IMAR-C

0.1

Generated by Doxygen 1.6.3

Wed Jul 17 23:46:47 2013

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 activitiesMap Class Reference

Correspondance between a label and an activity.

`#include <naomngt.h>`

Public Attributes

- int **label**
- std::string **activity**

4.1.1 Detailed Description

Correspondance between a label and an activity.

Definition at line 28 of file naomngt.h.

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

- naomngt.h

4.2 Box< T > Class Template Reference

Public Types

- typedef T **ValueType**
- typedef T **ValueType**
- typedef T **ValueType**
- typedef T **ValueType**
- typedef T **ValueType**

Public Member Functions

- **Box** (const Point< T > &topLeft, const Size< T > &size)
- **Box** (T left, T top, T width, T height)
- bool **isEmpty** () const
- bool **isValid** () const
- Box **normalized** () const
- T **getLeft** () const
- T **getTop** () const
- T **getRight** () const
- T **getBottom** () const
- Size< T > **getSize** () const
- T **getX** () const
- T **getY** () const
- void **setLeft** (T pos)
- void **setTop** (T pos)
- void **setRight** (T pos)
- void **setBottom** (T pos)
- void **setX** (T pos)
- void **setY** (T pos)
- Point< T > **getTopLeft** () const
- Point< T > **getBottomRight** () const
- Point< T > **getTopRight** () const
- Point< T > **getBottomLeft** () const
- Point< T > **getCenter** () const
- void **setTopLeft** (const Point< T > &p)
- void **setBottomRight** (const Point< T > &p)
- void **setTopRight** (const Point< T > &p)
- void **setBottomLeft** (const Point< T > &p)
- void **moveLeft** (T pos)
- void **moveTop** (T pos)
- void **moveRight** (T pos)
- void **moveBottom** (T pos)
- void **moveTopLeft** (const Point< T > &p)
- void **moveBottomRight** (const Point< T > &p)
- void **moveTopRight** (const Point< T > &p)
- void **moveBottomLeft** (const Point< T > &p)
- void **moveCenter** (const Point< T > &p)
- void **translate** (T dx, T dy)
- void **translate** (const Point< T > &p)

- Box **translated** (T dx, T dy) const
- Box **translated** (const Point< T > &p) const
- void **moveTo** (T x, T t)
- void **moveTo** (const Point< T > &p)
- void **setBox** (T x, T y, T w, T h)
- void **getBox** (T *x=0, T *y=0, T *w=0, T *h=0) const
- void **setCoords** (T x1, T y1, T x2, T y2)
- void **getCoords** (T *x1=0, T *y1=0, T *x2=0, T *y2=0) const
- template<typename T2 >
  void **scale** (T2 xScale, T2 yScale)
- T **getWidth** () const
- T **getHeight** () const
- void **setWidth** (T w)
- void **setHeight** (T h)
- void **setSize** (const Size< T > &s)
- T **getArea** () const
- Box< T > **operator|** (const Box &r) const
- Box< T > **operator&** (const Box &r) const
- Box< T > & **operator|=** (const Box &r)
- Box< T > & **operator&=** (const Box &r)
- template<typename T2 >
  Box< T > & **operator*=** (T2 c)
- template<typename T2 >
  Box< T > & **operator/=** (T2 c)
- template<typename T2 >
  bool **contains** (const Point< T2 > &p) const
- template<typename T2 >
  bool **contains** (T2 x, T2 y) const
- bool **contains** (const Box< T > &r) const
- Box< T > **unite** (const Box< T > &r) const
- Box< T > **intersect** (const Box< T > &r) const
- bool **intersects** (const Box< T > &r) const
- template<typename T2 >
  operator Box< T2 > () const
- **Box** (const Point< T > &topLeft, const Size< T > &size)
- **Box** (T left, T top, T width, T height)
- bool **isEmpty** () const
- bool **isValid** () const
- Box **normalized** () const
- T **getLeft** () const
- T **getTop** () const
- T **getRight** () const
- T **getBottom** () const
- Size< T > **getSize** () const
- T **getX** () const
- T **getY** () const
- void **setLeft** (T pos)
- void **setTop** (T pos)
- void **setRight** (T pos)
- void **setBottom** (T pos)

- void **setX** (T pos)
- void **setY** (T pos)
- Point< T > **getTopLeft** () const
- Point< T > **getBottomRight** () const
- Point< T > **getTopRight** () const
- Point< T > **getBottomLeft** () const
- Point< T > **getCenter** () const
- void **setTopLeft** (const Point< T > &p)
- void **setBottomRight** (const Point< T > &p)
- void **setTopRight** (const Point< T > &p)
- void **setBottomLeft** (const Point< T > &p)
- void **moveLeft** (T pos)
- void **moveTop** (T pos)
- void **moveRight** (T pos)
- void **moveBottom** (T pos)
- void **moveTopLeft** (const Point< T > &p)
- void **moveBottomRight** (const Point< T > &p)
- void **moveTopRight** (const Point< T > &p)
- void **moveBottomLeft** (const Point< T > &p)
- void **moveCenter** (const Point< T > &p)
- void **translate** (T dx, T dy)
- void **translate** (const Point< T > &p)
- Box **translated** (T dx, T dy) const
- Box **translated** (const Point< T > &p) const
- void **moveTo** (T x, T t)
- void **moveTo** (const Point< T > &p)
- void **setBox** (T x, T y, T w, T h)
- void **getBox** (T *x=0, T *y=0, T *w=0, T *h=0) const
- void **setCoords** (T x1, T y1, T x2, T y2)
- void **getCoords** (T *x1=0, T *y1=0, T *x2=0, T *y2=0) const
- template<typename T2 >
  void **scale** (T2 xScale, T2 yScale)
- T **getWidth** () const
- T **getHeight** () const
- void **setWidth** (T w)
- void **setHeight** (T h)
- void **setSize** (const Size< T > &s)
- T **getArea** () const
- Box< T > **operator|** (const Box &r) const
- Box< T > **operator&** (const Box &r) const
- Box< T > & **operator|=** (const Box &r)
- Box< T > & **operator&=** (const Box &r)
- template<typename T2 >
  Box< T > & **operator*=** (T2 c)
- template<typename T2 >
  Box< T > & **operator/=** (T2 c)
- template<typename T2 >
  bool **contains** (const Point< T2 > &p) const
- template<typename T2 >
  bool **contains** (T2 x, T2 y) const

- bool **contains** (const Box< T > &r) const
- Box< T > **unite** (const Box< T > &r) const
- Box< T > **intersect** (const Box< T > &r) const
- bool **intersects** (const Box< T > &r) const
- template<typename T2 >
  operator Box< T2 > () const
- **Box** (const Point< T > &topLeft, const Size< T > &size)
- **Box** (T left, T top, T width, T height)
- bool **isEmpty** () const
- bool **isValid** () const
- Box **normalized** () const
- T **getLeft** () const
- T **getTop** () const
- T **getRight** () const
- T **getBottom** () const
- Size< T > **getSize** () const
- T **getX** () const
- T **getY** () const
- void **setLeft** (T pos)
- void **setTop** (T pos)
- void **setRight** (T pos)
- void **setBottom** (T pos)
- void **setX** (T pos)
- void **setY** (T pos)
- Point< T > **getTopLeft** () const
- Point< T > **getBottomRight** () const
- Point< T > **getTopRight** () const
- Point< T > **getBottomLeft** () const
- Point< T > **getCenter** () const
- void **setTopLeft** (const Point< T > &p)
- void **setBottomRight** (const Point< T > &p)
- void **setTopRight** (const Point< T > &p)
- void **setBottomLeft** (const Point< T > &p)
- void **moveLeft** (T pos)
- void **moveTop** (T pos)
- void **moveRight** (T pos)
- void **moveBottom** (T pos)
- void **moveTopLeft** (const Point< T > &p)
- void **moveBottomRight** (const Point< T > &p)
- void **moveTopRight** (const Point< T > &p)
- void **moveBottomLeft** (const Point< T > &p)
- void **moveCenter** (const Point< T > &p)
- void **translate** (T dx, T dy)
- void **translate** (const Point< T > &p)
- Box **translated** (T dx, T dy) const
- Box **translated** (const Point< T > &p) const
- void **moveTo** (T x, T t)
- void **moveTo** (const Point< T > &p)
- void **setBox** (T x, T y, T w, T h)
- void **getBox** (T *x=0, T *y=0, T *w=0, T *h=0) const

- void **setCoords** (T x1, T y1, T x2, T y2)
- void **getCoords** (T *x1=0, T *y1=0, T *x2=0, T *y2=0) const
- template<typename T2 >
  void **scale** (T2 xScale, T2 yScale)
- T **getWidth** () const
- T **getHeight** () const
- void **setWidth** (T w)
- void **setHeight** (T h)
- void **setSize** (const Size< T > &s)
- T **getArea** () const
- Box< T > **operator|** (const Box &r) const
- Box< T > **operator&** (const Box &r) const
- Box< T > & **operator|=** (const Box &r)
- Box< T > & **operator&=** (const Box &r)
- template<typename T2 >
  Box< T > & **operator*=** (T2 c)
- template<typename T2 >
  Box< T > & **operator/=** (T2 c)
- template<typename T2 >
  bool **contains** (const Point< T2 > &p) const
- template<typename T2 >
  bool **contains** (T2 x, T2 y) const
- bool **contains** (const Box< T > &r) const
- Box< T > **unite** (const Box< T > &r) const
- Box< T > **intersect** (const Box< T > &r) const
- bool **intersects** (const Box< T > &r) const
- template<typename T2 >
  operator Box< T2 > () const
- **Box** (const Point< T > &topLeft, const Size< T > &size)
- **Box** (T left, T top, T width, T height)
- bool **isEmpty** () const
- bool **isValid** () const
- Box **normalized** () const
- T **getLeft** () const
- T **getTop** () const
- T **getRight** () const
- T **getBottom** () const
- Size< T > **getSize** () const
- T **getX** () const
- T **getY** () const
- void **setLeft** (T pos)
- void **setTop** (T pos)
- void **setRight** (T pos)
- void **setBottom** (T pos)
- void **setX** (T pos)
- void **setY** (T pos)
- Point< T > **getTopLeft** () const
- Point< T > **getBottomRight** () const
- Point< T > **getTopRight** () const
- Point< T > **getBottomLeft** () const

- Point< T > **getCenter** () const
- void **setTopLeft** (const Point< T > &p)
- void **setBottomRight** (const Point< T > &p)
- void **setTopRight** (const Point< T > &p)
- void **setBottomLeft** (const Point< T > &p)
- void **moveLeft** (T pos)
- void **moveTop** (T pos)
- void **moveRight** (T pos)
- void **moveBottom** (T pos)
- void **moveTopLeft** (const Point< T > &p)
- void **moveBottomRight** (const Point< T > &p)
- void **moveTopRight** (const Point< T > &p)
- void **moveBottomLeft** (const Point< T > &p)
- void **moveCenter** (const Point< T > &p)
- void **translate** (T dx, T dy)
- void **translate** (const Point< T > &p)
- Box **translated** (T dx, T dy) const
- Box **translated** (const Point< T > &p) const
- void **moveTo** (T x, T t)
- void **moveTo** (const Point< T > &p)
- void **setBox** (T x, T y, T w, T h)
- void **getBox** (T *x=0, T *y=0, T *w=0, T *h=0) const
- void **setCoords** (T x1, T y1, T x2, T y2)
- void **getCoords** (T *x1=0, T *y1=0, T *x2=0, T *y2=0) const
- template<typename T2 >
  void **scale** (T2 xScale, T2 yScale)
- T **getWidth** () const
- T **getHeight** () const
- void **setWidth** (T w)
- void **setHeight** (T h)
- void **setSize** (const Size< T > &s)
- T **getArea** () const
- Box< T > **operator|** (const Box &r) const
- Box< T > **operator&** (const Box &r) const
- Box< T > & **operator|=** (const Box &r)
- Box< T > & **operator&=** (const Box &r)
- template<typename T2 >
  Box< T > & **operator*=** (T2 c)
- template<typename T2 >
  Box< T > & **operator/=** (T2 c)
- template<typename T2 >
  bool **contains** (const Point< T2 > &p) const
- template<typename T2 >
  bool **contains** (T2 x, T2 y) const
- bool **contains** (const Box< T > &r) const
- Box< T > **unite** (const Box< T > &r) const
- Box< T > **intersect** (const Box< T > &r) const
- bool **intersects** (const Box< T > &r) const
- template<typename T2 >
  operator Box< T2 > () const

