

AFS

Generated by Doxygen 1.8.6

Fri Mar 6 2015 11:22:10

Contents

1	Module Index	1
1.1	Modules	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	Module Documentation	7
4.1	CUDA_PRIMALDUAL_OPTIMIZATION	7
4.1.1	Detailed Description	7
4.1.2	Function Documentation	7
4.1.2.1	__cudaSafeCall	7
4.1.2.2	call_binarization	8
4.1.2.3	call_segmentation	8
4.1.2.4	kernel_binarize_u	9
4.1.2.5	kernel_grad_ascent	9
4.1.2.6	kernel_grad_descent	9
5	Class Documentation	11
5.1	ColorFeatures Class Reference	11
5.1.1	Detailed Description	12
5.1.2	Constructor & Destructor Documentation	12
5.1.2.1	ColorFeatures	12
5.1.3	Member Function Documentation	13
5.1.3.1	extractFeatures	13
5.2	Corel Class Reference	13
5.2.1	Detailed Description	14
5.2.2	Member Function Documentation	14
5.2.2.1	Label2RGB	15
5.2.2.2	RGB2Label	16
5.3	CvRTreesMultiClass Class Reference	16

5.3.1	Detailed Description	17
5.3.2	Member Function Documentation	17
5.3.2.1	predict_multi_class	17
5.4	Dataset Class Reference	18
5.4.1	Detailed Description	21
5.4.2	Member Function Documentation	22
5.4.2.1	clearDir	22
5.4.2.2	createDir	22
5.4.2.3	findImage	22
5.4.2.4	getDatasetImages	23
5.4.2.5	getFullFeatureDim	24
5.4.2.6	Label2RGB	25
5.4.2.7	readImgsInDir	25
5.4.2.8	readSplitsToVectors	26
5.4.2.9	RGB2Label	27
5.5	DenseUnaryPixelPotential Class Reference	27
5.5.1	Detailed Description	28
5.5.2	Constructor & Destructor Documentation	28
5.5.2.1	DenseUnaryPixelPotential	28
5.5.3	Member Function Documentation	29
5.5.3.1	Evaluate	29
5.5.3.2	Train	30
5.6	DepthFeatures Class Reference	31
5.6.1	Detailed Description	32
5.6.2	Constructor & Destructor Documentation	32
5.6.2.1	DepthFeatures	33
5.6.3	Member Function Documentation	34
5.6.3.1	extractFeatures	34
5.7	eTrims Class Reference	34
5.7.1	Detailed Description	35
5.7.2	Member Function Documentation	35
5.7.2.1	Label2RGB	36
5.7.2.2	RGB2Label	37
5.8	Feature Class Reference	37
5.8.1	Detailed Description	39
5.8.2	Member Function Documentation	39
5.8.2.1	extractFeatures	39
5.8.2.2	getFeatureDim	40
5.8.2.3	getFullDim	40
5.8.2.4	getSubSample	40

5.8.2.5	loadFeature	41
5.8.2.6	saveFeature	41
5.8.2.7	setFeatureDim	42
5.8.2.8	setRectSize	42
5.8.2.9	setSubSample	43
5.9	HaarLikeFeatures Class Reference	44
5.9.1	Detailed Description	45
5.9.2	Constructor & Destructor Documentation	46
5.9.2.1	HaarLikeFeatures	46
5.9.3	Member Function Documentation	47
5.9.3.1	centerSurround	47
5.9.3.2	extractFeatures	47
5.9.3.3	fourSquare	48
5.9.3.4	horizontalEdge	49
5.9.3.5	horizontalLine	49
5.9.3.6	verticalEdge	50
5.9.3.7	verticalLine	50
5.10	Image Class Reference	50
5.10.1	Detailed Description	51
5.10.2	Constructor & Destructor Documentation	51
5.10.2.1	Image	51
5.10.3	Member Function Documentation	52
5.10.3.1	getHeight	52
5.10.3.2	getImage	53
5.10.3.3	getRGB	53
5.10.3.4	getWidth	53
5.10.3.5	isEmpty	54
5.11	ImageProc Class Reference	55
5.11.1	Detailed Description	55
5.11.2	Member Function Documentation	55
5.11.2.1	convertColor	55
5.11.2.2	getChannel	56
5.11.2.3	getEdgeImage	56
5.11.2.4	getIntegralImage	57
5.11.2.5	getPaddedImg	58
5.12	Learning Class Reference	59
5.12.1	Detailed Description	60
5.12.2	Member Function Documentation	60
5.12.2.1	Evaluate	60
5.12.2.2	getTrainTime	61

5.12.2.3 Train	61
5.13 LocationFeatures Class Reference	61
5.13.1 Detailed Description	62
5.13.2 Constructor & Destructor Documentation	62
5.13.2.1 LocationFeatures	63
5.13.3 Member Function Documentation	64
5.13.3.1 extractFeatures	64
5.14 Model Class Reference	64
5.14.1 Detailed Description	65
5.14.2 Member Function Documentation	65
5.14.2.1 ActivateFastComputation	65
5.14.2.2 Confusion	65
5.14.2.3 EvaluateFeatures	66
5.14.2.4 EvaluatePotentials	67
5.14.2.5 FindOptimalLambda	68
5.14.2.6 getPERFORMANCE	68
5.14.2.7 RankFeatures	69
5.14.2.8 SavePotentialMap	70
5.14.2.9 SelectFeatures	70
5.14.2.10 SetStructure	70
5.14.2.11 Solve	71
5.14.2.12 TrainPotentials	71
5.15 MSRC Class Reference	72
5.15.1 Detailed Description	73
5.15.2 Member Function Documentation	73
5.15.2.1 Label2RGB	73
5.15.2.2 RGB2Label	73
5.16 NYUv1 Class Reference	74
5.16.1 Detailed Description	75
5.16.2 Member Function Documentation	75
5.16.2.1 Label2RGB	75
5.16.2.2 RGB2Label	75
5.17 NYUv2 Class Reference	75
5.17.1 Detailed Description	76
5.17.2 Member Function Documentation	76
5.17.2.1 Label2RGB	77
5.17.2.2 RGB2Label	78
5.18 Optimization Class Reference	78
5.18.1 Detailed Description	78
5.18.2 Member Function Documentation	79

5.18.2.1	Solve	79
5.19	PedestrianParsing Class Reference	80
5.19.1	Detailed Description	81
5.19.2	Member Function Documentation	81
5.19.2.1	Label2RGB	81
5.19.2.2	RGB2Label	81
5.20	Performance Class Reference	82
5.20.1	Detailed Description	82
5.20.2	Constructor & Destructor Documentation	82
5.20.2.1	Performance	82
5.20.3	Member Function Documentation	83
5.20.3.1	operator+=	83
5.20.3.2	operator/=	83
5.20.3.3	operator=	83
5.20.3.4	set	83
5.21	Potential Class Reference	84
5.21.1	Detailed Description	86
5.21.2	Member Function Documentation	86
5.21.2.1	Evaluate	86
5.21.2.2	loadPotential	86
5.21.2.3	savePotential	87
5.21.2.4	Train	87
5.22	RandomForest Class Reference	88
5.22.1	Detailed Description	89
5.22.2	Constructor & Destructor Documentation	89
5.22.2.1	RandomForest	89
5.22.3	Member Function Documentation	90
5.22.3.1	Evaluate	90
5.22.3.2	Train	90
5.23	Sowerby Class Reference	91
5.23.1	Detailed Description	92
5.24	Test Class Reference	92
5.24.1	Detailed Description	93
5.25	TextonFeatures Class Reference	94
5.25.1	Detailed Description	95
5.25.2	Constructor & Destructor Documentation	95
5.25.2.1	TextonFeatures	95
5.25.3	Member Function Documentation	95
5.25.3.1	extractFeatures	95
5.26	Timer Class Reference	95

5.26.1 Detailed Description	96
5.26.2 Member Function Documentation	96
5.26.2.1 formatted_time	96
5.26.2.2 msg	96
5.26.2.3 num_digits	97
5.26.2.4 parse_formatted_time	97
5.26.2.5 stop	97
5.26.2.6 to_string	98
5.27 VariationalOptimization Class Reference	98
5.27.1 Detailed Description	99
5.27.2 Member Function Documentation	99
5.27.2.1 Solve	99
5.28 VOC2012 Class Reference	100
5.28.1 Detailed Description	101
5.28.2 Member Function Documentation	101
5.28.2.1 Label2RGB	102
5.28.2.2 RGB2Label	103
Index	104

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

CUDA_PRIMALDUAL_OPTIMIZATION	7
--	-------------------

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

CvRTrees	
CvRTreesMultiClass	16
Dataset	18
Corel	13
eTrims	34
MSRC	72
NYUv1	74
NYUv2	75
PedestrianParsing	80
Sowerby	91
Test	92
VOC2012	100
Feature	37
ColorFeatures	11
DepthFeatures	31
HaarLikeFeatures	44
LocationFeatures	61
TextonFeatures	94
Image	50
ImageProc	55
Learning	59
RandomForest	88
Model	64
Optimization	78
VariationalOptimization	98
Performance	82
Potential	84
DenseUnaryPixelPotential	27
Timer	95

Chapter 3

Class Index

3.1 Class List

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

ColorFeatures	
The ColorFeatures class	11
Corel	
The Corel class	13
CvRTreesMultiClass	
The CvRTreesMultiClass class This class has been taken from the following link: http://stackoverflow.com/questions/10358964/using-opencv-random-forests-is-there-an	
This function estimates the class probabilities given a feature vector. Each tree votes for one	
class label	16
Dataset	
The Dataset class	18
DenseUnaryPixelPotential	
The DenseUnaryPixelPotential class	27
DepthFeatures	
The DepthFeatures class	31
eTrims	
The eTrims class	34
Feature	
The Feature class	37
HaarLikeFeatures	
The HaarLikeFeatures class	44
Image	
The Image class	50
ImageProc	
The Image Processing class	55
Learning	
Virtual Class for Learning Algorithms	59
LocationFeatures	
The LocationFeatures class	61
Model	
The Model class	64
MSRC	
The MSRC class	72
NYUv1	
The NYUv1 class	74
NYUv2	
The NYUv2 class	75

Optimization	
The Optimization class	78
PedestrianParsing	
The PedestrianParsing class	80
Performance	
The Performance class	82
Potential	
The Potential class	84
RandomForest	
The RandomForest class	88
Sowerby	
The Sowerby class	91
Test	
The Test class	92
TextonFeatures	
The TextonFeatures class	94
Timer	
The Timer class	95
VariationalOptimization	
The VariationalOptimization class	98
VOC2012	
The VOC2012 class	100

Chapter 4

Module Documentation

4.1 CUDA_PRIMALDUAL_OPTIMIZATION

Functions

- `__global__ void kernel_binarize_u` (float *u, unsigned char *u_binary, int width, int height, int n, int pitch, int pitchUChar)
CUDA kernel function to binarize the relaxed solution u.
- void `call_binarization` (float *u, unsigned char *u_binary, int width, int height, int n)
call function to binarize the solution on CUDA
- void `__cudaSafeCall` (cudaError err, const char *file, const int line)
call function to print out CUDA errors
- `__global__ void kernel_grad_ascent` (float *u_bar, float *xi, float *psi, float *sum_u, float *g, int width, int height, int n, float lambda, int pitch)
CUDA kernel function for gradient ascent.
- `__global__ void kernel_grad_descent` (float *dataterm, float *u, float *u_bar, float *xi, float *psi, float *sum_u, int width, int height, int n, int pitch)
CUDA kernel function for gradient descent.
- void `call_segmentation` (float *dataterm, float *g, float *u, int width, int height, int n, float lambda, int &maxSteps, double &time_seg)
call function to run segmentation on gpu

4.1.1 Detailed Description

This group consists of CUDA kernel and c-call functions for primal-dual algorithm

4.1.2 Function Documentation

4.1.2.1 void `__cudaSafeCall` (cudaError err, const char * file, const int line) [inline]

call function to print out CUDA errors

Parameters

<i>err</i>	CUDA error
------------	------------

<i>file</i>	output file
<i>line</i>	output line

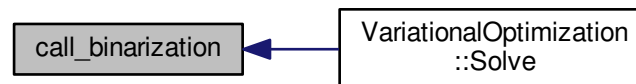
4.1.2.2 void call_binarization (float * *u*, unsigned char * *u_binary*, int *width*, int *height*, int *n*)

call function to binarize the solution on CUDA

Parameters

<i>u</i>	relaxed optimized solution
<i>u_binary</i>	binary solution
<i>width</i>	width of image
<i>height</i>	height of image
<i>n</i>	number of regions(classes)

Here is the caller graph for this function:



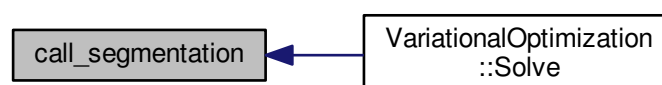
4.1.2.3 void call_segmentation (float * *dataterm*, float * *g*, float * *u*, int *width*, int *height*, int *n*, float *lambda*, int & *maxSteps*, double & *time_seg*)

call function to run segmentation on gpu

Parameters

<i>dataterm</i>	dataterm
<i>g</i>	edge detection function
<i>u</i>	region indicator variable
<i>width</i>	width of the image
<i>height</i>	height of the image
<i>n</i>	number of labels(classes)
<i>lambda</i>	weighting parameter
<i>maxSteps</i>	(maximum) number of iterations
<i>time_seg</i>	measures the runtime

Here is the caller graph for this function:



4.1.2.4 `__global__ void kernel_binarize_u (float * u, unsigned char * u_binary, int width, int height, int n, int pitch, int pitchUChar)`

CUDA kernel function to binarize the relaxed solution *u*.

Parameters

<i>u</i>	relaxed optimized solution
<i>u_binary</i>	binary solution
<i>width</i>	width of image
<i>height</i>	height of image
<i>n</i>	number of regions(classes)
<i>pitch</i>	CUDA memory management
<i>pitchUChar</i>	CUDA memory management

4.1.2.5 `__global__ void kernel_grad_ascent (float * u_bar, float * xi, float * psi, float * sum_u, float * g, int width, int height, int n, float lambda, int pitch)`

CUDA kernel function for gradient ascent.

Parameters

<i>u_bar</i>	overrelaxation of <i>u</i>
<i>xi</i>	dual variable
<i>psi</i>	Lagrange multiplier for simplex constraint
<i>sum_u</i>	variable for the simplex constraint
<i>g</i>	edge detection function
<i>width</i>	width of the image
<i>height</i>	height of the image
<i>n</i>	number of labels(classes)
<i>lambda</i>	weighting parameter
<i>pitch</i>	CUDA memory management

4.1.2.6 `__global__ void kernel_grad_descent (float * dataterm, float * u, float * u_bar, float * xi, float * psi, float * sum_u, int width, int height, int n, int pitch)`

CUDA kernel function for gradient descent.

Parameters

<i>dataterm</i>	dataterm
<i>u</i>	region indicator variable
<i>u_bar</i>	overrelaxation of <i>u</i>
<i>xi</i>	dual variable
<i>psi</i>	Lagrange multiplier for simplex constraint
<i>sum_u</i>	variable for the simplex constraint
<i>width</i>	width of the image
<i>height</i>	height of the image
<i>n</i>	number of labels(classes)
<i>pitch</i>	CUDA memory management

Chapter 5

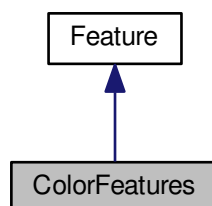
Class Documentation

5.1 ColorFeatures Class Reference

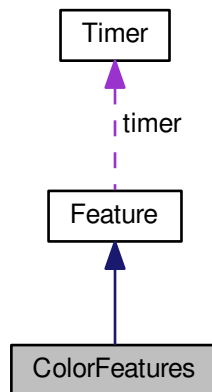
The [ColorFeatures](#) class.

```
#include <Feature.hpp>
```

Inheritance diagram for ColorFeatures:



Collaboration diagram for ColorFeatures:



Public Member Functions

- [ColorFeatures](#) (vector< Size > &all_patches, string colorFolder, string ext, int numSubSample)
Class Constructor.
- void [extractFeatures](#) (Image *im, Mat &features)
Extract features.

Public Attributes

- vector< Point2d > [rndLocs](#)
relative pixel positions
- vector< Point2d > [rndRelatives](#)
relative pixel positions
- vector< Rect > [rndPatches](#)
relative patch positions

Additional Inherited Members

5.1.1 Detailed Description

The [ColorFeatures](#) class.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 ColorFeatures::ColorFeatures (vector< Size > &all_patches, string colorFolder, string ext, int numSubSample)

Class Constructor.

