BDH_ICCV2013

1

Generated by Doxygen 1.8.7

Sat Jun 13 2015 13:43:55

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

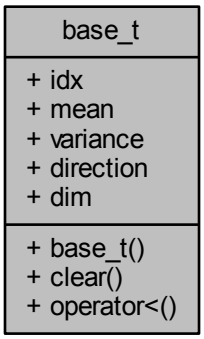
Class Documentation

3.1 base_t Struct Reference

base infomation, mainly PCA base.

`#include <baseset.h>`

Collaboration diagram for base_t:



Public Member Functions

- base_t ()
  default constructor
- void clear ()
  initialize all attributes default and release memory
- bool operator< (const base_t &obj)
  compare the variance

Public Attributes

- int idx
  index of base

- double mean

  mean at base direction

- double variance

  variance at base direction

- double * direction

  direction of base [dim]

Static Public Attributes

- static int dim

  dimension of data space (static member)

3.1.1 Detailed Description

base infomation, mainly PCA base.

destructor and copy constructor are not defined for fast sort.

3.1.2 Member Function Documentation

3.1.2.1 bool base_t::operator< (const base_t & *obj*) [inline]

compare the variance

Parameters

in	*obj*	object

References variance.

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

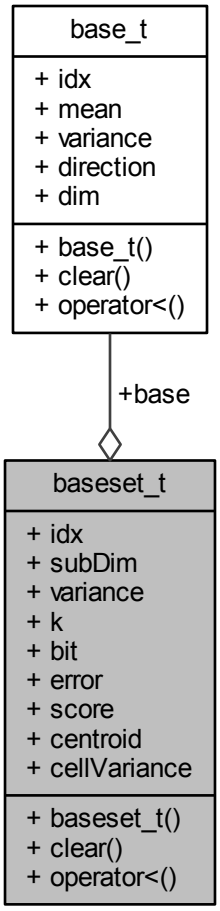
- C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/baseset.h

3.2 baseset_t Struct Reference

set of base_t

`#include <baseset.h>`

Collaboration diagram for baseset_t:



Public Member Functions

- baseset_t ()

  default constructor
- void clear ()

  initialize all attributes default and release memory
- bool operator< (const baseset_t &obj)

  compare the variance

Public Attributes

- int idx

  index of base set
- int subDim

  number of base
- double variance

sum of variance of base

- base_t * base

  set of base_t[subDim]

- int k

  number of centorid = 2^ bit

- double bit

  amount of infomation for this base set = log2(k)

- double error

  error of quantization = sum of cellVariance

- double score

  efficiency for incrementint the number of centroid

- double ** centroid

  centroid[k][subDim]

- double * cellVariance

  variance in cell[k]

3.2.1 Detailed Description

set of base_t

destructor and copy constructor are not defined for fast sort

3.2.2 Member Function Documentation

3.2.2.1 bool baseset_t::operator< (const baseset_t & *obj*) [inline]

compare the variance

Parameters

in	*obj*	object

References variance.

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

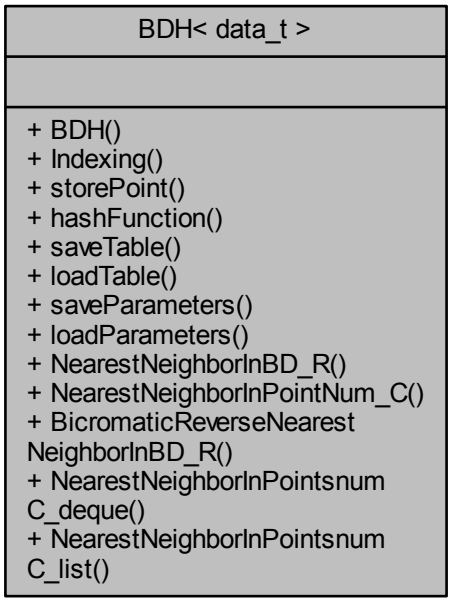
- C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/baseset.h

3.3 BDH< data_t > Class Template Reference

Bucket Distance Hashing.

`#include <BDH.h>`

Collaboration diagram for BDH< data_t >:



Public Types

- typedef unsigned index_t

  type of index for point

Public Member Functions

- BDH ()

  default constructor
- void Indexing (int dim, index_t num, data_t **const data, base_t *const base, int M, int P, int bit, double bit_step=1.0, double sampling_rate=1.0)

  Indexing parameters for hashing.
- void storePoint (index_t num, data_t **data)

  store point set into hash table
- size_t hashFunction (data_t *data)

  hash function
- bool saveTable (const string &path)

  save hash table
- bool loadTable (const string &path)

  load hash table
- bool saveParameters (const string &path)

  save pareameters for hashing
- bool loadParameters (const string &path)

  load pareameters for hashing

- int NearestNeighborInBD_R (data_t *query, point_t< data_t > *point, double Radius, int K=1, double epsilon=DBL_MAX) const

  search in Bucket Distance R from query
- int NearestNeighborInPointNum_C (data_t *query, point_t< data_t > *point, int Cnum, int K=1, double epsilon=DBL_MAX) const

  search nearest C points in accordance with Bucket Distance
- int BicromaticReverseNearestNeighborInBD_R (size_t nQuery, data_t **query, vector< point_t< data_t >> *point, double Radius, int K=1, double epsilon=DBL_MAX) const

  search in Bucket Distance R from query
- int NearestNeighborInPointsnumC_deque (data_t *query, point_t< data_t > *point, int Cnum, int K=1, double epsilon=DBL_MAX) const

  search nearest C points in accordance with Bucket Distance
- int NearestNeighborInPointsnumC_list (data_t *query, point_t< data_t > *point, int Cnum, int K=1, double epsilon=DBL_MAX) const

  search nearest C points in accordance with Bucket Distance

3.3.1 Detailed Description

template<typename data_t>class BDH< data_t >

Bucket Distance Hashing.

