

## GraphSpace with C++

Generated by Doxygen 1.8.17



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 distance< T > Class Template Reference	5
3.2 Distance::distanceHolder< T > Class Template Reference	5
3.3 euclidean< T > Class Template Reference	5
3.3.1 Detailed Description	6
3.3.2 Member Function Documentation	6
3.3.2.1 edge_dis()	6
3.3.2.2 get_Instance()	6
3.3.2.3 node_dis()	6
3.3.2.4 the_dis()	7
3.3.2.5 the_sim()	7
3.4 GA< T > Class Template Reference	7
3.4.1 Detailed Description	7
3.4.2 Constructor & Destructor Documentation	8
3.4.2.1 GA() [1/2]	8
3.4.2.2 GA() [2/2]	8
3.4.3 Member Function Documentation	8
3.4.3.1 cleanup()	8
3.4.3.2 initializeMatchMatrix()	8
3.4.3.3 isStable()	9
3.4.3.4 match()	9
3.4.3.5 setAssociationGraph()	9
3.5 gpc< T > Class Template Reference	9
3.5.1 Constructor & Destructor Documentation	10
3.5.1.1 gpc()	10
3.5.2 Member Function Documentation	10
3.5.2.1 align_geo()	10
3.5.2.2 est_pc()	10
3.5.2.3 get_barycenter()	10
3.5.2.4 get_barycenter_net()	10
3.5.2.5 get_gs()	11
3.5.2.6 gpc_aac()	11
3.5.2.7 set_barycenter()	11
3.6 Graph< T > Class Template Reference	11
3.6.1 Detailed Description	12
3.6.2 Constructor & Destructor Documentation	12
3.6.2.1 Graph() [1/3]	12

3.6.2.2 Graph() [2/3]	12
3.6.2.3 Graph() [3/3]	13
3.6.3 Member Function Documentation	13
3.6.3.1 add_edge()	13
3.6.3.2 add_vertex()	13
3.6.3.3 construct_adj()	13
3.6.3.4 get_adj()	13
3.6.3.5 get_edge_size()	14
3.6.3.6 get_graph_map()	14
3.6.3.7 get_keys()	14
3.6.3.8 get_n_nodes()	14
3.6.3.9 get_vertex_size()	14
3.6.3.10 get_vertices_id()	14
3.6.3.11 grow()	15
3.6.3.12 is_oriented()	15
3.6.3.13 isempty()	15
3.6.3.14 permute()	15
3.6.3.15 print_map()	15
3.6.3.16 scale()	15
3.7 GraphSet< T > Class Template Reference	16
3.7.1 Detailed Description	16
3.7.2 Constructor & Destructor Documentation	16
3.7.2.1 GraphSet() [1/2]	17
3.7.2.2 GraphSet() [2/2]	17
3.7.3 Member Function Documentation	17
3.7.3.1 add_graph()	17
3.7.3.2 align()	17
3.7.3.3 est()	17
3.7.3.4 get_aligned_GraphSet()	18
3.7.3.5 get_aligned_GraphSet_maps()	18
3.7.3.6 get_distance()	18
3.7.3.7 get_e_attr_max()	18
3.7.3.8 get_graphset()	18
3.7.3.9 get_graphset_maps()	18
3.7.3.10 get_matcher()	18
3.7.3.11 get_mean()	19
3.7.3.12 get_n_max()	19
3.7.3.13 get_permutation_vector()	19
3.7.3.14 get_v_attr_max()	19
3.7.3.15 is_oriented()	19
3.7.3.16 mean_aac()	19
3.7.3.17 permuted_graphset()	19

3.7.3.18 read_from_text()	20
3.7.3.19 save_aligned()	20
3.7.3.20 set_distance()	20
3.7.3.21 set_match()	20
3.7.3.22 set_permutation_vector()	20
3.7.3.23 to_matrix_with_attr()	20
3.8 ID< T > Class Template Reference	21
3.8.1 Detailed Description	21
3.8.2 Constructor & Destructor Documentation	21
3.8.2.1 ID() [1/2]	21
3.8.2.2 ID() [2/2]	21
3.8.3 Member Function Documentation	21
3.8.3.1 match()	22
3.9 matcher< T > Class Template Reference	22
3.9.1 Detailed Description	22
3.9.2 Constructor & Destructor Documentation	23
3.9.2.1 matcher() [1/2]	23
3.9.2.2 matcher() [2/2]	23
3.9.3 Member Function Documentation	23
3.9.3.1 get_dist()	23
3.9.3.2 get_distance()	23
3.9.3.3 get_f()	23
3.9.3.4 match()	24
3.9.3.5 set_dist()	24
3.9.3.6 the_dis()	24
3.9.4 Member Data Documentation	24
3.9.4.1 dist	24
3.9.4.2 distance	24
3.9.4.3 f	25
3.10 Matcher::matcherHolder< T > Class Template Reference	25
3.11 Munkres< T > Class Template Reference	25
<b>Index</b>	<b>27</b>