- **Box** (const Point< T > &topLeft, const Size< T > &size)
- **Box** (T left, T top, T width, T height)
- bool **isEmpty** () const
- bool **isValid** () const
- Box **normalized** () const
- T **getLeft** () const
- T **getTop** () const
- T **getRight** () const
- T **getBottom** () const
- Size< T > **getSize** () const
- T **getX** () const
- T **getY** () const
- void **setLeft** (T pos)
- void **setTop** (T pos)
- void **setRight** (T pos)
- void **setBottom** (T pos)
- void **setX** (T pos)
- void **setY** (T pos)
- Point< T > **getTopLeft** () const
- Point< T > **getBottomRight** () const
- Point< T > **getTopRight** () const
- Point< T > **getBottomLeft** () const
- Point< T > **getCenter** () const
- void **setTopLeft** (const Point< T > &p)
- void **setBottomRight** (const Point< T > &p)
- void **setTopRight** (const Point< T > &p)
- void **setBottomLeft** (const Point< T > &p)
- void **moveLeft** (T pos)
- void **moveTop** (T pos)
- void **moveRight** (T pos)
- void **moveBottom** (T pos)
- void **moveTopLeft** (const Point< T > &p)
- void **moveBottomRight** (const Point< T > &p)
- void **moveTopRight** (const Point< T > &p)
- void **moveBottomLeft** (const Point< T > &p)
- void **moveCenter** (const Point< T > &p)
- void **translate** (T dx, T dy)
- void **translate** (const Point< T > &p)
- Box **translated** (T dx, T dy) const
- Box **translated** (const Point< T > &p) const
- void **moveTo** (T x, T t)
- void **moveTo** (const Point< T > &p)
- void **setBox** (T x, T y, T w, T h)
- void **getBox** (T *x=0, T *y=0, T *w=0, T *h=0) const
- void **setCoords** (T x1, T y1, T x2, T y2)
- void **getCoords** (T *x1=0, T *y1=0, T *x2=0, T *y2=0) const
- template<typename T2 >
  void **scale** (T2 xScale, T2 yScale)
- T **getWidth** () const
- T **getHeight** () const

- void **setWidth** (T w)
- void **setHeight** (T h)
- void **setSize** (const Size< T > &s)
- T **getArea** () const
- Box< T > **operator|** (const Box &r) const
- Box< T > **operator&** (const Box &r) const
- Box< T > & **operator|=** (const Box &r)
- Box< T > & **operator&=** (const Box &r)
- template<typename T2 >
  Box< T > & **operator∗=** (T2 c)
- template<typename T2 >
  Box< T > & **operator/=** (T2 c)
- template<typename T2 >
  bool **contains** (const Point< T2 > &p) const
- template<typename T2 >
  bool **contains** (T2 x, T2 y) const
- bool **contains** (const Box< T > &r) const
- Box< T > **unite** (const Box< T > &r) const
- Box< T > **intersect** (const Box< T > &r) const
- bool **intersects** (const Box< T > &r) const
- template<typename T2 >
  operator Box< T2 > () const
- template<>
  void **scale** (float xScale, float yScale)
- template<>
  void **scale** (float xScale, float yScale)
- template<>
  void **scale** (double xScale, double yScale)
- template<>
  void **scale** (double xScale, double yScale)
- template<>
  void **scale** (float xScale, float yScale)
- template<>
  void **scale** (float xScale, float yScale)
- template<>
  void **scale** (double xScale, double yScale)
- template<>
  void **scale** (double xScale, double yScale)
- template<>
  void **scale** (float xScale, float yScale)
- template<>
  void **scale** (float xScale, float yScale)
- template<>
  void **scale** (double xScale, double yScale)
- template<>
  void **scale** (double xScale, double yScale)
- template<>
  void **scale** (float xScale, float yScale)
- template<>
  void **scale** (float xScale, float yScale)
- template<>
  void **scale** (double xScale, double yScale)

- template<>
  void **scale** (double xScale, double yScale)
- template<>
  void **scale** (float xScale, float yScale)
- template<>
  void **scale** (float xScale, float yScale)
- template<>
  void **scale** (double xScale, double yScale)
- template<>
  void **scale** (double xScale, double yScale)

Friends

- template<typename T2 >
  bool **operator==** (const Box< T2 > &, const Box< T2 > &)
- template<typename T2 >
  bool **operator!=** (const Box< T2 > &, const Box< T2 > &)
- template<typename T2 , typename T3 >
  Box< T2 > **operator*** (const Box< T2 > &, T3 c)
- template<typename T2 , typename T3 >
  Box< T2 > **operator*** (T3 c, const Box< T2 > &)
- template<typename T2 , typename T3 >
  Box< T2 > **operator/** (const Box< T2 > &, T3 c)
- template<typename T2 >
  bool **operator==** (const Box< T2 > &, const Box< T2 > &)
- template<typename T2 >
  bool **operator!=** (const Box< T2 > &, const Box< T2 > &)
- template<typename T2 , typename T3 >
  Box< T2 > **operator*** (const Box< T2 > &, T3 c)
- template<typename T2 , typename T3 >
  Box< T2 > **operator*** (T3 c, const Box< T2 > &)
- template<typename T2 , typename T3 >
  Box< T2 > **operator/** (const Box< T2 > &, T3 c)
- template<typename T2 >
  bool **operator==** (const Box< T2 > &, const Box< T2 > &)
- template<typename T2 >
  bool **operator!=** (const Box< T2 > &, const Box< T2 > &)
- template<typename T2 , typename T3 >
  Box< T2 > **operator*** (const Box< T2 > &, T3 c)
- template<typename T2 , typename T3 >
  Box< T2 > **operator*** (T3 c, const Box< T2 > &)
- template<typename T2 , typename T3 >
  Box< T2 > **operator/** (const Box< T2 > &, T3 c)
- template<typename T2 >
  bool **operator==** (const Box< T2 > &, const Box< T2 > &)
- template<typename T2 >
  bool **operator!=** (const Box< T2 > &, const Box< T2 > &)
- template<typename T2 , typename T3 >
  Box< T2 > **operator*** (const Box< T2 > &, T3 c)
- template<typename T2 , typename T3 >
  Box< T2 > **operator*** (T3 c, const Box< T2 > &)

- template<typename T2 , typename T3 >
  Box< T2 > **operator/** (const Box< T2 > &, T3 c)
- template<typename T2 >
  bool **operator==** (const Box< T2 > &, const Box< T2 > &)
- template<typename T2 >
  bool **operator!=** (const Box< T2 > &, const Box< T2 > &)
- template<typename T2 , typename T3 >
  Box< T2 > **operator*** (const Box< T2 > &, T3 c)
- template<typename T2 , typename T3 >
  Box< T2 > **operator*** (T3 c, const Box< T2 > &)
- template<typename T2 , typename T3 >
  Box< T2 > **operator/** (const Box< T2 > &, T3 c)

4.2.1 Detailed Description

template<typename T = double> class Box< T >

Definition at line 12 of file include/Box.h.

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

- include/Box.h
- include/densetrack/Box.h
- integration/densetrack/Box.h
- lib/densetrack/Box.h
- modules/densetrack/Box.h

4.3 Cache Class Reference

Classes

- struct **head_t**

Public Member Functions

- **Cache** (int l, long int size)
- int **get_data** (const int index, Qfloat ∗∗data, int len)
- void **swap_index** (int i, int j)

4.3.1 Detailed Description

Definition at line 67 of file svm.cpp.

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

- svm.cpp

4.4 decision_function Struct Reference

Public Attributes

- double * **alpha**
- double **rho**

4.4.1 Detailed Description

Definition at line 1641 of file svm.cpp.

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

- svm.cpp

4.5 DescInfo Struct Reference

Public Attributes

- int **nBins**
- int **fullOrientation**
- int **norm**
- float **threshold**
- int **flagThre**
- int **nxCells**
- int **nyCells**
- int **ntCells**
- int **dim**
- int **blockHeight**
- int **blockWidth**

4.5.1 Detailed Description

Definition at line 30 of file include/denseTrack.h.

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

- include/denseTrack.h
- naodensetrack.h
- DenseTrack.h
- modules/densetrack/denseTrack.h

4.6 DescMat Struct Reference

Public Attributes

- int **height**
- int **width**
- int **nBins**
- float ∗ **desc**

4.6.1 Detailed Description

Definition at line 45 of file include/denseTrack.h.

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

- include/denseTrack.h
- naodensetrack.h
- DenseTrack.h
- modules/densetrack/denseTrack.h

4.7 exec_time Struct Reference

Public Attributes

- int **begin**
- int **end**
- sTms **sbegin**
- sTms **send**

4.7.1 Detailed Description

Definition at line 15 of file integration.h.

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

- integration.h

4.8 IplImagePyramid Class Reference

Public Member Functions

- **IplImagePyramid** (const IplImagePyramid &pyramid)
- IplImagePyramid (IplImageWrapper image, double scaleFactor)
- IplImagePyramid (CvSize initSize, int depth, int nChannels, double scaleFactor)
- IplImagePyramid & **operator=** (const IplImagePyramid &pyramid)
- **operator const bool** () const
- **operator bool** ()
- std::size_t **numOfLevels** () const
- double **getScaleFactor** () const
- std::size_t getIndex (double scaleFactor, int round=0) const
- double **getScaleFactor** (std::size_t index) const
- double **getScaleFactorInv** (std::size_t index) const
- double **getXScaleFactor** (std::size_t index) const
- double **getXScaleFactorInv** (std::size_t index) const
- double **getYScaleFactor** (std::size_t index) const
- double **getYScaleFactorInv** (std::size_t index) const
- IplImageWrapper & **getImage** (std::size_t index)
- const IplImageWrapper & **getImage** (std::size_t index) const
- IplImageWrapper getImage (double scaleFactor, int round=0)
- const IplImageWrapper & **getImage** (double scaleFactor, int round=0) const
- void rebuild (IplImageWrapper image)
- **IplImagePyramid** (const IplImagePyramid &pyramid)
- IplImagePyramid (IplImageWrapper image, double scaleFactor)
- IplImagePyramid (CvSize initSize, int depth, int nChannels, double scaleFactor)
- IplImagePyramid & **operator=** (const IplImagePyramid &pyramid)
- **operator const bool** () const
- **operator bool** ()
- std::size_t **numOfLevels** () const
- double **getScaleFactor** () const
- std::size_t getIndex (double scaleFactor, int round=0) const
- double **getScaleFactor** (std::size_t index) const
- double **getScaleFactorInv** (std::size_t index) const
- double **getXScaleFactor** (std::size_t index) const
- double **getXScaleFactorInv** (std::size_t index) const
- double **getYScaleFactor** (std::size_t index) const
- double **getYScaleFactorInv** (std::size_t index) const
- IplImageWrapper & **getImage** (std::size_t index)
- const IplImageWrapper & **getImage** (std::size_t index) const
- IplImageWrapper getImage (double scaleFactor, int round=0)
- const IplImageWrapper & **getImage** (double scaleFactor, int round=0) const
- void rebuild (IplImageWrapper image)
- **IplImagePyramid** (const IplImagePyramid &pyramid)
- IplImagePyramid (IplImageWrapper image, double scaleFactor)
- IplImagePyramid (CvSize initSize, int depth, int nChannels, double scaleFactor)
- IplImagePyramid & **operator=** (const IplImagePyramid &pyramid)
- **operator const bool** () const
- **operator bool** ()

- std::size_t **numOfLevels** () const
- double **getScaleFactor** () const
- std::size_t getIndex (double scaleFactor, int round=0) const
- double **getScaleFactor** (std::size_t index) const
- double **getScaleFactorInv** (std::size_t index) const
- double **getXScaleFactor** (std::size_t index) const
- double **getXScaleFactorInv** (std::size_t index) const
- double **getYScaleFactor** (std::size_t index) const
- double **getYScaleFactorInv** (std::size_t index) const
- IplImageWrapper & **getImage** (std::size_t index)
- const IplImageWrapper & **getImage** (std::size_t index) const
- IplImageWrapper getImage (double scaleFactor, int round=0)
- const IplImageWrapper & **getImage** (double scaleFactor, int round=0) const
- void rebuild (IplImageWrapper image)
- **IplImagePyramid** (const IplImagePyramid &pyramid)
- IplImagePyramid (IplImageWrapper image, double scaleFactor)
- IplImagePyramid (CvSize initSize, int depth, int nChannels, double scaleFactor)
- IplImagePyramid & **operator=** (const IplImagePyramid &pyramid)
- **operator const bool** () const
- **operator bool** ()
- std::size_t **numOfLevels** () const
- double **getScaleFactor** () const
- std::size_t getIndex (double scaleFactor, int round=0) const
- double **getScaleFactor** (std::size_t index) const
- double **getScaleFactorInv** (std::size_t index) const
- double **getXScaleFactor** (std::size_t index) const
- double **getXScaleFactorInv** (std::size_t index) const
- double **getYScaleFactor** (std::size_t index) const
- double **getYScaleFactorInv** (std::size_t index) const
- IplImageWrapper & **getImage** (std::size_t index)
- const IplImageWrapper & **getImage** (std::size_t index) const
- IplImageWrapper getImage (double scaleFactor, int round=0)
- const IplImageWrapper & **getImage** (double scaleFactor, int round=0) const
- void rebuild (IplImageWrapper image)
- **IplImagePyramid** (const IplImagePyramid &pyramid)
- IplImagePyramid (IplImageWrapper image, double scaleFactor)
- IplImagePyramid (CvSize initSize, int depth, int nChannels, double scaleFactor)
- IplImagePyramid & **operator=** (const IplImagePyramid &pyramid)
- **operator const bool** () const
- **operator bool** ()
- std::size_t **numOfLevels** () const
- double **getScaleFactor** () const
- std::size_t getIndex (double scaleFactor, int round=0) const
- double **getScaleFactor** (std::size_t index) const
- double **getScaleFactorInv** (std::size_t index) const
- double **getXScaleFactor** (std::size_t index) const
- double **getXScaleFactorInv** (std::size_t index) const
- double **getYScaleFactor** (std::size_t index) const
- double **getYScaleFactorInv** (std::size_t index) const
- IplImageWrapper & **getImage** (std::size_t index)

- const IplImageWrapper & **getImage** (std::size_t index) const
- IplImageWrapper getImage (double scaleFactor, int round=0)
- const IplImageWrapper & **getImage** (double scaleFactor, int round=0) const
- void rebuild (IplImageWrapper image)

Protected Attributes

- std::vector< IplImageWrapper > **_imagePyramid**
- std::vector< double > **_scaleFactors**
- std::vector< double > **_scaleFactorsInv**
- std::vector< double > **_xScaleFactors**
- std::vector< double > **_xScaleFactorsInv**
- std::vector< double > **_yScaleFactors**
- std::vector< double > **_yScaleFactorsInv**
- double **_scaleFactor**
- double **_epsilon**

4.8.1 Detailed Description

Definition at line 19 of file include/densetrack/IplImagePyramid.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 IplImagePyramid::IplImagePyramid (IplImageWrapper *image*, double *scaleFactor*)

NOTE: the image is referenced on the lowest pyramid level!

