

## Segmentation RF

Generated by Doxygen 1.8.1

Wed Jun 5 2013 13:54:48



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	Features Class Reference . . . . .	5
3.1.1	Detailed Description . . . . .	6
3.1.2	Member Function Documentation . . . . .	6
3.1.2.1	invClsFreq . . . . .	6
3.1.2.2	invCoFreq . . . . .	6
3.1.2.3	labellmg . . . . .	6
3.1.2.4	labelSize . . . . .	6
3.1.2.5	push_backlmg . . . . .	6
3.1.2.6	push_backLab . . . . .	6
3.1.2.7	size . . . . .	6
3.1.2.8	updateClsCoFreq . . . . .	6
3.1.2.9	updateClsFreq . . . . .	7
3.1.3	Member Data Documentation . . . . .	7
3.1.3.1	clsFreq_ . . . . .	7
3.1.3.2	coFreq_ . . . . .	7
3.1.3.3	labellmg_ . . . . .	7
3.1.3.4	size_ . . . . .	7
3.1.3.5	vImg_ . . . . .	7
3.2	LabelLeafNode Struct Reference . . . . .	7
3.2.1	Constructor & Destructor Documentation . . . . .	8
3.2.1.1	LabelLeafNode . . . . .	8
3.2.2	Member Function Documentation . . . . .	8
3.2.2.1	display . . . . .	8
3.2.2.2	labelProb . . . . .	8
3.2.2.3	labelProb . . . . .	8

3.2.2.4	print	8
3.2.2.5	readLeafBin	8
3.2.2.6	readLeafTxt	8
3.2.2.7	showLeafBin	8
3.2.2.8	showLeafTxt	8
3.3	LabelPatchFeature< F > Class Template Reference	9
3.3.1	Detailed Description	9
3.3.2	Constructor & Destructor Documentation	9
3.3.2.1	LabelPatchFeature	9
3.3.3	Member Function Documentation	10
3.3.3.1	feat	10
3.3.3.2	feat	10
3.3.3.3	featRow	10
3.3.3.4	label	10
3.3.3.5	labH	10
3.3.3.6	labH	10
3.3.3.7	vecFeatRow	10
3.3.4	Member Data Documentation	10
3.3.4.1	featH_	10
3.3.4.2	featW_	10
3.3.4.3	imIndex_	10
3.3.4.4	labH_	11
3.3.4.5	labW_	11
3.4	node< U > Struct Template Reference	11
3.4.1	Detailed Description	11
3.4.2	Member Function Documentation	11
3.4.2.1	clone	11
3.4.2.2	nodeid	11
3.4.2.3	nodeid	12
3.5	Puzzle< P > Class Template Reference	12
3.5.1	Member Enumeration Documentation	12
3.5.1.1	METHOD	12
3.5.2	Member Function Documentation	12
3.5.2.1	checkConvergence	12
3.5.2.2	initialPick	12
3.5.2.3	proposeLabeling	12
3.5.2.4	selectPatches	13
3.5.2.5	solve	13
3.6	PuzzlePatch Struct Reference	13
3.6.1	Detailed Description	13

3.6.2	Constructor & Destructor Documentation . . . . .	13
3.6.2.1	PuzzlePatch . . . . .	13
3.6.3	Member Function Documentation . . . . .	14
3.6.3.1	agreement . . . . .	14
3.6.3.2	logProb . . . . .	14
3.6.3.3	logProb . . . . .	14
3.6.3.4	pos2pt . . . . .	14
3.6.4	Member Data Documentation . . . . .	14
3.6.4.1	center_ . . . . .	14
3.6.4.2	logProb_ . . . . .	14
3.6.4.3	piece_ . . . . .	14
3.7	RunRF< L, M, T, F, U > Class Template Reference . . . . .	14
3.7.1	Member Enumeration Documentation . . . . .	16
3.7.1.1	MODE . . . . .	16
3.7.2	Member Function Documentation . . . . .	16
3.7.2.1	binary . . . . .	16
3.7.2.2	binary . . . . .	16
3.7.2.3	detect . . . . .	17
3.7.2.4	getColorLabels . . . . .	17
3.7.2.5	run . . . . .	17
3.7.2.6	runDetect . . . . .	17
3.7.2.7	runExtract . . . . .	17
3.7.2.8	runTrain . . . . .	17
3.7.3	Member Data Documentation . . . . .	17
3.7.3.1	balance_ . . . . .	17
3.7.3.2	binary_ . . . . .	17
3.7.3.3	classInfo_ . . . . .	17
3.7.3.4	consideredCls_ . . . . .	18
3.7.3.5	entropy_ . . . . .	18
3.7.3.6	ext_ . . . . .	18
3.7.3.7	iterPerNode_ . . . . .	18
3.7.3.8	labHeight_ . . . . .	18
3.7.3.9	labWidth_ . . . . .	18
3.7.3.10	noPatches_ . . . . .	18
3.7.3.11	noTrees_ . . . . .	18
3.7.3.12	patchHeight_ . . . . .	18
3.7.3.13	patchWidth_ . . . . .	18
3.7.3.14	path2feat_ . . . . .	19
3.7.3.15	path2labs_ . . . . .	19
3.7.3.16	path2model_ . . . . .	19

3.7.3.17	<a href="#">path2results_</a>	19
3.7.3.18	<a href="#">path2test_</a>	19
3.7.3.19	<a href="#">path2train_</a>	19
3.7.3.20	<a href="#">predMethod_</a>	19
3.7.3.21	<a href="#">runName_</a>	19
3.7.3.22	<a href="#">step_</a>	19
3.7.3.23	<a href="#">trainSize_</a>	19
3.8	<a href="#">StructurePatch&lt; T, F &gt; Class Template Reference</a>	20
3.8.1	<a href="#">Member Function Documentation</a>	21
3.8.1.1	<a href="#">extractFeatures</a>	21
3.8.1.2	<a href="#">extractPatches</a>	21
3.8.1.3	<a href="#">getLabelSize</a>	21
3.8.1.4	<a href="#">getPatchChannels</a>	21
3.8.1.5	<a href="#">loadLabels</a>	21
3.8.1.6	<a href="#">loadPatches</a>	22
3.8.1.7	<a href="#">noImages</a>	22
3.8.1.8	<a href="#">noPatches</a>	22
3.8.1.9	<a href="#">noPatches</a>	22
3.8.1.10	<a href="#">pickRandomNames</a>	22
3.8.1.11	<a href="#">reset</a>	22
3.8.1.12	<a href="#">resizePatches</a>	22
3.8.1.13	<a href="#">savePatches</a>	22
3.8.2	<a href="#">Member Data Documentation</a>	22
3.8.2.1	<a href="#">balance_</a>	22
3.8.2.2	<a href="#">consideredCls_</a>	22
3.8.2.3	<a href="#">cvRNG_</a>	22
3.8.2.4	<a href="#">featH_</a>	23
3.8.2.5	<a href="#">features_</a>	23
3.8.2.6	<a href="#">featW_</a>	23
3.8.2.7	<a href="#">imName_</a>	23
3.8.2.8	<a href="#">labH_</a>	23
3.8.2.9	<a href="#">labW_</a>	23
3.8.2.10	<a href="#">noCls_</a>	23
3.8.2.11	<a href="#">noPatches_</a>	23
3.8.2.12	<a href="#">patches_</a>	23
3.8.2.13	<a href="#">step_</a>	23
3.8.2.14	<a href="#">trainingSize_</a>	23
3.9	<a href="#">StructureRF&lt; L, M, T, F, U &gt; Class Template Reference</a>	24
3.9.1	<a href="#">Constructor &amp; Destructor Documentation</a>	24
3.9.1.1	<a href="#">StructureRF</a>	24

3.9.2	Member Function Documentation	24
3.9.2.1	loadForest	24
3.9.2.2	loadForestBin	25
3.9.2.3	loadTree	25
3.9.2.4	loadTreeBin	25
3.9.2.5	noTrees	25
3.9.2.6	noTrees	25
3.9.2.7	regression	25
3.9.2.8	regressionPerTree	25
3.9.2.9	regressionPerTree	25
3.9.2.10	saveForest	25
3.9.2.11	saveForestBin	25
3.9.2.12	saveTree	26
3.9.2.13	saveTreeBin	26
3.9.2.14	trainForest	26
3.9.2.15	trainForestTree	26
3.9.2.16	treeClsFreq	26
3.9.3	Member Data Documentation	26
3.9.3.1	noTrees_	26
3.9.3.2	vTrees_	26
3.10	StructureRFdetector< L, M, T, F, U > Class Template Reference	26
3.10.1	Member Function Documentation	28
3.10.1.1	detectColor	28
3.10.1.2	detectPyramid	28
3.10.1.3	getFeatures	28
3.10.1.4	loadFeatures	28
3.10.1.5	saveFeatures	28
3.10.1.6	step	28
3.10.1.7	step	28
3.10.2	Member Data Documentation	29
3.10.2.1	forest_	29
3.10.2.2	height_	29
3.10.2.3	labH_	29
3.10.2.4	labW_	29
3.10.2.5	maxsize_	29
3.10.2.6	method_	29
3.10.2.7	noCls_	29
3.10.2.8	step_	29
3.10.2.9	storefeat_	29
3.10.2.10	width_	29

3.11 StructureTree< M, T, F, U > Class Template Reference . . . . .	30
3.11.1 Member Enumeration Documentation . . . . .	32
3.11.1.1 ENTROPY . . . . .	32
3.11.2 Constructor & Destructor Documentation . . . . .	32
3.11.2.1 StructureTree . . . . .	32
3.11.2.2 StructureTree . . . . .	32
3.11.3 Member Function Documentation . . . . .	32
3.11.3.1 applyTest . . . . .	32
3.11.3.2 consideredCls . . . . .	32
3.11.3.3 consideredCls . . . . .	32
3.11.3.4 evaluateTest . . . . .	33
3.11.3.5 generateTest . . . . .	33
3.11.3.6 getNoPatches . . . . .	33
3.11.3.7 getPatchProb . . . . .	33
3.11.3.8 grow . . . . .	33
3.11.3.9 growTree . . . . .	33
3.11.3.10 InfGain . . . . .	33
3.11.3.11 initDataSizes . . . . .	33
3.11.3.12 makeLeaf . . . . .	34
3.11.3.13 measureSet . . . . .	34
3.11.3.14 nEntropy1Cls . . . . .	34
3.11.3.15 nEntropy1ClsRnd . . . . .	34
3.11.3.16 nEntropy2Cls . . . . .	34
3.11.3.17 optimizeTest . . . . .	34
3.11.3.18 performSplit . . . . .	34
3.11.3.19 readTreeBin . . . . .	34
3.11.3.20 readTreeTxt . . . . .	35
3.11.3.21 regression . . . . .	35
3.11.3.22 saveTree . . . . .	35
3.11.3.23 saveTreeBin . . . . .	35
3.11.3.24 saveTreeTxt . . . . .	35
3.11.3.25 showLeaves . . . . .	35
3.11.3.26 split . . . . .	35
3.11.4 Member Data Documentation . . . . .	35
3.11.4.1 binary_ . . . . .	35
3.11.4.2 clsFreq_ . . . . .	35
3.11.4.3 coFreq_ . . . . .	36
3.11.4.4 consideredCls_ . . . . .	36
3.11.4.5 cvRNG_ . . . . .	36
3.11.4.6 entropy_ . . . . .	36



3.11.4.7	log_ . . . . .	36
3.11.4.8	maxDepth_ . . . . .	36
3.11.4.9	minSamples_ . . . . .	36
3.11.4.10	nodeSize_ . . . . .	36
3.11.4.11	patchCh_ . . . . .	36
3.11.4.12	patchH_ . . . . .	36
3.11.4.13	patchW_ . . . . .	36
3.12	Tree< U > Class Template Reference . . . . .	37
3.12.1	Detailed Description . . . . .	37
3.12.2	Member Enumeration Documentation . . . . .	38
3.12.2.1	SIDE . . . . .	38
3.12.3	Constructor & Destructor Documentation . . . . .	38
3.12.3.1	Tree . . . . .	38
3.12.4	Member Function Documentation . . . . .	38
3.12.4.1	addNode . . . . .	38
3.12.4.2	destroyTree . . . . .	38
3.12.4.3	preorderBin . . . . .	38
3.12.4.4	preorderTxt . . . . .	38
3.12.4.5	readNodeBin . . . . .	38
3.12.4.6	readNodeTxt . . . . .	38
3.12.4.7	saveTree . . . . .	38
3.12.4.8	saveTreeBin . . . . .	38
3.12.4.9	saveTreeTxt . . . . .	39
3.12.4.10	showTree . . . . .	39
3.12.4.11	treeld . . . . .	39
3.12.4.12	treeld . . . . .	39
3.12.5	Member Data Documentation . . . . .	39
3.12.5.1	binary_ . . . . .	39
3.12.5.2	path2models_ . . . . .	39
3.12.5.3	root_ . . . . .	39
3.12.5.4	treeld_ . . . . .	39



