RTG for efficient high-quality motion planning - Reference manual

Code by Michal Kleinbort and Oren Salzman

Generated by Doxygen 1.7.4
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Class Index</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Class Hierarchy</td>
<td>1</td>
</tr>
<tr>
<td>2 Class Index</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Class List</td>
<td>3</td>
</tr>
<tr>
<td>3 Class Documentation</td>
<td>5</td>
</tr>
<tr>
<td>3.1 RTG::BitSetPairsStruct Class Reference</td>
<td>5</td>
</tr>
<tr>
<td>3.1.1 Detailed Description</td>
<td>6</td>
</tr>
<tr>
<td>3.2 RTG::SingleGrid&lt;_PointDT, _SqrdDistanceFuncT&gt;::Cell_ind_hash Struct Reference</td>
<td>6</td>
</tr>
<tr>
<td>3.2.1 Detailed Description</td>
<td>6</td>
</tr>
<tr>
<td>3.3 RTG::EuclideanSqrdDistanceFunc&lt;_T&gt; Class Template Reference</td>
<td>6</td>
</tr>
<tr>
<td>3.3.1 Detailed Description</td>
<td>7</td>
</tr>
<tr>
<td>3.4 RTG::MultiRobotSqrdDistanceFunc&lt;_T&gt; Class Template Reference</td>
<td>7</td>
</tr>
<tr>
<td>3.4.1 Detailed Description</td>
<td>7</td>
</tr>
<tr>
<td>3.5 RTG::NearestNeighborsRTG&lt;_PointDT, _SqrdDistanceFuncT&gt; Class Template Reference</td>
<td>8</td>
</tr>
<tr>
<td>3.5.1 Detailed Description</td>
<td>9</td>
</tr>
<tr>
<td>3.6 RTG::PairsStruct Class Reference</td>
<td>9</td>
</tr>
<tr>
<td>3.6.1 Detailed Description</td>
<td>10</td>
</tr>
<tr>
<td>3.7 RTG::Point_d Class Reference</td>
<td>10</td>
</tr>
<tr>
<td>3.7.1 Detailed Description</td>
<td>11</td>
</tr>
<tr>
<td>3.8 RTG::Random_utils&lt;Point_d&gt; Class Template Reference</td>
<td>11</td>
</tr>
<tr>
<td>3.8.1 Detailed Description</td>
<td>11</td>
</tr>
<tr>
<td>3.9 RTG::RandomMatrixGen Class Reference</td>
<td>12</td>
</tr>
<tr>
<td>3.9.1 Detailed Description</td>
<td>12</td>
</tr>
</tbody>
</table>
Chapter 1

Class Index

1.1 Class Hierarchy

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

RTG::SingleGrid<_PointDT, _SqrdDistanceFuncT>::Cell_ind_hash . . . . . 6
RTG::EuclideanSqrdDistanceFunc<_T> . . . . . . . . . . . . . . . . . . . 6
RTG::MultiRobotSqrdDistanceFunc<_T> . . . . . . . . . . . . . . . . . . . 7
RTG::NearestNeighborsRTG<_PointDT, _SqrdDistanceFuncT> . . . . . 8
RTG::PairsStruct . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
   RTG::BitSetPairsStruct . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
   RTG::VecOfIntSetsPairsStruct . . . . . . . . . . . . . . . . . . . . . . . 14
RTG::Point_d . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
RTG::Random_utils<_Point_d> . . . . . . . . . . . . . . . . . . . . . . . . 11
RTG::RandomMatrixGen . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
RTG::RTG_utils . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
RTG::SingleGrid<_PointDT, _SqrdDistanceFuncT> . . . . . . . . . . . . . 13
2.1 Class List