Definition at line 219 of file lib/densetrack/IplImagePyramid.cpp.

4.8.2.2 IplImagePyramid::IplImagePyramid (CvSize *initSize*, int *depth*, int *nChannels*, double *scaleFactor*)

Build an empty pyramid (pixel values are set to zero).

Definition at line 226 of file lib/densetrack/IplImagePyramid.cpp.

4.8.2.3 IplImagePyramid::IplImagePyramid (IplImageWrapper *image*, double *scaleFactor*)

NOTE: the image is referenced on the lowest pyramid level!

4.8.2.4 IplImagePyramid::IplImagePyramid (CvSize *initSize*, int *depth*, int *nChannels*, double *scaleFactor*)

Build an empty pyramid (pixel values are set to zero).

4.8.2.5 IplImagePyramid::IplImagePyramid (IplImageWrapper *image*, double *scaleFactor*)

NOTE: the image is referenced on the lowest pyramid level!

4.8.2.6 IplImagePyramid::IplImagePyramid (CvSize *initSize*, int *depth*, int *nChannels*, double *scaleFactor*)

Build an empty pyramid (pixel values are set to zero).

4.8.2.7 IplImagePyramid::IplImagePyramid (IplImageWrapper *image*, double *scaleFactor*)

NOTE: the image is referenced on the lowest pyramid level!

4.8.2.8 IplImagePyramid::IplImagePyramid (CvSize *initSize*, int *depth*, int *nChannels*, double *scaleFactor*)

Build an empty pyramid (pixel values are set to zero).

4.8.2.9 IplImagePyramid::IplImagePyramid (IplImageWrapper *image*, double *scaleFactor*)

NOTE: the image is referenced on the lowest pyramid level!

4.8.2.10 IplImagePyramid::IplImagePyramid (CvSize *initSize*, int *depth*, int *nChannels*, double *scaleFactor*)

Build an empty pyramid (pixel values are set to zero).

4.8.3 Member Function Documentation

4.8.3.1 IplImageWrapper IplImagePyramid::getImage (double *scaleFactor*, int *round* = 0)

Parameters

round see getIndex()

4.8.3.2 IplImageWrapper IplImagePyramid::getImage (double *scaleFactor*, int *round* = 0)

Parameters

round see getIndex()

4.8.3.3 IplImageWrapper IplImagePyramid::getImage (double *scaleFactor*, int *round* = 0)

Parameters

round see getIndex()

4.8.3.4 IplImageWrapper IplImagePyramid::getImage (double *scaleFactor*, int *round* = 0)

Parameters

round see getIndex()

4.8.3.5 IplImageWrapper IplImagePyramid::getImage (double *scaleFactor*, int *round* = 0)

Parameters

round see getIndex()

Definition at line 334 of file lib/densetrack/IplImagePyramid.cpp.

4.8.3.6 std::size_t IplImagePyramid::getIndex (double *scaleFactor*, int *round* = 0) const

round == 0 => take the closest level (i.e., rounding) round < 0 => take the next level with a smaller factor (i.e., flooring) round > 0 => take the next level with a bigger factor (i.e., ceiling)

4.8.3.7 std::size_t IplImagePyramid::getIndex (double *scaleFactor*, int *round* = 0) const

round == 0 => take the closest level (i.e., rounding) round < 0 => take the next level with a smaller factor (i.e., flooring) round > 0 => take the next level with a bigger factor (i.e., ceiling)

4.8.3.8 std::size_t IplImagePyramid::getIndex (double *scaleFactor*, int *round* = 0) const

round == 0 => take the closest level (i.e., rounding) round < 0 => take the next level with a smaller factor (i.e., flooring) round > 0 => take the next level with a bigger factor (i.e., ceiling)

4.8.3.9 std::size_t IplImagePyramid::getIndex (double *scaleFactor*, int *round* = 0) const

round == 0 => take the closest level (i.e., rounding) round < 0 => take the next level with a smaller factor (i.e., flooring) round > 0 => take the next level with a bigger factor (i.e., ceiling)

4.8.3.10 std::size_t IplImagePyramid::getIndex (double *scaleFactor*, int *round* = 0) const

round == 0 => take the closest level (i.e., rounding) round < 0 => take the next level with a smaller factor (i.e., flooring) round > 0 => take the next level with a bigger factor (i.e., ceiling)

Definition at line 150 of file lib/densetrack/IplImagePyramid.cpp.

4.8.3.11 void IplImagePyramid::rebuild (IplImageWrapper *image*)

rebuilds the pyramid (re-using the already allocated space) with the given image NOTE: this image needs to have the exact sames size as the initial scale

4.8.3.12 void IplImagePyramid::rebuild (IplImageWrapper *image*)

rebuilds the pyramid (re-using the already allocated space) with the given image NOTE: this image needs to have the exact sames size as the initial scale

4.8.3.13 void IplImagePyramid::rebuild (IplImageWrapper *image*)

rebuilds the pyramid (re-using the already allocated space) with the given image NOTE: this image needs to have the exact sames size as the initial scale

4.8.3.14 void IplImagePyramid::rebuild (IplImageWrapper *image*)

rebuilds the pyramid (re-using the already allocated space) with the given image NOTE: this image needs to have the exact sames size as the initial scale

4.8.3.15 void IplImagePyramid::rebuild (IplImageWrapper *image*)

rebuilds the pyramid (re-using the already allocated space) with the given image NOTE: this image needs to have the exact sames size as the initial scale

Definition at line 190 of file lib/densetrack/IplImagePyramid.cpp.

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

- include/densetrack/IplImagePyramid.h
- include/IplImagePyramid.h
- integration/densetrack/IplImagePyramid.h
- lib/densetrack/IplImagePyramid.h
- modules/densetrack/IplImagePyramid.h
- lib/densetrack/IplImagePyramid.cpp
- modules/densetrack/IplImagePyramid.cpp
- src/IplImagePyramid.cpp

4.9 IplImageWrapper Class Reference

`#include <IplImageWrapper.h>`

Public Member Functions

- **IplImageWrapper** (IplImage *newImg=NULL, bool isOwner=true)
- **IplImageWrapper** (CvSize size, int depth, int channels)
- **IplImageWrapper** (std::string fileName)
- **IplImageWrapper** (const IplImageWrapper &newImg)
- IplImageWrapper **clone** () const
- IplImageWrapper & **operator=** (const IplImageWrapper &img)
- **operator IplImage** * ()
- **operator const IplImage** * () const
- **operator const bool** () const
- **operator bool** ()
- IplImage * **operator->** ()
- const IplImage * **operator->** () const
- IplImage * **getReference** ()
- const IplImage * **getReference** () const
- std::size_t **numOfReferences** () const
- bool **hasMask** () const
- Box< int > **getMask** () const
- void **setMask** (const Box< int > &mask)
- void **clearMask** ()
- **IplImageWrapper** (IplImage *newImg=NULL, bool isOwner=true)
- **IplImageWrapper** (CvSize size, int depth, int channels)
- **IplImageWrapper** (std::string fileName)
- **IplImageWrapper** (const IplImageWrapper &newImg)
- IplImageWrapper **clone** () const
- IplImageWrapper & **operator=** (const IplImageWrapper &img)
- **operator IplImage** * ()
- **operator const IplImage** * () const
- **operator const bool** () const
- **operator bool** ()
- IplImage * **operator->** ()
- const IplImage * **operator->** () const
- IplImage * **getReference** ()
- const IplImage * **getReference** () const
- std::size_t **numOfReferences** () const
- bool **hasMask** () const
- Box< int > **getMask** () const
- void **setMask** (const Box< int > &mask)
- void **clearMask** ()
- **IplImageWrapper** (IplImage *newImg=NULL, bool isOwner=true)
- **IplImageWrapper** (CvSize size, int depth, int channels)
- **IplImageWrapper** (std::string fileName)
- **IplImageWrapper** (const IplImageWrapper &newImg)
- IplImageWrapper **clone** () const

- IplImageWrapper & **operator=** (const IplImageWrapper &img)
- **operator IplImage** * ()
- **operator const IplImage** * () const
- **operator const bool** () const
- **operator bool** ()
- IplImage * **operator->** ()
- const IplImage * **operator->** () const
- IplImage * **getReference** ()
- const IplImage * **getReference** () const
- std::size_t **numOfReferences** () const
- bool **hasMask** () const
- Box< int > **getMask** () const
- void **setMask** (const Box< int > &mask)
- void **clearMask** ()
- **IplImageWrapper** (IplImage *newImg=NULL, bool isOwner=true)
- **IplImageWrapper** (CvSize size, int depth, int channels)
- **IplImageWrapper** (std::string fileName)
- **IplImageWrapper** (const IplImageWrapper &newImg)
- IplImageWrapper **clone** () const
- IplImageWrapper & **operator=** (const IplImageWrapper &img)
- **operator IplImage** * ()
- **operator const IplImage** * () const
- **operator const bool** () const
- **operator bool** ()
- IplImage * **operator->** ()
- const IplImage * **operator->** () const
- IplImage * **getReference** ()
- const IplImage * **getReference** () const
- std::size_t **numOfReferences** () const
- bool **hasMask** () const
- Box< int > **getMask** () const
- void **setMask** (const Box< int > &mask)
- void **clearMask** ()
- **IplImageWrapper** (IplImage *newImg=NULL, bool isOwner=true)
- **IplImageWrapper** (CvSize size, int depth, int channels)
- **IplImageWrapper** (std::string fileName)
- **IplImageWrapper** (const IplImageWrapper &newImg)
- IplImageWrapper **clone** () const
- IplImageWrapper & **operator=** (const IplImageWrapper &img)
- **operator IplImage** * ()
- **operator const IplImage** * () const
- **operator const bool** () const
- **operator bool** ()
- IplImage * **operator->** ()
- const IplImage * **operator->** () const
- IplImage * **getReference** ()
- const IplImage * **getReference** () const
- std::size_t **numOfReferences** () const
- bool **hasMask** () const
- Box< int > **getMask** () const
- void **setMask** (const Box< int > &mask)
- void **clearMask** ()

Protected Member Functions

- void **decrementAndFree** ()
- void **decrementAndFree** ()
- void **decrementAndFree** ()
- void **decrementAndFree** ()
- void **decrementAndFree** ()

Protected Attributes

- IplImage * **_img**
- std::size_t * **_nRefs**
- boost::optional< Box< int > > **_mask**

4.9.1 Detailed Description

Wrapper in order to use IplImages more easily in STL containers .. for that we need a destructor that releases an image correctly using cvReleaseImage(). We also want to have smart pointers with a reference count in order to copy containers.

NOTE: Do not use generic algorithms with this class! (See discussions on auto_ptr, similar thoughts/problems apply to this smart pointer!)

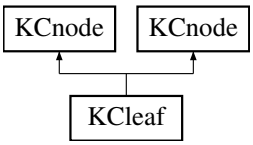
Definition at line 25 of file include/densetrack/IplImageWrapper.h.

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

- include/densetrack/IplImageWrapper.h
- include/IplImageWrapper.h
- integration/densetrack/IplImageWrapper.h
- lib/densetrack/IplImageWrapper.h
- modules/densetrack/IplImageWrapper.h
- lib/densetrack/IplImageWrapper.cpp
- modules/densetrack/IplImageWrapper.cpp
- src/IplImageWrapper.cpp

4.10 KCleaf Class Reference

Inheritance diagram for KCleaf:



Public Member Functions

- **KCleaf** (int dim, KMorthRect &bb, int n, KMdatIdxArray b)
- KMpoint **getPoint** ()
- virtual void **makeSums** (int &n, KMpoint &theSum, double &theSumSq)
- virtual void **getNeighbors** (KMctrIdxArray cands, int kCands)
- virtual void **getAssignments** (KMctrIdxArray cands, int kCands, KMctrIdxArray closeCtr, double ∗sqDist)
- virtual void **sampleCtr** (KMpoint c, KMorthRect &bb)
- virtual void **print** (int level)
- **KCleaf** (int dim, KMorthRect &bb, int n, KMdatIdxArray b)
- KMpoint **getPoint** ()
- virtual void **makeSums** (int &n, KMpoint &theSum, double &theSumSq)
- virtual void **getNeighbors** (KMctrIdxArray cands, int kCands)
- virtual void **getAssignments** (KMctrIdxArray cands, int kCands, KMctrIdxArray closeCtr, double ∗sqDist)
- virtual void **sampleCtr** (KMpoint c, KMorthRect &bb)
- virtual void **print** (int level)

Protected Attributes

- KMidxArray **bkt**

4.10.1 Detailed Description

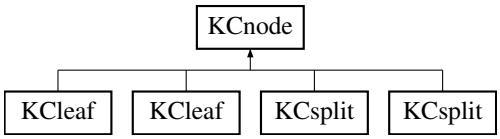
Definition at line 192 of file include/kmlocal/KCtree.h.