3.3.2 Member Function Documentation

3.3.2.1 template<typename data_t > int BDH< data_t >::BicromaticReverseNearestNeighborInBD_R (size_t *nQuery*, data_t ** *query*, vector< point_t< data_t >> * *point*, double *Radius*, int *K* = 1, double *epsilon* = DBL_MAX) const

search in Bucket Distance R from query

Returns

number of points in search area

Parameters

in	*nQuery*	number of query vecotr set
in	*query*	query point set
out	*point*	result reverse nearest neighbors
in	*Radius*	Search Parameter. Rudias of Search area
in	*K*	number of nearest neighbors
in	*epsilon*	search points near than epsilon

3.3.2.2 template<typename data_t > size_t BDH< data_t >::hashFunction (data_t * *data*)

hash function

Returns

hash value

Parameters

in	*data*	a point

3.3.2.3 template<typename data_t > void BDH< data_t >::Indexing (int *dim,* index_t *num,* data_t **const *data,* base_t *const *base,* int *M,* int *P,* int *bit,* double *bit_step =* 1.0, double *sampling_rate =* 1.0)

Indexing parameters for hashing.

Parameters

in	*dim*	dimension of data space
in	*num*	number of data points
in	*data*	sample points for training
in	*base*	base for projection
in	*M*	number of subspace
in	*P*	dimension of subspace
in	*bit*	bits num of hash table
in	*bit_step*	training parameter. 0 < bit_step <= bit.
in	*sampling_rate*	training parameter. 0 < rate <= 1.

3.3.2.4 template<typename data_t > bool BDH< data_t >::loadParameters (const string & *path*)

load pareameters for hashing

Returns

is file open ?

Parameters

in	*path*	file path

3.3.2.5 template<typename data_t > bool BDH< data_t >::loadTable (const string & *path*)

load hash table

Returns

is file open ?

Parameters

in	*path*	file path

3.3.2.6 template<typename data_t > int BDH< data_t >::NearestNeighborInBD_R (data_t * *query,* point_t< data_t > * *point,* double *Radius,* int *K =* 1, double *epsilon =* DBL_MAX) const

search in Bucket Distance R from query

Returns

number of points in search area

Parameters

in	*query*	query point
out	*point*	result nearest neighbors
in	*Radius*	Search Parameter, Radius of Search area
in	*K*	number of nearest neighbors
in	*epsilon*	search points near than epsilon

3.3.2.7 template<typename data_t > int BDH< data_t >::NearestNeighborInPointNum_C (data_t * *query*, point_t< data_t > * *point*, int *Cnum*, int *K* = 1, double *epsilon* = DBL_MAX) const

search nearest C points in accordance with Bucket Distance

Returns

number of points in search area

Parameters

in	*query*	query point
out	*point*	result nearest neighbors
in	*Cnum*	Search parameter, number of points in search
in	*K*	number of nearest neighbors
in	*epsilon*	search points near than epsilon

3.3.2.8 template<typename data_t > int BDH< data_t >::NearestNeighborInPointsnumC_deque (data_t * *query*, point_t< data_t > * *point*, int *Cnum*, int *K* = 1, double *epsilon* = DBL_MAX) const

search nearest C points in accordance with Bucket Distance

search with STL deque. this function doesn't work fast

Returns

number of points in search area

Parameters

in	*query*	query point
out	*point*	result nearest neighbors
in	*Cnum*	Search parameter, number of points in search
in	*K*	number of nearest neighbors
in	*epsilon*	search points near than epsilon

3.3.2.9 template<typename data_t > int BDH< data_t >::NearestNeighborInPointsnumC_list (data_t * *query*, point_t< data_t > * *point*, int *Cnum*, int *K* = 1, double *epsilon* = DBL_MAX) const

search nearest C points in accordance with Bucket Distance

search with STL list. this function work fast at large scale point set

Returns

number of points in search area

Parameters

in	*query*	query point
out	*point*	result nearest neighbors
in	*Cnum*	Search parameter, number of points in search
in	*K*	number of nearest neighbors
in	*epsilon*	search points near than epsilon

3.3.2.10 template<typename data_t > bool BDH< data_t >::saveParameters (const string & *path*)

save pareameters for hashing

Returns

is file open ?

Parameters

in	*path*	file path

3.3.2.11 template<typename data_t > bool BDH< data_t >::saveTable (const string & *path*)

save hash table

Returns

is file open ?

Parameters

in	*path*	file path

3.3.2.12 template<typename data_t > void BDH< data_t >::storePoint (index_t *num*, data_t ** *data*)

store point set into hash table

Parameters

in	*num*	number of data points.
in	*data*	data point set.

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

- C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/BDH.h

3.4 BDHtraining< data_t > Class Template Reference

training BDH parameters

`#include <BDHtraining.h>`

Collaboration diagram for BDHtraining< data_t >:

Public Member Functions

- BDHtraining ()

  default constructor
- ~BDHtraining ()

  destructor
- void training (int dim, int num, data_t **data, const base_t *const baseInput, int M, int P, int bit, double bit_step=1.0)

  training BDH parameters
- const baseset_t *const getBaseSet ()

  getter
- const baseset_t & getLestSet ()

  training BDH parameters
- bool saveParameters (const string &path)

  training BDH parameters

3.4.1 Detailed Description

template<typename data_t>class BDHtraining< data_t >

training BDH parameters

3.4.2 Member Function Documentation

3.4.2.1 template<typename data_t > const baseset_t* const BDHtraining< data_t >::getBaseSet ( ) [inline]

getter

Returns

baseSet

3.4.2.2 template<typename data_t > const baseset_t& BDHtraining< data_t >::getLestSet ( ) [inline]

training BDH parameters

Returns

lsetSet

3.4.2.3 template<typename data_t > bool BDHtraining< data_t >::saveParameters (const string & *path*)

training BDH parameters

Returns

is file open

Parameters

in	*path*	file path

3.4.2.4 template<typename data_t > void BDHtraining< data_t >::training (int *dim*, int *num*, data_t ** *data*, const base_t *const *baseInput*, int *M*, int *P*, int *bit*, double *bit_step* = 1.0)

training BDH parameters

Parameters

in	*dim*	dimension
in	*num*	number of sample
in	*data*	sample data set
in	*baseInput*	base
in	*M*	number of subspace
in	*P*	dimension of subspace
in	*bit*	bits num of hash table
in	*bit_step*	training parameter.