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

- RTG::BitSetPairsStruct (A class for storing the already reported pairs, which uses a bitset of the pairs) .............................. 5
- RTG::SingleGrid<_PointDT, _SqrdDistanceFuncT>::Cell_ind_hash (Hash function for cell indices) .............................. 6
- RTG::EuclideanSqrdDistanceFunc<_T> (A functor for computing the Euclidean squared distance of two given points) ............ 6
- RTG::MultiRobotSqrdDistanceFunc<_T> (A functor for computing the Multi Robot squared distance of two given points, representing two configurations) ............................................. 7
- RTG::NearestNeighborsRTG<_PointDT, _SqrdDistanceFuncT> (Randomly Translated Grids, a data structure for nearest neighbor search. Especially for all r-nearest neighbors) .......................... 8
- RTG::PairsStruct (An abstract class for the Pairs data structure) .......... 9
- RTG::Point_d (A class used for defining a d-dimensional point) .......... 10
- RTG::Random_utils<_Point_d> (A class used for generating random d-dimensional points) .................................................... 11
- RTG::RandomMatrixGen (A class used for generating a random d x d rotation matrix) .................................................... 12
- RTG::RTG_utils (Class of static methods) ....................................... 12
- RTG::SingleGrid<_PointDT, _SqrdDistanceFuncT> (A class used for defining a single grid) .................................................... 13
- RTG::VecOfIntSetsPairsStruct (A class for storing the already reported pairs, which uses a vector of unordered-sets (hash maps)) .......... 14
Chapter 3

Class Documentation

3.1 RTG::BitSetPairsStruct Class Reference

A class for storing the already reported pairs, which uses a bitset of the pairs.

```cpp
#include <bitset_pairs_struct.h>
```

Inheritance diagram for RTG::BitSetPairsStruct:

```
RTG::PairsStruct
   `-- RTG::BitSetPairsStruct
```

Public Types

- `typedef boost::dynamic_bitset Pairs`

Public Member Functions

- **BitSetPairsStruct** (unsigned int n, bool add_once=true)
  
  Constructor.

- virtual bool **pair_exists** (int i, int j) const
  
  Check whether a pair exists in the structure.

- virtual void **add_pair** (int i, int j)
  
  Add pair to the structure.

- virtual void **clear** (void)
  
  Clear the datastructure.

- virtual void **neighbors_of_i** (int i, std::vector<int> &neighborsVec) const
  
  places the indices of the neighbors of index i in the given vector
• virtual void all_pairs (std::vector<std::pair<int, int>> &pairsVec) const
  places the reported pairs of indices in the given vector

3.1.1 Detailed Description

A class for storing the already reported pairs, which uses a bitset of the pairs. The data structure is a bitset, where pair (i,j) is mapped to an index k. The documentation for this class was generated from the following file:

• bitset_pairs_struct.h

3.2 RTG::SingleGrid<_PointDT, _SqrdDistanceFuncT>::Cell_ind_hash Struct Reference

hash function for cell indices
#include <single_grid.h>

Public Member Functions

• std::size_t operator() (Cell_index const &cell) const

3.2.1 Detailed Description

template<typename _PointDT = Point_d, typename _SqrdDistanceFuncT = EuclideanSqrdDistanceFunc<_PointDT>> struct RTG::SingleGrid<_PointDT, _SqrdDistanceFuncT>::Cell_ind_hash

hash function for cell indices
The documentation for this struct was generated from the following file:

• single_grid.h

3.3 RTG::EuclideanSqrdDistanceFunc<T> Class Template Reference

A functor for computing the Euclidean squared distance of two given points.
#include <distance_functors.h>
Public Member Functions

- **EuclideanSqrdDistanceFunc** (unsigned int dim)
- double **operator()** (const _T &data1, const _T &data2) const

3.3.1 Detailed Description

template<typename _T> class RTG::EuclideanSqrdDistanceFunc<_T>

A functor for computing the Euclidean squared distance of two given points.

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

- distance_functors.h

3.4 RTG::MultiRobotSqrdDistanceFunc<_T> Class Template Reference

A functor for computing the Multi Robot squared distance of two given points, representing two configurations.