Parameters

<i>all_patches</i>	set of all patches
<i>colorFolder</i>	folder to store features
<i>ext</i>	extension of feature files
<i>numSubSample</i>	sub sample size

5.1.3 Member Function Documentation

5.1.3.1 void ColorFeatures::extractFeatures (Image * im, Mat & features) [virtual]

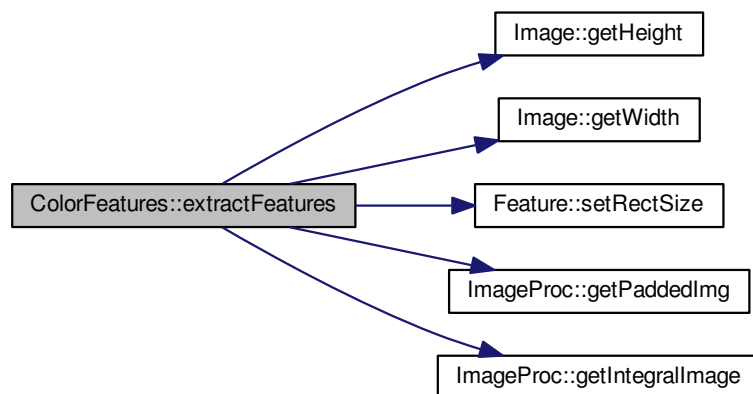
Extract features.

Parameters

<i>im</i>	image
<i>features</i>	feature matrix

Implements [Feature](#).

Here is the call graph for this function:



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

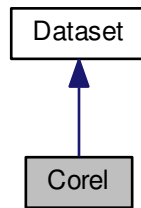
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Feature.hpp`
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Feature.cpp`

5.2 Corel Class Reference

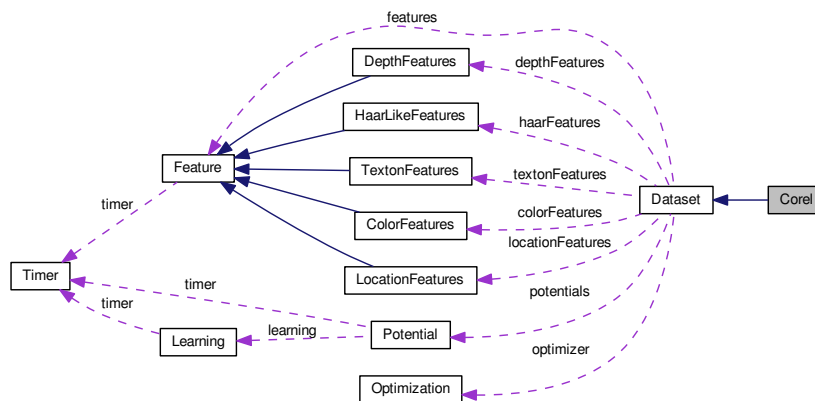
The [Corel](#) class.

```
#include <Dataset.hpp>
```

Inheritance diagram for Corel:



Collaboration diagram for Corel:



Public Member Functions

- [Corel](#) ()
Class Constructor.
- void [RGB2Label](#) (Mat rgb, Mat &labels)
Convert from RGB to Label.
- void [Label2RGB](#) (Mat labels, Mat &rgb)
Convert from label to RGB.

Additional Inherited Members

5.2.1 Detailed Description

The [Corel](#) class.

5.2.2 Member Function Documentation

5.2.2.1 void Corel::Label2RGB (Mat *labels*, Mat & *rgb*) [virtual]

Convert from label to RGB.

Parameters

<i>labels</i>	label matrix
<i>rgb</i>	RGB image

Reimplemented from [Dataset](#).

5.2.2.2 void Core::RGB2Label (Mat *rgb*, Mat & *labels*) [virtual]

Convert from RGB to Label.

Parameters

<i>rgb</i>	RGB image
<i>labels</i>	label matrix to store labels for each pixel

Reimplemented from [Dataset](#).

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

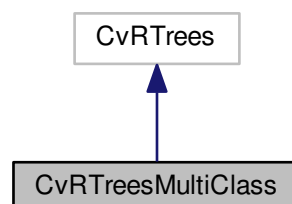
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Dataset.hpp
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Dataset.cpp

5.3 CvRTreesMultiClass Class Reference

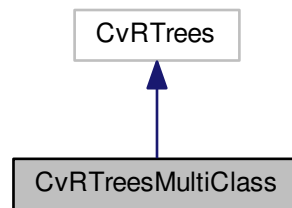
The [CvRTreesMultiClass](#) class This class has been taken from the following link: <http://stackoverflow.com/questions/10358964/using-opencv-random-forests-is-there-any-way-to-obtain-the-confidence>. This function estimates the class probabilities given a feature vector. Each tree votes for one class label.

```
#include <Learning.hpp>
```

Inheritance diagram for CvRTreesMultiClass:



Collaboration diagram for CvRTreesMultiClass:



Public Member Functions

- int [predict_multi_class](#) (const CvMat *sample, int out_votes[], const CvMat *missing=0) const
Predict unnormalized class probability for one test sample.

5.3.1 Detailed Description

The [CvRTreesMultiClass](#) class This class has been taken from the following link: <http://stackoverflow.com/questions/10358964/using-opencv-random-forests-is-there-any-way-to-obtain-the-confidence>
 This function estimates the class probabilities given a feature vector. Each tree votes for one class label.

5.3.2 Member Function Documentation

5.3.2.1 int CvRTreesMultiClass::predict_multi_class (const CvMat * *sample*, int *out_votes*[], const CvMat * *missing* = 0)
 const [inline]

Predict unnormalized class probability for one test sample.

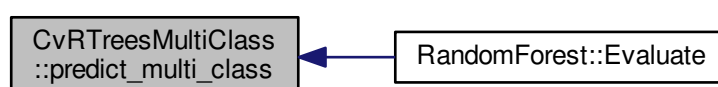
Parameters

<i>sample</i>	Test sample
<i>out_votes</i>	Unnormalized class probabilities for a given test sample
<i>missing</i>	Optional missing measurement mask of the sample (OpenCV doc)

Returns

Total number of trees in the forest

Here is the caller graph for this function:



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

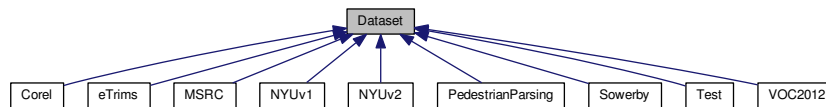
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Learning.hpp

5.4 Dataset Class Reference

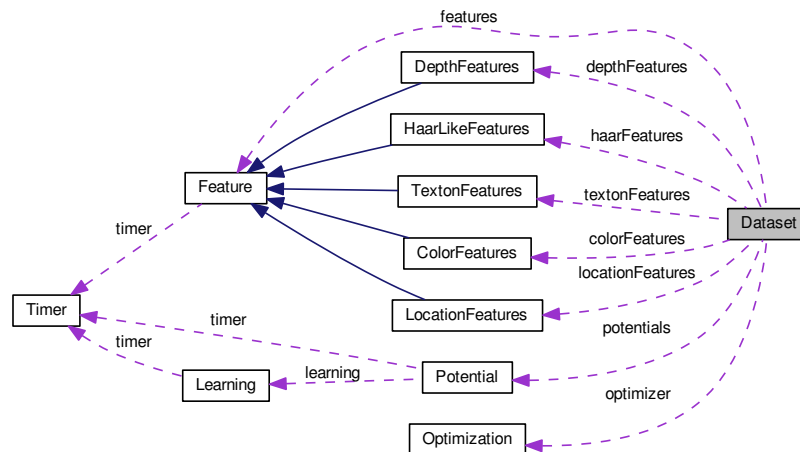
The [Dataset](#) class.

```
#include <Dataset.hpp>
```

Inheritance diagram for Dataset:



Collaboration diagram for Dataset:



Public Member Functions

- [Dataset](#) ()
Class Constructor.
- virtual [~Dataset](#) ()
Class Deconstructor.
- virtual void [RGB2Label](#) (Mat rgb, Mat &labels)
Convert from RGB to Label.
- virtual void [Label2RGB](#) (Mat labels, Mat &rgb)
Convert from label to RGB.
- virtual void [getDatasetImages](#) (int trainCount, int testCount)
Split Dataset.

- virtual void [readImgsInDir](#) (vector< string > &imgs, string folder, string ext)
Read [Dataset](#).
- virtual bool [readSplitsToVectors](#) ()
Read pre-split train/validation/test images.
- virtual void [writeSplitsToFile](#) ()
Write train/validation/test splits to files.
- virtual int [createDir](#) (string dirName)
Create directory.
- virtual int [clearDir](#) (string dirName)
Remove all file in directory.
- virtual int [findImage](#) (string name)
Find image index in testImgs.
- virtual void [createResultFolders](#) ()
Create Folders for Results.
- virtual int [getFullFeatureDim](#) ()
Get size of full feature set.
- virtual void [removeUnlabeledImages](#) ()
Remove Unlabeled Images from [Dataset](#).

Public Attributes

- vector< string > [allImages](#)
list of all images in dataset
- vector< string > [trainImgs](#)
list of training images
- vector< string > [testImgs](#)
list of test images
- vector< string > [validationImgs](#)
list of validation images
- vector< int > [selectedFeatures](#)
vector stores indices of selected features
- bool [isSplitDataset](#)
flag to split dataset randomly
- bool [savePotentials](#)
flag to save potentials
- bool [saveDetections](#)
flag to save detection results (argmax of potential for each pixel on image)
- bool [showEntropy](#)
flag to visualize entropy map of the image
- int [trainCount](#)
size of the training set
- int [testCount](#)
size of the test test
- double [trainEntropy](#)
entropy of the training data (balance of the classes in training set)
- double [testEntropy](#)
entropy of the test set (how certain the detection is)
- string [name_](#)
name of the dataset
- string [resultDir](#)

- folder to store the segmentation outputs*
- string [trainFolder](#)
 - folder to store classifiers*
- string [resultFolder](#)
 - folder to store results. (e.g. mainFolder + '/Result')*
- string [mainFolder](#)
 - main folder of dataset*
- string [imageFolder](#)
 - folder where RGB image files are*
- string [imageFileExt](#)
 - extension of RGB image files*
- string [grFolder](#)
 - folder where ground truth files are*
- string [grFileExt](#)
 - extension of ground truth files*
- string [depthFolder](#)
 - folder to store depth features*
- string [depthFileExt](#)
 - extension of depth feature files*
- string [haarFolder](#)
 - folder to store haar-like features*
- string [haarFeatureExt](#)
 - extension of haar-like feature files*
- string [colorFolder](#)
 - folder to store color features*
- string [colorFeatureExt](#)
 - extension of color feature files*
- string [locationFolder](#)
 - folder to store location features*
- string [locationFeatureExt](#)
 - extension of location feature files*
- string [textonFolder](#)
 - folder to store texton features*
- string [textonFeatureExt](#)
 - extension of texton feature files*
- string [depthFeatureFolder](#)
 - folder to store depth features*
- string [depthFeatureExt](#)
 - extension of depth feature files*
- string [potentialFolder](#)
 - folder to stor potentials*
- string [potentialExt](#)
 - extension of potential files*
- string [detectionFolder](#)
 - folder to store detection results*
- string [pMapFolder](#)
 - folder to store potential maps*
- int [numFeatureDims](#)
 - total size of current feature set*
- int [fcount_](#)
 - Number of features in the current(selected) feature set.*

- int `n`
maximum number of regions (== numClasses)
- int `maxSteps`
maximum iteration number for [VariationalOptimization](#)
- float `lambda`
smoothing parameter lambda for [VariationalOptimization](#)
- int `subSample`
sub sample size
- double `textonBandWidth`
bandwidth of kernels for [TextonFeautures](#)
- unsigned int `numClasses`
total number of classes in [Dataset](#)
- [HaarLikeFeatures](#) * `haarFeatures`
instance of [HaarLikeFeatures](#)
- [ColorFeatures](#) * `colorFeatures`
instance of [ColorFeatures](#)
- [LocationFeatures](#) * `locationFeatures`
instance of [LocationFeatures](#)
- [TextonFeatures](#) * `textonFeatures`
instance of [TextonFeatures](#)
- [DepthFeatures](#) * `depthFeatures`
instance of [DepthFeatures](#)
- const float * `class_weights`
Weights of each class to balance the training error.
- CvRTPParams `RFPParams`
Structure to store parameteres for Random Forest.
- CvTermCriteria `RFterm_crit`
termination criteria for Random Forests training
- int `RFmax_num_of_trees_in_the_forest`
maximum number of trees in the forest
- float `RFforest_accuracy`
sufficient accuracy, OOB error ([OpenCV docs](#))
- int `totalFeatureType`
total number of feature types
- int `numPotentials`
number of potentaills
- [Feature](#) ** `features`
pointer to the list of features
- [Potential](#) ** `potentials`
pointer to the list of potentials
- [Optimization](#) * `optimizer`
pointer to the optimizer

Protected Attributes

- int `index`
variable to store index

5.4.1 Detailed Description

The [Dataset](#) class.

5.4.2 Member Function Documentation

5.4.2.1 `int Dataset::clearDir (string dirName) [virtual]`

Remove all file in directory.

Parameters

<i>dirName</i>	directory to be cleared
----------------	-------------------------

Returns

1 if all files succesfully removed

5.4.2.2 `int Dataset::createDir (string dirName) [virtual]`

Create directory.

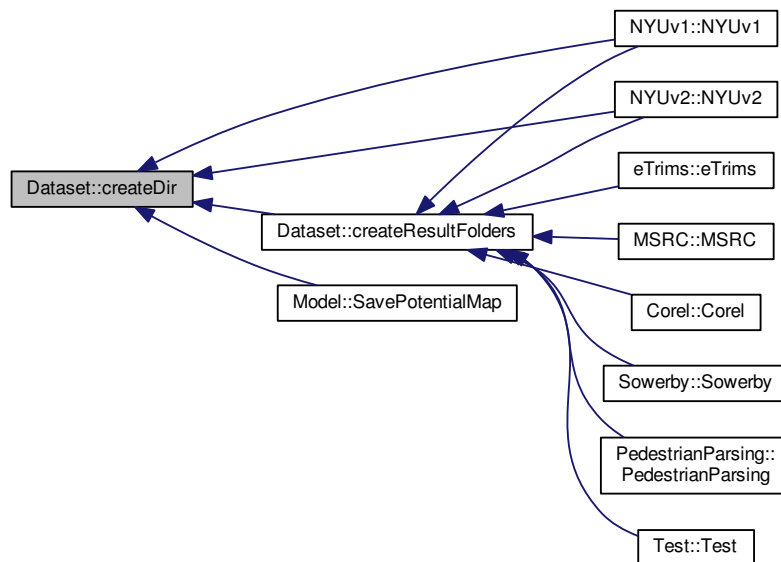
Parameters

<i>dirName</i>	name of directory
----------------	-------------------

Returns

1 if directory exist

Here is the caller graph for this function:



5.4.2.3 `int Dataset::findImage (string name) [virtual]`

Find image index in testImgs.

Parameters

<i>name</i>	name of the image
-------------	-------------------

Returns

index of the image in testImgs

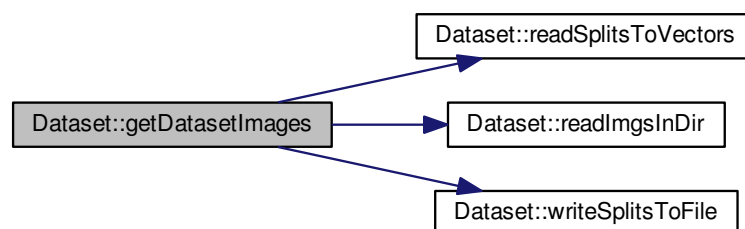
5.4.2.4 void Dataset::getDatasetImages (int *trainCount*, int *testCount*) [virtual]

Split [Dataset](#).

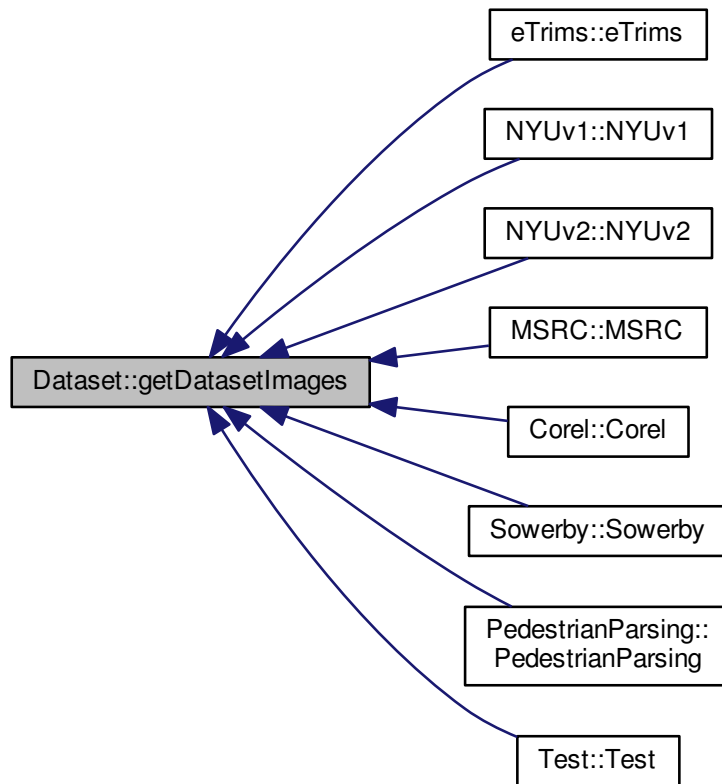
Parameters

<i>trainCount</i>	number of training samples, either a number or a proportion
<i>testCount</i>	number of test samples, either a number or a proportion

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.5 `int Dataset::getFullFeatureDim () [virtual]`

Get size of full feature set.