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

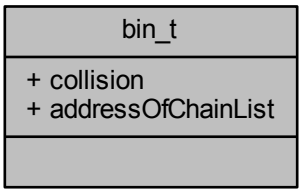
- C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/BDHtraining.h

3.5 bin_t Struct Reference

a chain list

`#include <HashTable.h>`

Collaboration diagram for bin_t:



Public Attributes

- collision_t collision

  collision
- address_t addressOfChainList

  head address of chain list

3.5.1 Detailed Description

a chain list

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

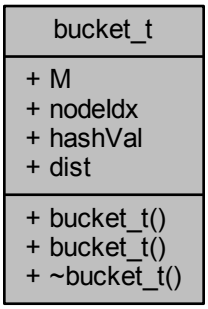
- C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/HashTable.h

3.6 bucket_t Struct Reference

status of neare bucket search

`#include <define.h>`

Collaboration diagram for bucket_t:



Public Member Functions

- bucket_t ()

  default constructor

- bucket_t (int M, int *nodeIdx, size_t hashVal, double dist)

  constructor. allocate memory of nodeIdx and deep copied.

- ~bucket_t ()

  destructor

Public Attributes

- int **M**
- int * nodeIdx

  index of nodes. size M.

- size_t hashVal

  hash value

- double dist

  distance

3.6.1 Detailed Description

status of neare bucket search

3.6.2 Constructor & Destructor Documentation

3.6.2.1 bucket_t::bucket_t (int *M,* int * *nodeIdx,* size_t *hashVal,* double *dist*) [inline]

constructor. allocate memory of nodeIdx and deep copied.

Parameters

M	index of layer
nodeIdx	index of nodes
hashVal	hash value
dist	distance

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

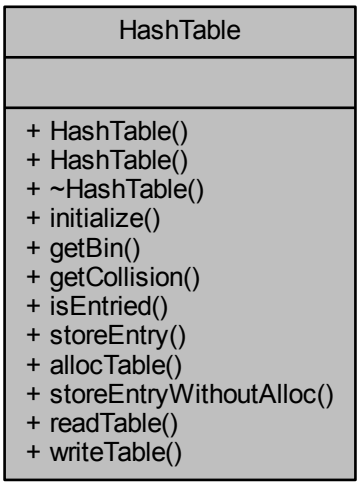
- C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/define.h

3.7 HashTable Class Reference

hash table

`#include <HashTable.h>`

Collaboration diagram for HashTable:



Public Member Functions

- HashTable ()

  default constructor
- HashTable (size_t pointSize, size_t hashSize)

  constructor
- ~HashTable ()

  destructor
- void initialize (size_t pointSize, size_t hashSize)

  initializer
- void getBin (size_t hashVal, bin_t &bin) const

  bin getter
- collision_t getCollision (size_t hashVal) const

  collision getter
- bool isEntried (size_t hashVal) const

  check is hash value available
- bool storeEntry (size_t HashValue, const address_t point)

  Storing a point into hash table.
- bool allocTable (unsigned ∗collision)

  allocation of hash table
- void storeEntryWithoutAlloc (size_t HashValue, const address_t data)

  you have to take care of whether allocation of hash table has already completed
- bool readTable (const string &tblFile)

  read hash table from binary file
- bool writeTable (const string &tblFile)

  write hash table from binary file

3.7.1 Detailed Description

hash table

3.7.2 Constructor & Destructor Documentation

3.7.2.1 HashTable::HashTable (size_t *pointSize,* size_t *hashSize*)

constructor

Parameters

in	*pointSize*	sizeof(data_t)∗dim;
in	*hashSize*	hash size = 1<<bit;

3.7.3 Member Function Documentation

3.7.3.1 bool HashTable::allocTable (unsigned ∗ *collision*)

allocation of hash table

Returns

is allocation complete ?

Parameters

in	*collision*	collision list of the hash table

3.7.3.2 void HashTable::getBin (size_t *hashVal,* bin_t & *bin*) const

bin getter

Parameters

in	*hashVal*	hash vlue
out	*bin*	bin

3.7.3.3 collision_t HashTable::getCollision (size_t *hashVal*) const [inline]

collision getter

Parameters

in	*hashVal*	hash vlue

3.7.3.4 void HashTable::initialize (size_t *pointSize,* size_t *hashSize*)

initializer

Parameters

in	*pointSize*	sizeof(data_t)∗dim;
in	*hashSize*	hash size = 1<<bit;

3.7.3.5 bool HashTable::isEntried (size_t *hashVal*) const [inline]

check is hash value available

Returns

is hashTable[hashVal] active ?

Parameters

in	*hashVal*	hash vlue

3.7.3.6 bool HashTable::readTable (const string & *tblFile*)

read hash table from binary file

Returns

is file open ?

Parameters

in	*tblFile*	file path

3.7.3.7 bool HashTable::storeEntry (size_t *HashValue,* const address_t *point*)

Storing a point into hash table.

Returns

is allocation complete ?

Parameters

in	*HashValue*	hash value of point data
in	*point*	the point stored into hash table

3.7.3.8 void HashTable::storeEntryWithoutAlloc (size_t *HashValue,* const address_t *data*)

you have to take care of whether allocation of hash table has already completed

Parameters

in	*HashValue*	hash value of point data
in	*data*	the point stored into hash table

3.7.3.9 bool HashTable::writeTable (const string & *tblFile*)

write hash table from binary file

Returns

is file open ?