# Chapter 1

## Class Index

### 1.1 Class Hierarchy

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

Features . . . . .	5
LabelLeafNode . . . . .	7
LabelPatchFeature< F > . . . . .	9
node< U > . . . . .	11
Puzzle< P > . . . . .	12
PuzzlePatch . . . . .	13
RunRF< L, M, T, F, U > . . . . .	14
StructurePatch< T, F > . . . . .	20
StructureRF< L, M, T, F, U > . . . . .	24
StructureRFdetector< L, M, T, F, U > . . . . .	26
Tree< U > . . . . .	37
StructureTree< M, T, F, U > . . . . .	30



## Chapter 2

# Class Index

### 2.1 Class List

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

Features	5
LabelLeafNode	7
LabelPatchFeature< F >	9
node< U >	11
Puzzle< P >	12
PuzzlePatch	13
RunRF< L, M, T, F, U >	14
StructurePatch< T, F >	20
StructureRF< L, M, T, F, U >	24
StructureRFdetector< L, M, T, F, U >	26
StructureTree< M, T, F, U >	30
Tree< U >	37



## Chapter 3

# Class Documentation

### 3.1 Features Class Reference

```
#include <StructurePatch.h>
```

#### Public Member Functions

- **Features** (const std::vector< cv::Mat > &labellmg, const std::vector< std::vector< IpImage \* > > &vimg)
- cv::Size **labelSize** ()
- void **push\_backImg** (std::vector< IpImage \* > &vimg)
- void **push\_backLab** (cv::Mat lab)
- void **updateClsCoFreq** (const std::vector< unsigned > &labels, unsigned center, unsigned nocls)
- void **updateClsFreq** (unsigned label, unsigned nocls)
- std::vector< std::vector< float > > **invCoFreq** () const
- std::vector< float > **invClsFreq** () const
- virtual unsigned **size** () const
- std::vector< cv::Mat > **labellmg** () const
- cv::Mat **labellmg** (unsigned pos) const
- std::vector< std::vector< IpImage \* > > **vimg** () const
- std::vector< IpImage \* > **vimg** (unsigned pos) const
- IpImage \* **vimg** (unsigned pos, unsigned ch) const
- std::vector< unsigned > **clsFreq** () const
- unsigned **clsFreq** (unsigned pos) const
- std::vector< std::vector< unsigned > > **coFreq** () const
- std::vector< unsigned > **coFreq** (unsigned pos) const
- unsigned **coFreq** (unsigned ctr, unsigned rnd) const
- void **size** (unsigned size)
- void **labellmg** (const std::vector< cv::Mat > &labellmg)
- void **labellmg** (unsigned pos, const cv::Mat labellmg)
- void **vimg** (const std::vector< std::vector< IpImage \* > > &vimg)
- void **vimg** (unsigned pos, const std::vector< IpImage \* > &vimg)
- void **vimg** (unsigned pos, unsigned ch, const IpImage \*vimg)
- void **clsFreq** (const std::vector< unsigned > &clsFreq)
- void **coFreq** (const std::vector< std::vector< unsigned > > &coFreq)

## Protected Attributes

- `std::vector< cv::Mat > labellmg_`
- `std::vector< std::vector< IplImage * > > vlmg_`
- unsigned `size_`
- `std::vector< unsigned > clsFreq_`
- `std::vector< std::vector< unsigned > > coFreq_`

### 3.1.1 Detailed Description

To keep inside all the vectors we need.

### 3.1.2 Member Function Documentation

**3.1.2.1** `std::vector<float> Features::invClsFreq ( ) const` `[inline]`

Computes and overwrites the frequencies with inverse frequencies.

**3.1.2.2** `std::vector<std::vector<float> > Features::invCoFreq ( ) const` `[inline]`

Computes and overwrites the co-frequencies with inverse co-frequencies.

**3.1.2.3** `std::vector<cv::Mat> Features::labellmg ( ) const` `[inline]`

Getter for class members.

**3.1.2.4** `cv::Size Features::labelSize ( )` `[inline]`

Returns the size of the label image.

**3.1.2.5** `void Features::push_backlmg ( std::vector< IplImage * > & vlmg )` `[inline]`

Adds a vector of images to the vector-of-vector of images at the end.

**3.1.2.6** `void Features::push_backLab ( cv::Mat lab )` `[inline]`

Adds a vector of images to the vector-of-vector of images at the end.

**3.1.2.7** `void Features::size ( unsigned size )` `[inline]`

Setter for class members.

**3.1.2.8** `void Features::updateClsCoFreq ( const std::vector< unsigned > & labels, unsigned center, unsigned nocl )` `[inline]`

Updates the co-frequencies of classes inside patches.



3.1.2.9 `void Features::updateClsFreq ( unsigned label, unsigned nocls )` `[inline]`

Updates the frequencies of classes in the training data.

### 3.1.3 Member Data Documentation

3.1.3.1 `Features::clsFreq_` `[protected]`

The class frequencies in the original training data for unbalanced data.

3.1.3.2 `Features::coFreq_` `[protected]`

The class co-frequencies in the original training data for unbalanced data.

3.1.3.3 `Features::lablmg_` `[protected]`

The grayscale images containing the labels.

3.1.3.4 `Features::size_` `[protected]`

The size of the data (number of images).

3.1.3.5 `Features::vlmg_` `[protected]`

The vectors of feature matrices – one per image.

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

- `StructurePatch.h`

## 3.2 LabelLeafNode Struct Reference

### Public Member Functions

- **LabelLeafNode** (const char \*path2models, long unsigned leafid, unsigned treeid, bool binary)
- virtual void [readLeafTxt](#) (const char \*path2models, long unsigned leafid, unsigned treeid)
- virtual void [readLeafBin](#) (const char \*path2models, long unsigned leafid, unsigned treeid)
- virtual void [showLeafTxt](#) (const char \*path2models, long unsigned leafid, unsigned treeid)
- virtual void [showLeafBin](#) (const char \*path2models, long unsigned leafid, unsigned treeid)
- virtual void [print](#) () const
- virtual void [display](#) (unsigned labW, unsigned labH, const std::map< cv::Vec3b, unsigned, vec3bCompare > &classinfo, const std::string &path2model) const
- float [labelProb](#) () const
- std::vector< unsigned > **vLabels** () const
- unsigned **vLabels** (unsigned pos) const
- void [labelProb](#) (float labelProb)
- void **vLabelsResize** (unsigned val)
- void **vLabels** (const std::vector< unsigned > &vLabels)
- void **vLabels** (unsigned pos, unsigned vLabels)
- [LabelLeafNode](#) ([LabelLeafNode](#) const &rhs)

## Protected Attributes

- float **labelProb\_**
- std::vector< unsigned > **vLabels\_**

## 3.2.1 Constructor & Destructor Documentation

3.2.1.1 `LabelLeafNode::LabelLeafNode ( LabelLeafNode const & rhs ) [inline]`

Copy constructors for trees (to put them in the forest). The assignment operator is done.

## 3.2.2 Member Function Documentation

3.2.2.1 `virtual void LabelLeafNode::display ( unsigned labW, unsigned labH, const std::map< cv::Vec3b, unsigned, vec3bCompare > & classinfo, const std::string & path2model ) const [inline],[virtual]`

Shows a leaf nicely with colors.

3.2.2.2 `float LabelLeafNode::labelProb ( ) const [inline]`

Getters for the class members.

3.2.2.3 `void LabelLeafNode::labelProb ( float labelProb ) [inline]`

Setters for the class members.

3.2.2.4 `virtual void LabelLeafNode::print ( ) const [inline],[virtual]`

Prints the leaf values to the screen.

3.2.2.5 `virtual void LabelLeafNode::readLeafBin ( const char * path2models, long unsigned leafid, unsigned treeid ) [inline],[virtual]`

Reads the leaf from a regular file.

3.2.2.6 `virtual void LabelLeafNode::readLeafTxt ( const char * path2models, long unsigned leafid, unsigned treeid ) [inline],[virtual]`

Reads the leaf from a regular file.

3.2.2.7 `virtual void LabelLeafNode::showLeafBin ( const char * path2models, long unsigned leafid, unsigned treeid ) [inline],[virtual]`

Writes the leaf info into an opened file.

3.2.2.8 `virtual void LabelLeafNode::showLeafTxt ( const char * path2models, long unsigned leafid, unsigned treeid ) [inline],[virtual]`

Writes the leaf info into an opened file.

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

- StructureTree.h

### 3.3 LabelPatchFeature< F > Class Template Reference

```
#include <StructurePatch.h>
```

#### Public Member Functions

- **LabelPatchFeature** (unsigned featW, unsigned featH, unsigned labW, unsigned labH, unsigned imIndex, const cv::Point &point)
- std::vector< unsigned > **label** (const F \*features) const
- std::vector< CvMat \* > **feat** (const F \*features) const
- CvMat \* **feat** (const F \*features, int channel) const
- cv::Mat **featRow** (const F \*features) const
- std::vector< cv::Mat > **vecFeatRow** (const F \*features) const
- unsigned **labH** () const
- unsigned **labW** () const
- unsigned **featH** () const
- unsigned **featW** () const
- unsigned **imIndex** () const
- cv::Point **point** () const
- void **labH** (unsigned labH)
- void **labW** (unsigned labW)
- void **featH** (unsigned featH)
- void **featW** (unsigned featW)
- void **imIndex** (unsigned imIndex)
- void **point** (const cv::Point &point)
- **LabelPatchFeature** (**LabelPatchFeature** const &rhs)
- **LabelPatchFeature** & **operator=** (**LabelPatchFeature** const &rhs)

#### Protected Attributes

- unsigned **featH\_**
- unsigned **featW\_**
- unsigned **labH\_**
- unsigned **labW\_**
- unsigned **imIndex\_**
- cv::Point **point\_**

#### 3.3.1 Detailed Description

```
template<class F>class LabelPatchFeature< F >
```

Patches are always relative to corner: top-left.

#### 3.3.2 Constructor & Destructor Documentation

```
3.3.2.1 template<class F > LabelPatchFeature< F >::LabelPatchFeature ( LabelPatchFeature< F > const & rhs ) [inline]
```

Copy constructors the default ones are not good with IplImages

### 3.3.3 Member Function Documentation

**3.3.3.1** `template<class F> std::vector< CvMat * > LabelPatchFeature< F >::feat ( const F * features ) const`  
[inline]

Gets the feature around the current pixel as a matrix.

**3.3.3.2** `template<class F> CvMat * LabelPatchFeature< F >::feat ( const F * features, int channel ) const`  
[inline]

Gets the feature around the current pixel as a matrix for the give channel value.

**3.3.3.3** `template<class F> cv::Mat LabelPatchFeature< F >::featRow ( const F * features ) const` [inline]

Gets the feature around the current pixel as a row-matrix.

**3.3.3.4** `template<class F> std::vector< unsigned > LabelPatchFeature< F >::label ( const F * features ) const`  
[inline]

Gets the label patch around the current pixel as a vector.

**3.3.3.5** `template<class F> unsigned LabelPatchFeature< F >::labH ( ) const` [inline]

Getters for the class members.

**3.3.3.6** `template<class F> void LabelPatchFeature< F >::labH ( unsigned labH )` [inline]

Setters for the class members.

**3.3.3.7** `template<class F> std::vector< cv::Mat > LabelPatchFeature< F >::vecFeatRow ( const F * features ) const`  
[inline]

Gets the feature around the current pixel as a vector of row-matrix.

