Robotics_ICE23_UNITN

Generated by Doxygen 1.9.7

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Module Documentation

5.1 Movement_commands

constants used to identify commands defined in movement module

Macros

- #define **no_command** 0
  no command
- #define **command_test** 1
  test
- #define **command_wait** 2
  wait
- #define **command_move** 3
  move
- #define **command_grasp** 4
  grasp
- #define **command_ungrasp** 5
  ungrasp
- #define **command_def_pos** 6
  define position
- #define **command_fast_catch** 7
  fast catch
- #define **command_catch** 8
  normal catch

5.1.1 Detailed Description

constants used to identify commands defined in movement module

5.2 Movement_security_commands

constants used to provide security in the communication

Macros

- #define **command_handshake** 9
  command to begin the handshake sequence
- #define **command_auth_key** 10
  command to request the authorization code from the movement module

5.2.1 Detailed Description

constants used to provide security in the communication

5.3 Planner_to_vision_commands

constants used to manage commands received from vision

Macros

- #define **no_command** 0
  no command
- #define **command_detect** 1
  detect command
- #define **command_quit** 2
  quit command

5.3.1 Detailed Description

constants used to manage commands received from vision

5.4 Planner_commands

Constants used to identify commands type to send to movement.

Macros

- #define **command_test** 1
  test
- #define **command_wait** 2
  wait
- #define **command_move** 3
  move
- #define **command_grasp** 4
  grasp
- #define **command_ungrasp** 5
  ungrasp
- #define **command_def_pos** 6
  define position
- #define **command_fast_catch** 7
  fast catch
- #define **command_catch** 8
  normal catch

5.4.1 Detailed Description

Constants used to identify commands type to send to movement.

5.5 Planner_security_commands

Constants used to provide security in the communication.

Macros

- #define **command_handshake** 9
  handshake command
- #define **command_auth_key** 10
  authentication key command

5.5.1 Detailed Description

Constants used to provide security in the communication.

5.6 Planner_result_type

Constants to identify the results type of an operation.

Macros

- #define **result_error** -1
  constant used to identify an execution error
- #define **result_unknown** 0
  constant used to identify an unknown result
- #define **result_completed** 1
  constant used to identify an execution success

5.6.1 Detailed Description

Constants to identify the results type of an operation.

5.7 Planner_relocation_classes

relocation classes

Macros

- #define **class_00_relocation** 0.42, -0.000, 0.82
  X1-Y1-Z2 x=0.913 y=0.288 z=0.869.
- #define **class_01_relocation** 0.42, -0.111, 0.82
  X1-Y2-Z1.
- #define **class_02_relocation** 0.42, -0.227, 0.82
  X1-Y2-Z2.
- #define **class_03_relocation** 0.42, -0.355, 0.82
  X1-Y2-Z2-CHAMFER x=0.913 y=0.693 //OK.
- #define **class_04_relocation** 0.3, -0.000, 0.82
  X1-Y2-Z2-TWINFILLET.
- #define **class_05_relocation** 0.3, -0.111, 0.82
  X1-Y3-Z2.
- #define **class_06_relocation** 0.3, -0.227, 0.82
  X1-Y3-Z2-FILLET.
- #define **class_07_relocation** 0.3, -0.355, 0.82
  X1-Y4-Z1.
- #define **class_08_relocation** 0.16, -0.000, 0.82
  X1-Y4-Z2 x=0.613 y=0.331.
- #define **class_09_relocation** 0.16, -0.227, 0.82
  X2-Y2-Z2.
- #define **class_10_relocation** 0.16, -0.355, 0.82
  X2-Y2-Z2-FILLET x=0.616 y=0.693.

5.7.1 Detailed Description

relocation classes

5.8 assignments

Functions

- void **assignment1** ()
  Execute the first assignment.
- void **assignment2** ()
  Execute the second assignment.
- void **assignment3** ()
  Execute the third assignment.
- void **assignment4** ()
  Execute the fourth assignment.