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

- include/kmlocal/KCtree.h
- lib/kmlocal-1.7.2/src/KCtree.h
- KCtree.cpp

4.11 KCnode Class Reference

Inheritance diagram for KCnode:



Public Member Functions

- **KCnode** (int dim, KMorthRect &bb)
- void **cellMidpt** (KMpoint pt)
- KMorthRect & **bndBox** ()
- virtual void **makeSums** (int &n, KMpoint &theSum, double &theSumSq)=0
- virtual void **getNeighbors** (KMctrIdxArray cands, int kCands)=0
- virtual void **getAssignments** (KMctrIdxArray cands, int kCands, KMctrIdxArray closeCtr, double ∗sqDist)=0
- virtual void **sampleCtr** (KMpoint c, KMorthRect &bb)=0
- virtual void **print** (int level)=0
- int **n_nodes** ()
- **KCnode** (int dim, KMorthRect &bb)
- void **cellMidpt** (KMpoint pt)
- KMorthRect & **bndBox** ()
- virtual void **makeSums** (int &n, KMpoint &theSum, double &theSumSq)=0
- virtual void **getNeighbors** (KMctrIdxArray cands, int kCands)=0
- virtual void **getAssignments** (KMctrIdxArray cands, int kCands, KMctrIdxArray closeCtr, double ∗sqDist)=0
- virtual void **sampleCtr** (KMpoint c, KMorthRect &bb)=0
- virtual void **print** (int level)=0
- int **n_nodes** ()

Protected Attributes

- const int **multCand**
- int **n_data**
- KMpoint **sum**
- double **sumSq**
- KMorthRect **bnd_box**

Friends

- class KCtree

4.11.1 Detailed Description

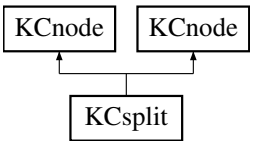
Definition at line 136 of file include/kmlocal/KCtree.h.

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

- include/kmlocal/KCtree.h
- lib/kmlocal-1.7.2/src/KCtree.h
- KCtree.cpp

4.12 KCsplit Class Reference

Inheritance diagram for KCsplit:



Public Member Functions

- **KCsplit** (int dim, KMorthRect &bb, int cd, KMcoord cv, KMcoord lv, KMcoord hv, KCptr lc=NULL, KCptr hc=NULL)
- virtual void **makeSums** (int &n, KMpoint &theSum, double &theSumSq)
- virtual void **getNeighbors** (KMctrIdxArray cands, int kCands)
- virtual void **getAssignments** (KMctrIdxArray cands, int kCands, KMctrIdxArray closeCtr, double *sqDist)
- virtual void **sampleCtr** (KMpoint c, KMorthRect &bb)
- virtual void **print** (int level)
- **KCsplit** (int dim, KMorthRect &bb, int cd, KMcoord cv, KMcoord lv, KMcoord hv, KCptr lc=NULL, KCptr hc=NULL)
- virtual void **makeSums** (int &n, KMpoint &theSum, double &theSumSq)
- virtual void **getNeighbors** (KMctrIdxArray cands, int kCands)
- virtual void **getAssignments** (KMctrIdxArray cands, int kCands, KMctrIdxArray closeCtr, double *sqDist)
- virtual void **sampleCtr** (KMpoint c, KMorthRect &bb)
- virtual void **print** (int level)

Protected Attributes

- int **cut_dim**
- KMcoord **cut_val**
- KMcoord **cd_bnds** [2]
- KCptr **child** [2]

4.12.1 Detailed Description

Definition at line 242 of file include/kmlocal/KCtree.h.

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

- include/kmlocal/KCtree.h
- lib/kmlocal-1.7.2/src/KCtree.h
- KCtree.cpp

4.13 KCtree Class Reference

Public Member Functions

- **KCtree** (KMdataArray pa, int n, int dd, int n_max=0, KMpoint bb_lo=NULL, KMpoint bb_-hi=NULL)
- void **getNeighbors** (KMfilterCenters &ctrs)
- void **getAssignments** (KMfilterCenters &ctrs, KMctrIdxArray closeCtr, double ∗sqDist)
- void **sampleCtr** (KMpoint c)
- void **print** (bool with_pts)
- **KCtree** (KMdataArray pa, int n, int dd, int n_max=0, KMpoint bb_lo=NULL, KMpoint bb_-hi=NULL)
- void **getNeighbors** (KMfilterCenters &ctrs)
- void **getAssignments** (KMfilterCenters &ctrs, KMctrIdxArray closeCtr, double ∗sqDist)
- void **sampleCtr** (KMpoint c)
- void **print** (bool with_pts)

Protected Member Functions

- void **skeletonTree** (KMdataArray pa, int n, int dd, int n_max, KMpoint bb_lo, KMpoint bb_hi, KMdatIdxArray pi)
- KCptr **buildKcTree** (KMdataArray pa, KMdatIdxArray pidx, int n, int dim, KMorthRect &bnd_-box)
- void **skeletonTree** (KMdataArray pa, int n, int dd, int n_max, KMpoint bb_lo, KMpoint bb_hi, KMdatIdxArray pi)
- KCptr **buildKcTree** (KMdataArray pa, KMdatIdxArray pidx, int n, int dim, KMorthRect &bnd_-box)

Protected Attributes

- int **dim**
- int **n_pts**
- int **max_pts**
- KMdataArray **pts**
- KMdatIdxArray **pidx**
- KCptr **root**
- KMorthRect **bnd_box**

4.13.1 Detailed Description

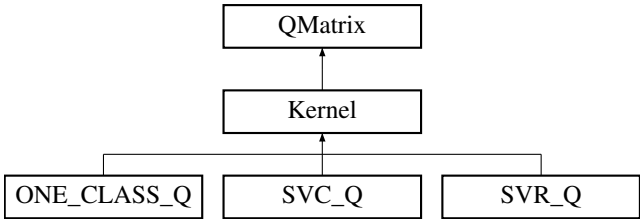
Definition at line 56 of file include/kmlocal/KCtree.h.

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

- include/kmlocal/KCtree.h
- lib/kmlocal-1.7.2/src/KCtree.h
- KCtree.cpp

4.14 Kernel Class Reference

Inheritance diagram for Kernel:



Public Member Functions

- **Kernel** (int l, svm_node *const *x, const svm_parameter &param)
- virtual Qfloat * **get_Q** (int column, int len) const =0
- virtual double * **get_QD** () const =0
- virtual void **swap_index** (int i, int j) const

Static Public Member Functions

- static double **k_function** (const svm_node *x, const svm_node *y, const svm_parameter &param)

Protected Attributes

- double(Kernel::* **kernel_function**)(int i, int j) const

4.14.1 Detailed Description

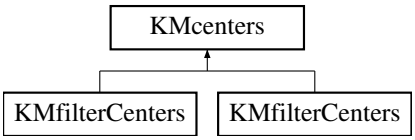
Definition at line 202 of file svm.cpp.

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

- svm.cpp

4.15 KMcenters Class Reference

Inheritance diagram for KMcenters:



Public Member Functions

- **KMcenters** (int k, KMdata &p)
- **KMcenters** (const KMcenters &s)
- KMcenters & **operator=** (const KMcenters &s)
- int **getDim** () const
- int **getNPts** () const
- int **getK** () const
- KMdata & **getData** ()
- KMpointArray **getDataPts** () const
- KMcenterArray **getCtrPts** () const
- KMcenter & **operator[ ]** (int i)
- const KMcenter & **operator[ ]** (int i) const
- void **resize** (int k)
- virtual void **print** (bool fancy=true)
- **KMcenters** (int k, KMdata &p)
- **KMcenters** (const KMcenters &s)
- KMcenters & **operator=** (const KMcenters &s)
- int **getDim** () const
- int **getNPts** () const
- int **getK** () const
- KMdata & **getData** ()
- KMpointArray **getDataPts** () const
- KMcenterArray **getCtrPts** () const
- KMcenter & **operator[ ]** (int i)
- const KMcenter & **operator[ ]** (int i) const
- void **resize** (int k)
- virtual void **print** (bool fancy=true)

Protected Attributes

- int **kCtrs**
- KMdata * **pts**
- KMcenterArray **ctrs**

4.15.1 Detailed Description

Definition at line 36 of file include/kmlocal/KMcenters.h.

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

- include/kmlocal/KMcenters.h
- lib/kmlocal-1.7.2/src/KMcenters.h
- KMcenters.cpp

4.16 KMdata Class Reference

Public Member Functions

- **KMdata** (int d, int n)
- int **getDim** () const
- int **getNPts** () const
- KMdataArray **getPts** () const
- KCtree * **getKcTree** () const
- KMdataPoint & **operator[ ]** (int i)
- const KMdataPoint & **operator[ ]** (int i) const
- void **setNPts** (int n)
- void **buildKcTree** ()
- virtual void **sampleCtr** (KMpoint sample)
- virtual void **sampleCtrs** (KMpointArray sample, int k, bool allowDuplicate)
- void **resize** (int d, int n)
- void **print** (bool fancy=true)
- **KMdata** (int d, int n)
- int **getDim** () const
- int **getNPts** () const
- KMdataArray **getPts** () const
- KCtree * **getKcTree** () const
- KMdataPoint & **operator[ ]** (int i)
- const KMdataPoint & **operator[ ]** (int i) const
- void **setNPts** (int n)
- void **buildKcTree** ()
- virtual void **sampleCtr** (KMpoint sample)
- virtual void **sampleCtrs** (KMpointArray sample, int k, bool allowDuplicate)
- void **resize** (int d, int n)
- void **print** (bool fancy=true)

4.16.1 Detailed Description

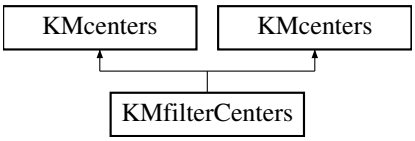
Definition at line 49 of file include/kmlocal/KMdata.h.

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

- include/kmlocal/KMdata.h
- lib/kmlocal-1.7.2/src/KMdata.h
- KMdata.cpp

4.17 KMfilterCenters Class Reference

Inheritance diagram for KMfilterCenters:



Public Member Functions

- **KMfilterCenters** (int k, KMdata &p, double df=1)
- **KMfilterCenters** (const KMfilterCenters &s)
- KMfilterCenters & **operator=** (const KMfilterCenters &s)
- KMpointArray **getSums** (bool autoUpdate=true)
- double * **getSumSqs** (bool autoUpdate=true)
- int * **getWeights** (bool autoUpdate=true)
- double **getDist** (bool autoUpdate=true)
- double **getAvgDist** (bool autoUpdate=true)
- double * **getDists** (bool autoUpdate=true)
- void **getAssignments** (KMctrIdxArray closeCtr, double *sqDist)
- void **genRandom** ()
- void **lloyd1Stage** ()
- void **swap1Stage** ()
- virtual void **print** (bool fancy=true)
- **KMfilterCenters** (int k, KMdata &p, double df=1)
- **KMfilterCenters** (const KMfilterCenters &s)
- KMfilterCenters & **operator=** (const KMfilterCenters &s)
- KMpointArray **getSums** (bool autoUpdate=true)
- double * **getSumSqs** (bool autoUpdate=true)
- int * **getWeights** (bool autoUpdate=true)
- double **getDist** (bool autoUpdate=true)
- double **getAvgDist** (bool autoUpdate=true)
- double * **getDists** (bool autoUpdate=true)
- void **getAssignments** (KMctrIdxArray closeCtr, double *sqDist)
- void **genRandom** ()
- void **lloyd1Stage** ()
- void **swap1Stage** ()
- virtual void **print** (bool fancy=true)

Protected Member Functions

- void **computeDistortion** ()
- void **moveToCentroid** ()
- void **swapOneCenter** (bool allowDuplicate=true)
- void **validate** ()
- void **invalidate** ()
- void **computeDistortion** ()

- void **moveToCentroid** ()
- void **swapOneCenter** (bool allowDuplicate=true)
- void **validate** ()
- void **invalidate** ()

Protected Attributes

- KMpointArray **sums**
- double * **sumSqs**
- int * **weights**
- double * **dists**
- double **currDist**
- bool **valid**
- double **dampFactor**

4.17.1 Detailed Description

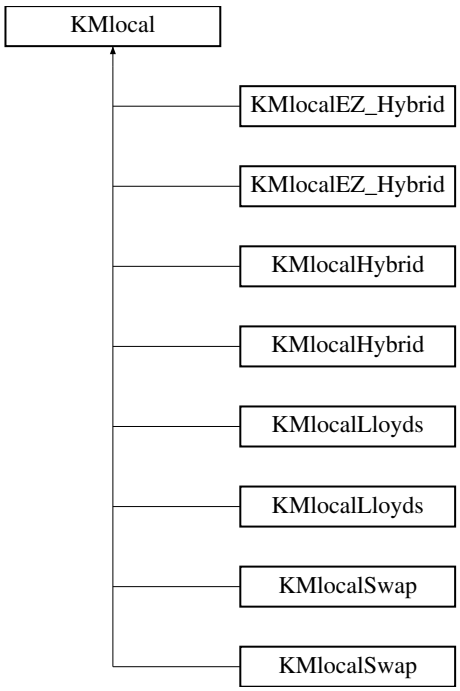
Definition at line 105 of file include/kmlocal/KMfilterCenters.h.