### 3.3.4 Member Data Documentation

**3.3.4.1** `template<class F> LabelPatchFeature< F >::featH_` [protected]

The height of the feature patch.

**3.3.4.2** `template<class F> LabelPatchFeature< F >::featW_` [protected]

The width of the feature patch.

**3.3.4.3** `template<class F> LabelPatchFeature< F >::imIndex_` [protected]

Index in the image-matrix and feature-matrix .

3.3.4.4 `template<class F> LabelPatchFeature< F >::labH_` `[protected]`

The height of the label patch.

3.3.4.5 `template<class F> LabelPatchFeature< F >::labW_` `[protected]`

The width of the label patch.

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

- StructurePatch.h

## 3.4 node< U > Struct Template Reference

```
#include <Tree.h>
```

### Public Member Functions

- **node** (const long double \*test, unsigned nodeSize, long unsigned [nodeid](#), const U \*leaf=NULL)
- virtual void **showNode** (std::ofstream &out, const char \*path2model, unsigned treeid, bool binary)
- virtual void **showNodeTxt** (std::ofstream &out, const char \*path2model, unsigned treeid)
- virtual void **showNodeBin** (std::ofstream &out, const char \*path2model, unsigned treeid)
- long unsigned [nodeid](#) () const
- U \* **leaf** () const
- long double \* **test** () const
- unsigned **nodeSize** () const
- [node](#)< U > \* **left** () const
- [node](#)< U > \* **right** () const
- void [nodeid](#) (long unsigned nodeid)
- void **leaf** (const U \*leaf)
- void **test** (const long double \*test)
- void **nodeSize** (unsigned nodeSize)
- void **left** ([node](#)< U > \*left)
- void **right** ([node](#)< U > \*right)
- [node](#) & **clone** ([node](#) const &rhs)

### 3.4.1 Detailed Description

```
template<class U>struct node< U >
```

The tree node structure to store the info in it.

### 3.4.2 Member Function Documentation

3.4.2.1 `template<class U> node& node< U >::clone ( node< U > const & rhs )` `[inline]`

Use default Copy and assignment constructors.

3.4.2.2 `template<class U> long unsigned node< U >::nodeid ( ) const` `[inline]`

Getters for the members.

3.4.2.3 `template<class U> void node< U >::nodeid ( long unsigned nodeid ) [inline]`

Setters for the members.

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

- Tree.h

## 3.5 Puzzle< P > Class Template Reference

### Public Types

- enum `METHOD` { `SIMPLE`, `PUZZLE` }

### Static Public Member Functions

- static `cv::Mat solve` (const `std::vector< std::vector< P > >` &patchLabels, const `cv::Size` &imsz, unsigned labW, unsigned labH, unsigned noCls, `std::vector< std::vector< float > >` &clsFreq, `Puzzle< P >::METHOD` method, unsigned maxIter=75)
- static `std::vector< P > initialPick` (const `std::vector< std::vector< P > >` &patchLabels, unsigned labW, unsigned labH, unsigned noCls, `std::vector< std::vector< float > >` &clsFreq)
- static `std::vector< P > selectPatches` (const `cv::Mat` &labeling, const `std::vector< std::vector< P > >` &patchLabels, unsigned labW, unsigned labH)
- static `cv::Mat proposeLabeling` (const `std::vector< P >` &candidates, const `cv::Size` &imsz, unsigned labW, unsigned labH, unsigned noCls)
- static bool `checkConvergence` (const `cv::Mat` &labeling, const `cv::Mat` &prevLabeling)

### 3.5.1 Member Enumeration Documentation

#### 3.5.1.1 `template<class P> enum Puzzle::METHOD`

Label selection method.

### 3.5.2 Member Function Documentation

3.5.2.1 `template<class P> bool Puzzle< P >::checkConvergence ( const cv::Mat & labeling, const cv::Mat & prevLabeling ) [static]`

Checks to see how much the labeling has changed between iterations.

3.5.2.2 `template<class P> std::vector< P > Puzzle< P >::initialPick ( const std::vector< std::vector< P > > & patchLabels, unsigned labW, unsigned labH, unsigned noCls, std::vector< std::vector< float > > & clsFreq ) [static]`

Picks the initial candidates as ones with maximum foreground probability.

3.5.2.3 `template<class P> cv::Mat Puzzle< P >::proposeLabeling ( const std::vector< P > & candidates, const cv::Size & featsize, unsigned labW, unsigned labH, unsigned noCls ) [static]`

Generates candidate labelings put of the proposed best patches.

3.5.2.4 `template<class P> std::vector< P > Puzzle< P >::selectPatches ( const cv::Mat & labeling, const std::vector< std::vector< P > > & patchLabels, unsigned labW, unsigned labH ) [static]`

Selects candidate best patches from the given patches

3.5.2.5 `template<class P> cv::Mat Puzzle< P >::solve ( const std::vector< std::vector< P > > & patchLabels, const cv::Size & featsize, unsigned labW, unsigned labH, unsigned noCls, std::vector< std::vector< float > > & clsFreq, Puzzle< P >::METHOD method, unsigned maxIter = 75 ) [static]`

It solves the recursive label optimization.

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

- `Puzzle.h`
- `Puzzle.cpp`

## 3.6 PuzzlePatch Struct Reference

```
#include <Puzzle.h>
```

### Public Member Functions

- **PuzzlePatch** (const cv::Point &center, const std::vector< unsigned > &piece, float [logProb](#))
- unsigned [agreement](#) (const std::vector< unsigned > &other) const
- virtual cv::Point [pos2pt](#) (unsigned pos, unsigned labW, unsigned labH) const
- float [logProb](#) () const
- cv::Point **center** () const
- std::vector< unsigned > **piece** () const
- unsigned **piece** (unsigned pos) const
- void [logProb](#) (float logProb)
- void **center** (const cv::Point &center)
- void **piece** (const std::vector< unsigned > &piece)
- void **piece** (unsigned pos, unsigned piece)
- [PuzzlePatch](#) ([PuzzlePatch](#) const &rhs)
- [PuzzlePatch](#) & **operator=** ([PuzzlePatch](#) const &rhs)

### Protected Attributes

- cv::Point [center\\_](#)
- std::vector< unsigned > [piece\\_](#)
- float [logProb\\_](#)

### 3.6.1 Detailed Description

[Puzzle.h](#) Author: Silvia-Laura Pintea For storing the possible label-ings.

### 3.6.2 Constructor & Destructor Documentation

3.6.2.1 `PuzzlePatch::PuzzlePatch ( PuzzlePatch const & rhs ) [inline]`

Copy constructors the default ones are not good with `IpImages`

### 3.6.3 Member Function Documentation

3.6.3.1 `unsigned PuzzlePatch::agreement ( const std::vector< unsigned > & other ) const` `[inline]`

Compares this piece with another piece.

3.6.3.2 `float PuzzlePatch::logProb ( ) const` `[inline]`

Getters for the class members.

3.6.3.3 `void PuzzlePatch::logProb ( float logProb )` `[inline]`

Setters for the class members.

3.6.3.4 `virtual cv::Point PuzzlePatch::pos2pt ( unsigned pos, unsigned labW, unsigned labH ) const` `[inline]`,  
`[virtual]`

Finds the corresponding position in the original image.

### 3.6.4 Member Data Documentation

3.6.4.1 `PuzzlePatch::center_` `[protected]`

The point on which the patch is centered.

3.6.4.2 `PuzzlePatch::logProb_` `[protected]`

The log of the patch probability from the RF.

3.6.4.3 `PuzzlePatch::piece_` `[protected]`

[Puzzle](#) piece — the label vector at this point.

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

- `Puzzle.h`

## 3.7 `RunRF< L, M, T, F, U >` Class Template Reference

### Public Types

- enum `MODE` { `TRAIN_RF`, `TEST_RF`, `TRAIN_TEST_RF`, `EXTRACT` }

### Public Member Functions

- `RunRF` (const char \*config)
- virtual void `run` (`RunRF::MODE` mode)
- virtual void `runDetect` ()
- virtual void `runExtract` ()
- virtual void `runTrain` ()
- virtual void `detect` (`StructureRFdetector< L, M, T, F, U >` &crDetect)



- bool **binary** () const
- unsigned **step** () const
- **Puzzle**< **PuzzlePatch** >::METHOD **predMethod** () const
- bool **balance** () const
- unsigned **trainSize** () const
- unsigned **noPatches** () const
- unsigned **iterPerNode** () const
- **StructureTree**< M, T, F, U >::ENTROPY **entropy** () const
- unsigned **consideredCls** () const
- std::string **ext** () const
- std::string **labTerm** () const
- unsigned **patchWidth** () const
- unsigned **patchHeight** () const
- unsigned **labWidth** () const
- unsigned **labHeight** () const
- std::string **path2train** () const
- std::string **path2labs** () const
- std::string **path2test** () const
- std::string **path2results** () const
- std::string **path2model** () const
- std::string **path2feat** () const
- unsigned **noTrees** () const
- std::vector< float > **pyrScales** () const
- unsigned **pyrScales** (unsigned pos) const
- std::map< cv::Vec3b, unsigned, vec3bCompare > **classInfo** ()
- unsigned **classInfo** (const cv::Vec3b &color)
- void **binary** (bool binary)
- void **step** (unsigned step)
- void **predMethod** (**Puzzle**< **PuzzlePatch** >::METHOD method)
- void **balance** (bool balance)
- void **trainSize** (unsigned trainSize)
- void **noPatches** (unsigned noPatches)
- void **iterPerNode** (unsigned iterPerNode)
- void **entropy** (typename **StructureTree**< M, T, F, U >::ENTROPY entropy)
- void **consideredCls** (unsigned consideredCls)
- void **ext** (const std::string &ext)
- void **labTerm** (const std::string &labTerm)
- void **patchWidth** (unsigned patchWidth)
- void **patchHeight** (unsigned patchHeight)
- void **labWidth** (unsigned labWidth)
- void **labHeight** (unsigned labHeight)
- void **path2train** (const std::string &path2train)
- void **path2labs** (const std::string &path2labs)
- void **path2test** (const std::string &path2test)
- void **path2results** (const std::string &path2results)
- void **path2model** (const std::string &path2model)
- void **path2feat** (const std::string &path2feat)
- void **noTrees** (unsigned noTrees)
- void **pyrScales** (const std::vector< float > &pyrScales)
- void **pyrScales** (unsigned pos, float pyrScales)
- void **classInfo** (const std::map< cv::Vec3b, unsigned, vec3bCompare > &classInfo)
- void **classInfo** (const cv::Vec3b &color, unsigned classInfo)

## Static Public Member Functions

- static cv::Mat [getColorLabels](#) (const cv::Mat &output, const std::map< cv::Vec3b, unsigned, vec3bCompare > &classinfo)

## Protected Attributes

- std::string [ext\\_](#)
- std::string [labTerm\\_](#)
- unsigned [patchWidth\\_](#)
- unsigned [patchHeight\\_](#)
- unsigned [labWidth\\_](#)
- unsigned [labHeight\\_](#)
- std::string [path2train\\_](#)
- std::string [path2labs\\_](#)
- std::string [path2test\\_](#)
- std::string [path2results\\_](#)
- std::string [path2model\\_](#)
- unsigned [noTrees\\_](#)
- std::vector< float > [pyrScales\\_](#)
- std::string [path2feat\\_](#)
- std::map< cv::Vec3b, unsigned, vec3bCompare > [classInfo\\_](#)
- std::string [runName\\_](#)
- unsigned [consideredCls\\_](#)
- bool [balance\\_](#)
- unsigned [trainSize\\_](#)
- unsigned [noPatches\\_](#)
- unsigned [iterPerNode\\_](#)
- [StructureTree](#)< M, T, F, U > ::ENTROPY [entropy\\_](#)
- [Puzzle](#)< [PuzzlePatch](#) > ::METHOD [predMethod\\_](#)
- unsigned [step\\_](#)
- bool [binary\\_](#)

### 3.7.1 Member Enumeration Documentation

- 3.7.1.1 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > enum RunRF::MODE`

Modes of running the RF code

### 3.7.2 Member Function Documentation

- 3.7.2.1 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > bool RunRF< L, M, T, F, U >::binary ( ) const [inline]`

Getters for the class members.