5.8.1 Detailed Description

Execute the assignment

5.9 specials

Functions

- void **special1** ()

  Execute the special 1.
- void **special2** ()

  Execute the special 2.

5.9.1 Detailed Description

Functions used to test the spawner and take photo of legos

Chapter 6

Namespace Documentation

6.1 vision Namespace Reference

Main vision script.

Functions

- subReceiveImage (data)

  Receives the images from zed camera and it will be converted to cv2 image.
- pointCloudCallBack (msg)

  Receives the msg from ZED camera and use it to find the point cloud.
- detectResulterCallback (ack_ready)

  Checks if the motion planner is ready to receive the position of the lego.
- **pubPlannerCommander** ()

  Sends the lego position to planner.
- **pubPlannerQuitter** ()

  Send quit command to planner.

Variables

- str **node_name** = ¨vision¨

  Node name
- str **sub_detect_resulter** = ¨/planner/detectResulter¨

  Subscriber to planner.
- str **pub_detect_commander** = ¨/vision/detectCommander¨

  Vision Publisher.
- str **sub_receive_image** = ¨/ur5/zed_node/left_raw/image_raw_color¨

  Subscriber to Zed Camera image.
- str **sub_receive_pointcloud** = ¨/ur5/zed_node/point_cloud/cloud_registered¨

  Subscriver to Zed Camera pointcloud
- int **is_real_robot** = 0

  flag for simulation or real world
- bool **allow_receive_image** = True

flag used to receive image

- bool **allow_receive_pointcloud** = False

  flag used to receive lego pointcloud
- bool **vision_ready** = False

  flag used to start vision
- list **lego_position_array** = [ ]

  contains all legos poses
- list **lego_list** = [ ]

  contains all legos detected
- **matrixVirtual** = numpy.matrix([[0.000, -0.499, 0.866], [-1.000, 0.000, 0.000], [0.000, -0.866, -0.499]])

  virtual matrix used to transform camera coordinates into robot coordinates
- **vectorVirtual** = numpy.array([-0.900, 0.240, -0.350])

  virtual vector used to transform camera coordinates into robot coordinates
- int **no_command** = 0

  not used
- int **command_detect** = 1

  used to define command for motion
- int **command_quit** = 2

  used to quit planner in case of exception
- int **default_queque_size** = 10

  default queue size for subscribers
- **base_offset** = numpy.array([0.500, 0.350, 1.750])

  used to transform camera coordinates into robot coordinates
- float **table_coord_z** = 0.860 + 0.100

  z table coordinate
- **file_path** = Path(__file__).resolve()

  path of file
- **root_path** = file_path.parents[0]

  directory which contains vision.py
- **pkg_vision_path** = os.path.abspath(os.path.join(root_path, ¨..¨))

  vision folder
- **zed_image_file** = os.path.join(pkg_vision_path, ¨../../src/vision/images/img_lego.png¨)

  path where img_lego is located
- **anonymous**
- **pos_pub** = rospy.Publisher(pub_detect_commander, legoFound, queue_size = default_queque_size)

  publisher for legoFound message
- **ack_sub** = rospy.Subscriber(sub_detect_resulter, eventResult, detectResulterCallback, queue_size = default_queque_size)

  subscriber used to receive ack from planner
- **image_sub** = rospy.Subscriber(sub_receive_image, Image, subReceiveImage, queue_size = default_queque_size)

  subscriber used to receive the image from camera
- **pointcloud_sub** = rospy.Subscriber(sub_receive_pointcloud, PointCloud2, pointCloudCallBack, queue_size = default_queque_size)

  subscriber used to receive the pointcloud
- **bridge** = CvBridge()

  cv2 bridge

6.1.1 Detailed Description

Main vision script.

6.1.2 Function Documentation

6.1.2.1 detectResulterCallback()

```
vision.detectResulterCallback (
              ack_ready )
```

Checks if the motion planner is ready to receive the position of the lego.