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

- include/kmlocal/KMfilterCenters.h
- lib/kmlocal-1.7.2/src/KMfilterCenters.h
- KMfilterCenters.cpp

4.18 KMlocal Class Reference

Inheritance diagram for KMlocal:



Public Member Functions

- **KMlocal** (const KMfilterCenters &sol, const KMterm &t)
- virtual KMfilterCenters **execute** ()
- int **getTotalStages** () const
- **KMlocal** (const KMfilterCenters &sol, const KMterm &t)
- virtual KMfilterCenters **execute** ()
- int **getTotalStages** () const

Protected Member Functions

- virtual void **printStageStats** ()
- virtual void **reset** ()
- virtual bool **isDone** () const
- virtual void **beginRun** ()
- virtual void **beginStage** ()
- virtual KMalg **selectMethod** ()=0
- virtual void **endStage** ()
- virtual bool **isRunDone** ()
- virtual void **endRun** ()
- virtual void **tryAcceptance** ()
- virtual void **printStageStats** ()
- virtual void **reset** ()

- virtual bool **isDone** () const
- virtual void **beginRun** ()
- virtual void **beginStage** ()
- virtual KMalg **selectMethod** ()=0
- virtual void **endStage** ()
- virtual bool **isRunDone** ()
- virtual void **endRun** ()
- virtual void **tryAcceptance** ()

Protected Attributes

- int **nPts**
- int **kCtrs**
- int **dim**
- KMterm **term**
- int **maxTotStage**
- int **stageNo**
- int **runInitStage**
- KMfilterCenters **curr**
- KMfilterCenters **best**

4.18.1 Detailed Description

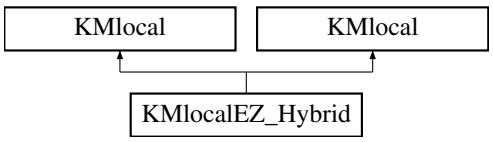
Definition at line 104 of file include/kmlocal/KMlocal.h.

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

- include/kmlocal/KMlocal.h
- lib/kmlocal-1.7.2/src/KMlocal.h
- KMlocal.cpp

4.19 KMlocalEZ_Hybrid Class Reference

Inheritance diagram for KMlocalEZ_Hybrid:



Public Member Functions

- **KMlocalEZ_Hybrid** (const KMfilterCenters &sol, const KMterm &t)
- **KMlocalEZ_Hybrid** (const KMfilterCenters &sol, const KMterm &t)

Protected Member Functions

- double **consecRDL** ()
- virtual void **printStageStats** ()
- virtual void **printRunStats** ()
- virtual void **reset** ()
- virtual void **beginStage** ()
- virtual KMalg **selectMethod** ()
- virtual void **endStage** ()
- virtual bool **isRunDone** ()
- virtual void **endRun** ()
- virtual void **tryAcceptance** ()
- double **consecRDL** ()
- virtual void **printStageStats** ()
- virtual void **printRunStats** ()
- virtual void **reset** ()
- virtual void **beginStage** ()
- virtual KMalg **selectMethod** ()
- virtual void **endStage** ()
- virtual bool **isRunDone** ()
- virtual void **endRun** ()
- virtual void **tryAcceptance** ()

4.19.1 Detailed Description

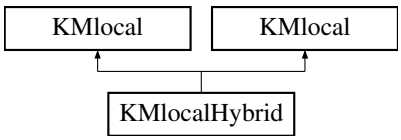
Definition at line 783 of file include/kmlocal/KMlocal.h.

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

- include/kmlocal/KMlocal.h
- lib/kmlocal-1.7.2/src/KMlocal.h

4.20 KMlocalHybrid Class Reference

Inheritance diagram for KMlocalHybrid:



Public Member Functions

- **KMlocalHybrid** (const KMfilterCenters &sol, const KMterm &t)
- **KMlocalHybrid** (const KMfilterCenters &sol, const KMterm &t)

Protected Member Functions

- double **accumRDL** ()
- double **consecRDL** ()
- virtual void **printStageStats** ()
- virtual void **printRunStats** ()
- int **nTrials** ()
- bool **simAnnealAccept** (double rdl)
- void **initTempRuns** ()
- bool **isTempRunDone** ()
- void **endTempRun** ()
- virtual void **reset** ()
- virtual void **beginStage** ()
- virtual KMalg **selectMethod** ()
- virtual void **endStage** ()
- virtual bool **isRunDone** ()
- virtual void **endRun** ()
- virtual void **tryAcceptance** ()
- double **accumRDL** ()
- double **consecRDL** ()
- virtual void **printStageStats** ()
- virtual void **printRunStats** ()
- int **nTrials** ()
- bool **simAnnealAccept** (double rdl)
- void **initTempRuns** ()
- bool **isTempRunDone** ()
- void **endTempRun** ()
- virtual void **reset** ()
- virtual void **beginStage** ()
- virtual KMalg **selectMethod** ()
- virtual void **endStage** ()
- virtual bool **isRunDone** ()
- virtual void **endRun** ()
- virtual void **tryAcceptance** ()

4.20.1 Detailed Description

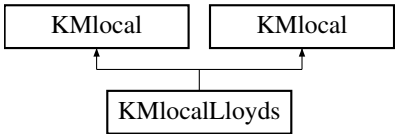
Definition at line 594 of file include/kmlocal/KMlocal.h.

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

- include/kmlocal/KMlocal.h
- lib/kmlocal-1.7.2/src/KMlocal.h

4.21 KMlocalLloyds Class Reference

Inheritance diagram for KMlocalLloyds:



Public Member Functions

- **KMlocalLloyds** (const KMfilterCenters &sol, const KMterm &t)
- **KMlocalLloyds** (const KMfilterCenters &sol, const KMterm &t)

Protected Member Functions

- double **accumRDL** ()
- virtual void **printStageStats** ()
- virtual void **printRunStats** ()
- virtual void **reset** ()
- virtual KMalg **selectMethod** ()
- virtual void **endStage** ()
- virtual bool **isRunDone** ()
- virtual void **endRun** ()
- double **accumRDL** ()
- virtual void **printStageStats** ()
- virtual void **printRunStats** ()
- virtual void **reset** ()
- virtual KMalg **selectMethod** ()
- virtual void **endStage** ()
- virtual bool **isRunDone** ()
- virtual void **endRun** ()

4.21.1 Detailed Description

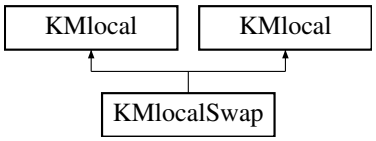
Definition at line 263 of file include/kmlocal/KMlocal.h.

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

- include/kmlocal/KMlocal.h
- lib/kmlocal-1.7.2/src/KMlocal.h

4.22 KMlocalSwap Class Reference

Inheritance diagram for KMlocalSwap:



Public Member Functions

- **KMlocalSwap** (const KMfilterCenters &sol, const KMterm &t, int p=1)
- **KMlocalSwap** (const KMfilterCenters &sol, const KMterm &t, int p=1)

Protected Member Functions

- virtual void **reset** ()
- virtual void **beginRun** ()
- virtual KMalg **selectMethod** ()
- virtual void **endStage** ()
- virtual bool **isRunDone** ()
- virtual void **endRun** ()
- virtual void **tryAcceptance** ()
- virtual void **reset** ()
- virtual void **beginRun** ()
- virtual KMalg **selectMethod** ()
- virtual void **endStage** ()
- virtual bool **isRunDone** ()
- virtual void **endRun** ()
- virtual void **tryAcceptance** ()

4.22.1 Detailed Description

Definition at line 363 of file include/kmlocal/KMlocal.h.

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

- include/kmlocal/KMlocal.h
- lib/kmlocal-1.7.2/src/KMlocal.h

4.23 KMorthRect Class Reference

Public Member Functions

- **KMorthRect** (int dd, KMcoord l=0, KMcoord h=0)
- **KMorthRect** (int dd, const KMorthRect &r)
- **KMorthRect** (int dd, KMpoint l, KMpoint h)
- bool **inside** (int dim, KMpoint p)
- void **expand** (int dim, double x, KMorthRect r)
- void **sample** (int dim, KMpoint p)
- **KMorthRect** (int dd, KMcoord l=0, KMcoord h=0)
- **KMorthRect** (int dd, const KMorthRect &r)
- **KMorthRect** (int dd, KMpoint l, KMpoint h)
- bool **inside** (int dim, KMpoint p)
- void **expand** (int dim, double x, KMorthRect r)
- void **sample** (int dim, KMpoint p)

Public Attributes

- KMpoint **lo**
- KMpoint **hi**

4.23.1 Detailed Description

Definition at line 380 of file include/kmlocal/KM_ANN.h.

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

- include/kmlocal/KM_ANN.h
- lib/kmlocal-1.7.2/src/KM_ANN.h
- KM_ANN.cpp

4.24 KMterm Class Reference

Public Member Functions

- **KMterm** (double a, double b, double c, double d, double mcr, double mar, int mrs, double ipa, int trl, double trf)
- void **setMaxTotStage** (int i, double val)
- void **setAbsMaxTotStage** (int s)
- int **getMaxTotStage** (int k, int n) const
- double **getMinConsecRDL** () const
- double **getMinAccumRDL** () const
- int **getMaxRunStage** () const
- void **setMinConsecRDL** (double rdl)
- void **setMinAccumRDL** (double rdl)
- void **setMaxRunStage** (int ms)
- double **getInitProbAccept** () const
- void **setInitProbAccept** (double ipa)
- int **getTempRunLength** () const
- void **setTempRunLength** (int trl)
- double **getTempReducFact** () const
- void **setTempReducFact** (double trf)
- **KMterm** (double a, double b, double c, double d, double mcr, double mar, int mrs, double ipa, int trl, double trf)
- void **setMaxTotStage** (int i, double val)
- void **setAbsMaxTotStage** (int s)
- int **getMaxTotStage** (int k, int n) const
- double **getMinConsecRDL** () const
- double **getMinAccumRDL** () const
- int **getMaxRunStage** () const
- void **setMinConsecRDL** (double rdl)
- void **setMinAccumRDL** (double rdl)
- void **setMaxRunStage** (int ms)
- double **getInitProbAccept** () const
- void **setInitProbAccept** (double ipa)
- int **getTempRunLength** () const
- void **setTempRunLength** (int trl)
- double **getTempReducFact** () const
- void **setTempReducFact** (double trf)

Protected Member Functions

- int **maxStage** (const double param[KM_TERM_VEC_LEN], int k, int n) const
- int **maxStage** (const double param[KM_TERM_VEC_LEN], int k, int n) const

4.24.1 Detailed Description

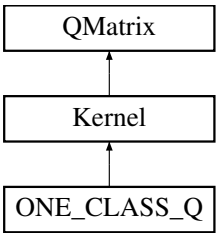
Definition at line 92 of file include/kmlocal/KMterm.h.

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

- include/kmlocal/KMterm.h
- lib/kmlocal-1.7.2/src/KMterm.h
- KMterm.cpp

4.25 ONE_CLASS_Q Class Reference

Inheritance diagram for ONE_CLASS_Q:



Public Member Functions

- **ONE_CLASS_Q** (const svm_problem &prob, const svm_parameter &param)
- Qfloat * **get_Q** (int i, int len) const
- double * **get_QD** () const
- void **swap_index** (int i, int j) const

4.25.1 Detailed Description

Definition at line 1316 of file svm.cpp.

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

- svm.cpp

4.26 Point< T > Class Template Reference

Public Types

- typedef T **ValueType**
- typedef T **ValueType**
- typedef T **ValueType**
- typedef T **ValueType**
- typedef T **ValueType**

Public Member Functions

- **Point** (T xpos, T ypos)
- bool **isNull** () const
- T **getX** () const
- T **getY** () const
- void **setX** (T x)
- void **setY** (T y)
- Point< T > & **operator+=** (const Point< T > &p)
- Point< T > & **operator-=** (const Point< T > &p)
- template<typename T2 >
  Point< T > & **operator*=** (T2 c)
- template<typename T2 >
  Point< T > & **operator/=** (T2 c)
- template<typename T2 >
  operator Point< T2 > () const
- **Point** (T xpos, T ypos)
- bool **isNull** () const
- T **getX** () const
- T **getY** () const
- void **setX** (T x)
- void **setY** (T y)
- Point< T > & **operator+=** (const Point< T > &p)
- Point< T > & **operator-=** (const Point< T > &p)
- template<typename T2 >
  Point< T > & **operator*=** (T2 c)
- template<typename T2 >
  Point< T > & **operator/=** (T2 c)
- template<typename T2 >
  operator Point< T2 > () const
- **Point** (T xpos, T ypos)
- bool **isNull** () const
- T **getX** () const
- T **getY** () const
- void **setX** (T x)
- void **setY** (T y)
- Point< T > & **operator+=** (const Point< T > &p)
- Point< T > & **operator-=** (const Point< T > &p)
- template<typename T2 >
  Point< T > & **operator*=** (T2 c)

- template<typename T2 >
  Point< T > & **operator/=** (T2 c)
- template<typename T2 >
  operator Point< T2 > () const
- **Point** (T xpos, T ypos)
- bool **isNull** () const
- T **getX** () const
- T **getY** () const
- void **setX** (T x)
- void **setY** (T y)
- Point< T > & **operator+=** (const Point< T > &p)
- Point< T > & **operator-=** (const Point< T > &p)
- template<typename T2 >
  Point< T > & **operator*=** (T2 c)
- template<typename T2 >
  Point< T > & **operator/=** (T2 c)
- template<typename T2 >
  operator Point< T2 > () const
- **Point** (T xpos, T ypos)
- bool **isNull** () const
- T **getX** () const
- T **getY** () const
- void **setX** (T x)
- void **setY** (T y)
- Point< T > & **operator+=** (const Point< T > &p)
- Point< T > & **operator-=** (const Point< T > &p)
- template<typename T2 >
  Point< T > & **operator*=** (T2 c)
- template<typename T2 >
  Point< T > & **operator/=** (T2 c)
- template<typename T2 >
  operator Point< T2 > () const

4.26.1 Detailed Description

template<typename T> class Point< T >

Definition at line 10 of file include/densetrack/Point.h.