- 3.7.2.2 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void RunRF< L, M, T, F, U >::binary ( bool binary ) [inline]`

Setters for the class members.

**3.7.2.3** `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U > void RunRF< L, M, T, F, U >::detect ( StructureRFdetector< L, M, T, F, U > & crDetect ) [virtual]`

Performs the RF detection on test images.

**3.7.2.4** `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U > cv::Mat RunRF< L, M, T, F, U >::getColorLabels ( const cv::Mat & output, const std::map< cv::Vec3b, unsigned, vec3bCompare > & classinfo ) [static]`

Gets the color labels for the image.

**3.7.2.5** `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U > void RunRF< L, M, T, F, U >::run ( RunRF< L, M, T, F, U >::MODE mode ) [virtual]`

Initialize and start training.

**3.7.2.6** `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U > void RunRF< L, M, T, F, U >::runDetect ( ) [virtual]`

Initialize and start detector on test set.

**3.7.2.7** `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U > void RunRF< L, M, T, F, U >::runExtract ( ) [virtual]`

Extracts feature/label patches from all the images.

Initialize and start training.

**3.7.2.8** `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U > void RunRF< L, M, T, F, U >::runTrain ( ) [virtual]`

Initialize and start training.

### 3.7.3 Member Data Documentation

**3.7.3.1** `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U > RunRF< L, M, T, F, U >::balance_ [protected]`

To balance the class data or not.

**3.7.3.2** `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U > RunRF< L, M, T, F, U >::binary_ [protected]`

If the tree is in a binary file or not.

**3.7.3.3** `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U > RunRF< L, M, T, F, U >::classInfo_ [protected]`

Maps class names to color to ids.

**3.7.3.4** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::consideredCls_` [protected]

The first x considered classes (they are taken in the order from the config file).

**3.7.3.5** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::entropy_` [protected]

The entropy type to be used (CENTER, RANDOM, ..)

**3.7.3.6** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::ext_` [protected]

Image extension used for reading images from dir.

Extra termination concatenated at the end of the label names.

**3.7.3.7** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::iterPerNode_` [protected]

Number of iterations per node for tree-training.

**3.7.3.8** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::labHeight_` [protected]

The height of our appearance/label patches.

**3.7.3.9** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::labWidth_` [protected]

The width of our appearance/label patches.

**3.7.3.10** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::noPatches_` [protected]

Number of patches to use for training (repeat some).

**3.7.3.11** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::noTrees_` [protected]

Number of trees.

**3.7.3.12** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::patchHeight_` [protected]

The height of our appearance/label patches.

**3.7.3.13** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::patchWidth_` [protected]

The width of our appearance/label patches.

**3.7.3.14** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::path2feat_ [protected]`

Path to features (if none, then extract).

**3.7.3.15** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::path2labs_ [protected]`

Path to labeled images.

**3.7.3.16** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::path2model_ [protected]`

Path to RF model.

**3.7.3.17** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::path2results_ [protected]`

Path results images (labeled).

**3.7.3.18** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::path2test_ [protected]`

Path to test images.

**3.7.3.19** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::path2train_ [protected]`

Path to train images.

**3.7.3.20** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::predMethod_ [protected]`

[Puzzle](#) or Simple prediction method.

**3.7.3.21** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::runName_ [protected]`

The name of the current run for the log-files.

**3.7.3.22** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::step_ [protected]`

The step on the grid for sampling patches.

**3.7.3.23** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > RunRF< L, M, T, F, U >::trainSize_ [protected]`

Number of images to be used for training 1 tree.

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

- RunRF.h
- RunRF.cpp

### 3.8 StructurePatch< T, F > Class Template Reference

#### Public Member Functions

- **StructurePatch** (CvRNG \*pRNG, unsigned patchW, unsigned patchH, unsigned noCls, unsigned labW, unsigned labH, unsigned trainSize, unsigned [noPatches](#), unsigned consideredCls, bool balance, unsigned step)
- cv::Mat \* [loadLabels](#) (const std::string &justname, const std::string &ext, const std::string &term, const std::string &labpath, const std::map< cv::Vec3b, unsigned, vec3bCompare > &classinfo, const std::string &featpath)
- void [resizePatches](#) ()
- unsigned [getPatchChannels](#) () const
- unsigned [noImages](#) () const
- unsigned [getLabelSize](#) () const
- virtual void [reset](#) ()
- virtual void [pickRandomNames](#) (const std::vector< std::string > &filenames)
- virtual void [extractPatches](#) (const std::string &imgpath, const std::string &labpath, const std::string &featpath, const std::vector< std::string > &vFileNames, const std::map< cv::Vec3b, unsigned, vec3bCompare > &classinfo, const std::string &term, const std::string &ext)
- virtual void [extractFeatures](#) (const cv::Mat \*labimg, IplImage \*img, const std::string &path2feat, bool showWhere=false)
- virtual void [savePatches](#) (const std::string &path2feat, unsigned pos)
- virtual void [loadPatches](#) (const std::string &path2feat, bool showWhere=false)
- unsigned [noPatches](#) () const
- bool **balance** () const
- F \* **features** () const
- CvRNG \* **cvRNG** () const
- unsigned **step** () const
- unsigned **trainingSize** () const
- unsigned **labW** () const
- unsigned **labH** () const
- unsigned **featW** () const
- unsigned **featH** () const
- unsigned **noCls** () const
- std::vector< std::string > **imName** () const
- std::string **imName** (unsigned pos) const
- std::vector< std::vector< const T \* > > **patches** () const
- std::vector< const T \* > **patches** (unsigned cls) const
- const T \* **patches** (unsigned cls, unsigned pt) const
- void [noPatches](#) (unsigned noPatches)
- void **balance** (bool balance)
- void **features** (F \*features)
- void **cvRNG** (CvRNG \*cvRNG)
- void **step** (unsigned step)
- void **trainingSize** (unsigned trainingSize)
- void **noCls** (unsigned noCls)
- void **labW** (unsigned labW)
- void **labH** (unsigned labH)
- void **featW** (unsigned featW)
- void **featH** (unsigned featH)
- void **imName** (const std::vector< std::string > &imName)

- void **imName** (unsigned pos, const std::string &imName)
- void **patches** (const std::vector< std::vector< const T \* > > &patches)
- void **patches** (unsigned cls, const std::vector< const T \* > &patches)
- void **patches** (unsigned cls, unsigned pt, const T \*patches)

### Protected Attributes

- unsigned **step\_**
- unsigned **trainingSize\_**
- std::vector< std::string > **imName\_**
- std::vector< std::vector< const T \* > > **patches\_**
- unsigned **noCls\_**
- unsigned **labW\_**
- unsigned **labH\_**
- CvRNG \* **cvRNG\_**
- unsigned **featW\_**
- unsigned **featH\_**
- unsigned **consideredCls\_**
- F \* **features\_**
- bool **balance\_**
- unsigned **noPatches\_**

### 3.8.1 Member Function Documentation

**3.8.1.1** `template<class T, class F> void StructurePatch< T, F >::extractFeatures ( const cv::Mat * labimg, IplImage * img, const std::string & path2feat, bool showWhere = false ) [virtual]`

Computes features if not there for loading.

**3.8.1.2** `template<class T, class F> void StructurePatch< T, F >::extractPatches ( const std::string & imgpath, const std::string & labpath, const std::string & featpath, const std::vector< std::string > & vFileNames, const std::map< cv::Vec3b, unsigned, vec3bCompare > & classinfo, const std::string & term, const std::string & ext ) [virtual]`

Extracts the feature patches but also the label patches. *imgpath* – path to the images *labpath* – path to labels *featpath* – path to features *vFileNames* – vector of image names *classinfo* – mapping from pixel color to label ID *labH* – label patch height *labW* – label patch width

**3.8.1.3** `template<class T, class F> unsigned StructurePatch< T, F >::getLabelSize ( ) const [inline]`

Gets the label-patch size.

**3.8.1.4** `template<class T, class F> unsigned StructurePatch< T, F >::getPatchChannels ( ) const`

Gets the number of feature channels.

**3.8.1.5** `template<class T, class F> cv::Mat * StructurePatch< T, F >::loadLabels ( const std::string & justname, const std::string & ext, const std::string & term, const std::string & labpath, const std::map< cv::Vec3b, unsigned, vec3bCompare > & classinfo, const std::string & featpath )`

Get image labels. Try loading it, if not there compute it (very slow).

**3.8.1.6** `template<class T, class F> void StructurePatch< T, F >::loadPatches ( const std::string & path2feat, bool showWhere = false ) [virtual]`

Loads the labels and the image features — 1 file per image.

**3.8.1.7** `template<class T, class F> unsigned StructurePatch< T, F >::noImages ( ) const [inline]`

Get the total number of training images.

**3.8.1.8** `template<class T, class F> unsigned StructurePatch< T, F >::noPatches ( ) const [inline]`

Getter for class members.

**3.8.1.9** `template<class T, class F> void StructurePatch< T, F >::noPatches ( unsigned noPatches ) [inline]`

Setter for patches.

**3.8.1.10** `template<class T, class F> void StructurePatch< T, F >::pickRandomNames ( const std::vector< std::string > & filenames ) [virtual]`

Randomly picks a subset of the images names to be used for training.

**3.8.1.11** `template<class T, class F> void StructurePatch< T, F >::reset ( ) [virtual]`

Resets the class members to add new patches.

**3.8.1.12** `template<class T, class F> void StructurePatch< T, F >::resizePatches ( )`

Resize patches to the number of patches to be used for training.

**3.8.1.13** `template<class T, class F> void StructurePatch< T, F >::savePatches ( const std::string & path2feat, unsigned pos ) [virtual]`

Saves the labels and the image features — for each image make one file.

## 3.8.2 Member Data Documentation

**3.8.2.1** `template<class T, class F> bool StructurePatch< T, F >::balance_ [protected]`

`balance_` To balance the class ratios or not.

**3.8.2.2** `template<class T, class F> StructurePatch< T, F >::consideredCls_ [protected]`

Number of classes we consider.

**3.8.2.3** `template<class T, class F> StructurePatch< T, F >::cvRNG_ [protected]`

For picking random stuff.



**3.8.2.4** `template<class T, class F> StructurePatch< T, F >::featH_` [protected]

The height of the feature patch.

**3.8.2.5** `template<class T, class F> F* StructurePatch< T, F >::features_` [protected]

feature\_ An instance of the class feature.

**3.8.2.6** `template<class T, class F> StructurePatch< T, F >::featW_` [protected]

The width of the feature patch.

**3.8.2.7** `template<class T, class F> StructurePatch< T, F >::imName_` [protected]

The original image name.

**3.8.2.8** `template<class T, class F> StructurePatch< T, F >::labH_` [protected]

Label patch height (redundant but how else?).

**3.8.2.9** `template<class T, class F> StructurePatch< T, F >::labW_` [protected]

Label patch width (redundant but how else?).

**3.8.2.10** `template<class T, class F> StructurePatch< T, F >::noCls_` [protected]

Number of classes.

**3.8.2.11** `template<class T, class F> unsigned StructurePatch< T, F >::noPatches_` [protected]

noPatches\_ Regardless of the number of features we can have a desired number of patches for training.

**3.8.2.12** `template<class T, class F> StructurePatch< T, F >::patches_` [protected]

The vector of LabelPatchFeatures per classes.

**3.8.2.13** `template<class T, class F> unsigned StructurePatch< T, F >::step_` [protected]

unsigned step\_ Lattice step for picking up features.

**3.8.2.14** `template<class T, class F> unsigned StructurePatch< T, F >::trainingSize_` [protected]

unsigned trainingSize\_ Number of images to sample for training one tree.