Parameters

ack_ready	msg received by Planner

6.1.2.2 pointCloudCallBack()

```
vision.pointCloudCallBack (
              msg )
```

Receives the msg from ZED camera and use it to find the point cloud.

Parameters

msg	msg taken from ZED camera

6.1.2.3 subReceiveImage()

```
vision.subReceiveImage (
              data )
```

Receives the images from zed camera and it will be converted to cv2 image.

Parameters

data	msg taken from ZED node

Chapter 7

Class Documentation

7.1 end_effector Struct Reference

Describes the end effector.

`#include <kinetics.h>`

Public Attributes

- Eigen::Vector3f **posit**
- Eigen::Matrix3f **orient**

7.1.1 Detailed Description

Describes the end effector.

Parameters

posit	position of end effector
orient	orientation of end effector

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

- kinetics.h

7.2 ExecutingTask Struct Reference

Structure used to coordinate messages between planner and movement.

Public Attributes

- int **command_id**
- ros::Time **start_moment**
- ros::Duration **interval_moment**
- bool **busy**

7.2.1 Detailed Description

Structure used to coordinate messages between planner and movement.

Parameters

command_id	used to understand which task movement must execute
start_moment	initial time
interval_moment	duration of the task
busy	shows if the movement is performing a task or not

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

- movement.cpp

7.3 recogniseLego.Lego Class Reference

class that represent legos

Public Member Functions

- __init__ (self, name, conf, x1, y1, x2, y2, img_source_path)
  Class constructor.
- **showImg** (self)
  Show lego info.

Public Attributes

- **name**
  lego name
- **class_id**
  id lego class
- **confidence**
  percentage of confidence
- **xmin**
  x min
- **ymin**
  y min
- **xmax**

x max

- **ymax**

  y max
- **img_source_path**

  image path
- **img_source**

  image
- **center_point**

  center point of lego
- **center_point_uv**

  point
- **point_cloud**

  camera point
- **point_world**

  camera point
- **img**

7.3.1 Detailed Description

class that represent legos

7.3.2 Constructor & Destructor Documentation

7.3.2.1 __init__()

```
recogniseLego.Lego.__init__ (
              self,
              name,
              conf,
              x1,
              y1,
              x2,
              y2,
              img_source_path )
```

Class constructor.

Parameters

name	lego name
conf	confidence
x1	x coordinate min
y1	y coordinate min
x2	x coordinate max
y2	y coordinate max
img_source_path	image path

Note

all lego's info detected

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

- recogniseLego.py

7.4 objectPositionOrientation Struct Reference

Describes the orientations of an object in the workspace.

`#include <kinetics.h>`

Public Attributes

- Eigen::Vector3f **position**
- Eigen::Vector3f **orientation**

7.4.1 Detailed Description

Describes the orientations of an object in the workspace.

Parameters

position	pose of object
orientation	orientation of object

Note

mainly used by movement module

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

- kinetics.h

7.5 recogniseArea.RecogniseArea Class Reference

Defines custom area.

Public Member Functions

- __init__ (self, image_path, output_path)

  Class constructor.
- execute (self)

  crops the pixels to outline the area

Public Attributes

- **img_path**
  input image path
- **output_path**
  output image path
- **img**
  loads an image from the input image path

7.5.1 Detailed Description

Defines custom area.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 __init__()

```
recogniseArea.RecogniseArea.__init__ (
              self,
              image_path,
              output_path )
```

Class constructor.

Parameters

img_path	path of input image
output_path	path of output image

7.5.3 Member Function Documentation

7.5.3.1 execute()

```
recogniseArea.RecogniseArea.execute (
              self )
```

crops the pixels to outline the area

Note

Creates a mask of the same size of the image
Checks if the image is from real or simulation camera
Defines the region without aliasing
Apply the mask to the image
Save the image on the output path

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

- recogniseArea.py

7.6 recogniseLego.RecogniseLego Class Reference

Class that uses custom trained weights and detect lego blocks with YOLOv5.