Parameters

in	*tblFile*	file path

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

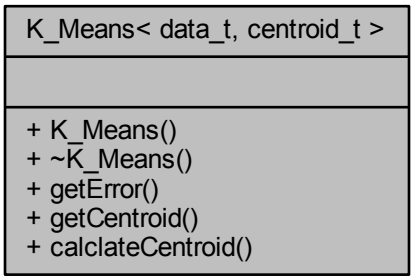
- C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/HashTable.h

3.8 K_Means< data_t, centroid_t > Class Template Reference

k-means

`#include <k_means.h>`

Collaboration diagram for K_Means< data_t, centroid_t >:



Public Member Functions

- K_Means ()

  default constructor
- ~K_Means ()

  destructor
- double getError ()

  error getter
- double **const getCentroid () const

  centroid getter
- void calclateCentroid (int dim, int num, data_t **point, int K)

  calculate centroid

3.8.1 Detailed Description

template<typename data_t, typename centroid_t>class K_Means< data_t, centroid_t >

k-means

3.8.2 Member Function Documentation

3.8.2.1 template<typename data_t , typename centroid_t > void K_Means< data_t, centroid_t >::calclateCentroid (int *dim,* int *num,* data_t ** *point,* int *K*)

calculate centroid

Parameters

in	*dim*	dimension
in	*num*	number of sample
in	*point*	sample point set
in	*K*	number of centroid

3.8.2.2 template<typename data_t, typename centroid_t> double** const K_Means< data_t, centroid_t >::getCentroid ( ) const [inline]

centroid getter

Returns

centroid

3.8.2.3 template<typename data_t, typename centroid_t> double K_Means< data_t, centroid_t >::getError ( ) [inline]

error getter

Returns

error

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

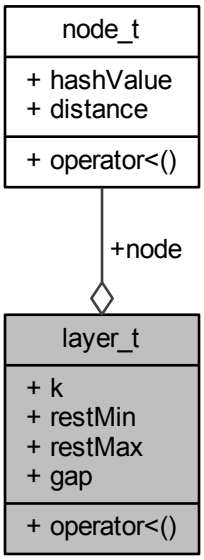
- C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/k_means.h

3.9 layer_t Struct Reference

type of layer. a node coresponde to a subspace

`#include <define.h>`

Collaboration diagram for layer_t:



Public Member Functions

- bool operator< (const layer_t &obj)

  compare the gap

Public Attributes

- int k

  number of nodes
- double restMin

  the minimam rest distance from this layer
- double restMax

  the maximam rest distance from this layer
- double gap

  the gap of distance between max and min
- node_t * node

  nodes.

3.9.1 Detailed Description

type of layer. a node coresponde to a subspace

3.9.2 Member Function Documentation

3.9.2.1 bool layer_t::operator< (const layer_t & *obj*) [inline]

compare the gap

Returns

is which gap larger ?

References gap.

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

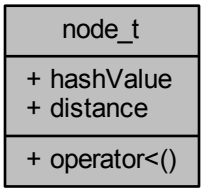
- C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/define.h

3.10 node_t Struct Reference

type of node. a node coresponde to a centroid

`#include <define.h>`

Collaboration diagram for node_t:



Public Member Functions

- bool operator< (const node_t &obj)

  compare the distance

Public Attributes

- size_t hashValue

  hash value
- double distance

  sub bucket distance

3.10.1 Detailed Description

type of node. a node coresponde to a centroid

3.10.2 Member Function Documentation

3.10.2.1 bool node_t::operator< (const node_t & *obj*) [inline]

compare the distance

Returns

is which lesser ?

Parameters

obj	compared object

References distance.

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

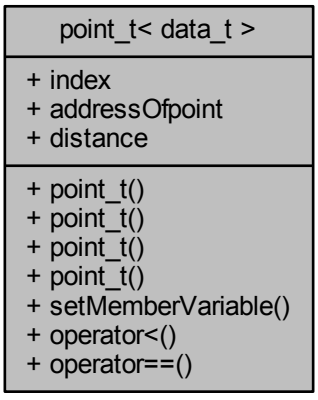
- C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/define.h

3.11 point_t< data_t > Struct Template Reference

this structure has propaties of a point

`#include <point.h>`

Collaboration diagram for point_t< data_t >:



Public Member Functions

- point_t ()
  default constructor
- point_t (const size_t &index, data_t *addressOfpoint, const double &distance)
  constructor
- point_t (const size_t &index, const double &distance)
  constructor

- point_t (const size_t &index, data_t *addressOfpoint)

  constructor
- void setMemberVariable (const size_t &index, data_t *addressOfpoint, const double &distance)

  set member variables
- bool operator< (const point_t &e) const

  compare the distance
- bool operator== (const point_t &e) const

  compare the distance

Public Attributes

- size_t index

  index of data
- data_t * addressOfpoint

  head address of a data
- double distance

  ditance from query

3.11.1 Detailed Description

template<typename data_t>struct point_t< data_t >

this structure has propaties of a point

3.11.2 Constructor & Destructor Documentation

3.11.2.1 template<typename data_t> point_t< data_t >::point_t (const size_t & *index*, data_t * *addressOfpoint*, const double & *distance*) [inline]

constructor

Parameters

in	*index*	the index of point
in	*addressOfpoint*	the address of point
in	*distance*	the distance from query

3.11.2.2 template<typename data_t> point_t< data_t >::point_t (const size_t & *index*, const double & *distance*) [inline]

constructor

Parameters

in	*index*	the index of point
in	*distance*	the distance from query

3.11.2.3 template<typename data_t> point_t< data_t >::point_t (const size_t & *index*, data_t * *addressOfpoint*) [inline]

constructor

Parameters

in	*index*	the index of point
in	*addressOfpoint*	the address of point

3.11.3 Member Function Documentation

3.11.3.1 template<typename data_t> bool point_t< data_t >::operator< (const point_t< data_t > & *e*) const [inline]

compare the distance

Returns

is my distance lessor than e's ?

Parameters

in	*e*	compare object

References point_t< data_t >::distance.

3.11.3.2 template<typename data_t> bool point_t< data_t >::operator== (const point_t< data_t > & *e*) const [inline]

compare the distance

Returns

is my distance equal to e's ?

Parameters

in	*e*	compare object

References point_t< data_t >::index.

3.11.3.3 template<typename data_t> void point_t< data_t >::setMemberVariable (const size_t & *index*, data_t * *addressOfpoint*, const double & *distance*) [inline]

set member variables

Parameters

in	*index*	the index of point
in	*addressOfpoint*	the address of point
in	*distance*	the distance from query

References point_t< data_t >::addressOfpoint, point_t< data_t >::distance, and point_t< data_t >::index.