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

- include/densetrack/Point.h
- include/Point.h
- integration/densetrack/Point.h
- lib/densetrack/Point.h
- modules/densetrack/Point.h

4.27 PointDesc Class Reference

Public Member Functions

- **PointDesc** (const DescInfo &hogInfo, const DescInfo &hofInfo, const DescInfo &mbhInfo, const CvPoint2D32f &point_)
- **PointDesc** (const DescInfo &hogInfo, const DescInfo &hofInfo, const DescInfo &mbhInfo, const CvPoint2D32f &point_)
- **PointDesc** (const DescInfo &hogInfo, const DescInfo &hofInfo, const DescInfo &mbhInfo, const CvPoint2D32f &point_)
- **PointDesc** (const DescInfo &hogInfo, const DescInfo &hofInfo, const DescInfo &mbhInfo, const CvPoint2D32f &point_)

Public Attributes

- std::vector< float > **hog**
- std::vector< float > **hof**
- std::vector< float > **mbhX**
- std::vector< float > **mbhY**
- CvPoint2D32f **point**

4.27.1 Detailed Description

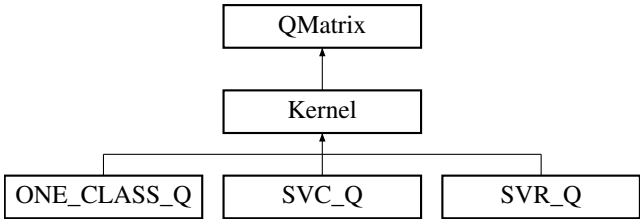
Definition at line 53 of file include/denseTrack.h.

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

- include/denseTrack.h
- naodensetrack.h
- DenseTrack.h
- modules/densetrack/denseTrack.h

4.28 QMatrix Class Reference

Inheritance diagram for QMatrix:



Public Member Functions

- virtual Qfloat * **get_Q** (int column, int len) const =0
- virtual double * **get_QD** () const =0
- virtual void **swap_index** (int i, int j) const =0

4.28.1 Detailed Description

Definition at line 194 of file svm.cpp.

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

- svm.cpp

4.29 Size< T > Class Template Reference

Public Types

- typedef T **ValueType**
- typedef T **ValueType**
- typedef T **ValueType**
- typedef T **ValueType**
- typedef T **ValueType**

Public Member Functions

- **Size** (T w, T h)
- bool **isEmpty** () const
- bool **isValid** () const
- T **getWidth** () const
- T **getHeight** () const
- void **setWidth** (T w)
- void **setHeight** (T h)
- void **transpose** ()
- T **getArea** () const
- Size< T > & **operator+=** (const Size< T > &)
- Size< T > & **operator-=** (const Size< T > &)
- template<typename T2 >
  Size< T > & **operator*=** (T2 c)
- template<typename T2 >
  Size< T > & **operator/=** (T2 c)
- template<typename T2 >
  operator Size< T2 > () const
- **Size** (T w, T h)
- bool **isEmpty** () const
- bool **isValid** () const
- T **getWidth** () const
- T **getHeight** () const
- void **setWidth** (T w)
- void **setHeight** (T h)
- void **transpose** ()
- T **getArea** () const
- Size< T > & **operator+=** (const Size< T > &)
- Size< T > & **operator-=** (const Size< T > &)
- template<typename T2 >
  Size< T > & **operator*=** (T2 c)
- template<typename T2 >
  Size< T > & **operator/=** (T2 c)
- template<typename T2 >
  operator Size< T2 > () const
- **Size** (T w, T h)
- bool **isEmpty** () const
- bool **isValid** () const
- T **getWidth** () const

- T **getHeight** () const
- void **setWidth** (T w)
- void **setHeight** (T h)
- void **transpose** ()
- T **getArea** () const
- Size< T > & **operator+=** (const Size< T > &)
- Size< T > & **operator-=** (const Size< T > &)
- template<typename T2 >
  Size< T > & **operator∗=** (T2 c)
- template<typename T2 >
  Size< T > & **operator/=** (T2 c)
- template<typename T2 >
  operator Size< T2 > () const
- **Size** (T w, T h)
- bool **isEmpty** () const
- bool **isValid** () const
- T **getWidth** () const
- T **getHeight** () const
- void **setWidth** (T w)
- void **setHeight** (T h)
- void **transpose** ()
- T **getArea** () const
- Size< T > & **operator+=** (const Size< T > &)
- Size< T > & **operator-=** (const Size< T > &)
- template<typename T2 >
  Size< T > & **operator∗=** (T2 c)
- template<typename T2 >
  Size< T > & **operator/=** (T2 c)
- template<typename T2 >
  operator Size< T2 > () const
- **Size** (T w, T h)
- bool **isEmpty** () const
- bool **isValid** () const
- T **getWidth** () const
- T **getHeight** () const
- void **setWidth** (T w)
- void **setHeight** (T h)
- void **transpose** ()
- T **getArea** () const
- Size< T > & **operator+=** (const Size< T > &)
- Size< T > & **operator-=** (const Size< T > &)
- template<typename T2 >
  Size< T > & **operator∗=** (T2 c)
- template<typename T2 >
  Size< T > & **operator/=** (T2 c)
- template<typename T2 >
  operator Size< T2 > () const

4.29.1 Detailed Description

template<typename T = double> class Size< T >

Definition at line 11 of file include/densetrack/Size.h.

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

- include/densetrack/Size.h
- include/Size.h
- integration/densetrack/Size.h
- lib/densetrack/Size.h
- modules/densetrack/Size.h

4.30 Solver::SolutionInfo Struct Reference

Public Attributes

- double **obj**
- double **rho**
- double **upper_bound_p**
- double **upper_bound_n**
- double **r**

4.30.1 Detailed Description

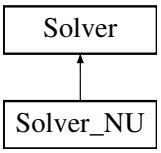
Definition at line 398 of file svm.cpp.

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

- svm.cpp

4.31 Solver Class Reference

Inheritance diagram for Solver:



Classes

- struct SolutionInfo

Public Member Functions

- void **Solve** (int l, const QMatrix &Q, const double *p_, const schar *y_, double *alpha_, double Cp, double Cn, double eps, SolutionInfo *si, int shrinking)

Protected Types

- enum { **LOWER_BOUND**, **UPPER_BOUND**, **FREE** }

Protected Member Functions

- double **get_C** (int i)
- void **update_alpha_status** (int i)
- bool **is_upper_bound** (int i)
- bool **is_lower_bound** (int i)
- bool **is_free** (int i)
- void **swap_index** (int i, int j)
- void **reconstruct_gradient** ()
- virtual int **select_working_set** (int &i, int &j)
- virtual double **calculate_rho** ()
- virtual void **do_shrinking** ()

Protected Attributes

- int **active_size**
- schar * **y**
- double * **G**
- char * **alpha_status**
- double * **alpha**
- const QMatrix * **Q**
- const double * **QD**
- double **eps**
- double **Cp**

- double **Cn**
- double ∗ **p**
- int ∗ **active_set**
- double ∗ **G_bar**
- int **l**
- bool **unshrink**

4.31.1 Detailed Description

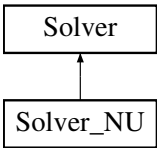
Definition at line 393 of file svm.cpp.

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

- svm.cpp

4.32 Solver_NU Class Reference

Inheritance diagram for Solver_NU:



Public Member Functions

- void **Solve** (int l, const QMatrix &Q, const double *p, const schar *y, double *alpha, double Cp, double Cn, double eps, SolutionInfo *si, int shrinking)

4.32.1 Detailed Description

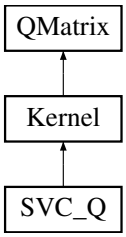
Definition at line 1009 of file svm.cpp.

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

- svm.cpp

4.33 SVC_Q Class Reference

Inheritance diagram for SVC_Q:



Public Member Functions

- **SVC_Q** (const svm_problem &prob, const svm_parameter &param, const schar *y_)
- Qfloat * **get_Q** (int i, int len) const
- double * **get_QD** () const
- void **swap_index** (int i, int j) const

4.33.1 Detailed Description

Definition at line 1266 of file svm.cpp.

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

- svm.cpp

4.34 svm_model Struct Reference

Public Attributes

- struct svm_parameter **param**
- int **nr_class**
- int **l**
- struct svm_node ** **SV**
- double ** **sv_coef**
- double * **rho**
- double * **probA**
- double * **probB**
- int * **sv_indices**
- int * **label**
- int * **nSV**
- int **free_sv**

4.34.1 Detailed Description

Definition at line 52 of file include/svm.h.

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

- include/svm.h
- lib/libsvm-3.17/svm.h

4.35 svm_node Struct Reference

Public Attributes

- int **index**
- double **value**

4.35.1 Detailed Description

Definition at line 12 of file include/svm.h.

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

- include/svm.h
- lib/libsvm-3.17/svm.h

4.36 svm_parameter Struct Reference

Public Attributes

- int **svm_type**
- int **kernel_type**
- int **degree**
- double **gamma**
- double **coef0**
- double **cache_size**
- double **eps**
- double **C**
- int **nr_weight**
- int * **weight_label**
- double * **weight**
- double **nu**
- double **p**
- int **shrinking**
- int **probability**

4.36.1 Detailed Description

Definition at line 28 of file include/svm.h.

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

- include/svm.h
- lib/libsvm-3.17/svm.h

4.37 svm_problem Struct Reference

Public Attributes

- int **l**
- double * **y**
- struct svm_node ** **x**

4.37.1 Detailed Description

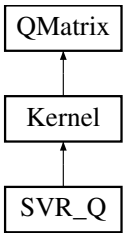
Definition at line 18 of file include/svm.h.

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

- include/svm.h
- lib/libsvm-3.17/svm.h

4.38 SVR_Q Class Reference

Inheritance diagram for SVR_Q:



Public Member Functions

- **SVR_Q** (const svm_problem &prob, const svm_parameter &param)
- void **swap_index** (int i, int j) const
- Qfloat * **get_Q** (int i, int len) const
- double * **get_QD** () const

4.38.1 Detailed Description

Definition at line 1362 of file svm.cpp.

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

- svm.cpp

4.39 Tactil Class Reference

Public Member Functions

- **Tactil** (boost::shared_ptr< AL::ALBroker > broker, const std::string &name)
- virtual void init ()
- void onFrontHeadTouched ()
- void **helloAnimation** ()

4.39.1 Detailed Description

Definition at line 32 of file tactil.h.

4.39.2 Member Function Documentation

4.39.2.1 void Tactil::init () [virtual]

Overloading ALModule::init(). This is called right after the module has been loaded

Definition at line 25 of file modules/libintegration/tactil.cpp.

4.39.2.2 void Tactil::onFrontHeadTouched ()

This method will be called every time the event FrontTactilTouched is raised.

Definition at line 50 of file modules/libintegration/tactil.cpp.

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

- tactil.h
- modules/libintegration/tactil.cpp
- src/tactil.cpp

4.40 temps_exec Struct Reference

Public Attributes

- int **debut**
- int **fin**
- sTms **sdebut**
- sTms **sfin**

4.40.1 Detailed Description

Definition at line 18 of file integration/main.c.

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

- integration/main.c

4.41 Track Class Reference

Public Member Functions

- **Track** (int maxNPoints_)
- void **addPointDesc** (const PointDesc &point)
- **Track** (int maxNPoints_)
- void **addPointDesc** (const PointDesc &point)
- **Track** (int maxNPoints_)
- void **addPointDesc** (const PointDesc &point)
- **Track** (int maxNPoints_)
- void **addPointDesc** (const PointDesc &point)

Public Attributes

- std::list< PointDesc > **pointDescs**
- int **maxNPoints**

4.41.1 Detailed Description

Definition at line 71 of file include/denseTrack.h.

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

- include/denseTrack.h
- naodensetrack.h
- DenseTrack.h
- modules/densetrack/denseTrack.h

4.42 TrackerInfo Struct Reference

Public Attributes

- int **trackLength**
- int **initGap**

4.42.1 Detailed Description

Definition at line 24 of file include/denseTrack.h.