#include <distance_functors.h>

Public Member Functions

- **MultiRobotSqrdDistanceFunc** (unsigned int num_of_robots, unsigned int dim=3)
- double **operator()** (const _T &data1, const _T &data2) const

3.4.1 Detailed Description

template<typename _T> class RTG::MultiRobotSqrdDistanceFunc<_T>

A functor for computing the Multi Robot squared distance of two given points, representing two configurations.

Each configuration represents the position of a set of robots. The distance according to multi-robot metric is the sum of distances traveled by each robot. This functor computes the squared distance of two configurations.

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

- distance_functors.h
3.5 RTG::NearestNeighborsRTG<typename PointDT, SqrdDistanceFuncT>
Class Template Reference

Randomly Translated Grids, a data structure for nearest neighbor search. Especially for all r-nearest neighbors.

#include <nearest_neighbors_RTG.h>

Public Types

- typedef typename PointDT PointD
- typedef typename SqrdDistanceFuncT SqrdDistanceFunc
- typedef SingleGrid<PointD, SqrdDistanceFunc> RTGSingleGrid

Public Member Functions

- NearestNeighborsRTG(double r, double cell_size, unsigned int grid_num, unsigned int num_samples, unsigned int dim, SqrdDistanceFunc *pDistFunc=NULL, PairsStruct *pairs=NULL, bool all_nn_only=true, int seed=1)
  Constructor.
- ~NearestNeighborsRTG() (void)
  Destructor.
- void set_pairs_struct(PairsStruct *pairs)
- void add(const PointD &data)
  Adds a data point into the existing array of points.
- void add(const std::vector<PointD> &data)
  Adds a vector of points into the existing array of points.
- void nearestR(int data_index, std::vector<PointD> &nbh)
  A method retrieving the r-nearest-neighbors of the point whose index in the data points array is data_index. The neighbors are inserted into the nbh vector of data points. This method is not supported when all_nn_only is set to true.
- void nearestR(int data_index, std::vector<int> &nbh_ind)
  A method retrieving the indices of r-nearest-neighbors of the point whose index in the data points array is data_index. The indices of neighbors are inserted into the nbh_ind vector of integers. This method is not supported when all_nn_only is set to true.
- void all_nearestR(std::vector<std::pair<PointD, PointD> > &nbh_pairs)
  A method retrieving all pairs r-nearest-neighbors. The pairs of data points are inserted into the nbh_pairs vector of pairs of points.
- void all_nearestR(std::vector<std::pair<int, int> > &nbh_ind_pairs)
  A method retrieving all pairs of indices of r-nearest-neighbors points. The pairs of indices are inserted into the nbh_ind_pairs vector of pairs of points indices.
- std::size_t size() const
  A method retrieving the number of data points in the structure.
3.6 RTG::PairsStruct Class Reference

3.5.1 Detailed Description

template<typename _PointDT = Point, typename _SqrdDistanceFuncT = EuclideanSqrdDistanceFunc<_PointDT>> class RTG::NearestNeighborsRTG<_PointDT, _SqrdDistanceFuncT>

Randomly Translated Grids, a data structure for nearest neighbor search. Especially for all r-nearest neighbors.

Based on:


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

- nearest_neighbors_RTG.h

3.6 RTG::PairsStruct Class Reference

An abstract class for the Pairs data structure.

#include <pairs.h>

Inheritance diagram for RTG::PairsStruct:

```
RTG::PairsStruct
  ↓
RTG::BitSetPairsStruct  RTG::VecOfIntSetsPairsStruct
```

Public Member Functions

- **PairsStruct** (unsigned int n, bool add_once=true)
  Constructor.
- virtual ~PairsStruct (void)
  Destructor.
- virtual bool pair_exists (int i, int j) const =0
  Check whether a pair exists in the structure.
- virtual void add_pair (int i, int j)=0
  Add pair to the structure.
- virtual void neighbors_of_i (int i, std::vector<int> &neighborsVec) const =0
  Output the neighbors of the i-th data point.
- virtual void all_pairs (std::vector<std::pair<int, int>> &pairsVec) const =0
  Output all neighboring pairs.
- virtual void clear (void)
  Clear the datastructure.
Protected Attributes