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

distance< T > . . . . .	5
euclidean< T > . . . . .	5
Distance::distanceHolder< T > . . . . .	5
gpc< T > . . . . .	9
Graph< T > . . . . .	11
GraphSet< T > . . . . .	16
matcher< T > . . . . .	22
GA< T > . . . . .	7
ID< T > . . . . .	21
Matcher::matcherHolder< T > . . . . .	25
Munkres< T > . . . . .	25





## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">distance&lt; T &gt;</a>	5
<a href="#">Distance::distanceHolder&lt; T &gt;</a>	5
<a href="#">euclidean&lt; T &gt;</a>	5
<a href="#">GA&lt; T &gt;</a>	7
<a href="#">gpc&lt; T &gt;</a>	9
<a href="#">Graph&lt; T &gt;</a>	11
<a href="#">GraphSet&lt; T &gt;</a>	16
<a href="#">ID&lt; T &gt;</a>	21
<a href="#">matcher&lt; T &gt;</a>	22
<a href="#">Matcher::matcherHolder&lt; T &gt;</a>	25
<a href="#">Munkres&lt; T &gt;</a>	25



## Chapter 3

# Class Documentation

### 3.1 `distance< T >` Class Template Reference

```
#include <distance.h>
```

Inheritance diagram for `distance< T >`:

### 3.2 `Distance::distanceHolder< T >` Class Template Reference

#### Public Member Functions

- **distanceHolder** (distances d\_id)
- void **setDistance** (distances d\_id)

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

- DistanceFactory.h

### 3.3 `euclidean< T >` Class Template Reference

```
#include <euclidean.h>
```

Inheritance diagram for `euclidean< T >`:

Collaboration diagram for `euclidean< T >`:

#### Public Member Functions

- T [the\\_sim](#) (attr\_type< T > x, attr\_type< T > y)
- T [the\\_dis](#) (attr\_type< T > x, attr\_type< T > y)
- T [node\\_dis](#) (attr\_type< T > x, attr\_type< T > y) override
- T [node\\_sim](#) (attr\_type< T > x, attr\_type< T > y) override
- T [edge\\_dis](#) (attr\_type< T > x, attr\_type< T > y) override
- T [edge\\_sim](#) (attr\_type< T > x, attr\_type< T > y) override
- std::string [get\\_Instance](#) () override

### 3.3.1 Detailed Description

```
template<class T>
class euclidean< T >
```

Class that inherit form the class distance and that implements the euclidean distance

### 3.3.2 Member Function Documentation

#### 3.3.2.1 edge\_dis()

```
template<class T >
T euclidean< T >::edge_dis (
    attr_type< T > x,
    attr_type< T > y ) [override], [virtual]
```

Method to compute the distance between two edges

Implements [distance< T >](#).

#### 3.3.2.2 get\_Instance()

```
template<class T >
std::string euclidean< T >::get_Instance [override], [virtual]
```

This method returns the name of the distance used

Implements [distance< T >](#).

#### 3.3.2.3 node\_dis()

```
template<class T >
T euclidean< T >::node_dis (
    attr_type< T > x,
    attr_type< T > y ) [override], [virtual]
```

Method to compute the distance between two nodes

Implements [distance< T >](#).

### 3.3.2.4 the\_dis()

```
template<class T >
T euclidean< T >::the_dis (
    attr_type< T > x,
    attr_type< T > y )
```

function to compute the distance between two nodes or two edges.

### 3.3.2.5 the\_sim()

```
template<class T >
T euclidean< T >::the_sim (
    attr_type< T > x,
    attr_type< T > y )
```

function to compute pointwise product

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

- euclidean.h

## 3.4 GA< T > Class Template Reference

```
#include <GA.h>
```

Inheritance diagram for GA< T >:

Collaboration diagram for GA< T >:

### Public Member Functions

- [GA](#) ()=default
- [GA](#) (Distance::distances\_d)
- void [match](#) (GraphPointer< T > first\_graph, GraphPointer< T > second\_graph) override
- bool [isStable](#) (GraphPointer< T > first\_graph, GraphPointer< T > second\_graph, MatrixXd M1, MatrixXd M2, double eps)
- void [initializeMatchMatrix](#) (GraphPointer< T > x, GraphPointer< T > y)
- void [cleanup](#) (GraphPointer< T > first\_graph, GraphPointer< T > second\_graph)
- void [setAssociationGraph](#) (GraphPointer< T > first\_graph, GraphPointer< T > second\_graph)

### Additional Inherited Members

#### 3.4.1 Detailed Description

```
template<class T>
class GA< T >
```

Class that implements the Graduate assigned algorithm to match two graph

### 3.4.2 Constructor & Destructor Documentation

#### 3.4.2.1 GA() [1/2]

```
template<class T >
GA< T >::GA ( ) [default]
```

Default constructor

#### 3.4.2.2 GA() [2/2]

```
template<class T >
GA< T >::GA (
    Distance::distances _d ) [inline]
```

Constructor if you want to set a specific distance

### 3.4.3 Member Function Documentation

#### 3.4.3.1 cleanup()

```
template<class T >
void GA< T >::cleanup (
    GraphPointer< T > first_graph,
    GraphPointer< T > second_graph )
```

accessory function used by the match method to compute the matching with the hungarian algorithm in the [Munkres](#) function