Public Member Functions

- __init__ (self, img_path)
  Class constructor.
- detectArea (self, img_path)
  uses RecogniseArea class to detect the area with lego
- detectLegos (self, img_path)
  opens the image and detect legos
- **showImg** (self)

Public Attributes

- **lego_list**
- **results**

7.6.1 Detailed Description

Class that uses custom trained weights and detect lego blocks with YOLOv5.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 __init__()

```
recogniseLego.RecogniseLego.__init__ (
              self,
              img_path )
```

Class constructor.

Parameters

img_path	path of input image

7.6.3 Member Function Documentation

7.6.3.1 detectArea()

```
recogniseLego.RecogniseLego.detectArea (
              self,
              img_path )
```

uses RecogniseArea class to detect the area with lego

Parameters

img_path	path of input image

Note

invoked in the constructor

7.6.3.2 detectLegos()

```
recogniseLego.RecogniseLego.detectLegos (
              self,
              img_path )
```

opens the image and detect legos

Parameters

img_path	path of input image

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

- recogniseLego.py

7.7 spawnedLego Struct Reference

Public Attributes

- string **name**
- int **class_id**
- double **coord_x**
- double **coord_y**
- double **coord_z**
- double **orient_x**
- double **orient_y**
- double **orient_z**
- double **orient_w**

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

- spawnLego.cpp

7.8 WaitingTask Struct Reference

used to coordinate messages between planner and movement

Public Attributes

- int **command_id**
- ros::Time **start_moment**
- ros::Duration **interval_moment**
- bool **busy**

7.8.1 Detailed Description

used to coordinate messages between planner and movement

Parameters

command_id	the id command to execute
start_moment	start time of the task
interval_moment	duration of the task

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

- planner.cpp

Chapter 8

File Documentation

8.1 kinetics.h File Reference

Provides some useful tools for planner and movement.

```
#include <iostream>
#include <cmath>
#include <complex>
#include <Eigen/Dense>
```

Classes

- struct objectPositionOrientation

  Describes the orientations of an object in the workspace.
- struct end_effector

  Describes the end effector.

Macros

- #define **null_vector** 0, 0, 0

  This is the vector having the norm = 0.

Functions

- Eigen::Matrix3f orient2matrix (Eigen::Vector3f eu_angles)

  trasforms Euler angles in a 3x3 matrix
- Eigen::Vector3f correctOrientation (Eigen::Matrix3f curr_orientation, Eigen::Matrix3f final_orientation)

  Calculates the orientation error between two given orientation matrices.
- Eigen::MatrixXf jacobMatrix (Eigen::VectorXf jo_ang)

  Calculates the jacobian Matrix related to the change of the reference system.
- Eigen::Matrix4f joint1Transf (float j1_angle)

  Calculates the transformation matrix for the joint 1 given an input angle between joint 0 and 1.
- Eigen::Matrix4f joint2Transf (float j2_angle)

  Calculates the transformation matrix for the joint 2 given an input angle between joint 1 and 2.

- Eigen::Matrix4f joint3Transf (float j3_angle)

  Calculates the transformation matrix for the joint 3 given an input angle between joint 2 and 3.
- Eigen::Matrix4f joint4Transf (float j4_angle)

  Calculates the transformation matrix for the joint 4 given an input angle between joint 3 and 4.
- Eigen::Matrix4f joint5Transf (float j5_angle)

  Calculates the transformation matrix for the joint 5 given an input angle between joint 4 and 5.
- Eigen::Matrix4f joint6Transf (float j6_angle)

  Calculates the transformation matrix for the joint 6 given an input angle between joint 5 and 6.
- end_effector directKinematic (Eigen::VectorXf j_angles)

  Calculates the position and orientation of the end effector.
- Eigen::MatrixXf inverseKinematic (end_effector &arm)

  Finds the joint motion as a function of the "desired" end-effector motion and configuration.

8.1.1 Detailed Description

Provides some useful tools for planner and movement.

