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 will be clearer after reading the “Robot Control” class.

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
  const double isFrontMsg = true
)

  Constructor to assign a model’s slope, intercept, and points, shifting them if the flag “isFrontMsg” is not set

  Parameters
  m is the model to be converted
  isFrontMsg is the flag to signalize if the points have to be rotated or not

- 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 ( ) const

  Gets the model’s slope

- ## getIntercept( )

  double WeightedModel::getIntercept ( ) const

  Gets the model's intercept

- ## getTotalCounter( )

  double WeightedModel::getCounter ( ) const

  Gets the model's back counter + front counter

- ## assignPoints( )

  void WeightedModel::assignPoints (const Model & m
  const double isFrontMsg = true
)

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

  Parameters

  m is the model to be assign
  isFrontMsg is the flag to signalize if the points have to be rotated or not

- ## checkIfSameModel( )

  bool WeightedModel::checkIfSameModel (const Model & m) const

  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
  const double isFrontMsg = true
)

  Fuse 'm' with the weighted model object.

  Parameters

  m is the model to be fused
  isFrontMsg is the flag to signalize if the points have to be rotated or not

- ## toModel( )

  Model WeightedModel::toModel ( ) const

  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