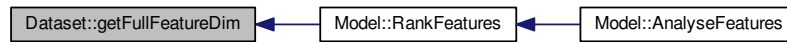
Returns

size of full feature set

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.6 void Dataset::Label2RGB (Mat *labels*, Mat & *rgb*) [virtual]

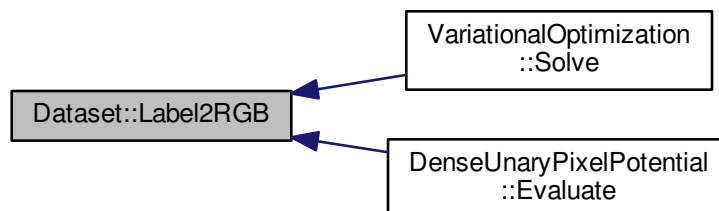
Convert from label to RGB.

Parameters

<i>labels</i>	label matrix
<i>rgb</i>	RGB image

Reimplemented in [PedestrianParsing](#), [VOC2012](#), [NYUv2](#), [NYUv1](#), [Corel](#), [MSRC](#), and [eTrims](#).

Here is the caller graph for this function:



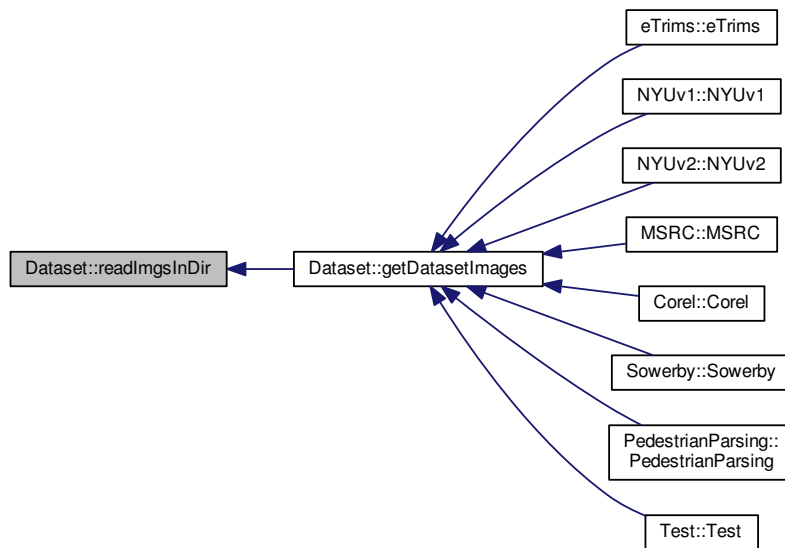
5.4.2.7 void Dataset::readImgsInDir (vector< string > & *imgs*, string *folder*, string *ext*) [virtual]

Read [Dataset](#).

Parameters

<i>imgs</i>	vector of image names
<i>folder</i>	folder to read images from
<i>ext</i>	extension of image files

Here is the caller graph for this function:



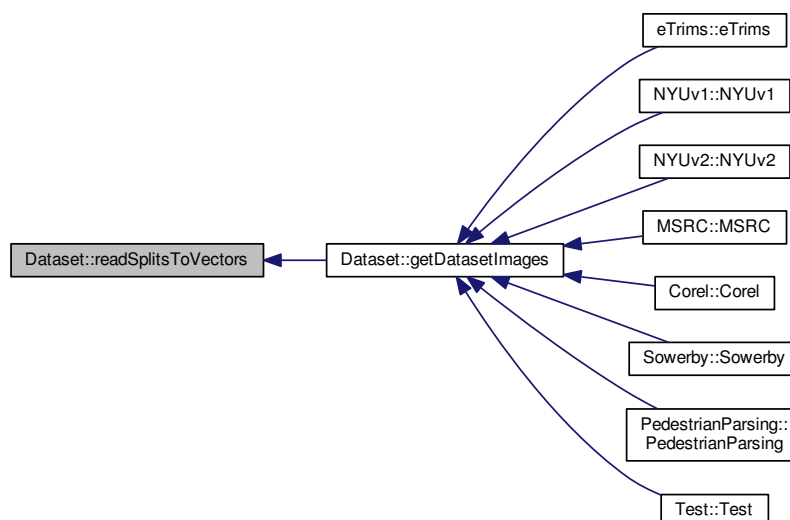
5.4.2.8 `bool Dataset::readSplitsToVectors () [virtual]`

Read pre-splitted train/validation/test images.

Returns

true if split files exist

Here is the caller graph for this function:



5.4.2.9 void Dataset::RGB2Label (Mat *rgb*, Mat & *labels*) [virtual]

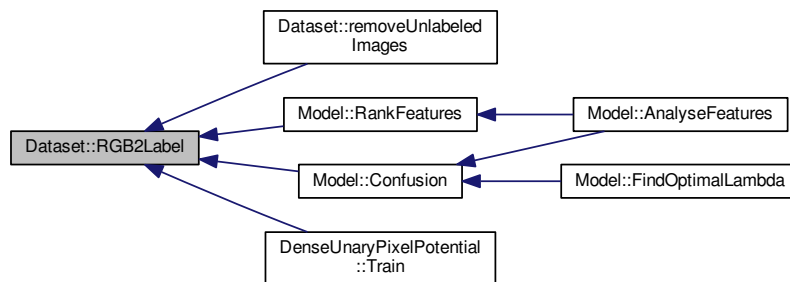
Convert from RGB to Label.

Parameters

<i>rgb</i>	RGB image
<i>labels</i>	label matrix to store labels for each pixel

Reimplemented in [PedestrianParsing](#), [VOC2012](#), [NYUv2](#), [NYUv1](#), [Corel](#), [MSRC](#), and [eTrims](#).

Here is the caller graph for this function:



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

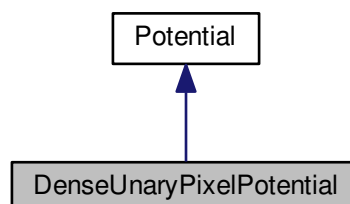
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Dataset.hpp`
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Dataset.cpp`

5.5 DenseUnaryPixelPotential Class Reference

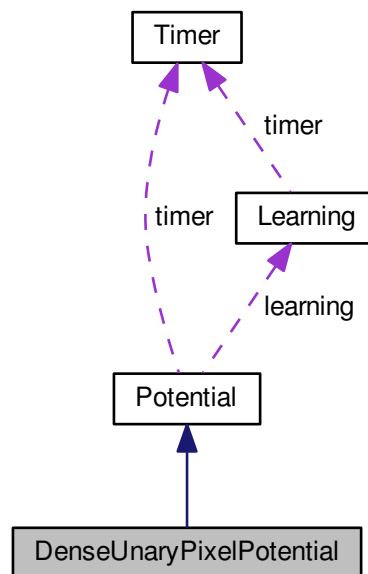
The [DenseUnaryPixelPotential](#) class.

```
#include <Potential.hpp>
```

Inheritance diagram for DenseUnaryPixelPotential:



Collaboration diagram for DenseUnaryPixelPotential:



Public Member Functions

- [DenseUnaryPixelPotential](#) (string dir, string fileExt)
Class Constructor.
- int [Train](#) ([Dataset](#) *dataset, vector< string > &imageList, int from, int to, bool FAST_COMPUTATION=false)
Train potential.
- int [Evaluate](#) ([Dataset](#) *dataset, vector< string > &imageList, int from, int to, bool FAST_COMPUTATION=false)
Evaluate potentials.

Additional Inherited Members

5.5.1 Detailed Description

The [DenseUnaryPixelPotential](#) class.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 DenseUnaryPixelPotential::DenseUnaryPixelPotential (string dir, string fileExt)

Class Constructor.

Parameters

<i>dir</i>	Output directory
<i>fileExt</i>	file extension

5.5.3 Member Function Documentation

5.5.3.1 `int DenseUnaryPixelPotential::Evaluate (Dataset * dataset, vector< string > & imageList, int from, int to, bool FAST_COMPUTATION = false) [virtual]`

Evaluate potentials.

Parameters

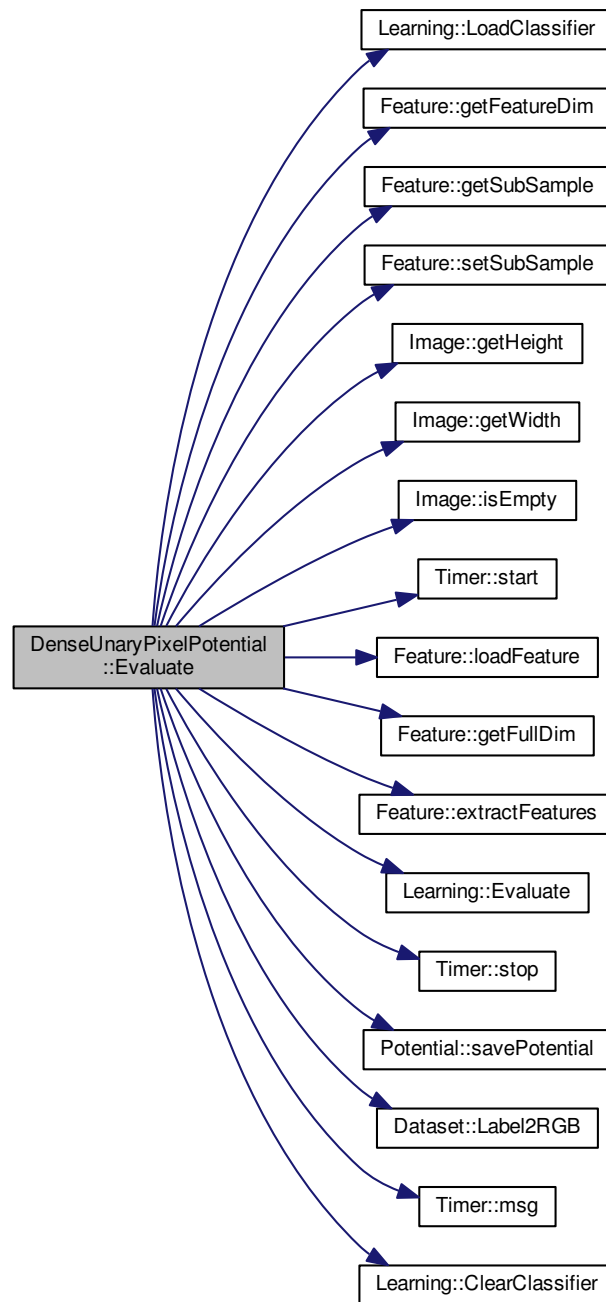
<i>dataset</i>	pointer to the current dataset
<i>imageList</i>	list of images
<i>from</i>	start index
<i>to</i>	stop index, set to -1 to process all images startin from "from"
<i>FAST_COMPUTATION</i>	flag to activate fast computation

Returns

estimated time per image in milliseconds

Implements [Potential](#).

Here is the call graph for this function:



```

5.5.3.2 int DenseUnaryPixelPotential::Train ( Dataset * dataset, vector< string > & imageList, int from, int to, bool
FAST_COMPUTATION = false ) [virtual]

```

Train potential.

Parameters

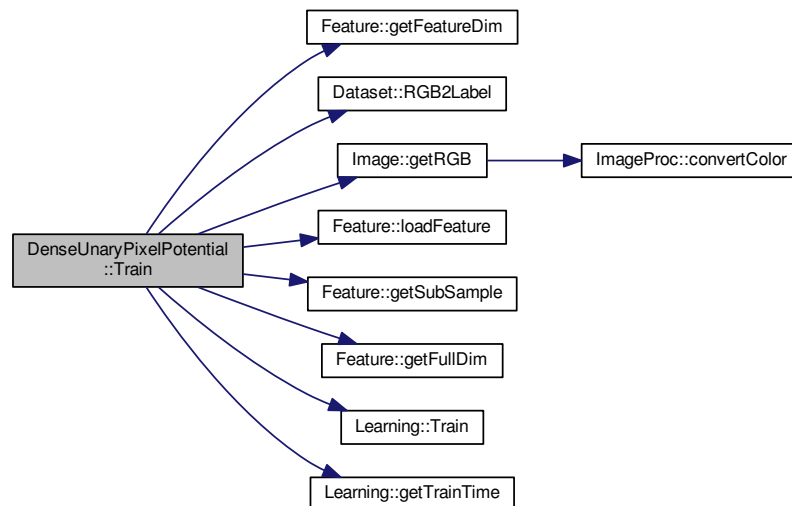
<i>dataset</i>	pointer to the current dataset
<i>imageList</i>	list of images
<i>from</i>	start index
<i>to</i>	stop index, set to -1 to process all images startin from "from"
<i>FAST_COMPUTATION</i>	flag to activate fast computation

Returns

estimated training time in miliseconds

Implements [Potential](#).

Here is the call graph for this function:



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

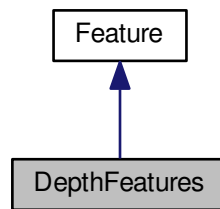
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Potential.hpp`
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Potential.cpp`

5.6 DepthFeatures Class Reference

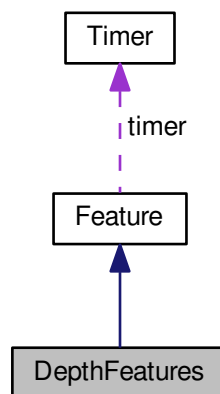
The [DepthFeatures](#) class.

```
#include <Feature.hpp>
```

Inheritance diagram for DepthFeatures:



Collaboration diagram for DepthFeatures:



Public Member Functions

- `DepthFeatures` (`vector< Size > &all_patches`, `string depthFolder`, `string ext`, `int numSubSample`)
Class Constructor.
- `void extractFeatures (Image *im, Mat &features)`
Extract features.

Additional Inherited Members

5.6.1 Detailed Description

The `DepthFeatures` class.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 DepthFeatures::DepthFeatures (vector< Size > & *all_patches*, string *depthFolder*, string *ext*, int *numSubSample*)

Class Constructor.

Parameters

<i>all_patches</i>	set of all patches
<i>depthFolder</i>	folder to store features
<i>ext</i>	extension of feature files
<i>numSubSample</i>	sub sample size

5.6.3 Member Function Documentation

5.6.3.1 void DepthFeatures::extractFeatures (Image * *im*, Mat & *features*) [virtual]

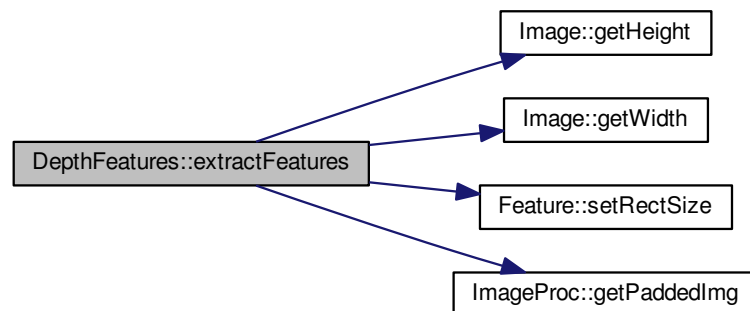
Extract features.

Parameters

<i>im</i>	image
<i>features</i>	feature matrix

Implements [Feature](#).

Here is the call graph for this function:



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

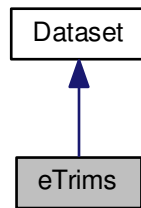
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Feature.hpp
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Feature.cpp

5.7 eTrims Class Reference

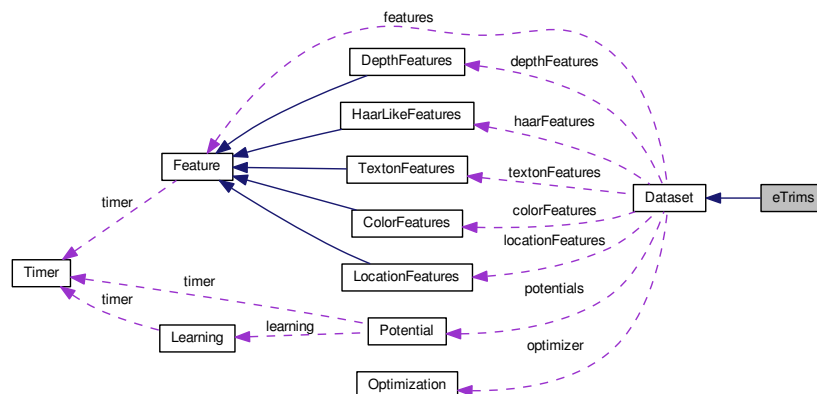
The [eTrims](#) class.

```
#include <Dataset.hpp>
```

Inheritance diagram for eTrims:



Collaboration diagram for eTrims:



Public Member Functions

- `eTrims` ()
eTrims
- void `RGB2Label` (Mat rgb, Mat &labels)
Convert from RGB to Label.
- void `Label2RGB` (Mat labels, Mat &rgb)
Convert from label to RGB.

Additional Inherited Members

5.7.1 Detailed Description

The `eTrims` class.

5.7.2 Member Function Documentation

5.7.2.1 void eTrims::Label2RGB (Mat *labels*, Mat & *rgb*) [virtual]

Convert from label to RGB.