#### 3.4.3.2 initializeMatchMatrix()

```
template<class T >
void GA< T >::initializeMatchMatrix (
    GraphPointer< T > x,
    GraphPointer< T > y )
```

accessory function used by the match method to setup all the variables needed

### 3.4.3.3 isStable()

```
template<class T >
bool GA< T >::isStable (
    GraphPointer< T > first_graph,
    GraphPointer< T > second_graph,
    MatrixXd M1,
    MatrixXd M2,
    double eps )
```

accessory function used by the match method to controll that all is going well

### 3.4.3.4 match()

```
template<class T >
void GA< T >::match (
    GraphPointer< T > first_graph,
    GraphPointer< T > second_graph ) [override], [virtual]
```

Method that takes as input two GraphPointer and compute the permutation vector according with the Graduate assigned algorithm. Vector that is stored in the attribute f of the class Matcher

Implements [matcher< T >](#).

### 3.4.3.5 setAssociationGraph()

```
template<class T >
void GA< T >::setAssociationGraph (
    GraphPointer< T > first_graph,
    GraphPointer< T > second_graph )
```

accessory function used by the match method to setup all the variables needed

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

- GA.h

## 3.5 gpc< T > Class Template Reference

### Public Member Functions

- [gpc](#) (const [GraphSet](#)< T > &\_gs)
- [GraphSet](#)< T > [get\\_gs](#) () const
- Eigen::Matrix< T, Eigen::Dynamic, 1 > [get\\_barycenter](#) () const
- GraphPointer< T > [get\\_barycenter\\_net](#) () const
- void [set\\_barycenter](#) (Eigen::Matrix< T, Eigen::Dynamic, Eigen::Dynamic >)
- std::tuple< Eigen::MatrixXd, Eigen::MatrixXd, Eigen::VectorXd > [est\\_pc](#) (int n\_comp, bool scale)
- void [align\\_geo](#) (const geodesic< T > &geo, bool scale, int s\_min, int s\_max)
- std::tuple< Eigen::MatrixXd, std::vector< std::map< std::pair< int, int >, attr\_type< double > > >, Eigen::VectorXd > [gpc\\_aac](#) (int max\_iterations, double tol, int n\_comp, bool scale, double s\_min, double s\_max)
- void [set\\_barycenter](#) (Eigen::Matrix< int, Eigen::Dynamic, Eigen::Dynamic > Mat)
- void [set\\_barycenter](#) (Eigen::Matrix< float, Eigen::Dynamic, Eigen::Dynamic > Mat)
- void [set\\_barycenter](#) (Eigen::Matrix< double, Eigen::Dynamic, Eigen::Dynamic > Mat)

### 3.5.1 Constructor & Destructor Documentation

#### 3.5.1.1 gpc()

```
template<class T >
gpc< T >::gpc (
    const GraphSet< T > & _gs ) [inline]
```

Constructor

### 3.5.2 Member Function Documentation

#### 3.5.2.1 align\_geo()

```
template<class T >
void gpc< T >::align_geo (
    const geodesic< T > & geo,
    bool scale,
    int s_min,
    int s_max )
```

This method aligns the graphset with respect to a given geodesic

#### 3.5.2.2 est\_pc()

```
template<class T >
std::tuple< Eigen::MatrixXd, Eigen::MatrixXd, Eigen::VectorXd > gpc< T >::est_pc (
    int n_comp,
    bool scale )
```

This method estimate the PCA with respect to a given alignment of the graphset. It is only an estimation, it is not the optimal one.

#### 3.5.2.3 get\_barycenter()

```
template<class T >
Eigen::Matrix< T, Eigen::Dynamic, 1 > gpc< T >::get_barycenter
```

Getter for the barycenter

#### 3.5.2.4 get\_barycenter\_net()

```
template<class T >
GraphPointer< T > gpc< T >::get_barycenter_net
```

Getter for the barycenter\_net



### 3.5.2.5 get\_gs()

```
template<class T >
GraphSet< T > gpc< T >::get_gs
```

Getter for the graphset

### 3.5.2.6 gpc\_aac()

```
template<class T >
std::tuple< Eigen::MatrixXd, std::vector< std::map< std::pair< int, int >, attr_type< double
> > >, Eigen::VectorXd > gpc< T >::gpc_aac (
    int max_iterations,
    double tol,
    int n_comp,
    bool scale,
    double s_min,
    double s_max )
```

This method compute the Geodesic Principal Components of the graphset with the Align All and Compute principle. It takes in input the maximum number of iterations(max\_iterations), the tollerance (tol), the number of principal components wanted to be estimated (n\_comp), a flag to indicate if you want to scale the PCA (scale) and the begin and end positionof the geodesic (?)