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

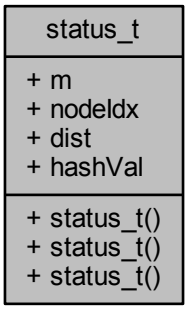
- C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/point.h

3.12 status_t Struct Reference

status of neare bucket search

`#include <define.h>`

Collaboration diagram for status_t:



Public Member Functions

- status_t ()
  default constructor
- status_t (const int &m, const int &nodeIdx, const size_t &hashVal, const double &dist)
  constructor
- status_t (const int &m)
  constructor

Public Attributes

- int m
  index of layer
- int nodeIdx
  index of nodes
- double dist
  distance
- size_t hashVal
  hash value

3.12.1 Detailed Description

status of neare bucket search

3.12.2 Constructor & Destructor Documentation

3.12.2.1 status_t::status_t (const int & *m,* const int & *nodeIdx,* const size_t & *hashVal,* const double & *dist*) [inline]

constructor

Parameters

m	index of layer
nodeIdx	index of nodes
hashVal	hash value
dist	distance

3.12.2.2 status_t::status_t (const int & *m*) [inline]

constructor

Parameters

m	index of layer

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

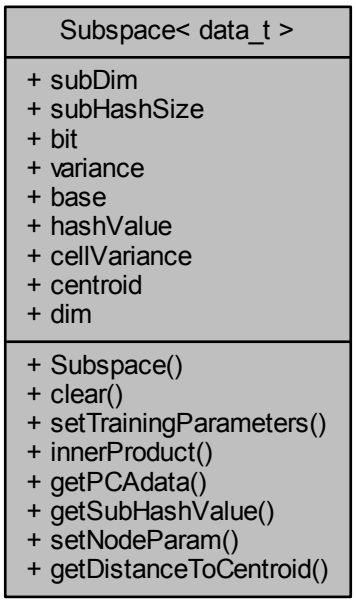
- C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/define.h

3.13 Subspace< data_t > Class Template Reference

manage subspace

`#include <Subspace.h>`

Collaboration diagram for Subspace< data_t >:



Public Member Functions

- Subspace ()

default constructor

- void clear ()

  initialize all member variables
- void setTrainingParameters (const baseset_t &baseSet)

  set training paramters
- double innerProduct (double *base, const data_t *data)

  inner product
- void getPCAdata (const data_t *data, double *PCAdata)

  project data into Principal Component space
- size_t getSubHashValue (const data_t *data)

  get sub hash value
- void setNodeParam (node_t *node, data_t *query)

  set node param
- double getDistanceToCentroid (double *PCAquery, int centroidIndex)

  get distance to centroid

Public Attributes

- int subDim

  dimension of subspace
- int subHashSize

  hash size at subspace = 1<<bit
- double bit

  information volume
- double variance

  sum of variance
- double ** base

  base direction[P][dim]
- size_t * hashValue

  hash value of bin corespond to centroid[subHashSize]
- double * cellVariance

  variance in cell[subHashSize]
- double ** centroid

  centroid[subHashSize][subDim]

Static Public Attributes

- static int dim

  dimension

3.13.1 Detailed Description

template<typename data_t>class Subspace< data_t >

manage subspace

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

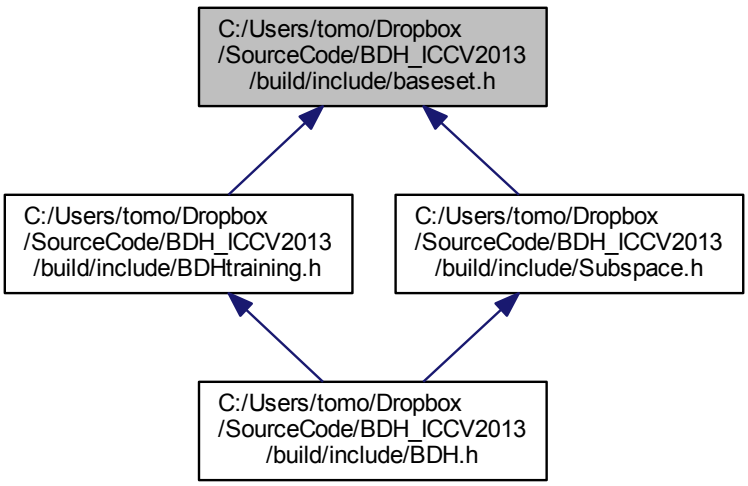
- C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/Subspace.h

Chapter 4

File Documentation

4.1 C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/baseset.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct base_t

  base infomation, mainly PCA base.
- struct baseset_t

  set of base_t

4.1.1 Detailed Description

Author

T.Sato

Date

2015.05.04

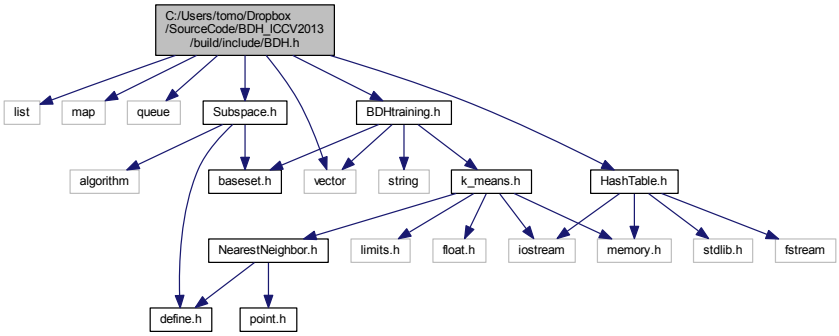
Version

1.0

4.2 C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/BDH.h File Reference

```
#include <list>
#include <map>
#include <queue>
#include <vector>
#include <Subspace.h>
#include <BDHtraining.h>
#include <HashTable.h>
```

Include dependency graph for BDH.h:



Classes

- class BDH< data_t >

  Bucket Distance Hashing.