- unsigned int num_of_points_
  number of data points
- bool add_once_
  A flag indicating whether to add a pair (i,j) once or twice (add (i,j) and (j,i))

3.6.1 Detailed Description

An abstract class for the Pairs data structure.
The documentation for this class was generated from the following file:

- pairs.h

3.7 RTG::Point_d Class Reference

A class used for defining a d-dimensional point.
#include <Point_d.h>

Public Member Functions

- template<class InputIterator>
  Point_d_(unsigned int dim, InputIterator first, InputIterator last)
  Constructor given an input iterator.
- Point_d_(const Point_d_ &p)
  Copy constructor.
- ~Point_d_ ()
  Destructor.
- double & operator[](unsigned int idx)
  retrieves the i-th coordinate of the d-dimensional point This method must be implemented
- const double & operator[](unsigned int idx) const
  retrieves the i-th coordinate of the d-dimensional point This method must be implemented
- unsigned int dimension () const
  returns the dimension of the point

Friends

- std::ostream & operator<<(std::ostream &output, const Point_d_ &p)
  operator<< for outputing the point
3.7.1 Detailed Description

A class used for defining a d-dimensional point.

A single transformed grid that defines a partition of the data into cells of a given size. Nearby points that lie in the same cell are reported.

**It is possible to use different classes for d-dimensional points.**

However, these classes must implement the following methods:
- `double& operator[](unsigned int idx)`
- `const double& operator[](unsigned int idx) const`

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

- `Point_d.h`

3.8 RTG::Random_utils< Point_d > Class Template Reference

A class used for generating random d dimensional points.

```cpp
#include <Random_utils.h>
```

### Public Types

- `typedef boost::random::mt19937 MTEng`
- `typedef boost::random::uniform_real_distribution< double > UniformDist`

### Public Member Functions

- `Random_utils (int seed=1, double min=-1, double max=1)`
  * Constructor.
- `double get_random_num () const`
  * returning a random real number according to a uniform distribution
- `Point_d get_random_point (int dim) const`
  * returning a random point in dim dimensions.
- `void get_random_vec (int dim, std::vector< double > &vec) const`
  * updating the given vec with dim random elements

3.8.1 Detailed Description

```cpp
template< typename Point_d > class RTG::Random_utils< Point_d >
```

A class used for generating random d dimensional points.

The documentation for this class was generated from the following file:
• Random_utils.h

3.9 RTG::RandomMatrixGen Class Reference

A class used for generating a random d x d rotation matrix.
#include <Random_matrix_gen.h>

Public Types

• typedef Eigen::MatrixXd MatrixXd
• typedef Eigen::VectorXd VectorXd

Public Member Functions

• boost::shared_ptr<MatrixXd> get_random_rotation_matrix (int d) const
  returns a shared pointer of a d x d random rotation matrix

3.9.1 Detailed Description

A class used for generating a random d x d rotation matrix.
The documentation for this class was generated from the following file:

• Random_matrix_gen.h

3.10 RTG::RTG_utils Class Reference

Class of static methods.
#include <RTG_utils.h>

Static Public Member Functions

• static double calculate_radius (int n, double d, double mu=1.0, double vol=1.0)
  returns the radius for r-NN based on the number n of points, the ratio between the free
  space volume and c-space volume, and the dimension d mu is the free space volume
  out of the c-space which is 1
• template<typename _T>
  static double get_bbox_volume (std::vector<_T> &points, unsigned int dim)
  computes the volume of the bounding box of a given point-set
3.11 RTG::SingleGrid::<PointDT, SqrdDistanceFuncT> Class Template Reference

3.10.1 Detailed Description

Class of static methods.

3.10.2 Member Function Documentation

3.10.2.1 static double RTG::RTG_utils::calculate_radius ( int n, double d, double mu = 1.0, double vol = 1.0 ) [inline, static]

returns the radius for r-NN based on the number n of points, the ratio between the free space volume and c-space volume, and the dimension d mu is the free space volume out of the c-space which is 1

Based on:


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

- RTG_utils.h

3.11 RTG::SingleGrid::<PointDT, SqrdDistanceFuncT> Class Template Reference

A class used for defining a single grid.