### 3.5.2.7 set\_barycenter()

```
template<class T >
void gpc< T >::set_barycenter (
    Eigen::Matrix< T, Eigen::Dynamic, Eigen::Dynamic > )
```

Setter of the barycenter

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

- gpc.h

## 3.6 Graph< T > Class Template Reference

```
#include <Graph.h>
```

## Public Member Functions

- [Graph](#) (bool \_oriented)
- [Graph](#) (const std::map< std::pair< int, int >, attr\_type< T >> &\_graph\_map, const bool \_oriented)
- [Graph](#) (std::map< std::pair< std::pair< int, int >, std::pair< int, int >>, double > product\_graph\_map, constructor, bool oriented)
- void [add\\_vertex](#) (const attr\_type< T > &attribute, const int id\_vertex)
- void [add\\_edge](#) (const int id\_vertex1, const int id\_vertex2, const attr\_type< T > &edge\_attribute)
- bool [is\\_oriented](#) () const
- bool [isempty](#) () const
- int [get\\_n\\_nodes](#) () const
- std::list< int > [get\\_vertices\\_id](#) () const
- std::map< std::pair< int, int >, attr\_type< T > > [get\\_graph\\_map](#) () const
- int [get\\_vertex\\_size](#) () const
- int [get\\_edge\\_size](#) () const
- std::vector< std::vector< int > > [get\\_adj](#) () const
- void [construct\\_adj](#) ()
- GraphPointer< T > [permute](#) (const std::vector< int > &f) const
- void [print\\_map](#) () const
- std::set< std::pair< int, int > > [get\\_keys](#) () const
- void [grow](#) (int size, const attr\_type< T > &new\_attribute)
- GraphPointer< T > [scale](#) (double a) const

### 3.6.1 Detailed Description

```
template<class T>
class Graph< T >
```

Class [Graph](#) is used to define a graph object

### 3.6.2 Constructor & Destructor Documentation

#### 3.6.2.1 Graph() [1/3]

```
template<class T >
Graph< T >::Graph (
    bool _oriented ) [inline]
```

Constructor

#### 3.6.2.2 Graph() [2/3]

```
template<class T >
Graph< T >::Graph (
    const std::map< std::pair< int, int >, attr_type< T >> &_graph_map,
    const bool _oriented )
```

Constructor

### 3.6.2.3 Graph() [3/3]

```
template<class T >
Graph< T >::Graph (
    std::map< std::pair< std::pair< int, int >, std::pair< int, int >>, double >
    product_graph_constructor,
    bool oriented )
```

Constructor

## 3.6.3 Member Function Documentation

### 3.6.3.1 add\_edge()

```
template<class T >
void Graph< T >::add_edge (
    const int id_vertex1,
    const int id_vertex2,
    const attr_type< T > & edge_attribute )
```

The method adds an edge to the graph

### 3.6.3.2 add\_vertex()

```
template<class T >
void Graph< T >::add_vertex (
    const attr_type< T > & attribute,
    const int id_vertex )
```

The methods adds a vertex to the graph

### 3.6.3.3 construct\_adj()

```
template<class T >
void Graph< T >::construct_adj
```

The method constructs the adjacency list of the graph and it stores the matrix in the attribute

### 3.6.3.4 get\_adj()

```
template<class T >
std::vector< std::vector< int > > Graph< T >::get_adj
```

The method returns the adjacency list of the graph

### 3.6.3.5 get\_edge\_size()

```
template<class T >
int Graph< T >::get_edge_size
```

The method returns the size of the edge attribute

### 3.6.3.6 get\_graph\_map()

```
template<class T >
std::map< std::pair< int, int >, attr_type< T > > Graph< T >::get_graph_map
```

The methods returns the map that describes the graph

### 3.6.3.7 get\_keys()

```
template<class T >
std::set< std::pair< int, int > > Graph< T >::get_keys
```

The methods returns a set that contains all the keys of the graph map, namely all the nodes couple present in the graph

### 3.6.3.8 get\_n\_nodes()

```
template<class T >
int Graph< T >::get_n_nodes
```

The methods return the number of the vertices of the graph

### 3.6.3.9 get\_vertex\_size()

```
template<class T >
int Graph< T >::get_vertex_size
```

The method returns the size of the vertex attribute

### 3.6.3.10 get\_vertices\_id()

```
template<class T >
std::list< int > Graph< T >::get_vertices_id
```

The methods returns a list of the graph vertices id

### 3.6.3.11 grow()

```
template<class T >
void Graph< T >::grow (
    int size,
    const attr_type< T > & new_attribute )
```

The method increases the size of the graph creating new vertex with a specified input, until the chosen size is reached. The new vertex created are not linked with other vertex already existing

### 3.6.3.12 is\_oriented()

```
template<class T >
bool Graph< T >::is_oriented
```

The methods returns the orientation of the graph

### 3.6.3.13 isempty()

```
template<class T >
bool Graph< T >::isempty
```

The methods returns a bool that indicates if the graph is empty

### 3.6.3.14 permute()

```
template<class T >
GraphPointer< T > Graph< T >::permute (
    const std::vector< int > & f ) const
```

