WeightedModel Class

The Weighted Model class implements an equivalent of the “Weighted point” class for models. The reason for implementing this is to implement the algorithm to select the lines. To synchronize both front and back LIDARs we chose to wait a constant amount of time and combine all the information found by the two line finding nodes. This way, we have to find a resulting model from all these messages, and the way we chose to handle it, is by using the average of all slopes and intercepts. Since we don’t know how many models will be used, just like weighted point, we have to store the model counter. This weighted model can be converted to normal model, and the points assigned to it are the first and last point (the negative-most and the positive-most, respectively).

Public Methods

◆ WeightedModel( ) [1/3]

```
WeightedModel::WeightedModel( )
```

Default constructor that assigns default value to slope, intercept and wight

◆ WeightedModel( ) [2/3]

```
WeightedModel::WeightedModel ( const Model & m )
```

Constructor to assign a model’s slope, intercept, negative and positive-most points.

Parameters

m is the model to be converted

◆ WeightedModel( ) [3/3]

```
WeightedModel::WeightedModel ( const double aa
                               const double bb
                             )
```

Constructor to assign slope and intercept to the weighted model

Parameters

aa is the slope
bb is the intercept

◆ getSlope( )

```
double WeightedModel::getSlope ( )
```

Gets the model’s slope

◆ getIntercept( )

```
double WeightedModel::getIntercept ( )
```

Gets the model’s intercept

- ## getCounter( )

  double WeightedModel::getCounter ( )

  Gets the model’s counter

- ## assignPoints( )

  void WeightedModel::assignPoints (const Model & m)

  Assign negative and positive-most points from model ‘m’ to this weighted model

  Parameters

  m is the model to be assign

- ## checkIfSameModel( )

  bool WeightedModel::checkIfSameModel (const Model & m)

  Checks if the weighted model object and ‘m’ is approximately the same

  Parameters

  m is the model to be compared

- ## fuseModels( )

  void WeightedModel::fuseModels (const Model & m)

  Fuse ‘m’ with the weighted model object.

  Parameters

  m is the model to be fused

- ## toModel( )

  Model WeightedModel::toModel ( )

  Converts this objects to a normal model

- ## friend operator << ( )

  std::ostream & operator << (std::ostream & out, const WeightedModel & wm)

  Print weighted model object

  Parameters

  out is where to print, normally terminal

  wm is the object to be printed