IMAR-C

0.1

Generated by Doxygen 1.6.3

Wed Jul 17 16:59:01 2013

Contents

Chapter 1

File Index

1.1 File List

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

Chapter 2

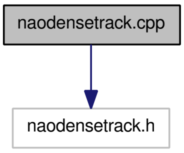
File Documentation

2.1 naodensetrack.cpp File Reference

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

`#include "naodensetrack.h"`

Include dependency graph for naodensetrack.cpp:



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.

2.1.1 Detailed Description

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

Author

LEAR

Date

05/07/2013

Definition in file naodensetrack.cpp.

2.1.2 Function Documentation

2.1.2.1 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.

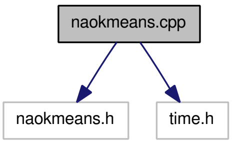
Definition at line 491 of file naodensetrack.cpp.

2.2 naokmeans.cpp File Reference

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

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

Include dependency graph for naokmeans.cpp:



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 (sepereted 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.

2.2.1 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.

2.2.2 Function Documentation

2.2.2.1 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.

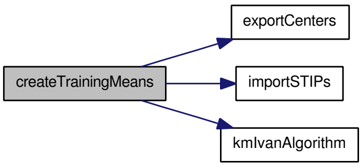
← ***maxPts*** The maximum number of data we can compute

← ***k*** The number of centers

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

Definition at line 262 of file naokmeans.cpp.

Here is the call graph for this function:



2.2.2.2 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.

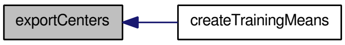
← ***dim*** Center's dimension.

← ***k*** Number of centers.

← ***ctrs*** The centers.

Definition at line 83 of file naokmeans.cpp.

Here is the caller graph for this function:



2.2.2.3 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.

2.2.2.4 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.

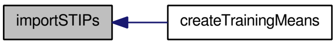
→ ***dataPts*** The KMlocal object which will be containing STIPs.

Returns

Number of points imported.

Definition at line 23 of file naokmeans.cpp.

Here is the caller graph for this function:



2.2.2.5 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.

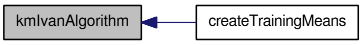
← ***k*** The number of centers.

→ ***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.

Here is the caller graph for this function:

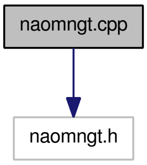


2.3 naomngt.cpp File Reference

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

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

Include dependency graph for naomngt.cpp:



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)
- bool **fileExist** (std::string file, std::string folder)
- void **addVideos** (std::string bddName, std::string activity, int nbVideos, std::string ∗videoPaths, int dim, int maxPts)
- string **inttostring** (int int2str)
- void **trainBdd** (std::string bddName, int dim, int maxPts, int k)
- void **addLabel** (int label, std::string file, int k)
- void **addActivity** (std::string activityName, std::string bddName)
- void **deleteActivity** (std::string activityName, std::string bddName)
- 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)

2.3.1 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.

2.3.2 Function Documentation

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

Creates a new BDD.

Parameters

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

Definition at line 494 of file naomngt.cpp.

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

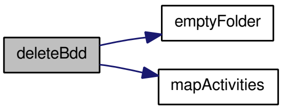
Deletes a BDD.

Parameters

← ***bddName*** The name of the bdd.

Definition at line 531 of file naomngt.cpp.

Here is the call graph for this function:



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

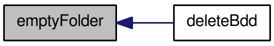
Deletes all files present in the folder.

Parameters

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

Definition at line 552 of file naomngt.cpp.

Here is the caller graph for this function:



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

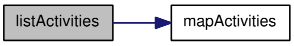
List activities present in the specified database.

Parameters

← ***bdd*** The name of the bdd.

Definition at line 34 of file naomngt.cpp.

Here is the call graph for this function:



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

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

Parameters

← ***path2bdd*** The path to the BDD

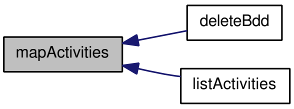
↔ ***am*** A pointer to an object activitiesMap.

Returns

The number of activities.

Definition at line 53 of file naomngt.cpp.

Here is the caller graph for this function:

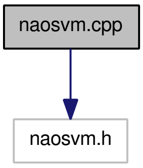


2.4 naosvm.cpp File Reference

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

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

Include dependency graph for naosvm.cpp:



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)

2.4.1 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.

2.4.2 Function Documentation

2.4.2.1 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.

2.4.2.2 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.

2.4.2.3 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