The method returns the permuted graph given the permutation to apply

### 3.6.3.15 print\_map()

```
template<class T >
void Graph< T >::print_map
```

The method prints the graph in the map form

### 3.6.3.16 scale()

```
template<class T >
GraphPointer< T > Graph< T >::scale (
    double a ) const
```

The method return a graph that has got multiplied attribute by the input constant

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

- gpc.h
- Graph.h

## 3.7 GraphSet< T > Class Template Reference

```
#include <GraphSet.h>
```

### Public Member Functions

- **GraphSet** (bool \_oriented)
- **GraphSet** (const std::vector< std::map< std::pair< int, int >, attr\_type< T >>> &graph\_maps, const bool orientation)
- **GraphSet** (const std::vector< std::map< std::pair< int, int >, attr\_type< T >>> &graph\_maps, const bool orientation, const Matcher::matchers \_m, const Distance::distances \_d)
- bool **is\_oriented** () const
- void **add\_graph** (GraphPointer< T > graph)
- std::vector< GraphPointer< T > > **get\_graphset** () const
- std::vector< std::map< std::pair< int, int >, attr\_type< T >>> **get\_graphset\_maps** () const
- std::vector< GraphPointer< T > > **get\_aligned\_GraphSet** () const
- std::vector< std::map< std::pair< int, int >, attr\_type< T >>> **get\_aligned\_GraphSet\_maps** () const
- std::vector< std::vector< int > > **get\_permutation\_vector** () const
- void **set\_permutation\_vector** (int index, std::vector< int > p)
- GraphPointer< T > **get\_mean** () const
- Matcher::matchers **get\_matcher** () const
- void **set\_match** (const Matcher::matchers \_m)
- Distance::distances **get\_distance** () const
- void **set\_distance** (const Distance::distances \_d)
- int **get\_n\_max** () const
- int **get\_v\_attr\_max** () const
- int **get\_e\_attr\_max** () const
- void **align** (GraphPointer< T > g)
- **GraphSet**< T > **permuted\_graphset** () const
- void **save\_aligned** ()
- Eigen::Matrix< T, Eigen::Dynamic, Eigen::Dynamic > **to\_matrix\_with\_attr** (bool aligned) const
- GraphPointer< T > **est** (const GraphPointer< T > &m1) const
- void **mean\_aac** (int max\_iteration, double tol)
- void **read\_from\_text** (std::string file\_name)

### 3.7.1 Detailed Description

```
template<class T>
class GraphSet< T >
```

Class **GraphSet** defines a set of graph. template T defines the type of data that is contained in the attributes A set of graph is defined as a vector of pointers to graph objects

### 3.7.2 Constructor & Destructor Documentation

### 3.7.2.1 GraphSet() [1/2]

```
template<class T >
GraphSet< T >::GraphSet (
    const std::vector< std::map< std::pair< int, int >, attr_type< T >>> & graph_↵
maps,
    const bool orientation )
```

Constructor

### 3.7.2.2 GraphSet() [2/2]

```
template<class T >
GraphSet< T >::GraphSet (
    const std::vector< std::map< std::pair< int, int >, attr_type< T >>> & graph_↵
maps,
    const bool orientation,
    const Matcher::matchers _m,
    const Distance::distances _d )
```

Constructor

## 3.7.3 Member Function Documentation

### 3.7.3.1 add\_graph()

```
template<class T >
void GraphSet< T >::add_graph (
    GraphPointer< T > graph )
```

The method adds a graph to the [GraphSet](#). A graph can be added to the graphset only if it has the same orientation as the graphset.

### 3.7.3.2 align()

```
template<class T >
void GraphSet< T >::align (
    GraphPointer< T > g )
```

The methods saves in the permutation\_vector the permutations that have to be done to align the [GraphSet](#) with the specified input

### 3.7.3.3 est()

```
template<class T >
GraphPointer< T > GraphSet< T >::est (
    const GraphPointer< T > & m1 ) const
```

The methods return an estimate, but not the final value, of the mean of the [GraphSet](#)

### 3.7.3.4 get\_aligned\_GraphSet()

```
template<class T >
std::vector< GraphPointer< T > > GraphSet< T >::get_aligned_GraphSet
```

The method returns the vector of aligned graphs

### 3.7.3.5 get\_aligned\_GraphSet\_maps()

```
template<class T >
std::vector< std::map< std::pair< int, int >, attr_type< T > > > GraphSet< T >::get_↵
aligned_GraphSet_maps
```