4.2.1 Detailed Description

Author

Tomokazu Sato

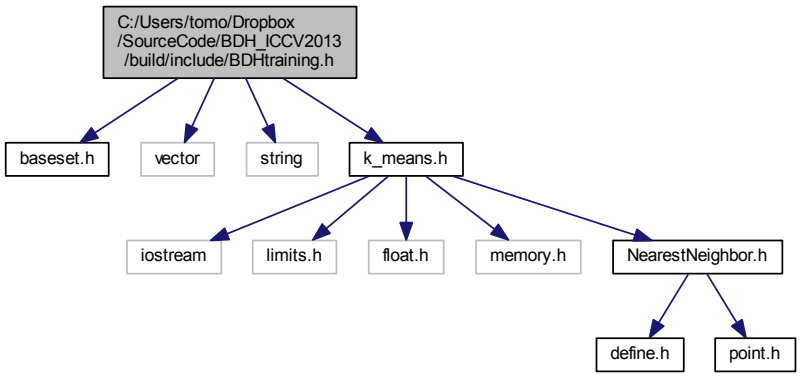
Date

2015/01/13

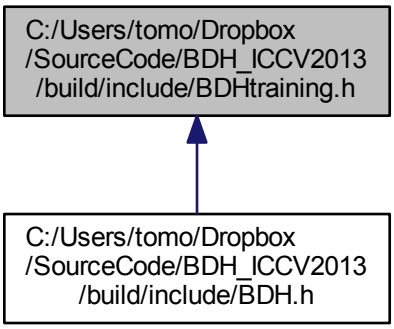
4.3 C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/BDHtraining.h File Reference

```
#include <baseset.h>
#include <vector>
#include <string>
#include "k_means.h"
```

Include dependency graph for BDHtraining.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BDHtraining< data_t >

  training BDH parameters

4.3.1 Detailed Description

Author

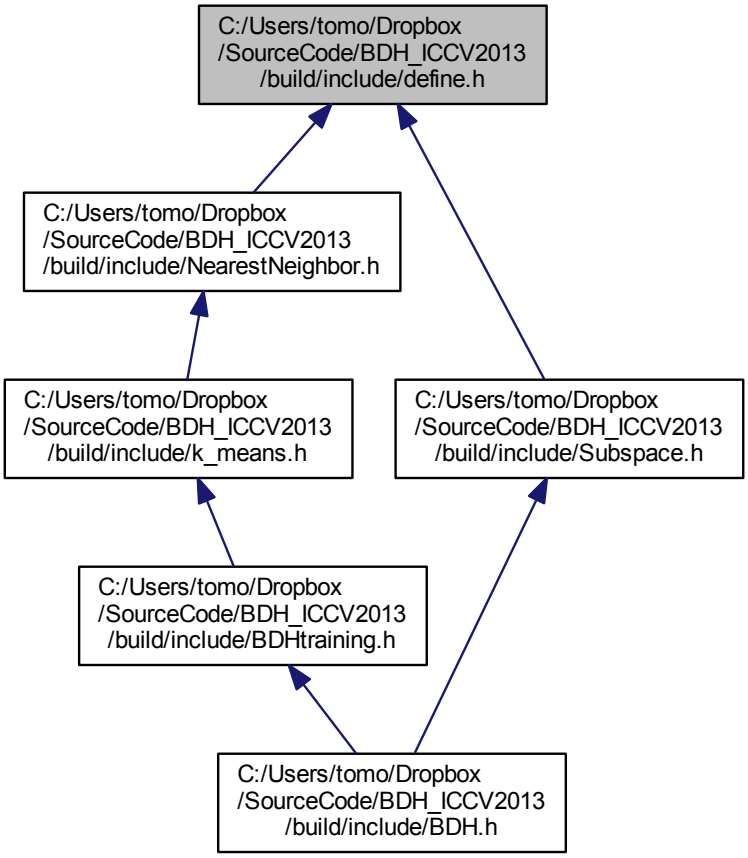
Tomokazu Sato

Date

2015/05/04

4.4 C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/define.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct node_t

  type of node. a node coresponde to a centroid
- struct layer_t

  type of layer. a node coresponde to a subspace
- struct status_t

  status of neare bucket search

- struct bucket_t

  status of neare bucket search

Functions

- double NORM (double x)

  norm for distance

4.4.1 Detailed Description

Author

Tomokazu Sato

Date

2015/05/05

4.4.2 Function Documentation

4.4.2.1 double NORM (double *x*) [inline]

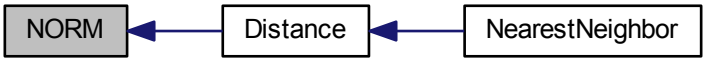
norm for distance

Returns

scalar distance

Referenced by Distance().

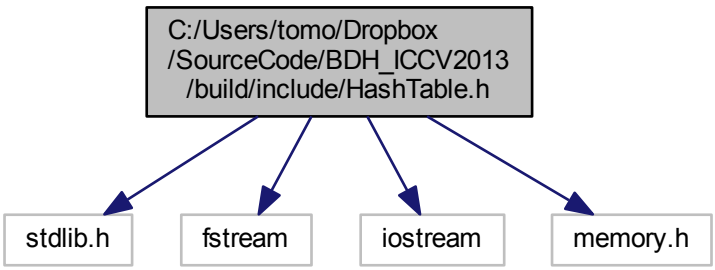
Here is the caller graph for this function:



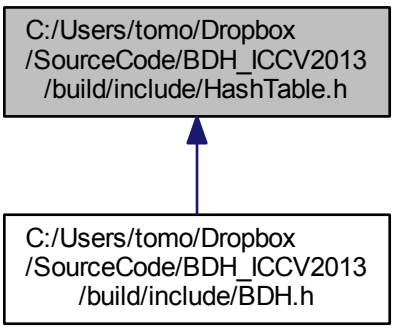
4.5 C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/HashTable.h File Reference

```
#include <stdlib.h>
#include <fstream>
#include <iostream>
#include <memory.h>
```