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

- StructurePatch.h
- StructurePatch.cpp

### 3.9 StructureRF< L, M, T, F, U > Class Template Reference

#### Public Member Functions

- **StructureRF** (int trees=0)
- void **loadForest** (std::string &filename, bool binary)
- void **trainForest** (int min\_s, int max\_d, CvRNG \*pRNG, const M &TrData, int samples, const char \*path2models, const std::string &runName, typename **StructureTree**< M, T, F, U >::ENTROPY entropy, unsigned consideredCls, bool binary)
- void **trainForestTree** (unsigned min\_s, unsigned max\_d, CvRNG \*pRNG, const M &TrData, unsigned samples, unsigned treeld, const char \*path2models, const std::string &runName, typename **StructureTree**< M, T, F, U >::ENTROPY entropy, unsigned consideredCls, bool binary)
- std::vector< std::vector< float > > **treeClsFreq** () const
- void **saveForest** (const char \*filename)
- void **saveTree** (const char \*filename, unsigned treeld)
- void **saveForestBin** (const char \*filename)
- void **saveTreeBin** (const char \*filename, unsigned treeld)
- void **loadForestBin** (std::string &filename, bool binary)
- void **loadTreeBin** (const char \*filename, unsigned treeld, bool binary)
- void **loadTree** (const char \*filename, unsigned treeld, bool binary)
- void **regressionPerTree** (std::vector< const U \* > &result, const T \*testPatch, const F \*features, const char \*filename, unsigned treeld) const
- virtual void **regression** (std::vector< const U \* > &result, const T \*testPatch, const F \*features) const
- virtual void **regressionPerTree** (const U \*result, const T \*testPatch, const F \*features, const char \*filename, unsigned treeld)
- unsigned **noTrees** () const
- std::vector< L< M, T, F, U > \* > **vTrees** () const
- L< M, T, F, U > \* **vTrees** (unsigned pos) const
- void **noTrees** (unsigned noTrees)
- void **vTrees** (const std::vector< L< M, T, F, U > \* > &vTrees)
- void **vTrees** (unsigned pos, const L< M, T, F, U > \*tree)
- **StructureRF** (const **StructureRF** &rhs)
- **StructureRF** & **operator=** (const **StructureRF** &rhs)

#### Protected Attributes

- unsigned **noTrees\_**
- std::vector< L< M, T, F, U > \* > **vTrees\_**

#### 3.9.1 Constructor & Destructor Documentation

- 3.9.1.1 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U> StructureRF< L, M, T, F, U >::StructureRF ( const StructureRF< L, M, T, F, U > & rhs ) [inline]`

Copy constructors because we use vectors of pointers to trees.

#### 3.9.2 Member Function Documentation

- 3.9.2.1 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U > void StructureRF< L, M, T, F, U >::loadForest ( std::string & filename, bool binary )`

Loads all the trees from directory.

3.9.2.2 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void  
StructureRF< L, M, T, F, U >::loadForestBin ( std::string & filename, bool binary )`

Loads all the trees from directory.

3.9.2.3 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void  
StructureRF< L, M, T, F, U >::loadTree ( const char * filename, unsigned treeld, bool binary )`

Loads a trees from directory.

3.9.2.4 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void  
StructureRF< L, M, T, F, U >::loadTreeBin ( const char * filename, unsigned treeld, bool binary )`

Loads a trees from directory.

3.9.2.5 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U> unsigned  
StructureRF< L, M, T, F, U >::noTrees ( ) const [inline]`

Getter for the trees.

3.9.2.6 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U> void  
StructureRF< L, M, T, F, U >::noTrees ( unsigned noTrees ) [inline]`

Setter for the trees.

3.9.2.7 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void  
StructureRF< L, M, T, F, U >::regression ( std::vector< const U * > & result, const T * testPatch, const F *  
features ) const [virtual]`

Predicts on 1 single test patch.

3.9.2.8 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void  
StructureRF< L, M, T, F, U >::regressionPerTree ( std::vector< const U * > & result, const T * testPatch, const F *  
features, const char * filename, unsigned treeld ) const`

Predicts on 1 single test patch.

3.9.2.9 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void  
StructureRF< L, M, T, F, U >::regressionPerTree ( const U * result, const T * testPatch, const F * features, const  
char * filename, unsigned treeld ) [virtual]`

Predicts on 1 single test patch.

3.9.2.10 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void  
StructureRF< L, M, T, F, U >::saveForest ( const char * filename )`

Writes the trees in the forest to files.

3.9.2.11 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void  
StructureRF< L, M, T, F, U >::saveForestBin ( const char * filename )`

Writes the trees in the forest to binary files.

**3.9.2.12** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void  
StructureRF< L, M, T, F, U >::saveTree ( const char * filename, unsigned treeld )`

Writes the trees in the forest to files.

**3.9.2.13** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void  
StructureRF< L, M, T, F, U >::saveTreeBin ( const char * filename, unsigned treeld )`

Writes the trees in the forest to binary files.

**3.9.2.14** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void  
StructureRF< L, M, T, F, U >::trainForest ( int min_s, int max_d, CvRNG * pRNG, const M & TrData, int samples,  
const char * path2models, const std::string & runName, typename StructureTree< M, T, F, U >::ENTROPY entropy,  
unsigned consideredCls, bool binary )`

Trains the forest on the given patches.

Forest training: training each tree. min\_s – minimum samples max\_d – maximum depth samples – total samples

**3.9.2.15** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void  
StructureRF< L, M, T, F, U >::trainForestTree ( unsigned min_s, unsigned max_d, CvRNG * pRNG, const M &  
TrData, unsigned samples, unsigned treeld, const char * path2models, const std::string & runName, typename  
StructureTree< M, T, F, U >::ENTROPY entropy, unsigned consideredCls, bool binary )`

Trains a specified tree in the forest on the given patches.

**3.9.2.16** `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > std::vector<  
std::vector< float > > StructureRF< L, M, T, F, U >::treeClsFreq ( ) const`

Returns a vector of class prior frequencies (one for each tree).

### 3.9.3 Member Data Documentation

**3.9.3.1** `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U> StructureRF< L,  
M, T, F, U >::noTrees_ [protected]`

The number of trees to use.

**3.9.3.2** `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U> StructureRF< L,  
M, T, F, U >::vTrees_ [protected]`

The vector of pointer to trees.

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

- StructureRF.h
- StructureRF.cpp

## 3.10 StructureRFdetector< L, M, T, F, U > Class Template Reference

## Public Member Functions

- **StructureRFdetector** ([StructureRF](#)< L, M, T, F, U > \*pRF, int w, int h, unsigned cls, unsigned labW, unsigned labH, [Puzzle](#)< [PuzzlePatch](#) >::METHOD method, unsigned [step](#))
- virtual void [detectColor](#) (const F \*features, cv::Mat &imgDetect, const std::vector< const T \* > &patches, const cv::Size &imsz) const
- virtual void [detectPyramid](#) (const std::string &imname, const std::string &path2img, const std::string &path2feat, const std::string &ext, const std::vector< float > &pyramid, std::vector< cv::Mat > &vImgDetect) const
- virtual std::vector< std::vector< [IplImage](#) \* > > [getFeatures](#) (const std::string &path2img, const std::vector< std::string > &path2feat, const std::vector< float > &pyr, std::vector< std::vector< const T \* > > &patches, cv::Size &origsz, bool showWhere=false) const
- virtual std::vector< std::vector< [IplImage](#) \* > > [loadFeatures](#) (const std::vector< std::string > &path2feat, std::vector< std::vector< const T \* > > &patches, bool showWhere=false) const
- virtual void [saveFeatures](#) (const std::vector< std::string > &path2feat, const std::vector< std::vector< [IplImage](#) \* > > &vImg) const
- unsigned [step](#) () const
- const [StructureRF](#)< L, M, T, F, U > \* **forest** () const
- int **width** () const
- int **height** () const
- unsigned **labW** () const
- unsigned **labH** () const
- unsigned **noCls** () const
- [Puzzle](#)< [PuzzlePatch](#) >::METHOD **method** () const
- unsigned **maxsize** () const
- void [step](#) (unsigned step)
- void **width** (int width)
- void **height** (int height)
- void **labH** (unsigned labH)
- void **labW** (unsigned labW)
- void **noCls** (unsigned noCls)
- void **forest** ([StructureRF](#)< L, M, T, F, U > \*forest)
- void **method** ([Puzzle](#)< [PuzzlePatch](#) >::METHOD method)
- void **maxsize** (unsigned maxsize)

## Protected Attributes

- unsigned [maxsize\\_](#)
- unsigned [step\\_](#)
- [StructureRF](#)< L, M, T, F, U > \* [forest\\_](#)
- int [width\\_](#)
- int [height\\_](#)
- unsigned [labW\\_](#)
- unsigned [labH\\_](#)
- unsigned [noCls\\_](#)
- [Puzzle](#)< [PuzzlePatch](#) >::METHOD [method\\_](#)
- bool [storefeat\\_](#)

### 3.10.1 Member Function Documentation

3.10.1.1 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void StructureRFdetector< L, M, T, F, U >::detectColor ( const F * features, cv::Mat & imgDetect, const std::vector< const T * > & patches, const cv::Size & imgsize ) const [virtual]`

Gets an input image and returns a detection image (RF regression). Given a set of predicted leafs for current pixel, get the final label: Simple: [1] Just get the most voted pixel label per position. [Puzzle](#): [2] Optimized the patch selection label **[kontschider]**.

Gets an input image and returns a detection image (pixel labels by RF regression). Given a set of predicted leafs for current pixel, get the final label: Simple: [1] Just get the most voted pixel label per position. [Puzzle](#): [2] Optimized the patch selection label **[kontschider]**.

3.10.1.2 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void StructureRFdetector< L, M, T, F, U >::detectPyramid ( const std::string & imname, const std::string & path2img, const std::string & path2feat, const std::string & ext, const std::vector< float > & pyramid, std::vector< cv::Mat > & vlmgDetect ) const [virtual]`

Scales the image at a number of sizes and it labels each scale [?].

3.10.1.3 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > std::vector< std::vector< IpImage * > > StructureRFdetector< L, M, T, F, U >::getFeatures ( const std::string & path2img, const std::vector< std::string > & path2feat, const std::vector< float > & pyr, std::vector< std::vector< const T * > > & patches, cv::Size & origsize, bool showWhere = false ) const [virtual]`

Extracts or loads the test features for the current test image.

3.10.1.4 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > std::vector< std::vector< IpImage * > > StructureRFdetector< L, M, T, F, U >::loadFeatures ( const std::vector< std::string > & path2feat, std::vector< std::vector< const T * > > & patches, bool showWhere = false ) const [virtual]`

Loads the test features from file for the current test image.

3.10.1.5 `template<template< class M, class T, class F, class U > class L, class M , class T , class F , class U > void StructureRFdetector< L, M, T, F, U >::saveFeatures ( const std::vector< std::string > & path2feat, const std::vector< std::vector< IpImage * > > & vlmg ) const [virtual]`

Loads the test features from file for the current test image.

3.10.1.6 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U> unsigned StructureRFdetector< L, M, T, F, U >::step ( ) const [inline]`

Getter for the class members.

3.10.1.7 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U> void StructureRFdetector< L, M, T, F, U >::step ( unsigned step ) [inline]`

Setter for the class members.

### 3.10.2 Member Data Documentation

3.10.2.1 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U>  
StructureRFdetector< L, M, T, F, U >::forest_ [protected]`

Pointer to the trained forest.

3.10.2.2 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U>  
StructureRFdetector< L, M, T, F, U >::height_ [protected]`

The height of the feature patch.

3.10.2.3 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U>  
StructureRFdetector< L, M, T, F, U >::labH_ [protected]`

The height of the feature patch.

3.10.2.4 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U>  
StructureRFdetector< L, M, T, F, U >::labW_ [protected]`

The width of the label patch.

3.10.2.5 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U>  
StructureRFdetector< L, M, T, F, U >::maxsize_ [protected]`

The maximum image size.

3.10.2.6 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U>  
StructureRFdetector< L, M, T, F, U >::method_ [protected]`

The label selection method.

3.10.2.7 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U>  
StructureRFdetector< L, M, T, F, U >::noCls_ [protected]`

The number of classes.

3.10.2.8 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U>  
StructureRFdetector< L, M, T, F, U >::step_ [protected]`

The step of the grid for sampling.

3.10.2.9 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U>  
StructureRFdetector< L, M, T, F, U >::storefeat_ [protected]`

If we store the features or not.

3.10.2.10 `template<template< class M, class T, class F, class U > class L, class M, class T, class F, class U>  
StructureRFdetector< L, M, T, F, U >::width_ [protected]`

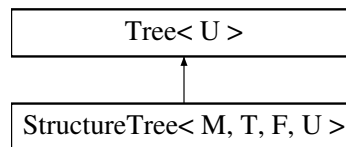
The width of the feature patch.

