# Robot Control Class

The robot control class is responsible for selecting two models from all the information found by both line finding nodes, and calculating an ideal velocity based in the finite state machine and a fuzzy controller. The logic is simple: first, receives a bunch of message coming from back and front nodes counting every model, the reasoning being that the correct models will appear more frequently than the bad ones. To select the best models in the set of received models, we search for the most frequent models with positive intercept for the left model and negative intercept for the right model. Then, using the finite state machine, we are able to calculate the wheels' command.

# Public Methods

- RobotControl( )

  RobotControl::RobotControl( )

  Default constructor that declares vector and finite state machine

- clearModels( )

  void RobotControl::clearModels ( )

  Clear all models attached

- frontMessage( )

  void RobotControl::frontMessage ( const visualization_msgs::Marker & msg )

  Receives message from front line finder node

  **Parameters**

  ***msg*** is message coming from front node

- backMessage( )

  void RobotControl::backMessage ( const visualization_msgs::Marker & msg )

  Receives message from back line finder node

  **Parameters**

  ***msg*** is message coming from back node

- selectModels( )

  std::pair<Model, Model> RobotControl::selectModels ( ) const

  Select left and right models based on which models appeared most frequently

- getWheelsCommand( )

  std::pair<std_msgs::Float64, std_msgs::Float64> RobotControl::getWheelsCommand ( const std::pair<Model, Model> & selectedModels )

  Uses the selectedModels and the robot's finite state machine to calculate the output

  **Parameters**

  ***selectedModels*** is the left and right models selected

# Private Methods

- addMsgModels( )

  void RobotControl::addMsgModels ( const std::vector<Model> & modelsInMsg )

  Increments, fuse and add new models to the weighted models vector

  **Parameters**

  ***modelsInMsg*** is the models you want to add to the object

- translateAxis( )

  visualization_msgs::Marker RobotControl::translateAxis
  (
  const visualization_msgs::Marker & msg,
  const double newOX,
  const double newOY
  ) const

  Translate points in 'msg' to new X and Y origin

  **Parameters**

  ***msg*** is message containing the points
  ***newOX*** is the new X coordinate origin
  ***newOY*** is the new Y coordinate origin

- rotateAxis( )

  visualization_msgs::Marker RobotControl::rotateAxis
  (
  const visualization_msgs::Marker & msg,
  const double angleRot
  ) const

  Rotate points in 'msg' in the Z-axis by the angle 'angleRot'

  **Parameters**

  ***msg*** is message containing the points
  ***angleRot*** is the angle amount to rotate

- getFoundLines( )

  std::vector<Model> RobotControl::getFoundLines
  ( const visualization_msgs::Marker & msg ) const

  Extract models from message points

  **Parameters**

  ***msg*** is message containing the points

- friend operator << ( )

  std::ostream & operator << (std::ostream & out, const RobotControl & rc )

  Print robot control object

  **Parameters**

  ***out*** is where to print, normally terminal

  ***rc*** is the object to be printed