Parameters

<i>labels</i>	label matrix
<i>rgb</i>	RGB image

Reimplemented from [Dataset](#).

5.7.2.2 void eTrims::RGB2Label (Mat *rgb*, Mat & *labels*) [virtual]

Convert from RGB to Label.

Parameters

<i>rgb</i>	RGB image
<i>labels</i>	label matrix to store labels for each pixel

Reimplemented from [Dataset](#).

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

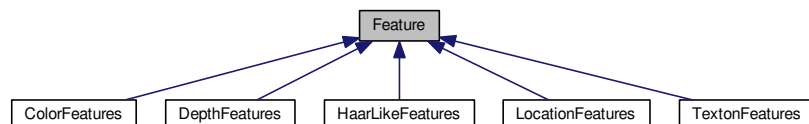
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Dataset.hpp
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Dataset.cpp

5.8 Feature Class Reference

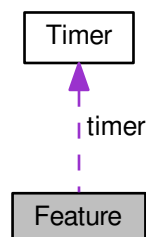
The [Feature](#) class.

```
#include <Feature.hpp>
```

Inheritance diagram for Feature:



Collaboration diagram for Feature:



Public Member Functions

- [Feature](#) ()
Class Constructor.
- virtual [~Feature](#) ()
Class Deconstructor.
- void [saveFeature](#) (string imName, Mat &features)
Save features.
- void [loadFeature](#) (string imName, Mat &features)
Load feature.
- virtual void [extractFeatures](#) (Image *im, Mat &features)=0
Extract features.
- void [setSubSample](#) (int ss)
Set sub sample size.
- int [getSubSample](#) ()
Get sub sample size.
- int [getFeatureDim](#) ()
Get size of feature set.
- int [getFullDim](#) ()
Get size of full feature set.
- void [setFeatureDim](#) (int dim)
Set size of feature set.

Public Attributes

- string [name_](#)
name of the feature
- Mat [isComputeFeature](#)
matrix to store feature indices indicate which features to be computed

Protected Member Functions

- void [setRectSize](#) (int height, int width)
Set rectangle size.

Protected Attributes

- FILE * [foperator](#)
File operator.
- string [ftFolder](#)
folder to store feature
- string [ext_](#)
feature file extension
- int [subSample](#)
size of sub sample
- int [featureSize](#)
size of feature
- int [featureDim](#)
size of selected feature set
- int [rHeight](#)

- patch height*
- int [rWidth](#)
patch width
- int [halfHeight](#)
half size of patch height
- int [halfWidth](#)
half size of patch width
- int [sixthOfHeight](#)
sixth of patch height
- int [sixthOfWidth](#)
sixth of patch width
- int [fourthOfHeight](#)
fourth of patch height
- int [fourthOfWidth](#)
fourth of patch width
- float [weight0](#)
weight of black region (Haar-like features)
- float [weight1](#)
weight of white region (Haar-like features)
- vector< Size > [allPatches](#)
vector of all patches
- [Timer](#) [timer](#)
timer to estimate the time

5.8.1 Detailed Description

The [Feature](#) class.

5.8.2 Member Function Documentation

5.8.2.1 virtual void Feature::extractFeatures ([Image](#) * *im*, [Mat](#) & *features*) [pure virtual]

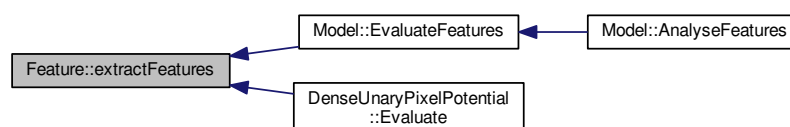
Extract features.

Parameters

<i>im</i>	image
<i>features</i>	feature matrix

Implemented in [TextonFeatures](#), [DepthFeatures](#), [LocationFeatures](#), [ColorFeatures](#), and [HaarLikeFeatures](#).

Here is the caller graph for this function:



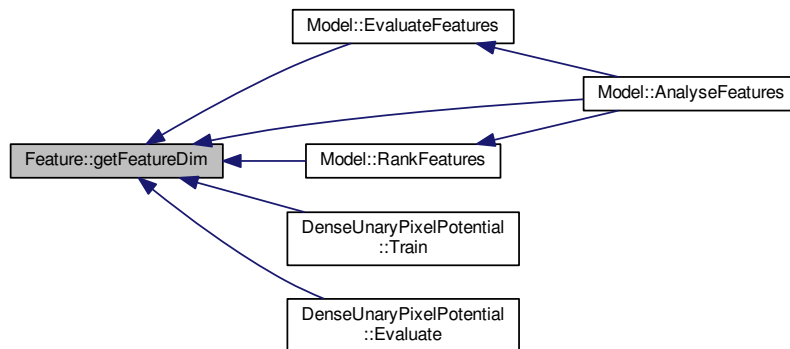
5.8.2.2 int Feature::getFeatureDim ()

Get size of feature set.

Returns

size of feature set

Here is the caller graph for this function:



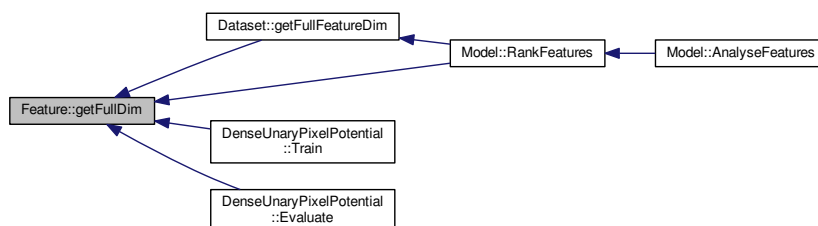
5.8.2.3 int Feature::getFullDim ()

Get size of full feature set.

Returns

size of full feature set

Here is the caller graph for this function:



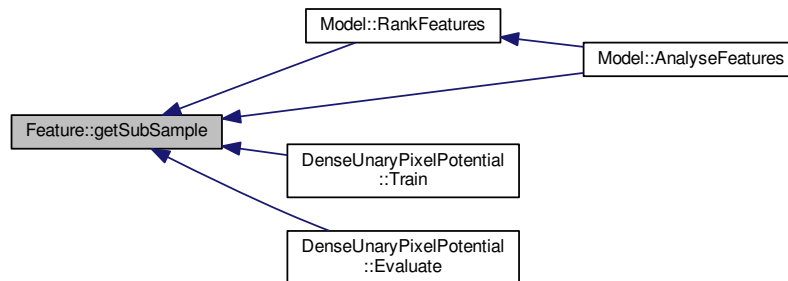
5.8.2.4 int Feature::getSubSample ()

Get sub sample size.

Returns

size of sub sample

Here is the caller graph for this function:



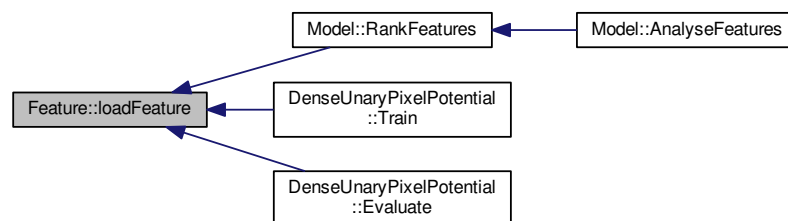
5.8.2.5 void Feature::loadFeature (string *imName*, Mat & *features*)

Load feature.

Parameters

<i>imName</i>	image name
<i>features</i>	feature matrix

Here is the caller graph for this function:



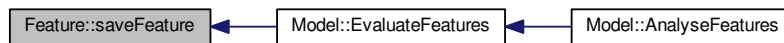
5.8.2.6 void Feature::saveFeature (string *imName*, Mat & *features*)

Save features.

Parameters

<i>imName</i>	image name
<i>features</i>	feature matrix

Here is the caller graph for this function:



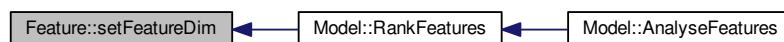
5.8.2.7 void Feature::setFeatureDim (int *dim*)

Set size of feature set.

Parameters

<i>dim</i>	size
------------	------

Here is the caller graph for this function:



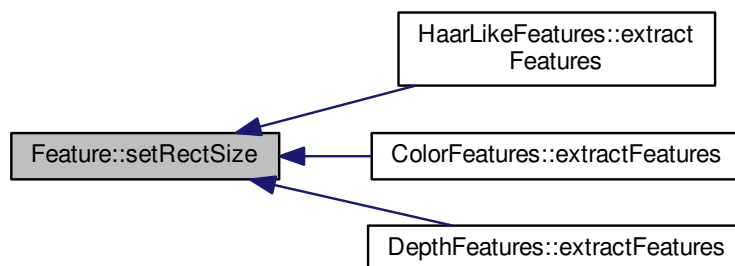
5.8.2.8 void Feature::setRectSize (int *height*, int *width*) [protected]

Set rectangle size.

Parameters

<i>height</i>	height of new rectangle
<i>width</i>	width of new rectangle

Here is the caller graph for this function:



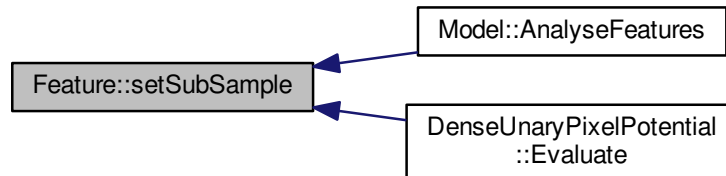
5.8.2.9 void Feature::setSubSample (int ss)

Set sub sample size.

Parameters

<code>ss</code>	sub sample size
-----------------	-----------------

Here is the caller graph for this function:



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

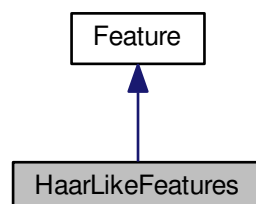
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Feature.hpp`
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Feature.cpp`

5.9 HaarLikeFeatures Class Reference

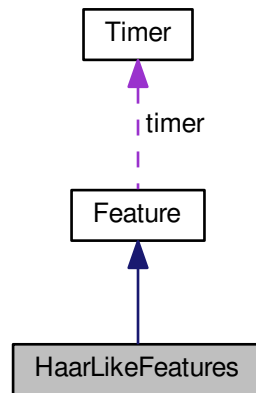
The [HaarLikeFeatures](#) class.

```
#include <Feature.hpp>
```

Inheritance diagram for HaarLikeFeatures:



Collaboration diagram for HaarLikeFeatures:



Public Member Functions

- [HaarLikeFeatures](#) (vector< Size > &all_patches, string haarFolder, string ext, int numSubSample)
Class Constructor.
- void [extractFeatures](#) (Image *im, Mat &features)
Extract features.

Protected Member Functions

- void [horizontalEdge](#) (Mat &im, int r, int c, float &fValue)
Horizontal Edge [Feature](#).
- void [verticalEdge](#) (Mat &im, int r, int c, float &fValue)
Vertical Edge [Feature](#).
- void [horizontalLine](#) (Mat &im, int r, int c, float &fValue)
Horizontal Line [Feature](#).
- void [verticalLine](#) (Mat &im, int r, int c, float &fValue)
Vertical Edge [Feature](#).
- void [centerSurround](#) (Mat &im, int r, int c, float &fValue)
Center Surround [Feature](#).
- void [fourSquare](#) (Mat &im, int r, int c, float &fValue)
Four Square [Feature](#).

Additional Inherited Members

5.9.1 Detailed Description

The [HaarLikeFeatures](#) class.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 HaarLikeFeatures::HaarLikeFeatures (vector< Size > & *all_patches*, string *haarFolder*, string *ext*, int *numSubSample*)

Class Constructor.

Parameters

<i>all_patches</i>	set of all patches
<i>haarFolder</i>	folder to store features
<i>ext</i>	extension of feature files
<i>numSubSample</i>	sub sample size

5.9.3 Member Function Documentation

5.9.3.1 void HaarLikeFeatures::centerSurround (Mat & *im*, int *r*, int *c*, float & *fValue*) [protected]

Center Surround [Feature](#).

Parameters

<i>im</i>	image
<i>r</i>	row of pixel at which the feature will be computed
<i>c</i>	column of pixel at which the feature will be computed
<i>fValue</i>	feature value

Here is the caller graph for this function:



5.9.3.2 void HaarLikeFeatures::extractFeatures (Image * *im*, Mat & *features*) [virtual]

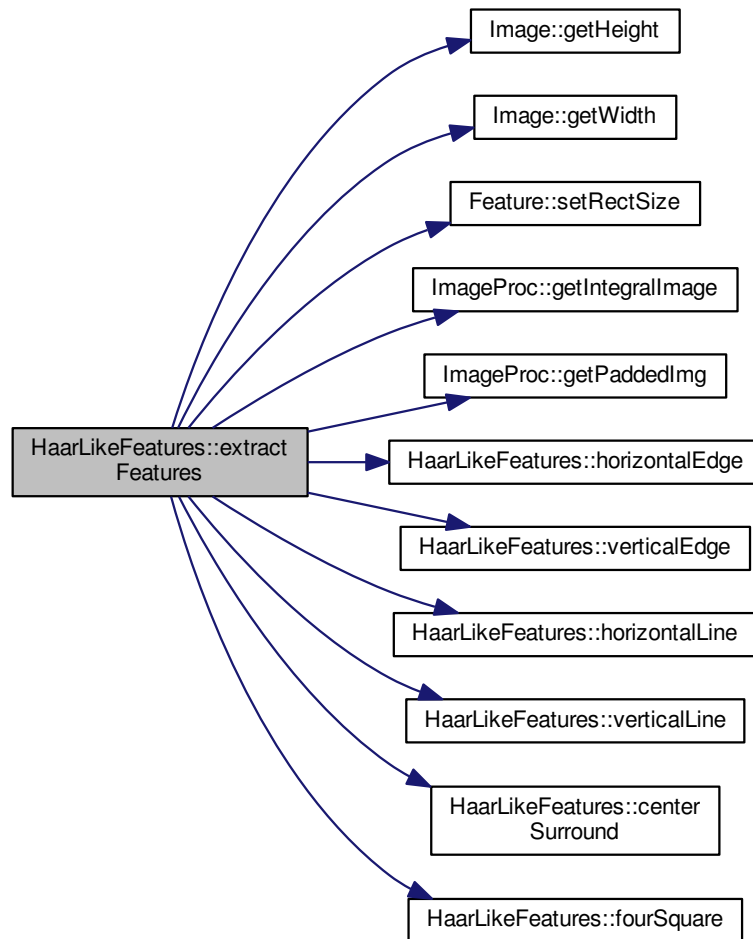
Extract features.

Parameters

<i>im</i>	image
<i>features</i>	feature matrix

Implements [Feature](#).

Here is the call graph for this function:



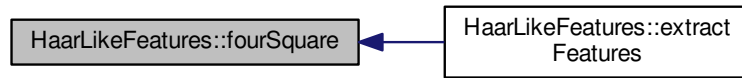
5.9.3.3 `void HaarLikeFeatures::fourSquare (Mat & im, int r, int c, float & fValue)` [protected]

Four Square [Feature](#).

Parameters

<i>im</i>	image
<i>r</i>	row of pixel at which the feature will be computed
<i>c</i>	column of pixel at which the feature will be computed
<i>fValue</i>	feature value

Here is the caller graph for this function:



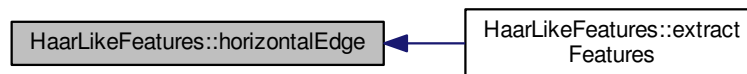
5.9.3.4 `void HaarLikeFeatures::horizontalEdge (Mat & im, int r, int c, float & fValue)` `[protected]`

Horizontal Edge [Feature](#).

Parameters

<i>im</i>	image
<i>r</i>	row of pixel at which the feature will be computed
<i>c</i>	column of pixel at which the feature will be computed
<i>fValue</i>	feature value

Here is the caller graph for this function:



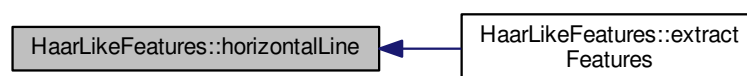
5.9.3.5 `void HaarLikeFeatures::horizontalLine (Mat & im, int r, int c, float & fValue)` `[protected]`

Horizontal Line [Feature](#).

Parameters

<i>im</i>	image
<i>r</i>	row of pixel at which the feature will be computed
<i>c</i>	column of pixel at which the feature will be computed
<i>fValue</i>	feature value

Here is the caller graph for this function:



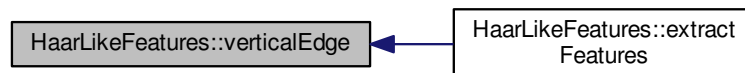
5.9.3.6 void HaarLikeFeatures::verticalEdge (Mat & *im*, int *r*, int *c*, float & *fValue*) [protected]

Vertical Edge [Feature](#).

Parameters

<i>im</i>	image
<i>r</i>	row of pixel at which the feature will be computed
<i>c</i>	column of pixel at which the feature will be computed
<i>fValue</i>	feature value

Here is the caller graph for this function:



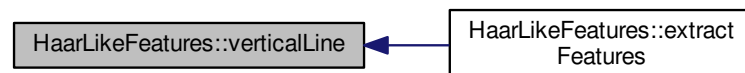
5.9.3.7 void HaarLikeFeatures::verticalLine (Mat & *im*, int *r*, int *c*, float & *fValue*) [protected]

Vertical Edge [Feature](#).