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

- StructureRFdetector.h
- StructureRFdetector.cpp

### 3.11 StructureTree< M, T, F, U > Class Template Reference

Inheritance diagram for StructureTree< M, T, F, U >:



#### Public Types

- enum [ENTROPY](#) {  
    **CENTER, RANDOM, CENTER\_RANDOM, MEAN\_DIFF,**  
    **APPROX\_MAGNI\_KERNEL, APPROX\_ANGLE\_KERNEL** }
- typedef std::vector  
    < std::vector< const T \* >  
    >::const\_iterator **vectConstIterT**
- typedef std::vector< const T \* >  
    ::const\_iterator **constIterT**
- typedef std::vector  
    < std::vector< const T \* >  
    >::iterator **vectIterT**
- typedef std::vector< const T \* >  
    ::iterator **IterT**

#### Public Member Functions

- [StructureTree](#) (const char \*filename, unsigned treeid, bool binary)
- **StructureTree** (unsigned minS, unsigned maxD, CvRNG \*pRNG, unsigned labSz, unsigned patchW, unsigned patchH, unsigned patchCh, unsigned [treeid](#), const char \*path2models, const std::string &runName, typename [StructureTree](#)< M, T, F, U >::ENTROPY entropy, unsigned [consideredCls](#), bool binary)
- void [readTreeBin](#) ()
- void [readTreeTxt](#) ()
- void [generateTest](#) (long double \*test, unsigned int max\_w, unsigned int max\_h, unsigned int max\_c)
- unsigned [getNoPatches](#) (const std::vector< std::vector< const T \* > > &trainSet)
- float [nEntropy1Cls](#) (const std::vector< std::vector< const T \* > > &SetA, float &sizeA)
- float [nEntropy1ClsRnd](#) (const std::vector< std::vector< const T \* > > &SetA, const F \*features, float &sizeA, unsigned pick)
- float [nEntropy2Cls](#) (const std::vector< std::vector< const T \* > > &SetA, const F \*features, unsigned pick, float &sizeA)
- float [performSplit](#) (std::vector< std::vector< const T \* > > &tmpA, std::vector< std::vector< const T \* > > &tmpB, const std::vector< std::vector< const T \* > > &TrainSet, const F \*features, const std::vector< std::vector< IntIndex > > &valSet, unsigned pick, int threshold, unsigned &sizeA, unsigned &sizeB)
- bool [optimizeTest](#) (std::vector< std::vector< const T \* > > &SetA, std::vector< std::vector< const T \* > > &SetB, const std::vector< std::vector< const T \* > > &TrainSet, const F \*features, long double \*test, unsigned int iter, unsigned pick, float &best)



- void [evaluateTest](#) (std::vector< std::vector< IntIndex > > &valSet, const long double \*test, const std::vector< std::vector< const T \* > > &TrainSet, const F \*features)
- bool [applyTest](#) (const long double \*test, const T \*testPatch, const F \*features) const
- void [split](#) (std::vector< std::vector< const T \* > > &SetA, std::vector< std::vector< const T \* > > &SetB, const std::vector< std::vector< const T \* > > &TrainSet, const std::vector< std::vector< IntIndex > > &valSet, int t)
- virtual const U \* [regression](#) (const T \*testPatch, const F \*features, [node](#)< U > \*[node](#))
- virtual void [showLeaves](#) (unsigned labWidth, unsigned labHeight, const std::map< cv::Vec3b, unsigned, vec3bCompare > &classinfo)
- virtual bool [saveTree](#) () const
- virtual bool [saveTreeBin](#) () const
- virtual bool [saveTreeTxt](#) () const
- virtual void [initDataSizes](#) (const M &trData)
- virtual void [growTree](#) (const M &trData, int samples)
- virtual void [grow](#) (const std::vector< std::vector< const T \* > > &trainSet, const F \*features, long double &nodeid, unsigned int depth, int samples, [node](#)< U > \*parent, typename [Tree](#)< U >::SIDE side, float &prev-InfGain)
- virtual void [makeLeaf](#) (const F \*features, const std::vector< std::vector< const T \* > > &trainSet, long double nodeid, int depth, [node](#)< U > \*parent, typename [Tree](#)< U >::SIDE side)
- virtual std::vector< std::vector< float > > [getPatchProb](#) (const std::vector< std::vector< const T \* > > &trainSet, const F \*features)
- virtual float [measureSet](#) (const std::vector< std::vector< const T \* > > &SetA, const std::vector< std::vector< const T \* > > &SetB, const F \*features, unsigned pick)
- virtual float [InfGain](#) (const std::vector< std::vector< const T \* > > &SetA, const std::vector< std::vector< const T \* > > &SetB, const F \*features, unsigned pick)
- unsigned [consideredCls](#) () const
- [ENTROPY](#) [entropy](#) () const
- unsigned [maxDepth](#) () const
- CvRNG \* [cvRNG](#) () const
- unsigned [minSamples](#) () const
- unsigned [nodeSize](#) () const
- unsigned [labSz](#) () const
- unsigned [patchW](#) () const
- unsigned [patchH](#) () const
- unsigned [patchCh](#) () const
- std::ofstream [log](#) () const
- std::vector< float > [clsFreq](#) () const
- float [clsFreq](#) (unsigned pos) const
- std::vector< std::vector< float > > [coFreq](#) () const
- std::vector< float > [coFreq](#) (unsigned pos) const
- float [coFreq](#) (unsigned ctr, unsigned rnd) const
- void [consideredCls](#) (unsigned consideredCls)
- void [entropy](#) ([ENTROPY](#) entropy)
- void [maxDepth](#) (unsigned maxDepth)
- void [cvRNG](#) (CvRNG \*cvRNG)
- void [minSamples](#) (unsigned minSamples)
- void [nodeSize](#) (unsigned nodeSize)
- void [labSz](#) (unsigned labSz)
- void [patchW](#) (unsigned patchW)
- void [patchH](#) (unsigned patchH)
- void [patchCh](#) (unsigned patchCh)
- void [clsFreq](#) (const std::vector< float > &clsFreq)
- void [coFreq](#) (const std::vector< std::vector< float > > &coFreq)
- [StructureTree](#) ([StructureTree](#) const &rhs)
- [StructureTree](#) & [operator=](#) ([StructureTree](#) const &rhs)

## Protected Attributes

- std::ofstream `log_`
- unsigned `labSz_`
- unsigned `patchW_`
- unsigned `patchH_`
- unsigned `patchCh_`
- unsigned `nodeSize_`
- unsigned `minSamples_`
- CvRNG \* `cvRNG_`
- unsigned `maxDepth_`
- `ENTROPY` `entropy_`
- unsigned `consideredCls_`
- std::vector< float > `clsFreq_`
- std::vector< std::vector< float > > `coFreq_`
- bool `binary_`

### 3.11.1 Member Enumeration Documentation

3.11.1.1 `template<class M, class T, class F, class U> enum StructureTree::ENTROPY`

On what do we evaluate the entropy.

### 3.11.2 Constructor & Destructor Documentation

3.11.2.1 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::StructureTree ( const char * filename, unsigned treeld, bool binary )`

Reads the tree from file.

3.11.2.2 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::StructureTree ( StructureTree< M, T, F, U > const & rhs ) [inline]`

Copy constructors for trees (to put them in the forest).

### 3.11.3 Member Function Documentation

3.11.3.1 `template<class M, class T, class F, class U> bool StructureTree< M, T, F, U >::applyTest ( const long double * test, const T * testPatch, const F * features ) const`

Applies a test to a patch.

3.11.3.2 `template<class M, class T, class F, class U> unsigned StructureTree< M, T, F, U >::consideredCls ( ) const [inline]`

Getters for the class members.

3.11.3.3 `template<class M, class T, class F, class U> void StructureTree< M, T, F, U >::consideredCls ( unsigned consideredCls ) [inline]`

Setters for the class members.

```
3.11.3.4 template<class M , class T , class F , class U > void StructureTree< M, T, F, U >::evaluateTest ( std::vector<
std::vector< IntIndex > > & valSet, const long double * test, const std::vector< std::vector< const T * > > &
TrainSet, const F * features )
```

Evaluates 1 test (given by 5 numbers: x1, y1, x2, y2, channel). It gets the feature channel and then it accesses it at the 2 randomly selected points and gets the difference between them.

```
3.11.3.5 template<class M , class T , class F , class U > void StructureTree< M, T, F, U >::generateTest ( long double *
test, unsigned int max_w, unsigned int max_h, unsigned int max_c )
```

Generates a random test of a random type.

```
3.11.3.6 template<class M , class T , class F , class U > unsigned StructureTree< M, T, F, U >::getNoPatches ( const
std::vector< std::vector< const T * > > & trainSet )
```

Just gets total number of patches regardless of class

Just gets total number of patches regardless of class.

```
3.11.3.7 template<class M , class T , class F , class U > std::vector< std::vector< float > > StructureTree< M, T, F, U
>::getPatchProb ( const std::vector< std::vector< const T * > > & trainSet, const F * features ) [virtual]
```

Computes the probabilities of each label-patch given the complete of patches. [1] For each label-patch get its prob as prod of pixel-label probs. [2] Find the label-patch with the maximum prob.

```
3.11.3.8 template<class M , class T , class F , class U > void StructureTree< M, T, F, U >::grow ( const std::vector<
std::vector< const T * > > & trainSet, const F * features, long double & nodeid, unsigned int depth, int samples,
node< U > * parent, typename Tree< U >::SIDE side, float & prevInfGain ) [virtual]
```

Creates the actual tree from the samples.

```
3.11.3.9 template<class M , class T , class F , class U > void StructureTree< M, T, F, U >::growTree ( const M & trData, int
samples ) [virtual]
```

Implementing the <<growTee>> with multiple labels.

```
3.11.3.10 template<class M , class T , class F , class U > float StructureTree< M, T, F, U >::InfGain ( const std::vector<
std::vector< const T * > > & SetA, const std::vector< std::vector< const T * > > & SetB, const F * features,
unsigned pick ) [virtual]
```

Classification information gain check. [1] Associate each app-patch with a random label we pick from the label-patch [2] Compute the negative entropy as:  $\sum_c p(c) \log p(c)$  [3] return:  $(\text{size}(A)\text{entropy}(A) + \text{size}(B)\text{entropy}(B)) / (\text{size}(A) + \text{size}(B))$ .

```
3.11.3.11 template<class M , class T , class F , class U > void StructureTree< M, T, F, U >::initDataSizes ( const M & trData
) [virtual]
```

Initializes the size of the labels, number of channels, etc.

```
3.11.3.12  template<class M , class T , class F , class U > void StructureTree< M, T, F, U >::makeLeaf ( const F * features,
const std::vector< std::vector< const T * > > & trainSet, long double nodeid, int depth, node< U > * parent,
typename Tree< U >::SIDE side ) [virtual]
```

Create leaf node from all patches corresponding to a class.