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

- include/denseTrack.h
- naodensetrack.h
- DenseTrack.h
- modules/densetrack/denseTrack.h

Chapter 5

File Documentation

5.1 naodensetrack.cpp File Reference

Set of function permiting to extract dense points and their trajectories.

`#include "naodensetrack.h"`

Functions

- CvScalar **getRect** (const CvPoint2D32f point, const CvSize size, const DescInfo descInfo)
- void **BuildDescMat** (const IplImage *xComp, const IplImage *yComp, DescMat *descMat, const DescInfo descInfo)
- std::vector< float > **getDesc** (const DescMat *descMat, CvScalar rect, DescInfo descInfo, float epsilon)
- void **HogComp** (IplImage *img, DescMat *descMat, DescInfo descInfo)
- void **HofComp** (IplImage *flow, DescMat *descMat, DescInfo descInfo)
- void **MbhComp** (IplImage *flow, DescMat *descMatX, DescMat *descMatY, DescInfo descInfo)
- void **OpticalFlowTracker** (IplImage *flow, std::vector< CvPoint2D32f > &points_in, std::vector< CvPoint2D32f > &points_out, std::vector< int > &status)
- int **isValid** (std::vector< CvPoint2D32f > &track, float &mean_x, float &mean_y, float &var_x, float &var_y, float &length, float min_var, float max_var, float max_dis)
- void **cvDenseSample** (IplImage *grey, IplImage *eig, std::vector< CvPoint2D32f > &points, const double quality, const double min_distance)
- void **cvDenseSample** (IplImage *grey, IplImage *eig, std::vector< CvPoint2D32f > &points_in, std::vector< CvPoint2D32f > &points_out, const double quality, const double min_distance)
- void **InitTrackerInfo** (TrackerInfo *tracker, int track_length, int init_gap)
- DescMat * **InitDescMat** (int height, int width, int nBins)
- void **ReleDescMat** (DescMat *descMat)
- void **InitDescInfo** (DescInfo *descInfo, int nBins, int flag, int orientation, int size, int nxy_cell, int nt_cell, float min_flow)
- void **usage** ()
- int extractSTIPs (std::string video, int dim, int maxPts, KMdata *dataPts)

  Permits to extract STIPs from a video .avi. It save the HOG and HOG of the trajectories in the object KMdata.

5.1.1 Detailed Description

Set of function permiting to extract dense points and their trajectories.

Author

LEAR

Date

05/07/2013

Definition in file naodensetrack.cpp.

5.1.2 Function Documentation

5.1.2.1 int extractSTIPs (std::string *video*, int *dim*, int *maxPts*, KMdata * *dataPts*)

Permits to extract STIPs from a video .avi. It save the HOG and HOG of the trajectories in the object KMdata.

Parameters

← ***stip*** Name of the video.

← ***dim*** STIPs dimension.

← ***maxPts*** Maximum number of points we want to use.

→ ***dataPts*** The object in which we save the STIPs.

Returns

Number of points extracted.

Definition at line 491 of file naodensetrack.cpp.

5.2 naokmeans.cpp File Reference

Set of functions permiting to execute KMeans algorithms using KMlocal classes.

```
#include "naokmeans.h"
#include <time.h>
```

Functions

- int importSTIPs (std::string stip, int dim, int maxPts, KMdata *dataPts)

  STIPs importation function in the format 1 point = 1 line. Each dimension are separated from one space (" ").

- void **exportSTIPs** (std::string stip, int dim, const KMdata &dataPts)
- void exportCenters (std::string centers, int dim, int k, KMfilterCenters ctrs)

  Export function to save KMfilterCenters in a file. One line corresponds to one point with dim value (separeted from one space " ").

- void importCenters (std::string centers, int dim, int k, KMfilterCenters *ctrs)

  Importation function saving external centers in the KMfilterCenters object. One line corresponds to one centers with its values (separeted from one space " ").

- void kmIvanAlgorithm (int ic, int dim, const KMdata &dataPts, int k, KMfilterCenters &ctrs)

  This is an optimized KMeans algorithm. Ivan's algorithm uses basic KMeans algorithm (here the Lloyd's one) and the idea was to initialize centers intelligently.

- void createTrainingMeans (std::string stipFile, int dim, int maxPts, int k, std::string meansFile)

  Import HOG and HOF from a file and compute KMeans algorithm to create the file training.means.

5.2.1 Detailed Description

Set of functions permiting to execute KMeans algorithms using KMlocal classes.

Author

Fabien ROUALDES (institut Mines-Télécom)

Date

02/07/2013

Definition in file naokmeans.cpp.

5.2.2 Function Documentation

5.2.2.1 void createTrainingMeans (std::string *stipFile*, int *dim*, int *maxPts*, int *k*, std::string *meansFile*)

Import HOG and HOF from a file and compute KMeans algorithm to create the file training.means.

Parameters

← ***stipFile*** The file containing the STIPs.

← ***dim*** Points and centers's dimension.

← ***maxPts*** The maximum number of data we can compute

← ***k*** The number of centers

→ ***meansFile*** The file in wich we will save the KMeans centers.

Definition at line 262 of file naokmeans.cpp.

5.2.2.2 void exportCenters (std::string *centers*, int *dim*, int *k*, KMfilterCenters *ctrs*)

Export function to save KMfilterCenters in a file. One line corresponds to one point with dim value (separeted from one space " ").

Parameters

← ***centers*** Name of the file which will be containing dimensions of each centers.

← ***dim*** Center's dimension.

← ***k*** Number of centers.

← ***ctrs*** The centers.

Definition at line 83 of file naokmeans.cpp.

5.2.2.3 void importCenters (std::string *centers*, int *dim*, int *k*, KMfilterCenters * *ctrs*)

Importation function saving external centers in the KMfilterCenters object. One line corresponds to one centers with its values (separeted from one space " ").

Parameters

← ***centers*** Name of the file which will be containing dimensions of each centers.

← ***dim*** Center's dimension.

← ***k*** Number of centers.

→ ***ctrs*** The centers.

Definition at line 109 of file naokmeans.cpp.

5.2.2.4 int importSTIPs (std::string *stip*, int *dim*, int *maxPts*, KMdata * *dataPts*)

STIPs importation function in the format 1 point = 1 line. Each dimension are separated from one space (" ").

Parameters

← ***stip*** Name of the file containing the STIPs.

← ***dim*** The STIPs dimension.

← ***maxPts*** The maximum number of points you want to import.

→ ***dataPts*** The KMlocal object which will be containing STIPs.

Returns

Number of points imported.

Definition at line 23 of file naokmeans.cpp.

5.2.2.5 void kmIvanAlgorithm (int *ic*, int *dim*, const KMdata & *dataPts*, int *k*, KMfilterCenters & *ctrs*)

This is an optimized KMeans algorithm. Ivan's algorithm uses basic KMeans algorithm (here the Lloyd's one) and the idea was to initialize centers intelligently.

Parameters

← ***ic*** The iteration coefficient will determine the number of iterations in each phases.

← ***dim*** Points and centers's dimension.

← ***dataPts*** The data we want to compute the centers.

← ***k*** The number of centers.

→ ***ctrs*** The centers.

The Ivan's algorithm is divided into 3 phases. The first phase is executed on 25 per cent of the data (randomly sampled). To begin, the centers are randomly generated. Then ic $*$ 4 iterations of a KMeans algorithm are executed. During the second part we cluster 50 per cent of the data using the older centroids. This step is computed ic $*$ 2 times. Finally, we make ic $*$ 1 iteration on all the data.

Definition at line 149 of file naokmeans.cpp.

5.3 naokmeans.h File Reference

```
#include <cstdlib>
#include <iostream>
#include <string.h>
#include <fstream>
#include "KMlocal.h"
```

Functions

- int importSTIPs (std::string stip, int dim, int maxPts, KMdata *dataPts)

  STIPs importation function in the format 1 point = 1 line. Each dimension are separated from one space (" ").

- void **exportSTIPs** (std::string stip, int dim, const KMdata &dataPts)
- void importCenters (std::string centers, int dim, int k, KMfilterCenters *ctrs)

  Importation function saving external centers in the KMfilterCenters object. One line corresponds to one centers with its values (separeted from one space " ").

- void exportCenters (std::string centers, int dim, int k, KMfilterCenters ctrs)

  Export function to save KMfilterCenters in a file. One line corresponds to one point with dim value (separeted from one space " ").

- void kmIvanAlgorithm (int ic, int dim, const KMdata &dataPts, int k, KMfilterCenters &ctrs)

  This is an optimized KMeans algorithm. Ivan's algorithm uses basic KMeans algorithm (here the Lloyd's one) and the idea was to initialize centers intelligently.

- void createTrainingMeans (std::string stipFile, int dim, int maxPts, int k, std::string meansFile)

  Import HOG and HOF from a file and compute KMeans algorithm to create the file training.means.

5.3.1 Detailed Description

Author

Fabien ROUALDES (institut Mines-Télécom)

Date

02/07/2013 Set of function permiting to execute KMeans algorithms

Definition in file naokmeans.h.

5.3.2 Function Documentation

5.3.2.1 void createTrainingMeans (std::string *stipFile*, int *dim*, int *maxPts*, int *k*, std::string *meansFile*)

Import HOG and HOF from a file and compute KMeans algorithm to create the file training.means.

Parameters

← ***stipFile*** The file containing the STIPs.

← ***dim*** Points and centers's dimension.

← ***maxPts*** The maximum number of data we can compute

← ***k*** The number of centers

→ ***meansFile*** The file in wich we will save the KMeans centers.

Definition at line 262 of file naokmeans.cpp.

5.3.2.2 void exportCenters (std::string *centers*, int *dim*, int *k*, KMfilterCenters *ctrs*)

Export function to save KMfilterCenters in a file. One line corresponds to one point with dim value (separeted from one space " ").

Parameters

← ***centers*** Name of the file which will be containing dimensions of each centers.

← ***dim*** Center's dimension.

← ***k*** Number of centers.

← ***ctrs*** The centers.

Definition at line 83 of file naokmeans.cpp.

5.3.2.3 void importCenters (std::string *centers*, int *dim*, int *k*, KMfilterCenters * *ctrs*)

Importation function saving external centers in the KMfilterCenters object. One line corresponds to one centers with its values (separeted from one space " ").

Parameters

← ***centers*** Name of the file which will be containing dimensions of each centers.

← ***dim*** Center's dimension.

← ***k*** Number of centers.

→ ***ctrs*** The centers.

Definition at line 109 of file naokmeans.cpp.

5.3.2.4 int importSTIPs (std::string *stip*, int *dim*, int *maxPts*, KMdata * *dataPts*)

STIPs importation function in the format 1 point = 1 line. Each dimension are separated from one space (" ").

Parameters

← ***stip*** Name of the file containing the STIPs.

← ***dim*** The STIPs dimension.

← ***maxPts*** The maximum number of points you want to import.

→ ***dataPts*** The KMlocal object which will be containing STIPs.

Returns

Number of points imported.

Definition at line 23 of file naokmeans.cpp.

5.3.2.5 void kmIvanAlgorithm (int *ic*, int *dim*, const KMdata & *dataPts*, int *k*, KMfilterCenters & *ctrs*)

This is an optimized KMeans algorithm. Ivan's algorithm uses basic KMeans algorithm (here the Lloyd's one) and the idea was to initialize centers intelligently.

Parameters

← ***ic*** The iteration coefficient will determine the number of iterations in each phases.

← ***dim*** Points and centers's dimension.

← ***dataPts*** The data we want to compute the centers.

← ***k*** The number of centers.

→ ***ctrs*** The centers.

The Ivan's algorithm is divided into 3 phases. The first phase is executed on 25 per cent of the data (randomly sampled). To begin, the centers are randomly generated. Then ic $*$ 4 iterations of a KMeans algorithm are executed. During the second part we cluster 50 per cent of the data using the older centroids. This step is computed ic $*$ 2 times. Finally, we make ic $*$ 1 iteration on all the data.

Definition at line 149 of file naokmeans.cpp.

5.4 naomngt.cpp File Reference

Set of functions permiting to manage the activity recognition BDD of Bag Of Words.

```
#include "naomngt.h"
```

Functions

- void listBdds ()
  List BDDs present in the global database.
- void listActivities (std::string bdd)
  List activities present in the specified database.
- int mapActivities (std::string path2bdd, activitiesMap **am)
  Fills the object activitiesMap which contain the equivalence Label-Activity.
- int nbOfFiles (std::string path)
  Counts the number of files in a folder.
- bool fileExist (std::string file, std::string folder)
  Checks if the file name does not exist.
- void addVideos (std::string bddName, std::string activity, int nbVideos, std::string *videoPaths, int dim, int maxPts)
  Adds a new video in the choosen activity of the specified BDD.
- std::string inttostring (int int2str)
  Converts an int into a string.
- void trainBdd (std::string bddName, int dim, int maxPts, int k)
  Trains the specified BDD.
- void addLabel (int label, std::string file, int k)
  Changes the label of the Bag Of Words.
- void addActivity (std::string activityName, std::string bddName)
  Creates a new activity in the specified BDD.
- void deleteActivity (std::string activityName, std::string bddName)
  Deletes an existant activity in the specified BDD.
- void addBdd (std::string bddName)
  Creates a new BDD.
- void deleteBdd (std::string bddName)
  Deletes a BDD.
- void emptyFolder (std::string folder)
  Deletes all files present in the folder.
- void **refreshBdd** (std::string bddName, int dim, int maxPts)

5.4.1 Detailed Description

Set of functions permiting to manage the activity recognition BDD of Bag Of Words.