Parameters

<i>im</i>	image
<i>r</i>	row of pixel at which the feature will be computed
<i>c</i>	column of pixel at which the feature will be computed
<i>fValue</i>	feature value

Here is the caller graph for this function:



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

- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Feature.hpp
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Feature.cpp

5.10 Image Class Reference

The [Image](#) class.

```
#include <Image.hpp>
```

Public Member Functions

- [Image](#) (string *impath*, string *depthpath*="")
Class Constructor.
- [~Image](#) ()
Class DeConstructor.
- int [getWidth](#) ()
get image width
- int [getHeight](#) ()
get image height
- Mat [getImage](#) ()
get image
- Mat [getRGB](#) ()
get RGB image
- bool [isEmpty](#) ()
is image empty ?

Public Attributes

- string [imName](#)
image path
- Mat [BGRImage_](#)
BGR image.
- Mat [LABImage_](#)
Lab image.
- Mat [L](#)
L channel.
- Mat [L8U](#)
L channel, unsigned 8 bit.
- Mat [a](#)
a channel
- Mat [b](#)
b channel
- Mat [depth](#)
depth image

5.10.1 Detailed Description

The [Image](#) class.

5.10.2 Constructor & Destructor Documentation

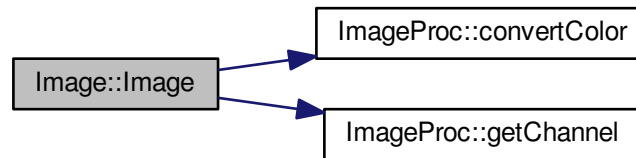
5.10.2.1 [Image::Image](#) (string *impath*, string *depthpath* = " ")

Class Constructor.

Parameters

<i>impath</i>	Path to the image
<i>depthpath</i>	Path to the depth image

Here is the call graph for this function:



5.10.3 Member Function Documentation

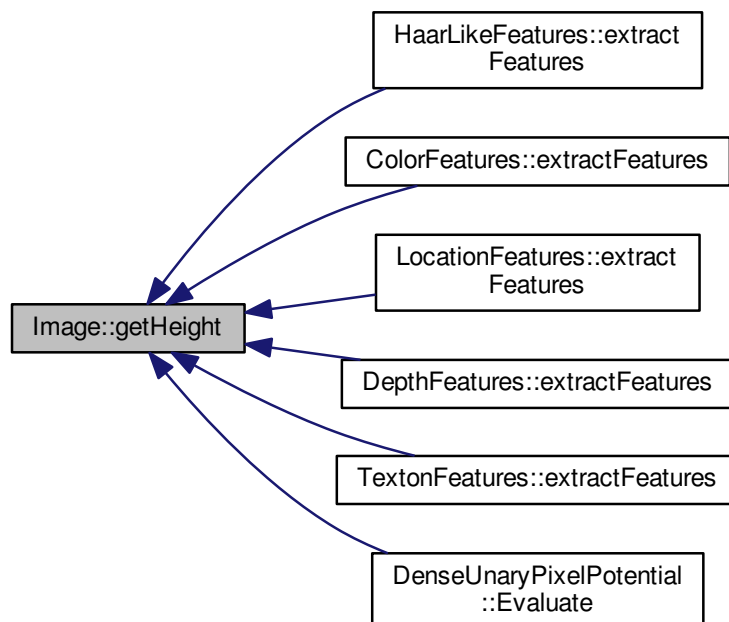
5.10.3.1 `int Image::getHeight ()`

get image height

Returns

image height

Here is the caller graph for this function:



5.10.3.2 Mat Image::getImage ()

get image

Returns

image

5.10.3.3 Mat Image::getRGB ()

get RGB image

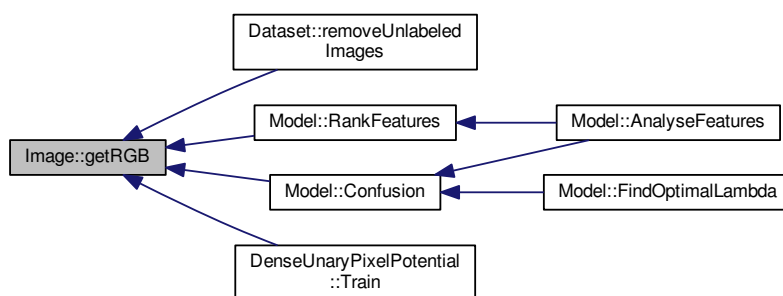
Returns

RGB image

Here is the call graph for this function:



Here is the caller graph for this function:



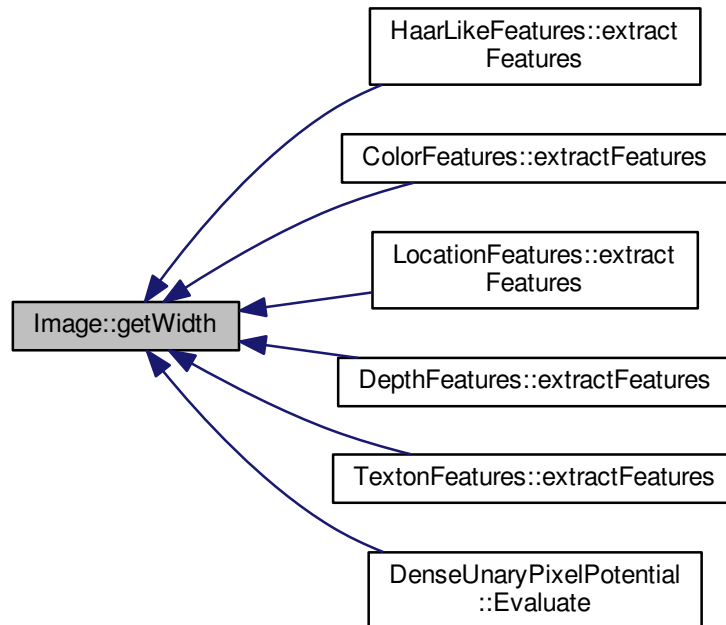
5.10.3.4 int Image::getWidth ()

get image width

Returns

image width

Here is the caller graph for this function:



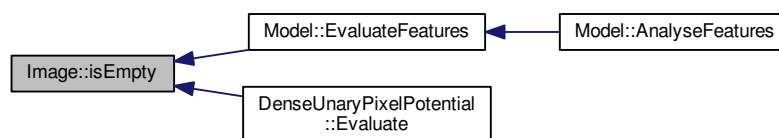
5.10.3.5 `bool Image::isEmpty ()`

is image empty ?

Returns

true or false

Here is the caller graph for this function:



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

- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Image.hpp`
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Image.cpp`

5.11 ImageProc Class Reference

The [Image](#) Processing class.

```
#include <ImageProc.hpp>
```

Public Member Functions

- [ImageProc](#) ()
Class Constructor.

Static Public Member Functions

- static Mat [getEdgeImage](#) (Mat input)
compute image edges
- static Mat [getIntegralImage](#) (Mat input)
compute integral image
- static Mat [getPaddedImg](#) (Mat input, int extTopRows, int extBotRows, int extLeftCols, int extRightCols)
zero pad to the image
- static Mat [convertColor](#) (Mat input, int code)
convert image color space
- static Mat [getChannel](#) (Mat input, unsigned char channel)
get specific channel of image

5.11.1 Detailed Description

The [Image](#) Processing class.

5.11.2 Member Function Documentation

5.11.2.1 Mat ImageProc::convertColor (Mat *input*, int *code*) [static]

convert image color space

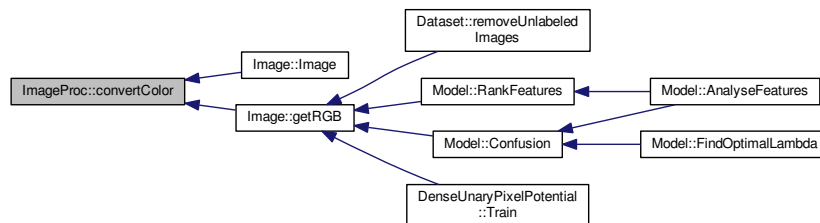
Parameters

<i>input</i>	image
<i>code</i>	OpenCV color conversion code

Returns

color converted image

Here is the caller graph for this function:



5.11.2.2 Mat ImageProc::getChannel (Mat *input*, unsigned char *channel* = 0) [static]

get specific channel of image

Parameters

<i>input</i>	image
<i>channel</i>	channel to return

Returns

image channel

Here is the caller graph for this function:



5.11.2.3 Mat ImageProc::getEdgeImage (Mat *input*) [static]

compute image edges

Parameters

<i>input</i>	RGB image
--------------	-----------

Returns

edge image

5.11.2.4 `Mat ImageProc::getIntegralImage (Mat input) [static]`

compute integral image

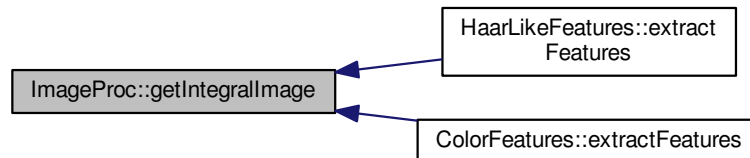
Parameters

<i>input</i>	RGB Image
--------------	---------------------------

Returns

integral image

Here is the caller graph for this function:



5.11.2.5 `Mat ImageProc::getPaddedImg (Mat input, int extTopRows, int extBotRows, int extLeftCols, int extRightCols)` [static]

zero pad to the image

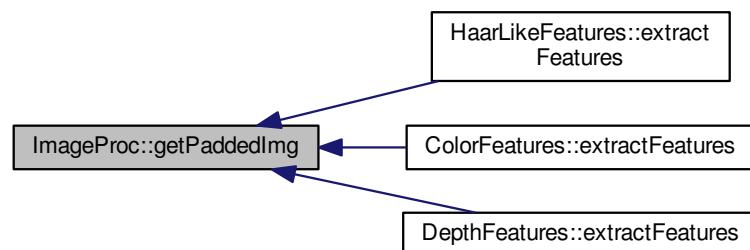
Parameters

<i>input</i>	image
<i>extTopRows</i>	number of top rows to pad zero
<i>extBotRows</i>	number of bot rows to pad zero
<i>extLeftCols</i>	number of left cols to pad zero
<i>extRightCols</i>	number of right cols to pad zero

Returns

zero padded image

Here is the caller graph for this function:



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

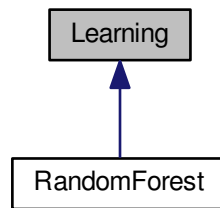
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/ImageProc.hpp
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/ImageProc.cpp

5.12 Learning Class Reference

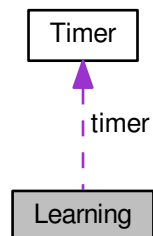
Virtual Class for [Learning](#) Algorithms.

```
#include <Learning.hpp>
```

Inheritance diagram for Learning:



Collaboration diagram for Learning:



Public Member Functions

- [Learning](#) ()
Class Constructor.
- virtual [~Learning](#) ()
Class Deconstructor.
- virtual void [Train](#) (Mat &trainData, const Mat &labels=Mat())=0
Train Classifier.
- virtual void [Evaluate](#) (Mat &testData, Mat &possibleLabels, Mat &labelProbs)=0
Predict class probabilities.
- virtual void [SaveClassifier](#) ()=0

- *Save Classifier.*
virtual void [LoadClassifier](#) ()=0
- *Load Classifier.*
virtual void [ClearClassifier](#) ()=0
- *Clear classifier object.*
int [getTrainTime](#) ()
Return estimated training time.

Public Attributes

- string [trainFile](#)
Path to the classifier file.
- string [name_](#)
Name of the classifier.

Protected Attributes

- uint [nClass](#)
Total number of classes.
- [Timer](#) [timer](#)
Timer to estimate the time.
- int [train_time](#)
Training time in milliseconds.

5.12.1 Detailed Description

Virtual Class for [Learning](#) Algorithms.

5.12.2 Member Function Documentation

5.12.2.1 virtual void [Learning::Evaluate](#) ([Mat](#) & *testData*, [Mat](#) & *possibleLabels*, [Mat](#) & *labelProbs*) [pure virtual]

Predict class probabilities.

Parameters

<i>testData</i>	Test data, size of {number of samples x number of features}
<i>possibleLabels</i>	argmax< class probabilities> for each test sample
<i>labelProbs</i>	Class probabilities for each test sample

Implemented in [RandomForest](#).

Here is the caller graph for this function:



5.12.2.2 `int Learning::getTrainTime ()`

Return estimated training time.

Returns

Estimated time in milliseconds

Here is the caller graph for this function:

5.12.2.3 `virtual void Learning::Train (Mat & trainData, const Mat & labels = Mat ()) [pure virtual]`

Train Classifier.

Parameters

<i>trainData</i>	Training data, size of {number of samples x number of features}
<i>labels</i>	Ground truth class labels, size of {1 x number of samples}

Implemented in [RandomForest](#).

Here is the caller graph for this function:



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

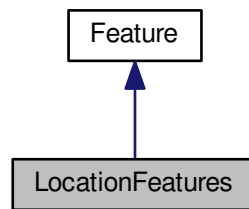
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Learning.hpp`
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Learning.cpp`

5.13 LocationFeatures Class Reference

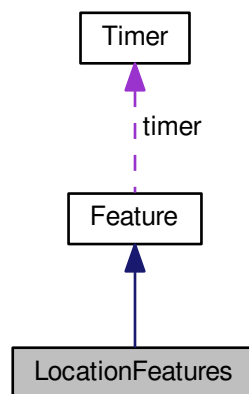
The [LocationFeatures](#) class.

```
#include <Feature.hpp>
```

Inheritance diagram for LocationFeatures:



Collaboration diagram for LocationFeatures:



Public Member Functions

- [LocationFeatures](#) (string LocFolder, string ext, int numSubSample)
Class Constructor.
- void [extractFeatures](#) (Image *im, Mat &features)
Extract features.

Additional Inherited Members

5.13.1 Detailed Description

The [LocationFeatures](#) class.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 LocationFeatures::LocationFeatures (string *LocFolder*, string *ext*, int *numSubSample*)

Class Constructor.

Parameters

<i>LocFolder</i>	folder to store features
<i>ext</i>	extension of feature files
<i>numSubSample</i>	sub sample size

5.13.3 Member Function Documentation

5.13.3.1 void LocationFeatures::extractFeatures (Image * *im*, Mat & *features*) [virtual]

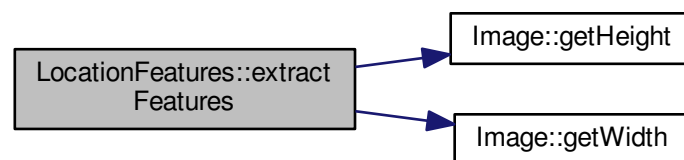
Extract features.

Parameters

<i>im</i>	image
<i>features</i>	feature matrix

Implements [Feature](#).

Here is the call graph for this function:



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

- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Feature.hpp
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Feature.cpp

5.14 Model Class Reference

The [Model](#) class.

```
#include <Model.hpp>
```

Public Member Functions

- [Model](#) ()
Class Constructor.
- void [SetStructure](#) ([Dataset](#) *dataset)
Set model structure.
- void [EvaluateFeatures](#) (vector< string > &imageList, int from=0, int to=-1, int &eval_ftr_time=DummyTime)
Evaluate features.
- void [TrainPotentials](#) (vector< string > &imageList, int from=0, int to=-1, int &train_pt_time=DummyTime)
Train potentials.

- void [EvaluatePotentials](#) (vector< string > &imageList, int from=0, int to=-1, int &eval_pt_time=DummyTime)
Evaluate potentials.
- void [Confusion](#) (vector< string > &imageList, string folder, string confFileName, int from=0, int to=-1)
Compute confusion matrix.
- void [Solve](#) (vector< string > &imageList, int from=0, int to=-1, int &solving_time=DummyTime)
Solve optimization.
- void [RankFeatures](#) (vector< string > &imageList, int from=0, int to=-1, bool isRank=true, int &ranking_time=DummyTime)
Rank features with mrmr.
- void [AnalyseFeatures](#) ()
Analyse features.
- int [SelectFeatures](#) (float alpha, float beta)
Select features.
- void [FindOptimalLambda](#) (vector< string > &imageList, int from=0, int to=-1)
Find optimal lambda for [VariationalOptimization](#).
- void [SavePotentialMap](#) (vector< string > &imageList, int from=0, int to=-1)
Save potential map for images.
- void [ActivateFastComputation](#) (bool isFast=true)
Activate fast computation Load features instead of computing on the fly.
- void [PrintDatasetInfo](#) ()
Print dataset info.
- [Performance](#) [getPERFORMANCE](#) ()
get current performance

5.14.1 Detailed Description

The [Model](#) class.

5.14.2 Member Function Documentation

5.14.2.1 void Model::ActivateFastComputation (bool *isFast* = true)

Activate fast computation Load features instead of computing on the fly.

Activate the loading only relevant features from pre-computed feature files.