Create leaf node from all patches corresponding to a class. [1] For each label-patch get its prob as prod of pixel-label probs. [2] Find the label-patch with the maximum prob [3] Add the best patch in the leaf.

```
3.11.3.13  template<class M , class T , class F , class U > float StructureTree< M, T, F, U >::measureSet ( const
std::vector< std::vector< const T * > > & SetA, const std::vector< std::vector< const T * > > & SetB, const F
* features, unsigned pick ) [virtual]
```

Overloading the function to carry around the labels matrices.

```
3.11.3.14  template<class M , class T , class F , class U > float StructureTree< M, T, F, U >::nEntropy1Cls ( const
std::vector< std::vector< const T * > > & SetA, float & totalFreqA )
```

Computes the negative entropy for 1 set wrt to the central pixel of the label patch.

```
3.11.3.15  template<class M , class T , class F , class U > float StructureTree< M, T, F, U >::nEntropy1ClsRnd ( const
std::vector< std::vector< const T * > > & SetA, const F * features, float & totalFreqA, unsigned pick )
```

Computes the negative entropy for 1 set wrt to a random pixel of the label patch.

```
3.11.3.16  template<class M , class T , class F , class U > float StructureTree< M, T, F, U >::nEntropy2Cls ( const
std::vector< std::vector< const T * > > & SetA, const F * features, unsigned pick, float & totalFreqA )
```

Computes the negative entropy for 1 set wrt to the central pixel of the label patch and a randomly picked pixel.

```
3.11.3.17  template<class M , class T , class F , class U > bool StructureTree< M, T, F, U >::optimizeTest ( std::vector<
std::vector< const T * > > & SetA, std::vector< std::vector< const T * > > & SetB, const std::vector<
std::vector< const T * > > & TrainSet, const F * features, long double * test, unsigned int iter, unsigned pick,
float & best )
```

Optimizes tests and thresholds. [1] Generate a 5 random values (for x1 y1 x2 y2 channel) in the <<test>> vector. [2] Evaluates the thresholds and finds the minimum and maximum index value [?]. [3] Iteratively generate random thresholds to split the index values [4] Split the data according to each threshold. [5] Find the best threshold and store it on the 6th position in <<test>>

```
3.11.3.18  template<class M , class T , class F , class U > float StructureTree< M, T, F, U >::performSplit ( std::vector<
std::vector< const T * > > & tmpA, std::vector< std::vector< const T * > > & tmpB, const std::vector<
std::vector< const T * > > & TrainSet, const F * features, const std::vector< std::vector< IntIndex > > & valSet,
unsigned pick, int threshold, unsigned & sizeA, unsigned & sizeB )
```

Just splits the data into subsets and makes sure the subsets are not empty

```
3.11.3.19  template<class M , class T , class F , class U > void StructureTree< M, T, F, U >::readTreeBin ( )
```

Reads the tree from a binary file.

3.11.3.20 `template<class M, class T, class F, class U> void StructureTree< M, T, F, U >::readTreeTxt ( )`

Reads the tree from a text file.

3.11.3.21 `template<class M, class T, class F, class U> const U * StructureTree< M, T, F, U >::regression ( const T * testPatch, const F * features, node< U > * node ) [virtual]`

Predicts on a one single test patch. A node contains: [0] – node type (0,1,-1), [1] – x1, [2] – y1, [3] – x2, [4] – y2, [5] – channel, [6] – threshold,

Predicts on a one single test patch. A node contains: [0] – node type (0,1,-1), [1] – x1, [2] – y1, [3] – x2, [4] – y2, [5] – channel, [6] – threshold, [7] – test type, [8] – node ID

3.11.3.22 `template<class M, class T, class F, class U> bool StructureTree< M, T, F, U >::saveTree ( ) const [virtual]`

Writes the current tree into a given file.

3.11.3.23 `template<class M, class T, class F, class U> bool StructureTree< M, T, F, U >::saveTreeBin ( ) const [virtual]`

Writes the current tree into a given file.

3.11.3.24 `template<class M, class T, class F, class U> bool StructureTree< M, T, F, U >::saveTreeTxt ( ) const [virtual]`

Writes the current tree into a given file.

3.11.3.25 `template<class M, class T, class F, class U> void StructureTree< M, T, F, U >::showLeaves ( unsigned labWidth, unsigned labHeight, const std::map< cv::Vec3b, unsigned, vec3bCompare > & classinfo ) [virtual]`

Displays the leaves of the tree.

3.11.3.26 `template<class M, class T, class F, class U> void StructureTree< M, T, F, U >::split ( std::vector< std::vector< const T * > > & SetA, std::vector< std::vector< const T * > > & SetB, const std::vector< std::vector< const T * > > & TrainSet, const std::vector< std::vector< IntIndex > > & valSet, int t )`

Splits the training samples into a left set and a right set.

### 3.11.4 Member Data Documentation

3.11.4.1 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::binary_ [protected]`

If it is binary tree than save it.

Reimplemented from [Tree< U >](#).

3.11.4.2 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::clsFreq_ [protected]`

The class frequencies in the original training data for unbalanced data.

3.11.4.3 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::coFreq_` [protected]

The class co-frequencies in the original training data for unbalanced data.

3.11.4.4 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::consideredCls_` [protected]

Considered classes (taken in the id order).

3.11.4.5 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::cvRNG_` [protected]

Pointer to a cv random number.

3.11.4.6 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::entropy_` [protected]

How to evaluate the entropy: central label, random label, both.

3.11.4.7 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::log_` [protected]

For output logging during training.

3.11.4.8 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::maxDepth_` [protected]

Maximum tree depth.

3.11.4.9 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::minSamples_` [protected]

Minimum samples to build a test node (else leaf).

3.11.4.10 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::nodeSize_` [protected]

The size of the test-nodes.

3.11.4.11 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::patchCh_` [protected]

The number of channels in the feature matrix.

3.11.4.12 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::patchH_` [protected]

The patch height.

3.11.4.13 `template<class M, class T, class F, class U> StructureTree< M, T, F, U >::patchW_` [protected]

The patch width.

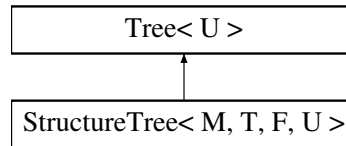
The documentation for this class was generated from the following files:

- StructureTree.h
- StructureTree.cpp

## 3.12 Tree< U > Class Template Reference

```
#include <Tree.h>
```

Inheritance diagram for Tree< U >:



### Public Types

- enum [SIDE](#) { [LEFT](#), [RIGHT](#), [ROOT](#) }

### Public Member Functions

- [Tree](#) (const char \*filename, bool binary)
- void [destroyTree](#) ([node](#)< U > \*anode)
- void [preorderTxt](#) ([node](#)< U > \*node, std::ofstream &out) const
- void [preorderBin](#) ([node](#)< U > \*node, std::ofstream &out) const
- void [showTree](#) ()
- virtual void [readNodeTxt](#) ([node](#)< U > \*parent, std::ifstream &in, [SIDE](#) side)
- virtual void [readNodeBin](#) ([node](#)< U > \*parent, std::ifstream &in, [SIDE](#) side)
- virtual [node](#)< U > \* [addNode](#) ([node](#)< U > \*parent, [SIDE](#) side, long double \*test, unsigned nodeSize, unsigned nodeid, const U \*leaf)
- virtual bool [saveTree](#) (const char \*filename) const
- virtual bool [saveTreeBin](#) (const char \*filename) const
- virtual bool [saveTreeTxt](#) (const char \*filename) const
- unsigned [treeld](#) () const
- [node](#)< U > \* [root](#) () const
- const char \* [path2models](#) () const
- bool [binary](#) () const
- void [treeld](#) (unsigned treeld)
- void [root](#) (const [node](#)< U > \*root)
- void [path2models](#) (const char \*path2models)
- void [binary](#) (bool binary)

### Protected Attributes

- [node](#)< U > \* [root\\_](#)
- unsigned [treeld\\_](#)
- const char \* [path2models\\_](#)
- bool [binary\\_](#)

#### 3.12.1 Detailed Description

```
template<class U>class Tree< U >
```

Standard binary tree class.

### 3.12.2 Member Enumeration Documentation

#### 3.12.2.1 `template<class U> enum Tree::SIDE`

Where to add the new node in the tree: 0 - left, 1 - right, 2 - root

### 3.12.3 Constructor & Destructor Documentation

#### 3.12.3.1 `template<class U> Tree< U >::Tree ( const char * filename, bool binary )`

Reads the tree from the file.

### 3.12.4 Member Function Documentation

#### 3.12.4.1 `template<class U> node< U > * Tree< U >::addNode ( node< U > * parent, SIDE side, long double * test, unsigned nodeSize, unsigned nodeid, const U * leaf ) [virtual]`

Adds a node to the tree given the parent node and the side.

#### 3.12.4.2 `template<class U> void Tree< U >::destroyTree ( node< U > * anode )`

Recursively destroys the nodes in the tree.

#### 3.12.4.3 `template<class U> void Tree< U >::preorderBin ( node< U > * node, std::ofstream & out ) const`

Traverses the tree in preorder OLR and displays the tests.

#### 3.12.4.4 `template<class U> void Tree< U >::preorderTxt ( node< U > * node, std::ofstream & out ) const`

Traverses the tree in inorder LOR and displays the tests.

Traverses the tree in preorder OLR and displays the tests.

#### 3.12.4.5 `template<class U> void Tree< U >::readNodeBin ( node< U > * parent, std::ifstream & in, SIDE side ) [virtual]`

Recursively read tree from binary file.

#### 3.12.4.6 `template<class U> void Tree< U >::readNodeTxt ( node< U > * parent, std::ifstream & in, SIDE side ) [virtual]`

Recursively read tree from file.

#### 3.12.4.7 `template<class U> bool Tree< U >::saveTree ( const char * filename ) const [virtual]`

Saves the tree to file.

#### 3.12.4.8 `template<class U> bool Tree< U >::saveTreeBin ( const char * filename ) const [virtual]`

Saves the tree to binary file.



3.12.4.9 `template<class U> bool Tree< U >::saveTreeTxt ( const char * filename ) const` [virtual]

Saves the tree in a txt file.

3.12.4.10 `template<class U> void Tree< U >::showTree ( )`

Show tree — traverses the tree in inorder LOR.

3.12.4.11 `template<class U> unsigned Tree< U >::treeld ( ) const` [inline]

Getters for the members.

3.12.4.12 `template<class U> void Tree< U >::treeld ( unsigned treeld )` [inline]

Setters for the members.

### 3.12.5 Member Data Documentation

3.12.5.1 `template<class U> Tree< U >::binary_` [protected]

If the tree should be binary or not

Reimplemented in [StructureTree< M, T, F, U >](#).

3.12.5.2 `template<class U> Tree< U >::path2models_` [protected]

The path to the Trees.

3.12.5.3 `template<class U> Tree< U >::root_` [protected]

The root node of the tree.

3.12.5.4 `template<class U> unsigned Tree< U >::treeld_` [protected]

treeld\_ [Tree](#) id to know which tree are we working on.

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

- Tree.h
- Tree.cpp

# Index

- addNode
  - Tree, [38](#)
- agreement
  - PuzzlePatch, [14](#)
- applyTest
  - StructureTree, [32](#)
- balance\_
  - RunRF, [17](#)
  - StructurePatch, [22](#)
- binary
  - RunRF, [16](#)
- binary\_
  - RunRF, [17](#)
  - StructureTree, [35](#)
  - Tree, [39](#)
- center\_
  - PuzzlePatch, [14](#)
- checkConvergence
  - Puzzle, [12](#)
- classInfo\_
  - RunRF, [17](#)
- clone
  - node, [11](#)
- clsFreq\_
  - Features, [7](#)
  - StructureTree, [35](#)
- coFreq\_
  - Features, [7](#)
  - StructureTree, [35](#)
- consideredCls
  - StructureTree, [32](#)
- consideredCls\_
  - RunRF, [17](#)
  - StructurePatch, [22](#)
  - StructureTree, [36](#)
- cvRNG\_
  - StructurePatch, [22](#)
  - StructureTree, [36](#)
- destroyTree
  - Tree, [38](#)
- detect
  - RunRF, [16](#)
- detectColor
  - StructureRFdetector, [28](#)
- detectPyramid
  - StructureRFdetector, [28](#)
- display
  - LabelLeafNode, [8](#)
- ENTROPY
  - StructureTree, [32](#)
- entropy\_
  - RunRF, [18](#)
  - StructureTree, [36](#)
- evaluateTest
  - StructureTree, [32](#)
- ext\_
  - RunRF, [18](#)
- extractFeatures
  - StructurePatch, [21](#)
- extractPatches
  - StructurePatch, [21](#)
- feat
  - LabelPatchFeature, [10](#)
- featH\_
  - LabelPatchFeature, [10](#)
  - StructurePatch, [22](#)
- featRow
  - LabelPatchFeature, [10](#)
- featW\_
  - LabelPatchFeature, [10](#)
  - StructurePatch, [23](#)
- Features, [5](#)
  - clsFreq\_, [7](#)
  - coFreq\_, [7](#)
  - invClsFreq, [6](#)
  - invCoFreq, [6](#)
  - labelImg, [6](#)
  - labelImg\_, [7](#)
  - labelSize, [6](#)
  - push\_backImg, [6](#)
  - push\_backLab, [6](#)
  - size, [6](#)
  - size\_, [7](#)
  - updateClsCoFreq, [6](#)
  - updateClsFreq, [6](#)
  - vImg\_, [7](#)
- features\_
  - StructurePatch, [23](#)
- forest\_
  - StructureRFdetector, [29](#)
- generateTest
  - StructureTree, [33](#)
- getColorLabels
  - RunRF, [17](#)

