypothesis::Object3D, 52
getSpine
GrapeSupervisor, 28
getUpperBound
FeatureMatcher, 23
gViewport2D
ManualStageWidget, 38
globalMakeLinks
Hypothesis::Scene3D, 58
GrapeSupervisor, 27
ggetSpine, 28
GrapeSupervisor, 27
GrapeSupervisor, 27
hasFinished, 28
setSpine, 28
hasFinished
GrapeSupervisor, 28
Supervisor, 68
hough
SphereGenerator, 65
HoughAction, 28
run, 29
HoughSolution, 29
Hypothesis::Ellipse, 19
calcPoints, 20
getParameters, 20
getPoints, 20
setParameters, 20
Hypothesis::Image2D, 30
addObject, 30
clear, 31
findObject, 31
getDistances, 31
getIndex, 31
getMask, 31
getMaskAroundObject, 31
getObject, 32
getObjectCount, 33
removeObject, 33
setDistances, 33
setObject, 34
sortObjectsVectorWithOrder, 34
Hypothesis::Node, 38
clearLinks, 39
findNode, 39
getLinkCount, 40
ID, 40
insertLink, 40
removeLink, 40, 41
setCost, 41
setID, 41
sortLinksVectorWithOrder, 41
type, 42
Hypothesis::Node::Link, 35
Hypothesis::Object2D, 42
addEllipse, 43
calcDistance, 44
calcHiddenIndex, 44
calcInFrontOf, 44
calcMinMax, 44
calcOverlappingBack, 44
calcOverlappingFront, 44
checkOverlapping, 44
clear, 45
copyFrom, 45
deleteEllipse, 45
fillEllipse, 45
color, 45
calcDistanceToCamera, 45
calEllipse, 45
calEllipseCount, 46
calHiddenIndex, 46
calIndex, 46
calNodeAsObject2D, 46
calOverlappingListBack, 46
calOverlappingListFront, 46
calOverlappingCalculation, 46
calColor, 48
calDistanceToCamera, 48
calHiddenIndex, 48
calIndex, 48
updateLinks, 48
Hypothesis::Object3D, 49
addSphere, 50
calDistance, 50
calDistanceToCamera, 50
calcDistanceToNeighbours, 51
calcIndex, 51
clear, 51
copyFrom, 51
deleteSphere, 51
getColor, 51
getIndex, 51
getNodeAsObject3D, 51
getPosition, 52
getSphere, 52
getSphereCount, 52
getSpheres, 52
setColor, 52
setPosition, 52
setIndex, 53
setPosition, 53
updateLinks, 53
Hypothesis::Scene3D, 56
addObject, 57
clear, 57
findObject, 57
getObject, 58
getObjectCount, 58
globalMakeLinks, 58
makeLinks, 58
removeObject, 58, 59
setObject, 59
sortObjectsVectorWithOrder, 59
Hypothesis::Sphere, 61
copyFrom, 62
getAbsCoords, 62
goCoords, 62
getID, 62
getParent, 62
getPosition, 63
getRadius, 63
setAbsCoords, 63
setCoords, 63
setId, 64
setParent, 64
setPosition, 64
setRadius, 64
ID
Hypothesis::Node, 40
IndexPair, 34
insertLink
Hypothesis::Node, 40
intermediateCallback
Supervisor, 68
isFactible
FeatureMatcher, 23
isInBackground
BitmapManager, 14
isSolution
FeatureMatcher, 24
keyPressEvent
ManualStageWidget, 38
loadCalibration
Viewport2D, 71
loadImage
Viewport2D, 71, 72
MainWindow, 35
makeGroups
ApplicationData, 11
makeLinks
Hypothesis::Scene3D, 58
ManualStageWidget, 36
addObject, 37
getViewport2D, 38
keyPressEvent, 38
ManualStageWidget, 37
ManualStageWidget, 37
matchLowestGrape
FeatureMatcher, 24
matchLowestPoint
FeatureMatcher, 24
object
ActionParams, 7
onStart
Supervisor, 68
OptSphereNLP, 54
calcIndex, 55
OptSpherePoints, 55
calcIndex, 56
optimize
SceneTreeOptimizer, 61
optimizeMatches
FeatureMatcher, 24
optimizeMatchesColor
FeatureMatcher, 24
OptimizeObjectAction, 53
run, 54
overlappingCalculation
Hypothesis::Object2D, 46
project
ApplicationData, 11
removeLink
Hypothesis::Node, 40, 41
removeObject
ApplicationData, 11
Hypothesis::Image2D, 33
Hypothesis::Scene3D, 58, 59
run
Action, 6
CreateBottomGrapeAction, 15
CreateRosetteAction, 16
HoughAction, 29
OptimizeObjectAction, 54
SceneTreeOptimizer, 60
addDependency, 61
exploreTreeManual, 61
optimize, 61
setAbsCoords
Hypothesis::Sphere, 63
setApplicationData
Supervisor, 69
setAutoCenter
Viewport3D, 74
setBitmapManager
Viewport2D, 72
setColor
Hypothesis::Object2D, 48
Hypothesis::Object3D, 52
setCoords
Hypothesis::Sphere, 63
setCost
Hypothesis::Node, 41
setDistanceToCamera
Hypothesis::Object2D, 48
setDistances
Hypothesis::Image2D, 33
setFeatureMatcher
Viewport2D, 72
setHiddenIndex
Hypothesis::Object2D, 48
setID
Hypothesis::Node, 41
Hypothesis::Sphere, 64
setImage
Viewport2D, 72
setIndex
Hypothesis::Object2D, 48
Hypothesis::Object3D, 53
setObject
Hypothesis::Image2D, 34
Hypothesis::Scene3D, 59
setParameters
Hypothesis::Ellipse, 20
setParent
Hypothesis::Sphere, 64
setPointMode
Viewport2D, 72
setPosition
Hypothesis::Object3D, 53
Hypothesis::Sphere, 64
setRadius
Hypothesis::Sphere, 64
setSelectedIndex
GLWidget, 27
setSphereMode
Viewport2D, 72
setSpine
GrapeSupervisor, 28
setSpineMode
Viewport2D, 72
setState
DrawableLayout, 19
showBorderHistograms
FeatureMatcher, 24
showImage
Viewport2D, 72
sortLinksVectorWithOrder
Hypothesis::Node, 41
sortObjectsVectorWithOrder
Hypothesis::Image2D, 34
Hypothesis::Scene3D, 59
sorter< T >, 61
SphereGenerator, 64
addHolePoint, 65
hough, 65
SphereGenerator, 65
SphereGenerator, 65
StateOptim, 66
Supervisor, 66
addAction, 67
addObject, 67
getAddress, 67
getBitmapManager, 68
getFeatureMatcher, 68
getParameters, 68
getScene, 68
hasFinished, 68
intermediateCallback, 68
onStart, 68
setApplicationData, 69
Supervisor, 67
TreeNode, 69
type
Hypothesis::Node, 42
unProjectPoint
FeatureMatcher, 24
updateLinks
Hypothesis::Object2D, 48
Hypothesis::Object3D, 53
updateSingleSphere
ApplicationData, 11
updateValues
Viewport3D, 74
Viewport2D, 69
getParent, 71
getSourceImage, 71
loadCalibration, 71
loadImage, 71, 72
setBitmapManager, 72
setFeatureMatcher, 72
setImage, 72
setPointMode, 72
setSphereMode, 72
setSpineMode, 72
showImage, 72
Viewport3D, 73
setAutoCenter, 74
updateValues, 74
writeData
FeatureMatcher, 25