IMAR-C

0.1

Generated by Doxygen 1.6.3

Sun Jul 21 16:12:44 2013

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 _bow Struct Reference

Public Attributes

- int **label**
- float * **histogram**

3.1.1 Detailed Description

Definition at line 23 of file imstatistics.c.

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

- imstatistics.c

3.2 _covarianceMatrix Struct Reference

Public Attributes

- int **label**
- double ** **matrix**

3.2.1 Detailed Description

Definition at line 28 of file imstatistics.c.

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

- imstatistics.c

Chapter 4

File Documentation

4.1 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.

4.1.1 Detailed Description

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

Author

LEAR

Date

05/07/2013

Definition in file naodensetrack.cpp.

4.1.2 Function Documentation

4.1.2.1 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.

4.2 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)

  STIPs exportation function in the format 1 point = 1 line. Each dimension are separated 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 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.

4.2.1 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.

4.2.2 Function Documentation

4.2.2.1 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 266 of file naokmeans.cpp.

4.2.2.2 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 84 of file naokmeans.cpp.

4.2.2.3 void exportSTIPs (std::string *stip*, int *dim*, const KMdata & *dataPts*)

STIPs exportation 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.

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

Definition at line 56 of file naokmeans.cpp.

4.2.2.4 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 111 of file naokmeans.cpp.

4.2.2.5 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.

4.2.2.6 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 152 of file naokmeans.cpp.

4.3 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)
  Deletes all files excepted videos and extracts STIPs again.
- void transferBdd (std::string bddName, std::string login, std::string robotIP, std::string password)
  Transfers the file svm.model, training.means and mapping.txt on the robot Nao.

4.3.1 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.

4.3.2 Function Documentation

4.3.2.1 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 438 of file naomngt.cpp.

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

Creates a new BDD.

Parameters

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

Definition at line 547 of file naomngt.cpp.

4.3.2.3 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 382 of file naomngt.cpp.

4.3.2.4 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.

4.3.2.5 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 486 of file naomngt.cpp.

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

Deletes a BDD.

Parameters

← ***bddName*** The name of the bdd.

Definition at line 584 of file naomngt.cpp.

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

Deletes all files present in the folder.

Parameters

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

Definition at line 606 of file naomngt.cpp.

4.3.2.8 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.

4.3.2.9 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.

4.3.2.10 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.

4.3.2.11 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.

4.3.2.12 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.

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

Deletes all files excepted videos and extracts STIPs again.

Parameters

← ***bddName*** The name of the BDD containing videos.

← ***dim*** The STIPs dimension.

← ***maxPts*** The maximum number of STIPs we can extract.

Definition at line 632 of file naomngt.cpp.

4.3.2.14 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.

4.3.2.15 void transferBdd (std::string *bddName*, std::string *login*, std::string *roboIP*, std::string *password*)

Transfers the file svm.model, training.means and mapping.txt on the robot Nao.

Parameters

← ***bddName*** The name of te BDD to transfer.

← ***login*** Your user name on the robot.

← ***robotIP*** The IP adress of the robot.

← ***password*** The password of the user on the robot.

Training files are sent on the robot Nao via ftp so you have to precise your username on the robot and yout password.

The server ftp must be configured with the following folders: "/data" and "/data/activity_recognition".

Then the program uses the library ftplib to send the training files. It sends mapping.txt, training.means and svm.model which are present in the folder "./bdd/<bdd_name>".

Definition at line 722 of file naomngt.cpp.

4.4 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)

  SVM Importation function. It read a file in the following format: label 1:value 2:value 3:value (each lines).

- void exportProblem (struct svm_problem svmProblem, std::string file)

  SVM Exporting function. It writes in a file in the following format: label 1:value 2:value 3:value (each lines).

- void exportProblemZero (struct svm_problem svmProblem, std::string file, int k)

  SVM Exporting function. It writes in a file in the following format: label 1:value 2:value 3:value (each lines). It is different of exportProblem because it writes the null values.

- struct svm_problem computeBOW (int label, const KMdata &dataPts, KMfilterCenters &ctrs)

  Converts the KMdata into a Bag Of Words histogram in the SVM format: label 1:value 2:value 3:value (each lines).

- 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)

  Create the SVM model of the activities present in a file.

4.4.1 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.

4.4.2 Function Documentation

4.4.2.1 struct svm_problem computeBOW (int *label*, const KMdata & *dataPts*, KMfilterCenters & *ctrs*) [read]

Converts the KMdata into a Bag Of Words histogram in the SVM format: label 1:value 2:value 3:value (each lines).

Parameters

← ***dataPts*** The KMdata.

← ***ctrs*** The centers.

Returns

The svm problem in a structure.

Definition at line 185 of file naosvm.cpp.

4.4.2.2 struct svm_model * createSvmModel (std::string *bowFile*, int *k*) [read]

Create the SVM model of the activities present in a file.

Parameters

← ***bowFile*** The name of the file containing the BOWs.

← ***k*** The number of clusters (dimension of a BOW).

Returns

The SVM model.

Definition at line 425 of file naosvm.cpp.

4.4.2.3 exportProblem (struct svm_problem *svmProblem*, std::string *file*)

SVM Exporting function. It writes in a file in the following format: label 1:value 2:value 3:value (each lines).

Parameters

← ***svmProblem*** The SVM problem to export.

→ ***file*** The file which will contain the Bag Of Words.

Definition at line 115 of file naosvm.cpp.

4.4.2.4 exportProblemZero (struct svm_problem *svmProblem*, std::string *file*, int *k*)

SVM Exporting function. It writes in a file in the following format: label 1:value 2:value 3:value (each lines). It is different of exportProblem because it writes the null values.

Parameters

← ***svmProblem*** The SVM problem to export.

→ ***file*** The file which will contain the Bag Of Words.
← ***The*** dimension of the STIPs.

Definition at line 145 of file naosvm.cpp.

4.4.2.5 struct svm_problem importProblem (std::string *file*, int *k*) [read]

SVM Importation function. It read a file in the following format: label 1:value 2:value 3:value (each lines).

Parameters

← ***file*** File containing the svm problem.
← ***k*** The number of clusters.

Returns

The svm problem in a structure.

Definition at line 18 of file naosvm.cpp.

4.4.2.6 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 290 of file naosvm.cpp.

4.4.2.7 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 312 of file naosvm.cpp.

4.4.2.8 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 253 of file naosvm.cpp.

Index