Include dependency graph for HashTable.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct bin_t

  a chain list
- class HashTable

  hash table

Typedefs

- typedef unsigned collision_t

  type of collision
- typedef char * address_t

  type of address

4.5.1 Detailed Description

Author

Tomokazu Sato

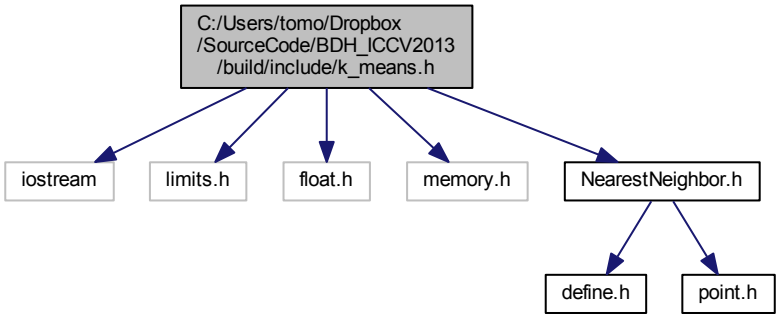
Date

2015/05/05

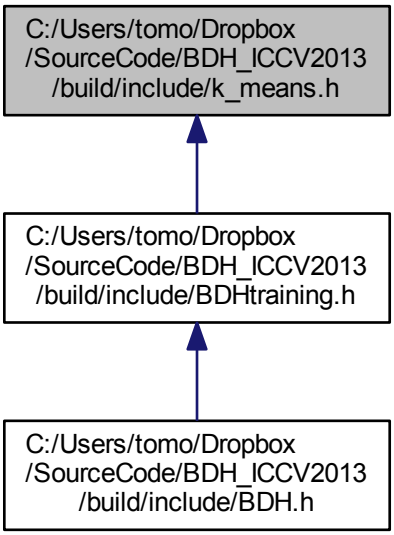
4.6 C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/k_means.h File Reference

```
#include <iostream>
#include <limits.h>
#include <float.h>
#include <memory.h>
#include "NearestNeighbor.h"
```

Include dependency graph for k_means.h:



This graph shows which files directly or indirectly include this file:



Classes

- class K_Means< data_t, centroid_t >

  k-means

4.6.1 Detailed Description

Author

T.Sato

Date

2015.05.05

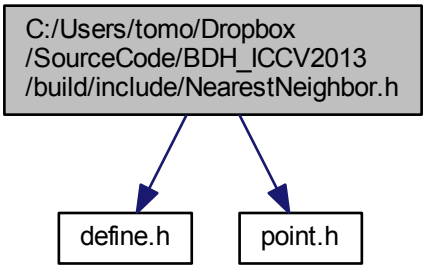
Version

1.0

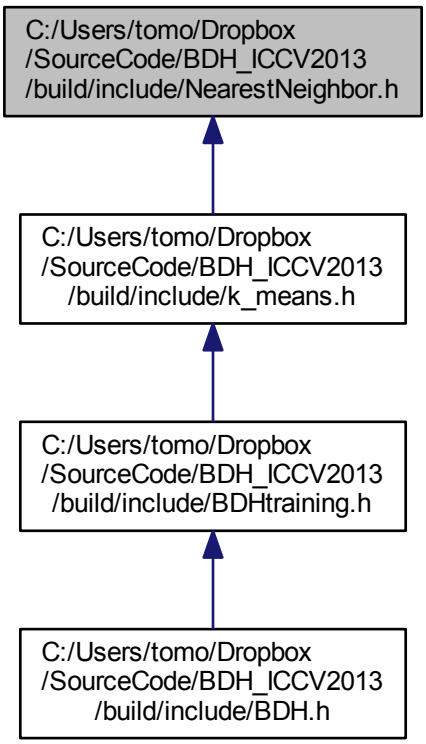
4.7 C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/NearestNeighbor.h File Reference

```
#include "define.h"
#include "point.h"
```

Include dependency graph for NearestNeighbor.h:



This graph shows which files directly or indirectly include this file:



Functions

- template<typename data_t , typename query_t >
  double Distance (int dim, data_t *sample, query_t *query)

distance calculation

- template<typename data_t , typename query_t >
  double Distance (int dim, data_t *sample, query_t *query, double Limit)

  distance calculation

- template<typename data_t , typename query_t >
  int NearestNeighbor (int dim, int num, data_t **sample, query_t *query)

  k-means

- template<typename data_t , typename query_t >
  void NearestNeighbor (int dim, int num, data_t **sample, query_t *query, point_t< data_t > &NNpoint)

  k-means

4.7.1 Detailed Description

Author

T.Sato

Date

2015.05.05

Version

1.0

4.7.2 Function Documentation

4.7.2.1 template<typename data_t , typename query_t > double Distance (int *dim*, data_t * *sample*, query_t * *query*)

distance calculation

Returns

distance

Parameters

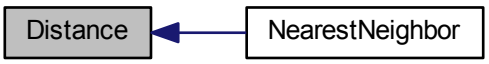
in	*dim*	dimension
in	*sample*	data
in	*query*	query

References NORM().

Referenced by NearestNeighbor().

Here is the call graph for this function:

Here is the caller graph for this function:



4.7.2.2 template<typename data_t , typename query_t > double Distance (int *dim*, data_t * *sample*, query_t * *query*, double *Limit*)

distance calculation

Returns

distance

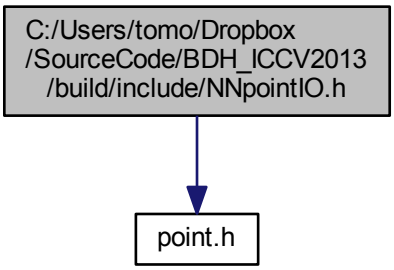
References NORM().

Here is the call graph for this function:

4.8 C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/NNpointIO.h File Reference

```
#include "point.h"
```

Include dependency graph for NNpointIO.h:



Functions

- template<typename data_t >
  bool SaveReverseNearestNeighbor (const string &path, unsigned Qnum, int K, vector< point_t< data_t >> *RNNpoint, double time=0.0, size_t memory=0)

  save the result of reverse nearest neighbor search.
- template<typename data_t >
  bool LoadReverseNearestNeighbor (const string &path, unsigned &Qnum, int &K, vector< point_t< data_t >> *RNNpoint, double &time, size_t &memory)

  load the result of reverse nearest neighbor search.
- template<typename data_t >
  bool SaveNearestNeighbor (const string &path, unsigned Qnum, int K, point_t< data_t > **NNpoint, double time=0.0, size_t memory=0)

  save the result of reverse nearest neighbor search.
- template<typename data_t >
  bool LoadNearestNeighbor (const string &path, unsigned &Qnum, int &K, point_t< data_t > **&NNpoint, double &time, size_t &memory)

  load the result of reverse nearest neighbor search.