Parameters

<i>isFast</i>	flag to activate fast computation
---------------	-----------------------------------

5.14.2.2 void Model::Confusion (vector< string > & *imageList*, string *folder*, string *confFileName*, int *from* = 0, int *to* = -1)

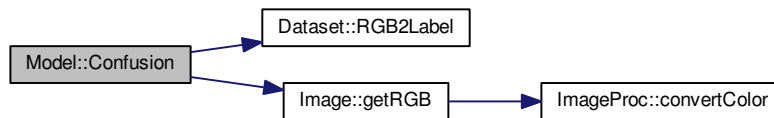
Compute confusion matrix.

Parameters

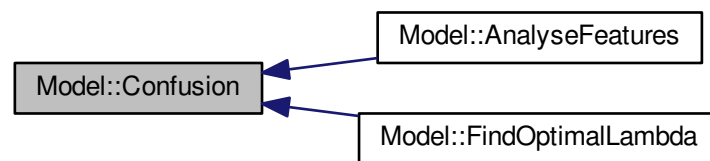
<i>imageList</i>	list of images
<i>folder</i>	path to the folder to save confusion matrix
<i>confFileName</i>	file name for confusion matrix

<i>from</i>	start index
<i>to</i>	stop index, set to -1 to segment all images starting from "from"

Here is the call graph for this function:



Here is the caller graph for this function:



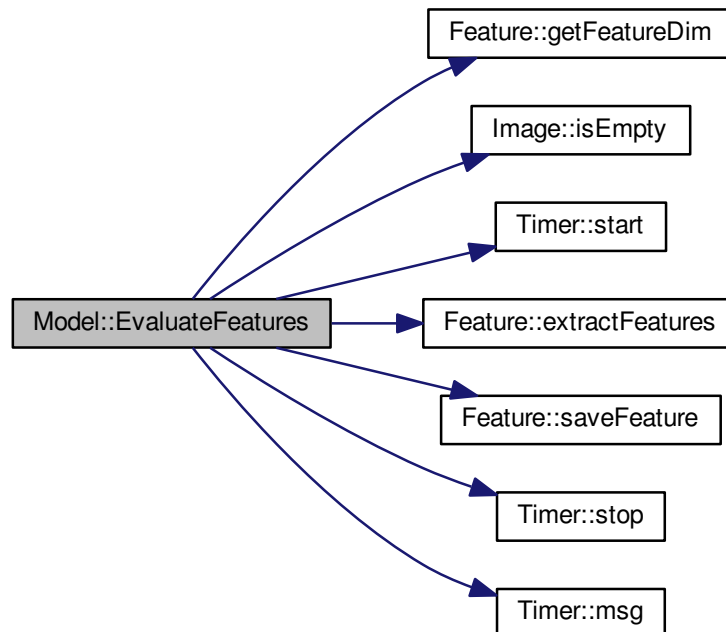
5.14.2.3 `void Model::EvaluateFeatures (vector< string > & imageList, int from = 0, int to = -1, int & eval_ftr_time = DummyTime)`

Evaluate features.

Parameters

<i>imageList</i>	list of images
<i>from</i>	start index
<i>to</i>	stop index, set to -1 to segment all images starting from "from"
<i>eval_ftr_time</i>	estimated time per image in milliseconds

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.4 `void Model::EvaluatePotentials (vector< string > & imageList, int from = 0, int to = -1, int & eval_pt_time = DummyTime)`

Evaluate potentials.

Parameters

<i>imageList</i>	list of images
<i>from</i>	start index
<i>to</i>	stop index, set to -1 to segment all images starting from "from"

<i>eval_pt_time</i>	estimated time per image in milliseconds
---------------------	--

Here is the call graph for this function:



Here is the caller graph for this function:



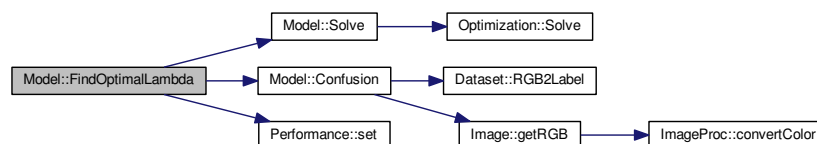
5.14.2.5 `void Model::FindOptimalLambda (vector< string > & imageList, int from = 0, int to = -1)`

Find optimal lambda for [VariationalOptimization](#).

Parameters

<i>imageList</i>	list of images
<i>from</i>	start index
<i>to</i>	stop index, set to -1 to segment all images starting from "from"

Here is the call graph for this function:



5.14.2.6 **Performance** `Model::getPERFORMANCE ()`

get current performance

Returns

current performance

Here is the caller graph for this function:



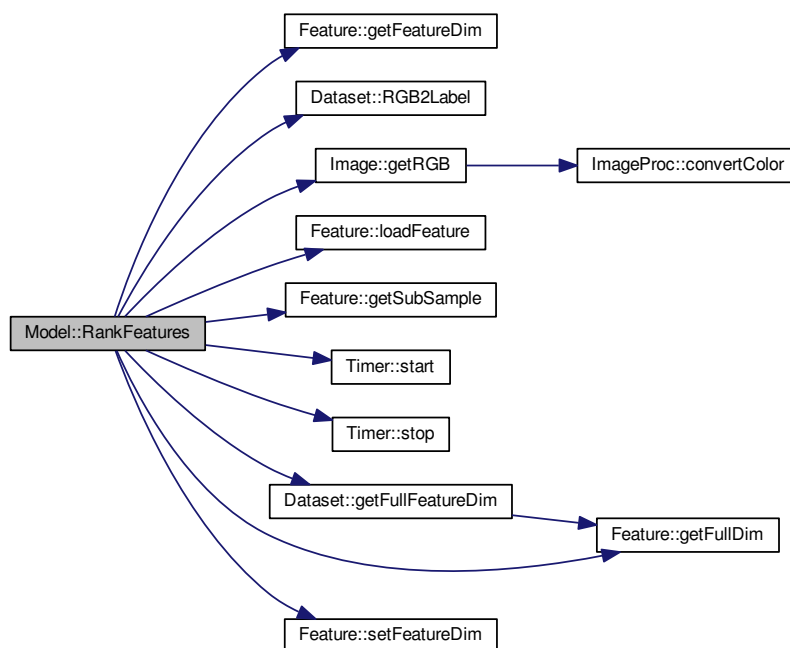
5.14.2.7 `void Model::RankFeatures (vector< string > & imageList, int from = 0, int to = -1, bool isRank = true, int & ranking_time = DummyTime)`

Rank features with mrmr.

Parameters

<i>imageList</i>	list of images
<i>from</i>	start index
<i>to</i>	stop index, set to -1 to segment all images starting from "from"
<i>isRank</i>	flag to rank features
<i>ranking_time</i>	estimated time to rank features

Here is the call graph for this function:



Here is the caller graph for this function:



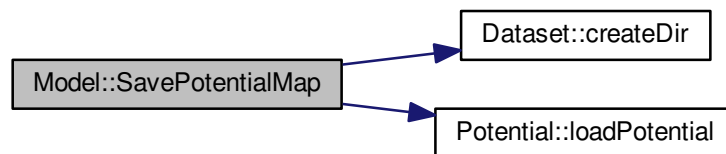
5.14.2.8 `void Model::SavePotentialMap (vector< string > & imageList, int from = 0, int to = -1)`

Save potential map for images.

Parameters

<i>imageList</i>	list of images
<i>from</i>	start index
<i>to</i>	stop index, set to -1 to segment all images starting from "from"

Here is the call graph for this function:



5.14.2.9 `int Model::SelectFeatures (float alpha, float beta)`

Select features.

Parameters

<i>alpha</i>	alpha
<i>beta</i>	beta

Returns

number of selected features

5.14.2.10 `void Model::SetStructure (Dataset * dataset)`

Set model structure.

Parameters

<i>dataset</i>	pointer to the dataset
----------------	------------------------

5.14.2.11 `void Model::Solve (vector< string > & imageList, int from = 0, int to = -1, int & solving_time = DummyTime)`

Solve optimization.

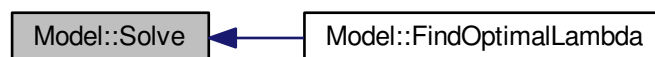
Parameters

<i>imageList</i>	list of images
<i>from</i>	start index
<i>to</i>	stop index, set to -1 to segment all images starting from "from"
<i>solving_time</i>	estimated time per image in milliseconds

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.12 `void Model::TrainPotentials (vector< string > & imageList, int from = 0, int to = -1, int & train_pt_time = DummyTime)`

Train potentials.

Parameters

<i>imageList</i>	list of images
<i>from</i>	start index
<i>to</i>	stop index, set to -1 to segment all images starting from "from"
<i>train_pt_time</i>	estimated time in milliseconds

Here is the call graph for this function:



Here is the caller graph for this function:



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

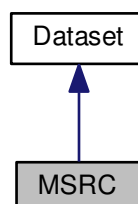
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Model.hpp`
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Model.cpp`

5.15 MSRC Class Reference

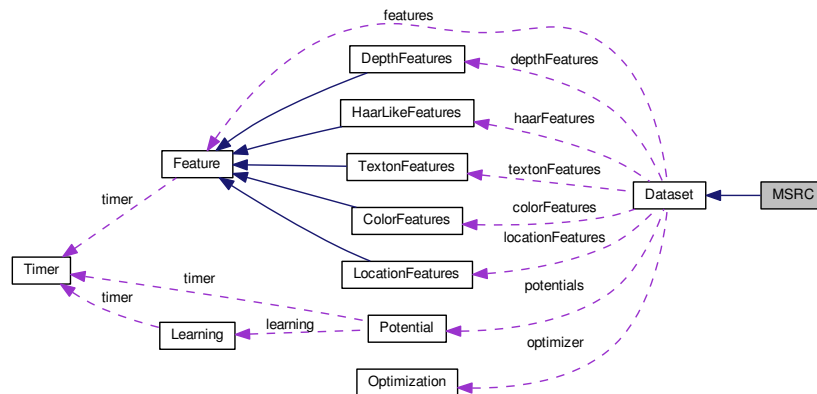
The [MSRC](#) class.

```
#include <Dataset.hpp>
```

Inheritance diagram for MSRC:



Collaboration diagram for MSRC:



Public Member Functions

- [MSRC](#) ()
MSRC.
- void [RGB2Label](#) (Mat rgb, Mat &labels)
Convert from RGB to Label.
- void [Label2RGB](#) (Mat labels, Mat &rgb)
Convert from label to RGB.

Additional Inherited Members

5.15.1 Detailed Description

The [MSRC](#) class.

5.15.2 Member Function Documentation

5.15.2.1 void MSRC::Label2RGB (Mat labels, Mat & rgb) [virtual]

Convert from label to RGB.

Parameters

<i>labels</i>	label matrix
<i>rgb</i>	RGB image

Reimplemented from [Dataset](#).

5.15.2.2 void MSRC::RGB2Label (Mat rgb, Mat & labels) [virtual]

Convert from RGB to Label.

Parameters

<i>rgb</i>	RGB image
<i>labels</i>	label matrix to store labels for each pixel

Reimplemented from [Dataset](#).

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

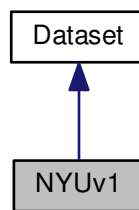
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Dataset.hpp
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Dataset.cpp

5.16 NYUv1 Class Reference

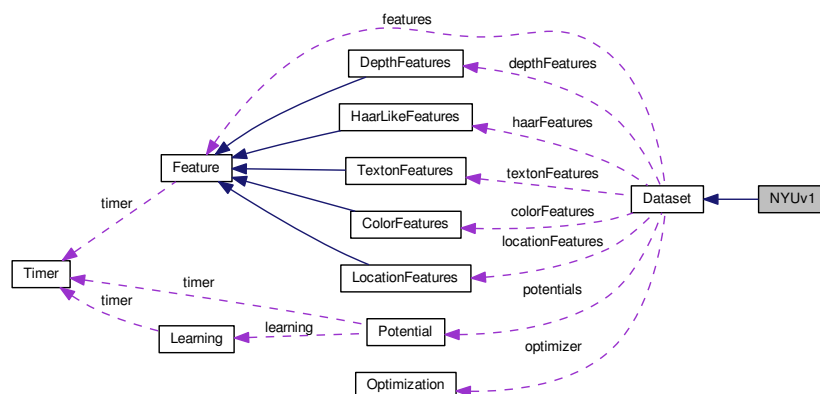
The [NYUv1](#) class.

```
#include <Dataset.hpp>
```

Inheritance diagram for NYUv1:



Collaboration diagram for NYUv1:



Public Member Functions

- [NYUv1](#) ()

Class Constructor.

- void [RGB2Label](#) (Mat rgb, Mat &labels)

Convert from RGB to Label.

- void [Label2RGB](#) (Mat labels, Mat &rgb)

Convert from label to RGB.

Additional Inherited Members

5.16.1 Detailed Description

The [NYUv1](#) class.

5.16.2 Member Function Documentation

5.16.2.1 void NYUv1::Label2RGB (Mat *labels*, Mat & *rgb*) [virtual]

Convert from label to RGB.

Parameters

<i>labels</i>	label matrix
<i>rgb</i>	RGB image

Reimplemented from [Dataset](#).

5.16.2.2 void NYUv1::RGB2Label (Mat *rgb*, Mat & *labels*) [virtual]

Convert from RGB to Label.

Parameters

<i>rgb</i>	RGB image
<i>labels</i>	label matrix to store labels for each pixel

Reimplemented from [Dataset](#).

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

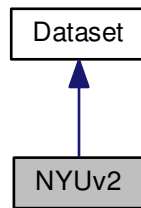
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Dataset.hpp
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Dataset.cpp

5.17 NYUv2 Class Reference

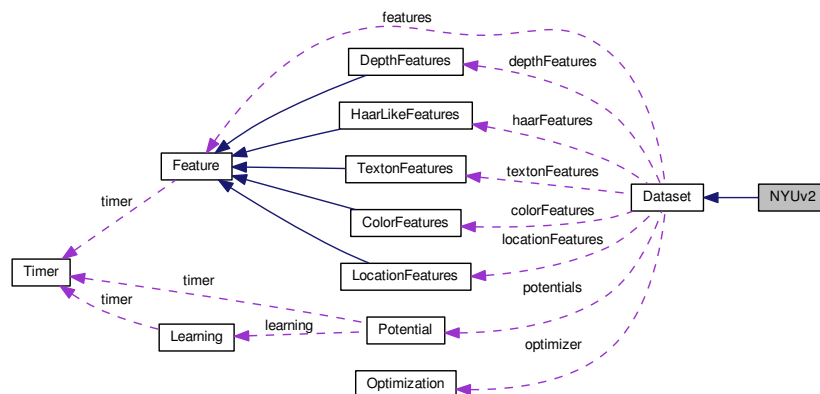
The [NYUv2](#) class.

```
#include <Dataset.hpp>
```

Inheritance diagram for NYUv2:



Collaboration diagram for NYUv2:



Public Member Functions

- [NYUv2](#) ()
Class Constructor.
- void [RGB2Label](#) (Mat rgb, Mat &labels)
Convert from RGB to Label.
- void [Label2RGB](#) (Mat labels, Mat &rgb)
Convert from label to RGB.

Additional Inherited Members

5.17.1 Detailed Description

The [NYUv2](#) class.

5.17.2 Member Function Documentation

5.17.2.1 void NYUv2::Label2RGB (Mat *labels*, Mat & *rgb*) [virtual]

Convert from label to RGB.

Parameters

<i>labels</i>	label matrix
<i>rgb</i>	RGB image

Reimplemented from [Dataset](#).

5.17.2.2 void NYUv2::RGB2Label (Mat *rgb*, Mat & *labels*) [virtual]

Convert from RGB to Label.

Parameters

<i>rgb</i>	RGB image
<i>labels</i>	label matrix to store labels for each pixel

Reimplemented from [Dataset](#).

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

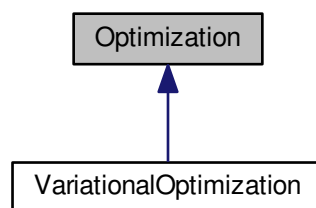
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Dataset.hpp
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Dataset.cpp

5.18 Optimization Class Reference

The [Optimization](#) class.

```
#include <Optimization.hpp>
```

Inheritance diagram for Optimization:

**Public Member Functions**

- virtual int [Solve](#) ([Dataset](#) *dataset, vector< string > &imageList, int from, int to, bool printTime=true)=0
optimize solution

5.18.1 Detailed Description

The [Optimization](#) class.

5.18.2 Member Function Documentation

5.18.2.1 `virtual int Optimization::Solve (Dataset * dataset, vector< string > & imageList, int from, int to, bool printTime = true) [pure virtual]`

optimize solution

Parameters

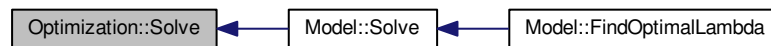
<i>dataset</i>	pointer to the current dataset
<i>imageList</i>	list of images to segment
<i>from</i>	start index
<i>to</i>	stop index, set to -1 to segment all images starting from "from"
<i>printTime</i>	flag to print out the estimated time on output

Returns

estimated time per image in milliseconds

Implemented in [VariationalOptimization](#).