Author

Fabien ROUALDES (institut Mines-Télécom)

Date

17/07/2013

Definition in file naomngt.cpp.

5.4.2 Function Documentation

5.4.2.1 void addActivity (std::string *activityName*, std::string *bddName*)

Creates a new activity in the specified BDD.

Parameters

← ***activityName*** The name of the new activity.

← ***bddName*** The name of the BDD.

Definition at line 437 of file naomngt.cpp.

5.4.2.2 void addBdd (std::string *bddName*)

Creates a new BDD.

Parameters

← ***bddName*** The name of the BDD we want to create.

Definition at line 546 of file naomngt.cpp.

5.4.2.3 void addLabel (int *label*, std::string *file*, int *k*)

Changes the label of the Bag Of Words.

Parameters

← ***label*** The label.

← ***file*** The file containing the Bag Of Words.

← ***k*** The dimension of the Bag Of Words.

Definition at line 381 of file naomngt.cpp.

5.4.2.4 void addVideos (std::string *bddName*, std::string *activity*, int *nbVideos*, std::string * *videoPaths*, int *dim*, int *maxPts*)

Adds a new video in the choosen activity of the specified BDD.

Parameters

← ***bddName*** The name of the BDD.

← ***activity*** The name of the activity.

← ***nbVideos*** The number of videos we want to add.

← ***videoPaths*** The different paths to the videos.

← ***dim*** The dimension of the HOG and HOF.

← ***maxPts*** The maximum vectors we want to compute.

Definition at line 156 of file naomngt.cpp.

5.4.2.5 void deleteActivity (std::string *activityName*, std::string *bddName*)

Deletes an existant activity in the specified BDD.

Parameters

← ***activityName*** The name of the activity to delete.

← ***bddName*** The name of the BDD.

Definition at line 485 of file naomngt.cpp.

5.4.2.6 void deleteBdd (std::string *bddName*)

Deletes a BDD.

Parameters

← ***bddName*** The name of the bdd.

Definition at line 583 of file naomngt.cpp.

5.4.2.7 void emptyFolder (std::string *folder*)

Deletes all files present in the folder.

Parameters

← ***folder*** The path to the folder.

Definition at line 605 of file naomngt.cpp.

5.4.2.8 bool fileExist (std::string *file*, std::string *folder*)

Checks if the file name does not exist.

Parameters

← ***file*** The path to the file.

← ***folder*** The path to the folder.

Returns

True or false.

Definition at line 126 of file naomngt.cpp.

5.4.2.9 std::string inttostring (int *int2str*)

Converts an int into a string.

Parameters

← ***int2str*** The int to convert.

Returns

The string converted.

Definition at line 206 of file naomngt.cpp.

5.4.2.10 void listActivities (std::string *bdd*)

List activities present in the specified database.

Parameters

← ***bdd*** The name of the bdd.

Definition at line 34 of file naomngt.cpp.

5.4.2.11 int mapActivities (std::string *path2bdd*, activitiesMap ** *am*)

Fills the object activitiesMap which contain the equivalence Label-Activity.

Parameters

← ***path2bdd*** The path to the BDD

↔ ***am*** A pointer to an object activitiesMap.

Returns

The number of activities.

Definition at line 53 of file naomngt.cpp.

5.4.2.12 int nbOfFiles (std::string *path*)

Counts the number of files in a folder.

Parameters

← ***path*** The path to the folder.

Returns

The number of files.

Definition at line 101 of file naomngt.cpp.

5.4.2.13 void trainBdd (std::string *bddName*, int *dim*, int *maxPts*, int *k*)

Trains the specified BDD.

Parameters

← ***bddName*** The name of the BDD.

← ***dim*** The dimension of the STIPs.

← ***maxPts*** The maximum number of points we want to compute.

← ***k*** The number of cluster (means).

Definition at line 225 of file naomngt.cpp.

5.5 naomngt.h File Reference

```
#include <stdlib.h>
#include <dirent.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include "naokmeans.h"
#include "naosvm.h"
#include "naodensetrack.h"
```

Classes

- class activitiesMap

  Correspondance between a label and an activity.

Functions

- void listBdds ()

  List BDDs present in the global database.

- void listActivities (std::string bdd)

  List activities present in the specified database.

- int mapActivities (std::string path2bdd, activitiesMap **am)

  Fills the object activitiesMap which contain the equivalence Label-Activity.

- int nbOfFiles (std::string path)

  Counts the number of files in a folder.

- bool fileExist (std::string file, std::string folder)

  Checks if the file name does not exist.

- void addVideos (std::string bddName, std::string activity, int nbVideos, std::string *videoPaths, int dim, int maxPts)

  Adds a new video in the choosen activity of the specified BDD.

- std::string inttostring (int int2str)

  Converts an int into a string.

- void trainBdd (std::string bddName, int dim, int maxPts, int k)

  Trains the specified BDD.

- void addLabel (int label, std::string file, int k)

  Changes the label of the Bag Of Words.

- void addBdd (std::string bddName)

  Creates a new BDD.

- void addActivity (std::string activityName, std::string bddName)

  Creates a new activity in the specified BDD.

- void deleteBdd (std::string bddName)

  Deletes a BDD.

- void deleteActivity (std::string activityName, std::string bddName)

  Deletes an existant activity in the specified BDD.

- void emptyFolder (std::string folder)

  Deletes all files present in the folder.

- void **refreshBdd** (std::string bddName, int dim, int maxPts)

5.5.1 Detailed Description

Author

Fabien ROUALDES (Institut Mines-Télécom)

Date

17/07/2013

Definition in file naomngt.h.

5.5.2 Function Documentation

5.5.2.1 void addActivity (std::string *activityName*, std::string *bddName*)

Creates a new activity in the specified BDD.

Parameters

← ***activityName*** The name of the new activity.

← ***bddName*** The name of the BDD.

Definition at line 437 of file naomngt.cpp.

5.5.2.2 void addBdd (std::string *bddName*)

Creates a new BDD.

Parameters

← ***bddName*** The name of the BDD we want to create.

Definition at line 546 of file naomngt.cpp.

5.5.2.3 void addLabel (int *label*, std::string *file*, int *k*)

Changes the label of the Bag Of Words.

Parameters

← ***label*** The label.

← ***file*** The file containing the Bag Of Words.

← ***k*** The dimension of the Bag Of Words.

Definition at line 381 of file naomngt.cpp.

5.5.2.4 void addVideos (std::string *bddName*, std::string *activity*, int *nbVideos*, std::string * *videoPaths*, int *dim*, int *maxPts*)

Adds a new video in the choosen activity of the specified BDD.

Parameters

← ***bddName*** The name of the BDD.

← ***activity*** The name of the activity.

← ***nbVideos*** The number of videos we want to add.

← ***videoPaths*** The different paths to the videos.

← ***dim*** The dimension of the HOG and HOF.

← ***maxPts*** The maximum vectors we want to compute.

Definition at line 156 of file naomngt.cpp.

5.5.2.5 void deleteActivity (std::string *activityName*, std::string *bddName*)

Deletes an existant activity in the specified BDD.

Parameters

← ***activityName*** The name of the activity to delete.

← ***bddName*** The name of the BDD.

Definition at line 485 of file naomngt.cpp.

5.5.2.6 void deleteBdd (std::string *bddName*)

Deletes a BDD.

Parameters

← ***bddName*** The name of the bdd.

Definition at line 583 of file naomngt.cpp.

5.5.2.7 void emptyFolder (std::string *folder*)

Deletes all files present in the folder.

Parameters

← ***folder*** The path to the folder.

Definition at line 605 of file naomngt.cpp.

5.5.2.8 bool fileExist (std::string *file*, std::string *folder*)

Checks if the file name does not exist.

Parameters

← ***file*** The path to the file.

← ***folder*** The path to the folder.

Returns

True or false.

Definition at line 126 of file naomngt.cpp.

5.5.2.9 std::string inttostring (int *int2str*)

Converts an int into a string.

Parameters

← ***int2str*** The int to convert.

Returns

The string converted.

Definition at line 206 of file naomngt.cpp.

5.5.2.10 void listActivities (std::string *bdd*)

List activities present in the specified database.

Parameters

← ***bdd*** The name of the bdd.

Definition at line 34 of file naomngt.cpp.

5.5.2.11 int mapActivities (std::string *path2bdd*, activitiesMap ∗∗ *am*)

Fills the object activitiesMap which contain the equivalence Label-Activity.

Parameters

← ***path2bdd*** The path to the BDD

↔ ***am*** A pointer to an object activitiesMap.

Returns

The number of activities.

Definition at line 53 of file naomngt.cpp.

5.5.2.12 int nbOfFiles (std::string *path*)

Counts the number of files in a folder.

Parameters

← ***path*** The path to the folder.

Returns

The number of files.

Definition at line 101 of file naomngt.cpp.

5.5.2.13 void trainBdd (std::string *bddName*, int *dim*, int *maxPts*, int *k*)

Trains the specified BDD.

Parameters

← ***bddName*** The name of the BDD.

← ***dim*** The dimension of the STIPs.

← ***maxPts*** The maximum number of points we want to compute.

← ***k*** The number of cluster (means).

Definition at line 225 of file naomngt.cpp.

5.6 naosvm.cpp File Reference

Set of functions permiting to import/ predict a svm problem, import/create a svm model.

```
#include "naosvm.h"
```

Functions

- struct svm_problem **importProblem** (std::string file, int k)
- void **exportProblem** (struct svm_problem svmProblem, std::string file)
- void **exportProblemZero** (struct svm_problem svmProblem, std::string file, int k)
- struct svm_problem **computeBOW** (int label, const KMdata &dataPts, KMfilterCenters &ctrs)
- void printProblem (struct svm_problem svmProblem)

  It permits to print the SVM problem in the standard output.

- int nrOfLines (std::string filename)

  A function returning the number of lines (which correspond to the number of activities).

- void printProbability (struct svm_model *pModel, struct svm_node *nodes)

  Print for each labels the probability of the activity (stored in the SVM node structure).

- struct svm_model * **createSvmModel** (std::string bowFile, int k)

5.6.1 Detailed Description

Set of functions permiting to import/ predict a svm problem, import/create a svm model.

Author

Fabien ROUALDES (institut Mines-Télécom)

Date

17/07/2013

Definition in file naosvm.cpp.

5.6.2 Function Documentation

5.6.2.1 int nrOfLines (std::string *filename*)

A function returning the number of lines (which correspond to the number of activities).

Parameters

←*fileName* The file we want to count the number of lines.

Returns

The number of lines of the file.

Definition at line 282 of file naosvm.cpp.

5.6.2.2 void printProbability (struct svm_model * *pModel*, struct svm_node * *nodes*)

Print for each labels the probability of the activity (stored in the SVM node structure).

Parameters

← ***pModel*** A pointer to the SVM model.

← ***nodes*** The activity stored in SVM nodes.

Definition at line 304 of file naosvm.cpp.

5.6.2.3 void printProblem (struct svm_problem *svmProblem*)

It permits to print the SVM problem in the standard output.

Parameters

← ***svmProblem*** It is the structure containing the SVM problem.

Definition at line 245 of file naosvm.cpp.

5.7 naosvm.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include "svm.h"
#include "KMlocal.h"
```

Functions

- struct svm_model * **createSvmModel** (std::string bowFile, int k)
- void printProbability (struct svm_model *pModel, struct svm_node *nodes)

  Print for each labels the probability of the activity (stored in the SVM node structure).

- struct svm_problem **importProblem** (std::string file, int k)
- struct svm_problem **computeBOW** (int label, const KMdata &dataPts, KMfilterCenters &ctrs)
- void **exportProblem** (struct svm_problem svmProblem, std::string file)
- void **exportProblemZero** (struct svm_problem svmProblem, std::string file, int k)
- void printProblem (struct svm_problem svmProblem)

  It permits to print the SVM problem in the standard output.

- int nrOfLines (std::string filename)

  A function returning the number of lines (which correspond to the number of activities).

5.7.1 Detailed Description

Author

Fabien ROUALDES (institut Mines-Télécom)

Date

09/07/2013 Set of function permiting to import/ predict a svm problem, import/create a svm model

Definition in file naosvm.h.

5.7.2 Function Documentation

5.7.2.1 int nrOfLines (std::string *filename*)

A function returning the number of lines (which correspond to the number of activities).

Parameters

← ***fileName*** The file we want to count the number of lines.

Returns

The number of lines of the file.

Definition at line 282 of file naosvm.cpp.

5.7.2.2 void printProbability (struct svm_model ∗ *pModel*, struct svm_node ∗ *nodes*)

Print for each labels the probability of the activity (stored in the SVM node structure).

Parameters

← ***pModel*** A pointer to the SVM model.

← ***nodes*** The activity stored in SVM nodes.

Definition at line 304 of file naosvm.cpp.

5.7.2.3 void printProblem (struct svm_problem *svmProblem*)

It permits to print the SVM problem in the standard output.

Parameters

← ***svmProblem*** It is the structure containing the SVM problem.

Definition at line 245 of file naosvm.cpp.

Index