4.8.1 Detailed Description

Author

T.Sato

Date

2015.05.03

Version

1.0

4.8.2 Function Documentation

4.8.2.1 template<typename data_t > bool LoadNearestNeighbor (const string & *path,* unsigned & *Qnum,* int & *K,* point_t< data_t > **& *NNpoint,* double & *time,* size_t & *memory*)

load the result of reverse nearest neighbor search.

Returns

is file open ?

Parameters

in	*path*	
out	*Qnum*	number of query point set
out	*K*	number of nearest neighbors
out	*NNpoint*	nearest neighbors
out	*time*	
out	*memory*	

References point_t< data_t >::distance, and point_t< data_t >::index.

4.8.2.2 template<typename data_t > bool LoadReverseNearestNeighbor (const string & *path,* unsigned & *Qnum,* int & *K,* vector< point_t< data_t >> * *RNNpoint,* double & *time,* size_t & *memory*)

load the result of reverse nearest neighbor search.

Returns

is file open ?

Parameters

in	*path*	file path
out	*Qnum*	number of query point set
out	*K*	number of nearest neighbors
out	*RNNpoint*	revers nearest neighbors
out	*time*	query time
out	*memory*	work memory

4.8.2.3 template<typename data_t > bool SaveNearestNeighbor (const string & *path,* unsigned *Qnum,* int *K,* point_t< data_t > ** *NNpoint,* double *time* = 0.0, size_t *memory* = 0)

save the result of reverse nearest neighbor search.

Returns

is file open ?

Parameters

in	*path*	
in	*Qnum*	number of query point set

in	*K*	number of nearest neighbors
in	*NNpoint*	nearest neighbors
in	*time*	
in	*memory*	

References point_t< data_t >::distance, and point_t< data_t >::index.

4.8.2.4 template<typename data_t > bool SaveReverseNearestNeighbor (const string & *path*, unsigned *Qnum*, int *K*, vector< point_t< data_t >> * *RNNpoint*, double *time* = 0.0, size_t *memory* = 0)

save the result of reverse nearest neighbor search.

Returns

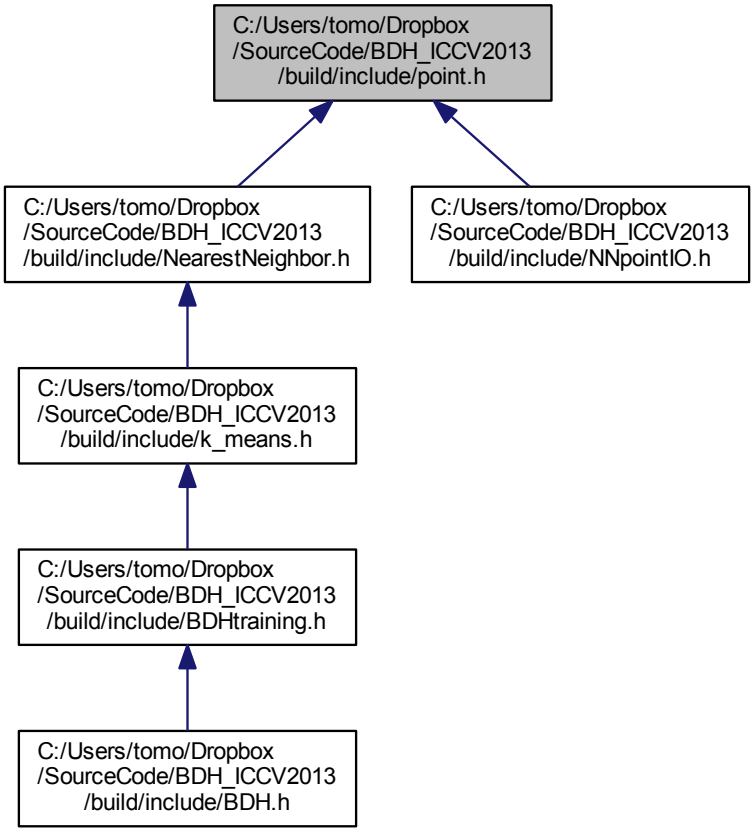
is file open ?

Parameters

in	*path*	file path
in	*Qnum*	number of query point set
in	*K*	number of nearest neighbors
in	*RNNpoint*	revers nearest neighbors
in	*time*	query time
in	*memory*	work memory

4.9 C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/point.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct point_t< data_t >

  this structure has propaties of a point

4.9.1 Detailed Description

Author

T.Sato

Date

2015.04.28

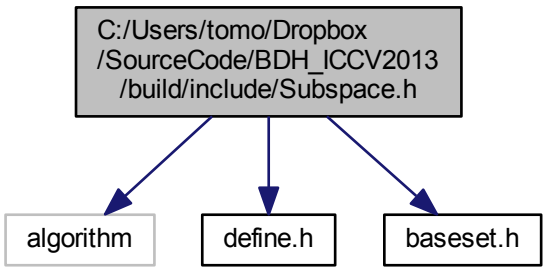
Version

1.0

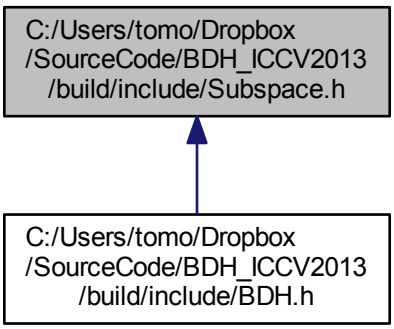
4.10 C:/Users/tomo/Dropbox/SourceCode/BDH_ICCV2013/build/include/Subspace.h File Reference

```
#include <algorithm>
#include "define.h"
#include "baseset.h"
```

Include dependency graph for Subspace.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Subspace< data_t >

  manage subspace

4.10.1 Detailed Description

Author

Tomokazu Sato

Date

2015/05/06