Here is the caller graph for this function:



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

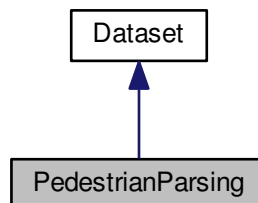
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Optimization.hpp

5.19 PedestrianParsing Class Reference

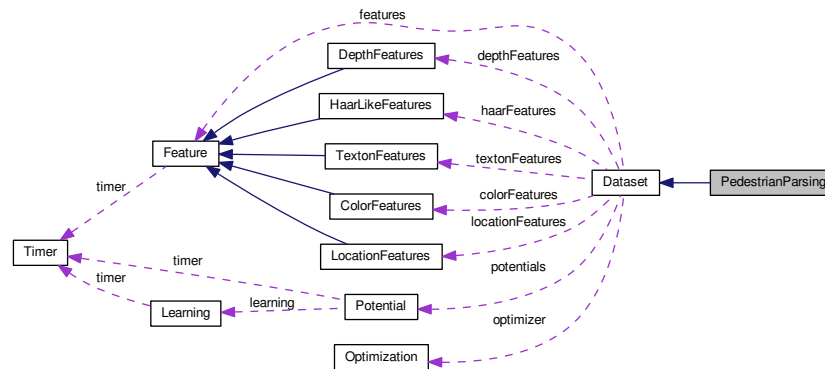
The [PedestrianParsing](#) class.

```
#include <Dataset.hpp>
```

Inheritance diagram for PedestrianParsing:



Collaboration diagram for PedestrianParsing:



Public Member Functions

- [PedestrianParsing](#) ()
Class Constructor.
- void [RGB2Label](#) (Mat rgb, Mat &labels)
Convert from RGB to Label.
- void [Label2RGB](#) (Mat labels, Mat &rgb)
Convert from label to RGB.

Additional Inherited Members

5.19.1 Detailed Description

The [PedestrianParsing](#) class.

5.19.2 Member Function Documentation

5.19.2.1 void PedestrianParsing::Label2RGB (Mat labels, Mat & rgb) [virtual]

Convert from label to RGB.

Parameters

<i>labels</i>	label matrix
<i>rgb</i>	RGB image

Reimplemented from [Dataset](#).

5.19.2.2 void PedestrianParsing::RGB2Label (Mat rgb, Mat & labels) [virtual]

Convert from RGB to Label.

Parameters

<i>rgb</i>	RGB image
<i>labels</i>	label matrix to store labels for each pixel

Reimplemented from [Dataset](#).

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

- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Dataset.hpp
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Dataset.cpp

5.20 Performance Class Reference

The [Performance](#) class.

```
#include <common.hpp>
```

Public Member Functions

- [Performance](#) (double init=0)
Class Constructor.
- void [set](#) ([Performance](#) perf)
set performance
- [Performance](#) & [operator=](#) (const [Performance](#) &p)
operator =
- [Performance](#) & [operator+=](#) (const [Performance](#) &p)
operator +=
- [Performance](#) & [operator/=](#) (const int &t)
operator /=

Public Attributes

- double [overall](#)
overall score
- double [average](#)
average score
- double [waverage](#)
weighted average score

5.20.1 Detailed Description

The [Performance](#) class.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 [Performance::Performance](#) (double *init* = 0) [inline]

Class Constructor.

Parameters

<i>init</i>	initial performance score
-------------	---------------------------

5.20.3 Member Function Documentation

5.20.3.1 Performance& Performance::operator+= (const Performance & *p*) [inline]

operator +=

Parameters

<i>p</i>	pointer to performance
----------	------------------------

Returns

new performance

5.20.3.2 Performance& Performance::operator/= (const int & *t*) [inline]

operator /=

Parameters

<i>t</i>	number
----------	--------

Returns

new performance

5.20.3.3 Performance& Performance::operator= (const Performance & *p*) [inline]

operator =

Parameters

<i>p</i>	pointer to performance
----------	------------------------

Returns

new performance

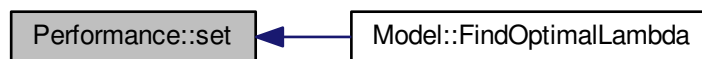
5.20.3.4 void Performance::set (Performance *perf*) [inline]

set performance

Parameters

<i>perf</i>	performance
-------------	-------------

Here is the caller graph for this function:



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

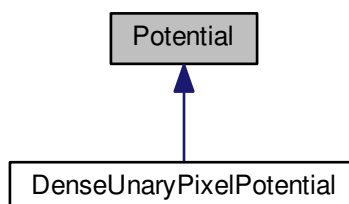
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/common.hpp`

5.21 Potential Class Reference

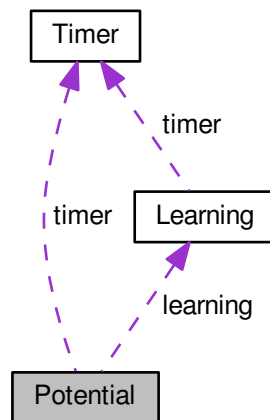
The [Potential](#) class.

```
#include <Potential.hpp>
```

Inheritance diagram for Potential:



Collaboration diagram for Potential:



Public Member Functions

- **Potential** ()
Class Constructor.
- virtual **~Potential** ()
Class Deconstructor.
- virtual int **Train** (**Dataset** *dataset, vector< string > &imageList, int from, int to, bool FAST_COMPUTATION)=0
Train potential.
- virtual int **Evaluate** (**Dataset** *dataset, vector< string > &imageList, int from, int to, bool FAST_COMPUTATION)=0
Evaluate potentials.
- void **savePotential** (string imName, Mat &potentials, int classNo)
Save potential.
- void **loadPotential** (string imName, Mat &potentials)
Load Potential.

Public Attributes

- string **name_**
Potential name.
- **Learning** * **learning**
pointer to the learning algorithm

Protected Attributes

- FILE * **foperator**
file operator
- string **folder**

potential output folder

- string [ext](#)

potential extension

- [Timer](#) *timer*

timer to estimate the time

5.21.1 Detailed Description

The [Potential](#) class.

5.21.2 Member Function Documentation

5.21.2.1 `virtual int Potential::Evaluate (Dataset * dataset, vector< string > & imageList, int from, int to, bool FAST_COMPUTATION)` [pure virtual]

Evaluate potentials.

Parameters

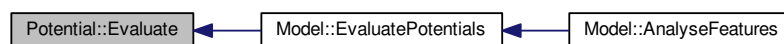
<i>dataset</i>	pointer to the current dataset
<i>imageList</i>	list of images
<i>from</i>	start index
<i>to</i>	stop index, set to -1 to process all images startin from "from"
<i>FAST_COMPUTATION</i>	flag to activate fast computation

Returns

estimated time per image in milliseconds

Implemented in [DenseUnaryPixelPotential](#).

Here is the caller graph for this function:



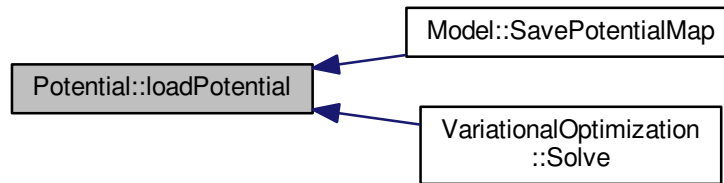
5.21.2.2 `void Potential::loadPotential (string imName, Mat & potentials)`

Load [Potential](#).

Parameters

<i>imName</i>	path to the image
<i>potentials</i>	potentials

Here is the caller graph for this function:



5.21.2.3 void Potential::savePotential (string *imName*, Mat & *potentials*, int *classNo*)

Save potential.

Parameters

<i>imName</i>	path to the image
<i>potentials</i>	potentials
<i>classNo</i>	total number of classes

Here is the caller graph for this function:



5.21.2.4 virtual int Potential::Train (Dataset * *dataset*, vector< string > & *imageList*, int *from*, int *to*, bool *FAST_COMPUTATION*) [pure virtual]

Train potential.

Parameters

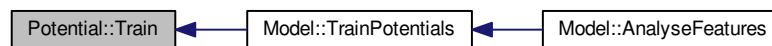
<i>dataset</i>	pointer to the current dataset
<i>imageList</i>	list of images
<i>from</i>	start index
<i>to</i>	stop index, set to -1 to process all images startin from "from"
<i>FAST_COMPUTATION</i>	flag to activate fast computation

Returns

estimated training time in milliseconds

Implemented in [DenseUnaryPixelPotential](#).

Here is the caller graph for this function:



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

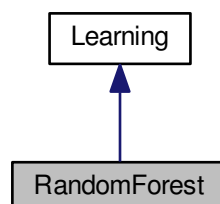
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Potential.hpp`
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Potential.cpp`

5.22 RandomForest Class Reference

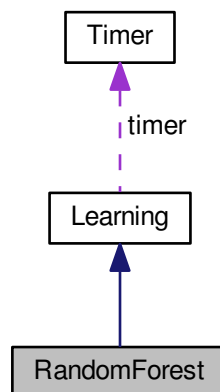
The [RandomForest](#) class.

```
#include <Learning.hpp>
```

Inheritance diagram for RandomForest:



Collaboration diagram for RandomForest:



Public Member Functions

- [RandomForest](#) (string *clsFile*, CvRTPParams *params*, unsigned int *numClasses*)
Class Constructor.
- void [Train](#) (Mat &trainData, const Mat &labels=Mat())
Train Classifier.
- void [Evaluate](#) (Mat &testData, Mat &possibleLabels, Mat &labelProbs)
Predict class probabilities.
- void [SaveClassifier](#) ()
Save Classifier.
- void [LoadClassifier](#) ()
Load Classifier.
- void [ClearClassifier](#) ()
Clear classifier object.

Additional Inherited Members

5.22.1 Detailed Description

The [RandomForest](#) class.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 RandomForest::RandomForest (string *clsFile*, CvRTPParams *params*, unsigned int *numClasses*)

Class Constructor.

Parameters

<i>clsFile</i>	Path to the cassifier file
<i>params</i>	Structure for classifier parameters
<i>numClasses</i>	Total number of classes

5.22.3 Member Function Documentation

5.22.3.1 `void RandomForest::Evaluate (Mat & testData, Mat & possibleLabels, Mat & labelProbs)` [virtual]

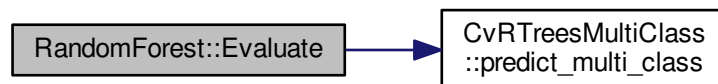
Predict class probabilities.

Parameters

<i>testData</i>	Test data, size of {number of samples x number of features}
<i>possibleLabels</i>	argmax< class probabilities> for each test sample
<i>labelProbs</i>	Class probabilities for each test sample

Implements [Learning](#).

Here is the call graph for this function:



5.22.3.2 `void RandomForest::Train (Mat & trainData, const Mat & labels = Mat ())` [virtual]

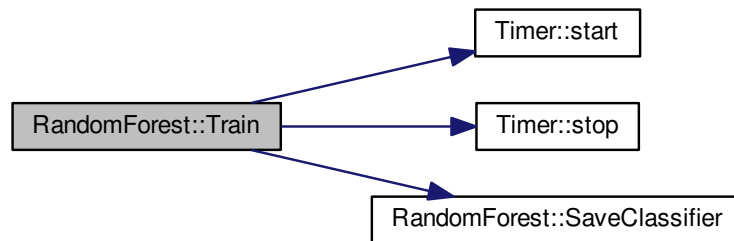
Train Classifier.

Parameters

<i>trainData</i>	Training data, size of {number of samples x number of features}
<i>labels</i>	Ground truth class labels, size of {1 x number of samples}

Implements [Learning](#).

Here is the call graph for this function:



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

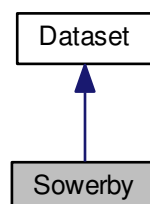
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Learning.hpp`
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Learning.cpp`

5.23 Sowerby Class Reference

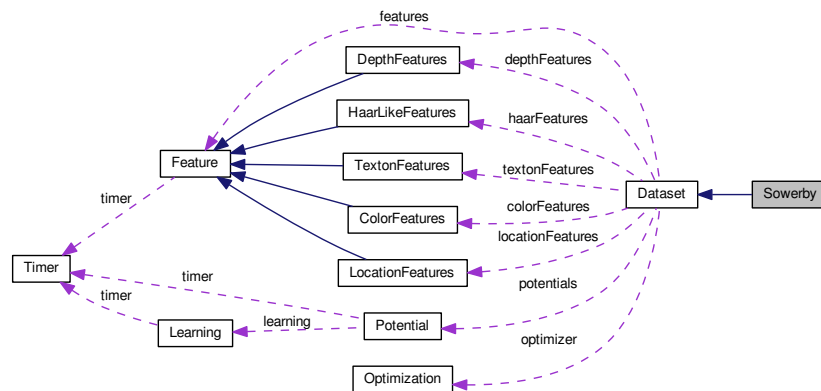
The [Sowerby](#) class.

```
#include <Dataset.hpp>
```

Inheritance diagram for Sowerby:



Collaboration diagram for Sowerby:



Public Member Functions

- [Sowerby](#) ()

Class Constructor.

Additional Inherited Members

5.23.1 Detailed Description

The [Sowerby](#) class.

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

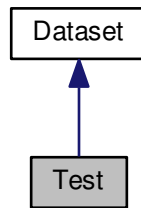
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Dataset.hpp
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Dataset.cpp

5.24 Test Class Reference

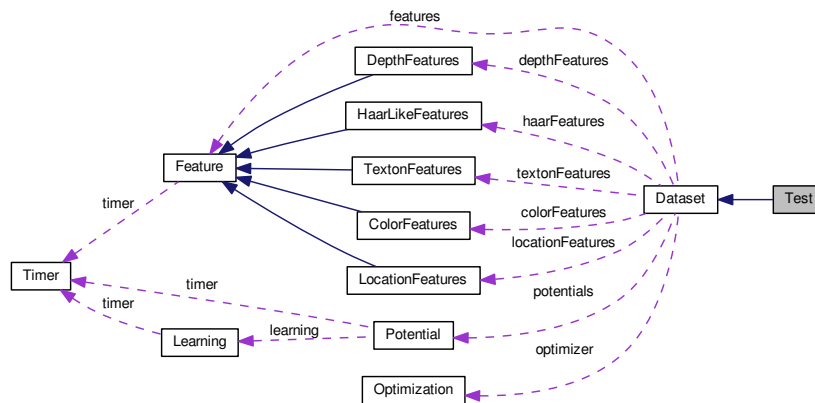
The [Test](#) class.

```
#include <Dataset.hpp>
```

Inheritance diagram for Test:



Collaboration diagram for Test:



Public Member Functions

- [Test \(\)](#)
Class Constructor.

Additional Inherited Members

5.24.1 Detailed Description

The [Test](#) class.

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

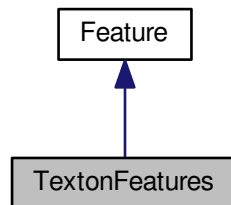
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Dataset.hpp
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Dataset.cpp

5.25 TextonFeatures Class Reference

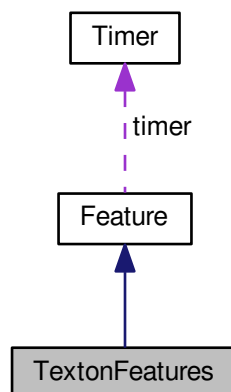
The [TextonFeatures](#) class.

```
#include <Feature.hpp>
```

Inheritance diagram for TextonFeatures:



Collaboration diagram for TextonFeatures:



Public Member Functions

- [TextonFeatures](#) (string TextonFolder, string ext, int numSubSample, double bandWidth)
Class Constructor.
- void [extractFeatures](#) ([Image](#) *im, Mat &filterResponses)
Extract features.

Additional Inherited Members

5.25.1 Detailed Description

The [TextonFeatures](#) class.

5.25.2 Constructor & Destructor Documentation

5.25.2.1 TextonFeatures::TextonFeatures (string *TextonFolder*, string *ext*, int *numSubSample*, double *bandWidth*)

Class Constructor.

Parameters

<i>TextonFolder</i>	folder to store features
<i>ext</i>	extension of feature files
<i>numSubSample</i>	sub sample size
<i>bandWidth</i>	bandwidth of texton kernels

5.25.3 Member Function Documentation

5.25.3.1 void TextonFeatures::extractFeatures (Image * *im*, Mat & *features*) [virtual]

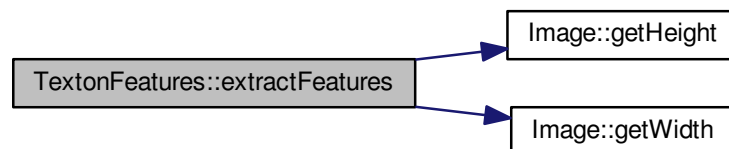
Extract features.

Parameters

<i>im</i>	image
<i>features</i>	feature matrix

Implements [Feature](#).

Here is the call graph for this function:



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

- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Feature.hpp
- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Feature.cpp

5.26 Timer Class Reference

The [Timer](#) class.

```
#include <common.hpp>
```

Public Member Functions

- void `start` ()
start clock
- int `stop` (string `msg`="")
stop clock
- void `msg` (int msec, string `msg`)
print out message

Static Public Member Functions

- static string `formatted_time` (int msec)
format time
- static int `parse_formatted_time` (string `time_stream`)
parse formatted time
- static int `num_digits` (int num)
number of digits
- static string `to_string` (int num)
convert number to string

5.26.1 Detailed Description

The `Timer` class.