Authors

Filippo Conti, Mattia Meneghin e Nicola Gianuzzi

Version

0.1

Copyright

8.1.2 Function Documentation

8.1.2.1 correctOrientation()

```
Eigen::Vector3f correctOrientation (
            Eigen::Matrix3f curr_orientation,
            Eigen::Matrix3f final_orientation )
```

Calculates the orientation error between two given orientation matrices.

Parameters

curr_orientation	current orientation
final_orientation	desired orientation

Returns

Eigen::Vector3f

8.1.2.2 directKinematic()

```
end_effector directKinematic (
            Eigen::VectorXf j_angles )
```

Calculates the position and orientation of the end effector.

Parameters

j_angles	joint angles

Returns

end_effector

8.1.2.3 inverseKinematic()

```
Eigen::MatrixXf inverseKinematic (
            end_effector & arm )
```

Finds the joint motion as a function of the "desired" end-effector motion and configuration.

Parameters

arm	is the desired end effector

Returns

Eigen::MatrixXf

8.1.2.4 jacobMatrix()

```
Eigen::MatrixXf jacobMatrix (
            Eigen::VectorXf jo_ang )
```

Calculates the jacobian Matrix related to the change of the reference system.

Parameters

jo_ang	

Returns

Eigen::MatrixXf

8.1.2.5 joint1Transf()

```
Eigen::Matrix4f joint1Transf (
```

```
            float j1_angle )
```

Calculates the transformation matrix for the joint 1 given an input angle between joint 0 and 1.

Parameters

j1_angle	joint 1 angle

Returns

Eigen::Matrix4f

8.1.2.6 joint2Transf()

```
Eigen::Matrix4f joint2Transf (
            float j2_angle )
```

Calculates the transformation matrix for the joint 2 given an input angle between joint 1 and 2.

Parameters

j2_angle	joint 2 angle

Returns

Eigen::Matrix4f

8.1.2.7 joint3Transf()

```
Eigen::Matrix4f joint3Transf (
            float j3_angle )
```

Calculates the transformation matrix for the joint 3 given an input angle between joint 2 and 3.

Parameters

j3_angle	joint 3 angle

Returns

Eigen::Matrix4f

8.1.2.8 joint4Transf()

```
Eigen::Matrix4f joint4Transf (
            float j4_angle )
```

Calculates the transformation matrix for the joint 4 given an input angle between joint 3 and 4.

Parameters

j4_angle	joint 4 angle

Returns

Eigen::Matrix4f

8.1.2.9 joint5Transf()

```
Eigen::Matrix4f joint5Transf (
            float j5_angle )
```

Calculates the transformation matrix for the joint 5 given an input angle between joint 4 and 5.

Parameters

j5_angle	joint 5 angle

Returns

Eigen::Matrix4f

8.1.2.10 joint6Transf()

```
Eigen::Matrix4f joint6Transf (
            float j6_angle )
```

Calculates the transformation matrix for the joint 6 given an input angle between joint 5 and 6.

Parameters

j6_angle	joint 6 angle

Returns

Eigen::Matrix4f

8.1.2.11 orient2matrix()

```
Eigen::Matrix3f orient2matrix (
            Eigen::Vector3f eu_angles )
```

