**Содержание:**

1. Про библиотеку
2. Использование и протокол
3. Список функций
   1. casmLineExtractor
   2. basmLineExtractor
   3. rdpMinimization
   4. lsLineApproximation
   5. lsRDPApproximation
   6. statisticalDistanceFilter
   7. statisticalFilter
   8. reduceMedianFilter
   9. naiveBreakpointDetector
   10. euclideanClusterExctraction
   11. adaptiveRDP
   12. adaptiveRDPStD
4. Что-то там еще

**1. Про библиотеку**

RDAL (Rangefinder Data Analysis Library) – библиотека, содержащая функции для обработки двумерных данных, полученных с помощью сенсора дальномера мобильным роботом.

Библиотека представлена в виде dll. Написана на языке С++.

**2. Использование и протокол**

Библиотека рассчитана для использования в проектах, написанных на языках, которые поддерживают dll. Так как библиотека написана на языке С++, то передача высокоуровневых параметров(классы и т.п.) в функции библиотеки из клиентских приложений, написанных на других языках, усложнено. Поэтому все параметры и возвращаемые значения библиотечных функций представлены простыми типами данных, которые могут передаваться между разными языками. Это так называемые [скалярные типы](https://msdn.microsoft.com/en-us/library/94z15h2c.aspx) (более подробно про использования С++ dll в других языках, в частности Delphi написано [здесь](rvelthuis.de/articles-cppobjs.html)). Так как библиотека написана на С++, то для обмена данными между клиентскими приложениями, написанными на других языках, предусмотрен протокол. Протокол определяет шаблон сигнатуры функций, а также формат входных и выходных данных.

**Шаблон сигнатуры функций.**

Первый параметр каждой функции – входные данные, указатель на начало массива чисел. Содержит координаты точек, передаваемых для обработки.

Предпоследний – выходные данные, указатель на указатель на начало массива чисел. По сути - двойной массив чисел, каждый из которых содержит координаты точек после обработки входных данных. Каждый отдельный массив содержит координаты тех точек, которые выделены в отдельный кластер, сформированный по определенному критерию, зависящий от вызываемой функции.

Последний – выходной параметр, ссылка на целое число. Содержит возвращаемое количество таких массивов в предыдущем параметре.

Общий шаблон функций выглядит следующим образом:

void someFunction(double\* input, …, double\*\* output, int& size);

Между первым и предпоследним параметрами находятся параметры конкретной функции. Все типа double или int.

**Формат входных и выходных данных**

Входные данные (input) передаются в виде указателя на массив, который организован следующим образом:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | X1 | Y1 | Z1 | R1 | D1 | X2 | Y2 | Z2 | … |

N – количество чисел, выделенных для массива

X1 – x-координата первой точки

Y1 – y-координата первой точки

Z1 – z-координата первой точки

R1 – r-координата первой точки

D1 – расстояние от дальномера до первой точки

*Пример:*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 11 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 1 |

N – 11 (первый элемент массива – число, содержащее размер массива)

1-ая точка Р1(1;1;1), расстояние до нее 1;

2-ая точка Р2(2;2;2), расстояние до нее 1;

Каждый из выходных массивов (output) организован таким же образом. Интерпретация выходных данных зависит от вызываемой функции. Выходные массивы могут интерпретироваться как множество точек, как множество линий (каждая пара точек – начало и конец линии), а так же как ломаная.

**3.Список функций**

**3.1 casmLineExtractor**

*void* casmLineExtractor(double\* input,

*double* clustering\_eps,

*int* clustering\_minPts,

*double* min\_rdp\_eps,

*double* max\_dist,

*int* min\_part\_size,

*double* merge\_dist,

*double* merge\_angle,

*int* filter\_kN,

*double* filter\_treshold,

*double\*\*&* output,

*int&* clusters\_size)

**3.2 basmLineExtractor**

*void* basmLineExtractor(double\* input,

*int* statistical\_kn,

*double* statistical\_threashold,

*int* min\_segm\_points,

*double* max\_dist\_diff,

*int* rmed\_window\_size,

*int* min\_rdp\_eps,

*int* min\_rdp\_size,

*double\*\*&* output,

*int&* clusters\_size)

**3.3 rdpMinimization**

*void* rdpMinimization(double\* input,

*double* threshold,

*double\*\*&* output,

*int&* clusters\_size)

Алгоритм Рамера-Дугласа-Пекера. Уменьшает число точек кривой, аппроксимированной большей серией точек.

threshold - порог. Если расстояние от отрезка, соединяющего крайние точки, до наиболее удаленной промежуточной точки больше порога, то происходит излом в данной точке.

Выходные данный интерпретируются как набор линий. Все данный помещаются в один кластер.

**3.4 lsLineApproximation**

void lsLineApproximation(double\* input, double\*\*& output, int& clusters\_size)

**3.5 lsRDPApproximation**

**3.6 statisticalDistanceFilter**

**3.7 statisticalFilter**

**3.8 reduceMedianFilter**

**3.9 naiveBreakpointDetector**

**3.10 rdpMinimization**

**3.11 adaptiveRDP**

**3.12 adaptiveRDPStD**