#include <single_grid.h>

Classes

- struct Cell_ind_hash
  
  hash function for cell indices

Public Types

- typedef _SqrdDistanceFuncT SqrdDistanceFunc
- typedef _PointDT PointD

Public Member Functions

- SingleGrid (std::vector<double> &shift, double cell_size, double r_sq, const std::vector<PointD> &points, unsigned int dim, boost::shared_ptr<Eigen::MatrixXd> rotation_matrix, SqrdDistanceFunc *pSqrDistFunc)
14 Class Documentation

constructor of a single grid with rotation

- **SingleGrid** (std::vector< double > shift, double cell_size, double r_sq, const std::vector<PointD> &points, unsigned int dim, SqrdDistanceFunc *pSqrDistFunc)

constructor of a single grid without rotation

- ~SingleGrid ()

destructor

- void find_all_pairs_single (PairsStruct *pairs)

  find all r-near pairs and update the auxiliary pairs structure

Protected Types

- typedef std::vector<int> **Cell_index**

- typedef int **Compact_cell_index**

- typedef boost::unordered_map<Cell_index, Compact_cell_index, Cell_ind_hash> **Cell_index_to_compact_cell_index**

  The type of the map between cell index to compact ones.

- typedef std::vector<std::vector<int>> **Compact_cell_index_to_int_vec**

  The type of the map between compact cell index to vector of integers (the indices of the data points)

3.11.1 Detailed Description

```
template<typename PointDT = PointD, typename SqrdDistanceFuncT = EuclideanSqrdDistanceFunc<
                 PointDT, SqrdDistanceFuncT > > class RTG::SingleGrid<PointDT, SqrdDistanceFuncT >
```

A class used for defining a single grid.

A single transformed grid that defines a partition of the data into cells of a given size. Nearby points that lie in the same cell are reported.

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

- single_grid.h

3.12 RTG::VecOfIntSetsPairsStruct Class Reference

A class for storing the already reported pairs, which uses a vector of unordered-sets (hash maps).

```
#include <vec_of_intsets_pairs_struct.h>
```

Inheritance diagram for RTG::VecOfIntSetsPairsStruct:
3.12 RTG::VecOfIntSetsPairsStruct Class Reference

Public Types

- typedef std::unordered_set<int> Int_set
- typedef std::vector<Int_set> Pairs

Public Member Functions

- VecOfIntSetsPairsStruct (unsigned int n, bool add_once=true) Constructor.
- virtual bool pair_exists (int i, int j) const Check whether a pair exists in the structure.
- virtual void add_pair (int i, int j) Add pair to the structure.
- virtual void clear (void) Clear the datastructure.
- virtual void neighbors_of_i (int i, std::vector<int> &neighborsVec) const places the indices of the neighbors of index i in the given vector
- virtual void all_pairs (std::vector<std::pair<int, int>> &pairsVec) const places the reported pairs of indices in the given vector

3.12.1 Detailed Description

A class for storing the already reported pairs, which uses a vector of unordered-sets (hash maps).

The data structure is a vector of unordered-sets. The i-th entry of the vector stores the set of reported neighbors of the i-th data point.

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

- vec_of_intsets_pairs.struct.h
Index

calculate_radius
   RTG::RTG_utils, 13

RTG::BitSetPairsStruct, 5
RTG::EuclideanSqrDistDistanceFunc, 6
RTG::MultiRobotSqrDistDistanceFunc, 7
RTG::NearestNeighborsRTG, 8
RTG::PairsStruct, 9
RTG::Point_d_, 10
RTG::Random_utils, 11
RTG::RandomMatrixGen, 12
RTG::RTG_utils, 12
   calculate_radius, 13
RTG::SingleGrid, 13
RTG::SingleGrid::Cell_ind_hash, 6
RTG::VecOfIntSetsPairsStruct, 14