5.26.2 Member Function Documentation

5.26.2.1 static string `Timer::formatted_time` (int `msecs`) `[inline]`, `[static]`

format time

Parameters

<code>msecs</code>	milliseconds
--------------------	--------------

Returns

formatted time

Here is the caller graph for this function:



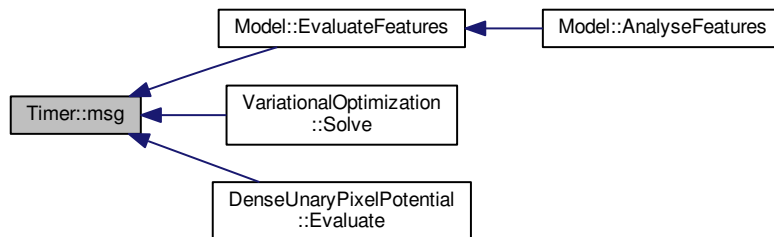
5.26.2.2 void `Timer::msg` (int `msecs`, string `msg`) `[inline]`

print out message

Parameters

<i>msecs</i>	milliseconds
<i>msg</i>	message

Here is the caller graph for this function:



5.26.2.3 static int Timer::num_digits (int *num*) [inline],[static]

number of digits

Parameters

<i>num</i>	number
------------	--------

Returns

total number of digits

5.26.2.4 static int Timer::parse_formatted_time (string *time_stream*) [inline],[static]

parse formatted time

Parameters

<i>time_stream</i>	time stream
--------------------	-------------

Returns

time in milliseconds

5.26.2.5 int Timer::stop (string *msg* = " ") [inline]

stop clock

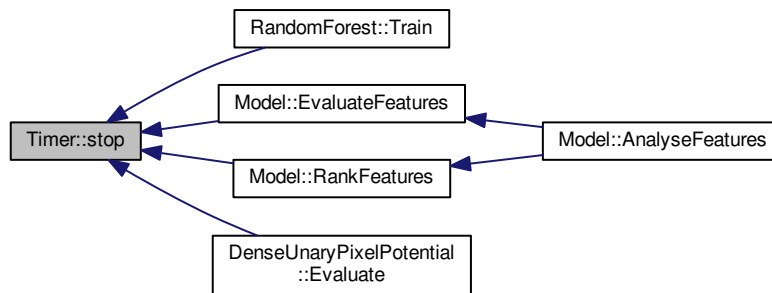
Parameters

<i>msg</i>	message to print out
------------	----------------------

Returns

time in milliseconds

Here is the caller graph for this function:



5.26.2.6 `static string Timer::to_string (int num)` `[inline]`, `[static]`

convert number to string

Parameters

<i>num</i>	number
------------	--------

Returns

string

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

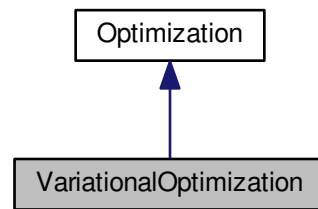
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/common.hpp`

5.27 VariationalOptimization Class Reference

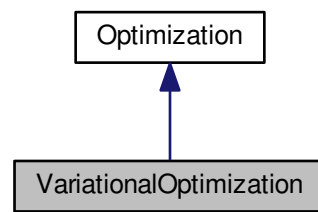
The [VariationalOptimization](#) class.

```
#include <Optimization.hpp>
```

Inheritance diagram for VariationalOptimization:



Collaboration diagram for VariationalOptimization:



Public Member Functions

- [VariationalOptimization](#) ()
Class Constructor.
- int [Solve](#) ([Dataset](#) *dataset, vector< string > &imageList, int from, int to, bool printTime=true)
optimize solution

5.27.1 Detailed Description

The [VariationalOptimization](#) class.

5.27.2 Member Function Documentation

5.27.2.1 int `VariationalOptimization::Solve (Dataset * dataset, vector< string > & imageList, int from, int to, bool printTime =true)` [virtual]

optimize solution

Parameters

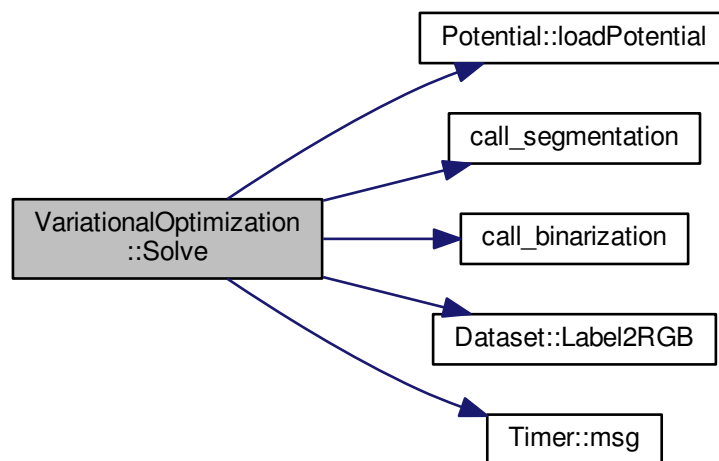
<i>dataset</i>	pointer to the current dataset
<i>imageList</i>	list of images to segment
<i>from</i>	start index
<i>to</i>	stop index, set to -1 to segment all images starting from "from"
<i>printTime</i>	flag to print out the estimated time on output

Returns

estimated time per image in milliseconds

Implements [Optimization](#).

Here is the call graph for this function:



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

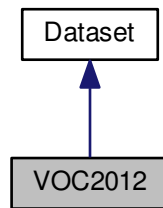
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Optimization.hpp`
- `/usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/src/Optimization.cpp`

5.28 VOC2012 Class Reference

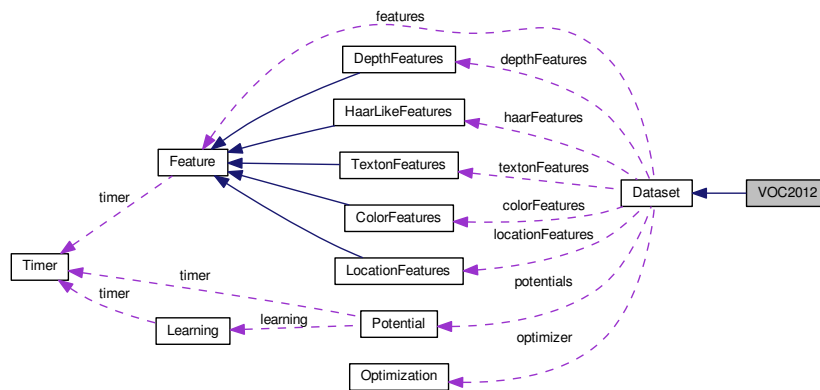
The [VOC2012](#) class.

```
#include <Dataset.hpp>
```

Inheritance diagram for VOC2012:



Collaboration diagram for VOC2012:



Public Member Functions

- [VOC2012](#) ()
Class Constructor.
- void [RGB2Label](#) (Mat rgb, Mat &labels)
Convert from RGB to Label.
- void [Label2RGB](#) (Mat labels, Mat &rgb)
Convert from label to RGB.

Additional Inherited Members

5.28.1 Detailed Description

The [VOC2012](#) class.

5.28.2 Member Function Documentation

5.28.2.1 void VOC2012::Label2RGB (Mat *labels*, Mat & *rgb*) [virtual]

Convert from label to RGB.

Parameters

<i>labels</i>	label matrix
<i>rgb</i>	RGB image

Reimplemented from [Dataset](#).

5.28.2.2 void VOC2012::RGB2Label (Mat *rgb*, Mat & *labels*) [virtual]

Convert from RGB to Label.

Parameters

<i>rgb</i>	RGB image
<i>labels</i>	label matrix to store labels for each pixel

Reimplemented from [Dataset](#).

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

- /usr/wiss/hazirbas/Work/Projects/thesis-hazibas/AFS/include/Dataset.hpp

Index

- [__cudaSafeCall](#)
 - [CUDA_PRIMALDUAL_OPTIMIZATION, 7](#)
- [ActivateFastComputation](#)
 - [Model, 65](#)
- [call_binarization](#)
 - [CUDA_PRIMALDUAL_OPTIMIZATION, 8](#)
- [call_segmentation](#)
 - [CUDA_PRIMALDUAL_OPTIMIZATION, 8](#)
- [centerSurround](#)
 - [HaarLikeFeatures, 47](#)
- [clearDir](#)
 - [Dataset, 22](#)
- [ColorFeatures, 11](#)
 - [ColorFeatures, 12](#)
 - [ColorFeatures, 12](#)
 - [extractFeatures, 13](#)
- [Confusion](#)
 - [Model, 65](#)
- [convertColor](#)
 - [ImageProc, 55](#)
- [Corel, 13](#)
 - [Label2RGB, 14](#)
 - [RGB2Label, 16](#)
- [createDir](#)
 - [Dataset, 22](#)
- [CvRTreesMultiClass, 16](#)
 - [predict_multi_class, 17](#)
- [Dataset, 18](#)
 - [clearDir, 22](#)
 - [createDir, 22](#)
 - [findImage, 22](#)
 - [getDatasetImages, 23](#)
 - [getFullFeatureDim, 24](#)
 - [Label2RGB, 25](#)
 - [RGB2Label, 26](#)
 - [readImgsInDir, 25](#)
 - [readSplitsToVectors, 26](#)
- [DenseUnaryPixelPotential, 27](#)
 - [DenseUnaryPixelPotential, 28](#)
 - [DenseUnaryPixelPotential, 28](#)
 - [Evaluate, 29](#)
 - [Train, 30](#)
- [DepthFeatures, 31](#)
 - [DepthFeatures, 32](#)
 - [DepthFeatures, 32](#)
 - [extractFeatures, 34](#)
- [eTrims, 34](#)
- [Label2RGB, 35](#)
- [RGB2Label, 37](#)
- [Evaluate](#)
 - [DenseUnaryPixelPotential, 29](#)
 - [Learning, 60](#)
 - [Potential, 86](#)
 - [RandomForest, 90](#)
- [EvaluateFeatures](#)
 - [Model, 66](#)
- [EvaluatePotentials](#)
 - [Model, 67](#)
- [extractFeatures](#)
 - [ColorFeatures, 13](#)
 - [DepthFeatures, 34](#)
 - [Feature, 39](#)
 - [HaarLikeFeatures, 47](#)
 - [LocationFeatures, 64](#)
 - [TextonFeatures, 95](#)
- [Feature, 37](#)
 - [extractFeatures, 39](#)
 - [getFeatureDim, 39](#)
 - [getFullDim, 40](#)
 - [getSubSample, 40](#)
 - [loadFeature, 41](#)
 - [saveFeature, 41](#)
 - [setFeatureDim, 42](#)
 - [setRectSize, 42](#)
 - [setSubSample, 42](#)
- [findImage](#)
 - [Dataset, 22](#)
- [FindOptimalLambda](#)
 - [Model, 68](#)
- [formatted_time](#)
 - [Timer, 96](#)
- [fourSquare](#)
 - [HaarLikeFeatures, 48](#)
- [getChannel](#)
 - [ImageProc, 56](#)
- [getDatasetImages](#)
 - [Dataset, 23](#)
- [getEdgeImage](#)
 - [ImageProc, 56](#)
- [getFeatureDim](#)
 - [Feature, 39](#)
- [getFullDim](#)
 - [Feature, 40](#)
- [getFullFeatureDim](#)
 - [Dataset, 24](#)

- getHeight
 - Image, [52](#)
- getImage
 - Image, [52](#)
- getIntegralImage
 - ImageProc, [56](#)
- getPERFORMANCE
 - Model, [68](#)
- getPaddedImg
 - ImageProc, [58](#)
- getRGB
 - Image, [53](#)
- getSubSample
 - Feature, [40](#)
- getTrainTime
 - Learning, [60](#)
- getWidth
 - Image, [53](#)
- HaarLikeFeatures, [44](#)
 - centerSurround, [47](#)
 - extractFeatures, [47](#)
 - fourSquare, [48](#)
 - HaarLikeFeatures, [46](#)
 - HaarLikeFeatures, [46](#)
 - horizontalEdge, [49](#)
 - horizontalLine, [49](#)
 - verticalEdge, [49](#)
 - verticalLine, [50](#)
- horizontalEdge
 - HaarLikeFeatures, [49](#)
- horizontalLine
 - HaarLikeFeatures, [49](#)
- Image, [50](#)
 - getHeight, [52](#)
 - getImage, [52](#)
 - getRGB, [53](#)
 - getWidth, [53](#)
 - Image, [51](#)
 - isEmpty, [54](#)
- ImageProc, [55](#)
 - convertColor, [55](#)
 - getChannel, [56](#)
 - getEdgeImage, [56](#)
 - getIntegralImage, [56](#)
 - getPaddedImg, [58](#)
- isEmpty
 - Image, [54](#)
- kernel_binarize_u
 - CUDA_PRIMALDUAL_OPTIMIZATION, [9](#)
- kernel_grad_ascent
 - CUDA_PRIMALDUAL_OPTIMIZATION, [9](#)
- kernel_grad_descent
 - CUDA_PRIMALDUAL_OPTIMIZATION, [9](#)
- Label2RGB
 - Corel, [14](#)
 - Dataset, [25](#)
 - eTrims, [35](#)
 - MSRC, [73](#)
 - NYUv1, [75](#)
 - NYUv2, [76](#)
 - PedestrianParsing, [81](#)
 - VOC2012, [101](#)
- Learning, [59](#)
 - Evaluate, [60](#)
 - getTrainTime, [60](#)
 - Train, [61](#)
- loadFeature
 - Feature, [41](#)
- loadPotential
 - Potential, [86](#)
- LocationFeatures, [61](#)
 - extractFeatures, [64](#)
 - LocationFeatures, [62](#)
 - LocationFeatures, [62](#)
- MSRC, [72](#)
 - Label2RGB, [73](#)
 - RGB2Label, [73](#)
- Model, [64](#)
 - ActivateFastComputation, [65](#)
 - Confusion, [65](#)
 - EvaluateFeatures, [66](#)
 - EvaluatePotentials, [67](#)
 - FindOptimalLambda, [68](#)
 - getPERFORMANCE, [68](#)
 - RankFeatures, [69](#)
 - SavePotentialMap, [70](#)
 - SelectFeatures, [70](#)
 - SetStructure, [70](#)
 - Solve, [71](#)
 - TrainPotentials, [71](#)
- msg
 - Timer, [96](#)
- NYUv1, [74](#)
 - Label2RGB, [75](#)
 - RGB2Label, [75](#)
- NYUv2, [75](#)
 - Label2RGB, [76](#)
 - RGB2Label, [78](#)
- num_digits
 - Timer, [97](#)
- operator+=
 - Performance, [83](#)
- operator/=
 - Performance, [83](#)
- operator=
 - Performance, [83](#)
- Optimization, [78](#)
 - Solve, [79](#)
- parse_formatted_time
 - Timer, [97](#)

- PedestrianParsing, 80
 - Label2RGB, 81
 - RGB2Label, 81
- Performance, 82
 - operator+=, 83
 - operator/=: 83
 - operator=: 83
 - Performance, 82
 - set, 83
- Potential, 84
 - Evaluate, 86
 - loadPotential, 86
 - savePotential, 87
 - Train, 87
- predict_multi_class
 - CvRTreesMultiClass, 17
- RGB2Label
 - Corel, 16
 - Dataset, 26
 - eTrims, 37
 - MSRC, 73
 - NYUv1, 75
 - NYUv2, 78
 - PedestrianParsing, 81
 - VOC2012, 103
- RandomForest, 88
 - Evaluate, 90
 - RandomForest, 89
 - RandomForest, 89
 - Train, 90
- RankFeatures
 - Model, 69
- readImgsInDir
 - Dataset, 25
- readSplitsToVectors
 - Dataset, 26
- saveFeature
 - Feature, 41
- savePotential
 - Potential, 87
- SavePotentialMap
 - Model, 70
- SelectFeatures
 - Model, 70
- set
 - Performance, 83
- setFeatureDim
 - Feature, 42
- setRectSize
 - Feature, 42
- SetStructure
 - Model, 70
- setSubSample
 - Feature, 42
- Solve
 - Model, 71
 - Optimization, 79
 - VariationalOptimization, 99
- Sowerby, 91
- stop
 - Timer, 97
- Test, 92
- TextonFeatures, 94
 - extractFeatures, 95
 - TextonFeatures, 95
 - TextonFeatures, 95
- Timer, 95
 - formatted_time, 96
 - msg, 96
 - num_digits, 97
 - parse_formatted_time, 97
 - stop, 97
 - to_string, 98
- to_string
 - Timer, 98
- Train
 - DenseUnaryPixelPotential, 30
 - Learning, 61
 - Potential, 87
 - RandomForest, 90
- TrainPotentials
 - Model, 71
- VOC2012, 100
 - Label2RGB, 101
 - RGB2Label, 103
- VariationalOptimization, 98
 - Solve, 99
- verticalEdge
 - HaarLikeFeatures, 49
- verticalLine
 - HaarLikeFeatures, 50