The method returns a vector of maps, related to the aligned graphs

### 3.7.3.6 get\_distance()

```
template<class T >
Distance::distances GraphSet< T >::get_distance
```

The method returns the enum correspondent to the distance used in the Graphset

### 3.7.3.7 get\_e\_attr\_max()

```
template<class T >
int GraphSet< T >::get_e_attr_max
```

The methods returns the maximum dimension of edge attributes among all graphs

### 3.7.3.8 get\_graphset()

```
template<class T >
std::vector< GraphPointer< T > > GraphSet< T >::get_graphset
```

The method returns the vector of graph pointers

### 3.7.3.9 get\_graphset\_maps()

```
template<class T >
std::vector< std::map< std::pair< int, int >, attr_type< T > > > GraphSet< T >::get_↵
graphset_maps
```

The method returns a vector of maps, related to the graphs contained in the [GraphSet](#)

### 3.7.3.10 get\_matcher()

```
template<class T >
Matcher::matchers GraphSet< T >::get_matcher
```

The method returns the enum correspondent to the matcher used in the Graphset



### 3.7.3.11 get\_mean()

```
template<class T >
GraphPointer< T > GraphSet< T >::get_mean
```

The method returns the mean of the [GraphSet](#)

### 3.7.3.12 get\_n\_max()

```
template<class T >
int GraphSet< T >::get_n_max
```

The method returns the maximum number of nodes that a graph in the [GraphSet](#) has.

### 3.7.3.13 get\_permutation\_vector()

```
template<class T >
std::vector< std::vector< int > > GraphSet< T >::get_permutation_vector
```

The method returns the vector of permutation

### 3.7.3.14 get\_v\_attr\_max()

```
template<class T >
int GraphSet< T >::get_v_attr_max
```

The methods returns the maximum dimension of vertex attributes among all graphs

### 3.7.3.15 is\_oriented()

```
template<class T >
bool GraphSet< T >::is_oriented
```

The methods return a bool that indicates if the graphs in the graphset are oriented or not

### 3.7.3.16 mean\_aac()

```
template<class T >
void GraphSet< T >::mean_aac (
    int max_iteration,
    double tol )
```

The methods returns the Fréchet Mean of the [GraphSet](#)

### 3.7.3.17 permuted\_graphset()

```
template<class T >
GraphSet< T > GraphSet< T >::permuted_graphset
```

The method returns the permuted graphset correspondent to the permutation vector

**3.7.3.18 read\_from\_text()**

```
template<class T >
void GraphSet< T >::read_from_text (
    std::string file_name )
```

This methods reads a graph from a corrected formatted text file

**3.7.3.19 save\_aligned()**

```
template<class T >
void GraphSet< T >::save_aligned
```

The method permutes the graphset and saves the permuted graphs obtained in the aligned\_graphset

**3.7.3.20 set\_distance()**

```
template<class T >
void GraphSet< T >::set_distance (
    const Distance::distances _d )
```

The method sets the enum of the distance

**3.7.3.21 set\_match()**

```
template<class T >
void GraphSet< T >::set_match (
    const Matcher::matchers _m )
```

The methods set the enum of the matcher

**3.7.3.22 set\_permutation\_vector()**

```
template<class T >
void GraphSet< T >::set_permutation_vector (
    int index,
    std::vector< int > p )
```

The method allows to modify one component of the permutation vector with the permutation given as input

**3.7.3.23 to\_matrix\_with\_attr()**

```
template<class T >
Eigen::Matrix< T, Eigen::Dynamic, Eigen::Dynamic > GraphSet< T >::to_matrix_with_attr (
    bool aligned ) const
```

The methods return a matrix associated to the graphset Every row corresponds to a graph Every columns correspond to a connection of the graph

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

- GraphSet.h

## 3.8 ID< T > Class Template Reference

```
#include <ID.h>
```

Inheritance diagram for ID< T >:

Collaboration diagram for ID< T >:

### Public Member Functions

- [ID](#) ()=default
- [ID](#) (Distance::distances \_d)
- void [match](#) (GraphPointer< T > first\_graph, GraphPointer< T > second\_graph) override

### Additional Inherited Members

#### 3.8.1 Detailed Description

```
template<class T>  
class ID< T >
```

Class that implements the [ID](#) matcher

#### 3.8.2 Constructor & Destructor Documentation

##### 3.8.2.1 ID() [1/2]

```
template<class T >  
ID< T >::ID ( ) [default]
```

Default constructor

##### 3.8.2.2 ID() [2/2]

```
template<class T >  
ID< T >::ID (   
    Distance::distances _d ) [inline]
```

Constructor if you want to set a specific distance

#### 3.8.3 Member Function Documentation

### 3.8.3.1 match()

```
template<class T >
void ID< T >::match (
    GraphPointer< T > first_graph,
    GraphPointer< T > second_graph ) [override], [virtual]
```

Method that takes as input two GraphPointer and compute the permutation vector that is the identity. Vector that is stored in the attribute f of the class Matcher

Implements [matcher< T >](#).