- getFeatures
  - StructureRFdetector, 28
- getLabelSize
  - StructurePatch, 21
- getNoPatches
  - StructureTree, 33
- getPatchChannels
  - StructurePatch, 21
- getPatchProb
  - StructureTree, 33
- grow
  - StructureTree, 33
- growTree
  - StructureTree, 33
- height\_
  - StructureRFdetector, 29
- imIndex\_
  - LabelPatchFeature, 10
- imName\_
  - StructurePatch, 23
- InfGain
  - StructureTree, 33
- initDataSizes
  - StructureTree, 33
- initialPick
  - Puzzle, 12
- invClsFreq
  - Features, 6
- invCoFreq
  - Features, 6
- iterPerNode\_
  - RunRF, 18
- labH
  - LabelPatchFeature, 10
- labH\_
  - LabelPatchFeature, 10
  - StructurePatch, 23
  - StructureRFdetector, 29
- labHeight\_
  - RunRF, 18
- labW\_
  - LabelPatchFeature, 11
  - StructurePatch, 23
  - StructureRFdetector, 29
- labWidth\_
  - RunRF, 18
- label
  - LabelPatchFeature, 10
- labelImg
  - Features, 6
- labelImg\_
  - Features, 7
- LabelLeafNode, 7
  - display, 8
  - LabelLeafNode, 8
  - labelProb, 8
- LabelLeafNode, 8
  - print, 8
  - readLeafBin, 8
  - readLeafTxt, 8
  - showLeafBin, 8
  - showLeafTxt, 8
- LabelPatchFeature
  - feat, 10
  - featH\_, 10
  - featRow, 10
  - featW\_, 10
  - imIndex\_, 10
  - labH, 10
  - labH\_, 10
  - labW\_, 11
  - label, 10
  - LabelPatchFeature, 9
  - LabelPatchFeature, 9
  - vecFeatRow, 10
- LabelPatchFeature< F >, 9
- labelProb
  - LabelLeafNode, 8
- labelSize
  - Features, 6
- loadFeatures
  - StructureRFdetector, 28
- loadForest
  - StructureRF, 24
- loadForestBin
  - StructureRF, 24
- loadLabels
  - StructurePatch, 21
- loadPatches
  - StructurePatch, 21
- loadTree
  - StructureRF, 25
- loadTreeBin
  - StructureRF, 25
- log\_
  - StructureTree, 36
- logProb
  - PuzzlePatch, 14
- logProb\_
  - PuzzlePatch, 14
- METHOD
  - Puzzle, 12
- MODE
  - RunRF, 16
- makeLeaf
  - StructureTree, 33
- maxDepth\_
  - StructureTree, 36
- maxsize\_
  - StructureRFdetector, 29
- measureSet
  - StructureTree, 34
- method\_
  - StructureRFdetector, 29

- minSamples\_
  - StructureTree, 36
- nEntropy1Cls
  - StructureTree, 34
- nEntropy1ClsRnd
  - StructureTree, 34
- nEntropy2Cls
  - StructureTree, 34
- noCls\_
  - StructurePatch, 23
  - StructureRFdetector, 29
- noImages
  - StructurePatch, 22
- noPatches
  - StructurePatch, 22
- noPatches\_
  - RunRF, 18
  - StructurePatch, 23
- noTrees
  - StructureRF, 25
- noTrees\_
  - RunRF, 18
  - StructureRF, 26
- node
  - clone, 11
  - nodeid, 11
- node < U >, 11
- nodeSize\_
  - StructureTree, 36
- nodeid
  - node, 11
- optimizeTest
  - StructureTree, 34
- patchCh\_
  - StructureTree, 36
- patchH\_
  - StructureTree, 36
- patchHeight\_
  - RunRF, 18
- patchW\_
  - StructureTree, 36
- patchWidth\_
  - RunRF, 18
- patches\_
  - StructurePatch, 23
- path2feat\_
  - RunRF, 18
- path2labs\_
  - RunRF, 19
- path2model\_
  - RunRF, 19
- path2models\_
  - Tree, 39
- path2results\_
  - RunRF, 19
- path2test\_
  - RunRF, 19
- path2train\_
  - RunRF, 19
- performSplit
  - StructureTree, 34
- pickRandomNames
  - StructurePatch, 22
- piece\_
  - PuzzlePatch, 14
- pos2pt
  - PuzzlePatch, 14
- predMethod\_
  - RunRF, 19
- preorderBin
  - Tree, 38
- preorderTxt
  - Tree, 38
- print
  - LabelLeafNode, 8
- proposeLabeling
  - Puzzle, 12
- push\_backImg
  - Features, 6
- push\_backLab
  - Features, 6
- Puzzle
  - checkConvergence, 12
  - initialPick, 12
  - METHOD, 12
  - proposeLabeling, 12
  - selectPatches, 12
  - solve, 13
- Puzzle < P >, 12
- PuzzlePatch, 13
  - agreement, 14
  - center\_, 14
  - logProb, 14
  - logProb\_, 14
  - piece\_, 14
  - pos2pt, 14
  - PuzzlePatch, 13
  - PuzzlePatch, 13
- readLeafBin
  - LabelLeafNode, 8
- readLeafTxt
  - LabelLeafNode, 8
- readNodeBin
  - Tree, 38
- readNodeTxt
  - Tree, 38
- readTreeBin
  - StructureTree, 34
- readTreeTxt
  - StructureTree, 34
- regression
  - StructureRF, 25
  - StructureTree, 35
- regressionPerTree

- StructureRF, 25
- reset
  - StructurePatch, 22
- resizePatches
  - StructurePatch, 22
- root\_
  - Tree, 39
- run
  - RunRF, 17
- runDetect
  - RunRF, 17
- runExtract
  - RunRF, 17
- runName\_
  - RunRF, 19
- RunRF
  - balance\_, 17
  - binary, 16
  - binary\_, 17
  - classInfo\_, 17
  - consideredCls\_, 17
  - detect, 16
  - entropy\_, 18
  - ext\_, 18
  - getColorLabels, 17
  - iterPerNode\_, 18
  - labHeight\_, 18
  - labWidth\_, 18
  - MODE, 16
  - noPatches\_, 18
  - noTrees\_, 18
  - patchHeight\_, 18
  - patchWidth\_, 18
  - path2feat\_, 18
  - path2labs\_, 19
  - path2model\_, 19
  - path2results\_, 19
  - path2test\_, 19
  - path2train\_, 19
  - predMethod\_, 19
  - run, 17
  - runDetect, 17
  - runExtract, 17
  - runName\_, 19
  - runTrain, 17
  - step\_, 19
  - trainSize\_, 19
- RunRF< L, M, T, F, U >, 14
- runTrain
  - RunRF, 17
- SIDE
  - Tree, 38
- saveFeatures
  - StructureRFdetector, 28
- saveForest
  - StructureRF, 25
- saveForestBin
  - StructureRF, 25
- savePatches
  - StructurePatch, 22
- saveTree
  - StructureRF, 25
  - StructureTree, 35
  - Tree, 38
- saveTreeBin
  - StructureRF, 26
  - StructureTree, 35
  - Tree, 38
- saveTreeTxt
  - StructureTree, 35
  - Tree, 38
- selectPatches
  - Puzzle, 12
- showLeafBin
  - LabelLeafNode, 8
- showLeafTxt
  - LabelLeafNode, 8
- showLeaves
  - StructureTree, 35
- showTree
  - Tree, 39
- size
  - Features, 6
- size\_
  - Features, 7
- solve
  - Puzzle, 13
- split
  - StructureTree, 35
- step
  - StructureRFdetector, 28
- step\_
  - RunRF, 19
  - StructurePatch, 23
  - StructureRFdetector, 29
- storefeat\_
  - StructureRFdetector, 29
- StructurePatch
  - balance\_, 22
  - consideredCls\_, 22
  - cvRNG\_, 22
  - extractFeatures, 21
  - extractPatches, 21
  - featH\_, 22
  - featW\_, 23
  - features\_, 23
  - getLabelSize, 21
  - getPatchChannels, 21
  - imName\_, 23
  - labH\_, 23
  - labW\_, 23
  - loadLabels, 21
  - loadPatches, 21
  - noCls\_, 23
  - noImages, 22
  - noPatches, 22

- noPatches\_, 23
- patches\_, 23
- pickRandomNames, 22
- reset, 22
- resizePatches, 22
- savePatches, 22
- step\_, 23
- trainingSize\_, 23
- StructurePatch< T, F >, 20
- StructureRF
  - loadForest, 24
  - loadForestBin, 24
  - loadTree, 25
  - loadTreeBin, 25
  - noTrees, 25
  - noTrees\_, 26
  - regression, 25
  - regressionPerTree, 25
  - saveForest, 25
  - saveForestBin, 25
  - saveTree, 25
  - saveTreeBin, 26
  - StructureRF, 24
  - StructureRF, 24
  - trainForest, 26
  - trainForestTree, 26
  - treeClsFreq, 26
  - vTrees\_, 26
- StructureRF< L, M, T, F, U >, 24
- StructureRFdetector
  - detectColor, 28
  - detectPyramid, 28
  - forest\_, 29
  - getFeatures, 28
  - height\_, 29
  - labH\_, 29
  - labW\_, 29
  - loadFeatures, 28
  - maxsize\_, 29
  - method\_, 29
  - noCls\_, 29
  - saveFeatures, 28
  - step, 28
  - step\_, 29
  - storefeat\_, 29
  - width\_, 29
- StructureRFdetector< L, M, T, F, U >, 26
- StructureTree
  - applyTest, 32
  - binary\_, 35
  - clsFreq\_, 35
  - coFreq\_, 35
  - consideredCls, 32
  - consideredCls\_, 36
  - cvRNG\_, 36
  - ENTROPY, 32
  - entropy\_, 36
  - evaluateTest, 32
  - generateTest, 33
  - getNoPatches, 33
  - getPatchProb, 33
  - grow, 33
  - growTree, 33
  - InfGain, 33
  - initDataSizes, 33
  - log\_, 36
  - makeLeaf, 33
  - maxDepth\_, 36
  - measureSet, 34
  - minSamples\_, 36
  - nEntropy1Cls, 34
  - nEntropy1ClsRnd, 34
  - nEntropy2Cls, 34
  - nodeSize\_, 36
  - optimizeTest, 34
  - patchCh\_, 36
  - patchH\_, 36
  - patchW\_, 36
  - performSplit, 34
  - readTreeBin, 34
  - readTreeTxt, 34
  - regression, 35
  - saveTree, 35
  - saveTreeBin, 35
  - saveTreeTxt, 35
  - showLeaves, 35
  - split, 35
  - StructureTree, 32
  - StructureTree, 32
- StructureTree< M, T, F, U >, 30
- trainForest
  - StructureRF, 26
- trainForestTree
  - StructureRF, 26
- trainSize\_
  - RunRF, 19
- trainingSize\_
  - StructurePatch, 23
- Tree
  - addNode, 38
  - binary\_, 39
  - destroyTree, 38
  - path2models\_, 39
  - preorderBin, 38
  - preorderTxt, 38
  - readNodeBin, 38
  - readNodeTxt, 38
  - root\_, 39
  - SIDE, 38
  - saveTree, 38
  - saveTreeBin, 38
  - saveTreeTxt, 38
  - showTree, 39
  - Tree, 38
  - treeld, 39
  - treeld\_, 39

---

Tree< U >, [37](#)  
treeClsFreq  
    StructureRF, [26](#)  
treeld  
    Tree, [39](#)  
treeld\_  
    Tree, [39](#)  
  
updateClsCoFreq  
    Features, [6](#)  
updateClsFreq  
    Features, [6](#)  
  
vImg\_  
    Features, [7](#)  
vTrees\_  
    StructureRF, [26](#)  
vecFeatRow  
    LabelPatchFeature, [10](#)  
  
width\_  
    StructureRFdetector, [29](#)