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

- ID.h

## 3.9 matcher< T > Class Template Reference

```
#include <matcher.h>
```

Inheritance diagram for matcher< T >:

### Public Member Functions

- [matcher](#) ()
- [matcher](#) (Distance::distances \_distance)
- virtual void [match](#) (GraphPointer< T > first\_graph, GraphPointer< T > second\_graph)=0
- double [the\\_dis](#) (GraphPointer< T > X, GraphPointer< T > Y)
- std::string [get\\_distance](#) ()
- std::vector< int > [get\\_f](#) ()
- double [get\\_dist](#) ()
- void [set\\_dist](#) (Distance::distances \_d)

### Protected Attributes

- double [dist](#)
- std::vector< int > [f](#)
- DistancePointer< T > [distance](#)

### 3.9.1 Detailed Description

```
template<class T>
class matcher< T >
```

Base abstract class for matchers

## 3.9.2 Constructor & Destructor Documentation

### 3.9.2.1 matcher() [1/2]

```
template<class T >  
matcher< T >::matcher ( ) [inline]
```

Constructor

### 3.9.2.2 matcher() [2/2]

```
template<class T >  
matcher< T >::matcher (   
    Distance::distances _distance ) [inline]
```

Constructor

## 3.9.3 Member Function Documentation

### 3.9.3.1 get\_dist()

```
template<class T >  
double matcher< T >::get_dist
```

Getter for dist

### 3.9.3.2 get\_distance()

```
template<class T >  
std::string matcher< T >::get_distance
```

The method returns the name of the distance used

### 3.9.3.3 get\_f()

```
template<class T >  
std::vector< int > matcher< T >::get_f
```

Getter for f

### 3.9.3.4 match()

```
template<class T >
virtual void matcher< T >::match (
    GraphPointer< T > first_graph,
    GraphPointer< T > second_graph ) [pure virtual]
```

The methods matches two graphs, saving in f the best permutation

Implemented in [GA< T >](#), and [ID< T >](#).

### 3.9.3.5 set\_dist()

```
template<class T >
void matcher< T >::set_dist (
    Distance::distances _d )
```

Setter of distance pointer

### 3.9.3.6 the\_dis()

```
template<class T >
double matcher< T >::the_dis (
    GraphPointer< T > X,
    GraphPointer< T > Y )
```

The methods sets dist equal to distance of the graphs X and Y based on the permutation f. In order to have the right distance, you have to be sure that in f there is the right permutation vector, so before running this function you have to do the matching.

## 3.9.4 Member Data Documentation

### 3.9.4.1 dist

```
template<class T >
double matcher< T >::dist [protected]
```

When match method is called, in this variable is stored the distance between the two matched graphs

### 3.9.4.2 distance

```
template<class T >
DistancePointer<T> matcher< T >::distance [protected]
```

Poiter to the distance that is used in the match

## 3.9.4.3 f

```
template<class T >
std::vector<int> matcher< T >::f [protected]
```

Permutation of X to get close to Y

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

- matcher.h

## 3.10 Matcher::matcherHolder&lt; T &gt; Class Template Reference

## Public Member Functions

- **matcherHolder** (matchers m\_id)
- void **setMatcher** (matchers m\_id)

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

- MatcherFactory.h

## 3.11 Munkres&lt; T &gt; Class Template Reference

## Public Member Functions

- Matrix< T > **pad\_matrix** (Matrix< T > &Matrix, T pad\_value)
- std::vector< std::pair< int, int > > **compute** (Matrix< T > cost\_matrix)
- Matrix< int > **\_make\_matrix** (int n, T value)
- void **\_clear\_covers** ()
- std::pair< int, int > **\_find\_a\_zero** (int i0, int j0)
- T **\_find\_smallest** ()
- int **\_find\_star\_in\_row** (int row)
- int **\_find\_star\_in\_col** (int col)
- int **\_find\_prime\_in\_row** (int row)
- void **\_convert\_path** (std::vector< std::vector< int > > \_path, int count)
- void **\_erase\_primes** ()
- int **step1** ()
- int **step2** ()
- int **step3** ()
- int **step4** ()
- int **step5** ()
- int **step6** ()

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

- Munkres.h





# Index

- add\_edge
  - Graph< T >, 13
- add\_graph
  - GraphSet< T >, 17
- add\_vertex
  - Graph< T >, 13
- align
  - GraphSet< T >, 17
- align\_geo
  - gpc< T >, 10
- cleanup
  - GA< T >, 8
- construct\_adj
  - Graph< T >, 13
- dist
  - matcher< T >, 24
- distance
  - matcher< T >, 24
- distance< T >, 5
- Distance::distanceHolder< T >, 5
- edge\_dis
  - euclidean< T >, 6
- est
  - GraphSet< T >, 17
- est\_pc
  - gpc< T >, 10
- euclidean< T >, 5
  - edge\_dis, 6
  - get\_Instance, 6
  - node\_dis, 6
  - the\_dis, 6
  - the\_sim, 7
- f
  - matcher< T >, 24
- GA
  - GA< T >, 8
- GA< T >, 7
  - cleanup, 8
  - GA, 8
  - initializeMatchMatrix, 8
  - isStable, 8
  - match, 9
  - setAssociationGraph, 9
- get\_adj
  - Graph< T >, 13
- get\_aligned\_GraphSet
  - GraphSet< T >, 17
- get\_aligned\_GraphSet\_maps
  - GraphSet< T >, 18
- get\_barycenter
  - gpc< T >, 10
- get\_barycenter\_net
  - gpc< T >, 10
- get\_dist
  - matcher< T >, 23
- get\_distance
  - GraphSet< T >, 18
  - matcher< T >, 23
- get\_e\_attr\_max
  - GraphSet< T >, 18
- get\_edge\_size
  - Graph< T >, 13
- get\_f
  - matcher< T >, 23
- get\_graph\_map
  - Graph< T >, 14
- get\_graphset
  - GraphSet< T >, 18
- get\_graphset\_maps
  - GraphSet< T >, 18
- get\_gs
  - gpc< T >, 10
- get\_Instance
  - euclidean< T >, 6
- get\_keys
  - Graph< T >, 14
- get\_matcher
  - GraphSet< T >, 18
- get\_mean
  - GraphSet< T >, 18
- get\_n\_max
  - GraphSet< T >, 19
- get\_n\_nodes
  - Graph< T >, 14
- get\_permutation\_vector
  - GraphSet< T >, 19
- get\_v\_attr\_max
  - GraphSet< T >, 19
- get\_vertex\_size
  - Graph< T >, 14
- get\_vertices\_id
  - Graph< T >, 14
- gpc
  - gpc< T >, 10
- gpc< T >, 9

- align\_geo, 10
- est\_pc, 10
- get\_barycenter, 10
- get\_barycenter\_net, 10
- get\_gs, 10
- gpc, 10
- gpc\_aac, 11
- set\_barycenter, 11
- gpc\_aac
  - gpc< T >, 11
- Graph
  - Graph< T >, 12
- Graph< T >, 11
  - add\_edge, 13
  - add\_vertex, 13
  - construct\_adj, 13
  - get\_adj, 13
  - get\_edge\_size, 13
  - get\_graph\_map, 14
  - get\_keys, 14
  - get\_n\_nodes, 14
  - get\_vertex\_size, 14
  - get\_vertices\_id, 14
  - Graph, 12
  - grow, 14
  - is\_oriented, 15
  - isempty, 15
  - permute, 15
  - print\_map, 15
  - scale, 15
- GraphSet
  - GraphSet< T >, 16, 17
- GraphSet< T >, 16
  - add\_graph, 17
  - align, 17
  - est, 17
  - get\_aligned\_GraphSet, 17
  - get\_aligned\_GraphSet\_maps, 18
  - get\_distance, 18
  - get\_e\_attr\_max, 18
  - get\_graphset, 18
  - get\_graphset\_maps, 18
  - get\_matcher, 18
  - get\_mean, 18
  - get\_n\_max, 19
  - get\_permutation\_vector, 19
  - get\_v\_attr\_max, 19
  - GraphSet, 16, 17
  - is\_oriented, 19
  - mean\_aac, 19
  - permuted\_graphset, 19
  - read\_from\_text, 19
  - save\_aligned, 20
  - set\_distance, 20
  - set\_match, 20
  - set\_permutation\_vector, 20
  - to\_matrix\_with\_attr, 20
- grow
  - Graph< T >, 14
- ID
  - ID< T >, 21
- ID< T >, 21
  - ID, 21
  - match, 21
- initializeMatchMatrix
  - GA< T >, 8
- is\_oriented
  - Graph< T >, 15
  - GraphSet< T >, 19
- isempty
  - Graph< T >, 15
- isStable
  - GA< T >, 8
- match
  - GA< T >, 9
  - ID< T >, 21
  - matcher< T >, 23
- matcher
  - matcher< T >, 23
- matcher< T >, 22
  - dist, 24
  - distance, 24
  - f, 24
  - get\_dist, 23
  - get\_distance, 23
  - get\_f, 23
  - match, 23
  - matcher, 23
  - set\_dist, 24
  - the\_dis, 24
- Matcher::matcherHolder< T >, 25
- mean\_aac
  - GraphSet< T >, 19
- Munkres< T >, 25
- node\_dis
  - euclidean< T >, 6
- permute
  - Graph< T >, 15
- permuted\_graphset
  - GraphSet< T >, 19
- print\_map
  - Graph< T >, 15
- read\_from\_text
  - GraphSet< T >, 19
- save\_aligned
  - GraphSet< T >, 20
- scale
  - Graph< T >, 15
- set\_barycenter
  - gpc< T >, 11
- set\_dist
  - matcher< T >, 24

set\_distance  
    GraphSet< T >, [20](#)  
set\_match  
    GraphSet< T >, [20](#)  
set\_permutation\_vector  
    GraphSet< T >, [20](#)  
setAssociationGraph  
    GA< T >, [9](#)  
  
the\_dis  
    euclidean< T >, [6](#)  
    matcher< T >, [24](#)  
the\_sim  
    euclidean< T >, [7](#)  
to\_matrix\_with\_attr  
    GraphSet< T >, [20](#)