trasforms Euler angles in a 3x3 matrix

Parameters

eu_angles	Euler angles

Returns

Eigen::Matrix3f

8.2 kinetics.h

Go to the documentation of this file.

```
00001
00009 #ifndef KINETICS_H
00010 #define KINETICS_H
00011
00012 #include <iostream>
00013 #include <cmath>
00014
00016 #include <complex>
00017
00018 // Dense module of the Eigen library
00019 // includes the necessary classes and functions for dense matrix operations
00020 // provides a wide range of functionalities, including matrix arithmetic operations
00021 #include <Eigen/Dense>
00022
00023 using namespace std;
00024
00028 #define null_vector 0, 0, 0 // this is the vector having the norm = 0
00029
00036 struct objectPositionOrientation {
00037         Eigen::Vector3f position;
00038         Eigen::Vector3f orientation;
00039 };
00040
00046 struct end_effector {
00047     Eigen::Vector3f posit;
00048     Eigen::Matrix3f orient;
00049 };
00050
00057 Eigen::Matrix3f orient2matrix(Eigen::Vector3f eu_angles) {
00058
00059     Eigen::Matrix3f resMatrix;
00060     resMatrix = Eigen::AngleAxisf(eu_angles(0), Eigen::Vector3f::UnitZ()) *
      Eigen::AngleAxisf(eu_angles(1), Eigen::Vector3f::UnitY()) * Eigen::AngleAxisf(eu_angles(2),
      Eigen::Vector3f::UnitX());
00061     return resMatrix;
00062 }
00063
00064 // It calculates the orientation error between two given orientation matrices
00072 Eigen::Vector3f correctOrientation(Eigen::Matrix3f curr_orientation, Eigen::Matrix3f
      final_orientation) {
00073
00074         Eigen::Matrix3f relative_or_mtx;
00075         relative_or_mtx = final_orientation.transpose() * curr_orientation;
00076
00077         Eigen::MatrixXf aux_mtx(3, 2);
00078         aux_mtx << relative_or_mtx(2, 1), -relative_or_mtx(1, 2), relative_or_mtx(0, 2),
      -relative_or_mtx(2, 0), relative_or_mtx(1, 0), -relative_or_mtx(0, 1);
00079
00080         float delta_angle_sin = (pow(aux_mtx(0, 0), 2) + pow(aux_mtx(0, 1), 2) + pow(aux_mtx(1, 0), 2)
      + pow(aux_mtx(1, 1), 2) + pow(aux_mtx(2, 0), 2) + pow(aux_mtx(2, 1), 2)) * 0.5;
00081         float delta_angle_cos = (relative_or_mtx(0, 0) + relative_or_mtx(1, 1) + relative_or_mtx(2, 2)
      - 1) / 2;
00082         float tan_angle = atan2(delta_angle_sin, delta_angle_cos);
00083
00084         Eigen::Vector3f axis_v;
00085         Eigen::Vector3f or_error;
00086
00087         if (tan_angle == 0) { or_error << null_vector; }
00088
00089         else {
00090                 axis_v = 1 / (2 * delta_angle_sin) * Eigen::Vector3f(relative_or_mtx(2, 1) -
      relative_or_mtx(1, 2), relative_or_mtx(0, 2) - relative_or_mtx(2, 0), relative_or_mtx(1, 0) -
      relative_or_mtx(0, 1));
00091                 or_error = final_orientation * tan_angle * axis_v;
00092         }
```

```
00093
00094         return or_error;
00095 }
00096
00103 Eigen::MatrixXf jacobMatrix(Eigen::VectorXf jo_ang) {
00104
00105     Eigen::VectorXf v_1(6);
00106     v_1 << 0, -0.425, -0.3922, 0, 0, 0;
00107
00108     Eigen::VectorXf v_2(6);
00109     v_2 << 0.1625, 0, 0, 0.1333, 0.0997, 0.0996 + 0.14;
00110
00111     Eigen::MatrixXf jac_mtx_1(6, 1);
00112     jac_mtx_1 <<    v_2(4) * (cos(jo_ang(0)) * cos(jo_ang(4)) + cos(jo_ang(1) + jo_ang(2) + jo_ang(3))
      * sin(jo_ang(0)) * sin(jo_ang(4))) + v_2(2) * cos(jo_ang(0)) + v_2(3) * cos(jo_ang(0)) - v_1(2) *
      cos(jo_ang(1) + jo_ang(2)) * sin(jo_ang(0)) - v_1(1) * cos(jo_ang(1)) * sin(jo_ang(0)) - v_2(4) *
      sin(jo_ang(1) + jo_ang(2) + jo_ang(3)) * sin(jo_ang(0)),
00113                     v_2(4) * (cos(jo_ang(4)) * sin(jo_ang(0)) - cos(jo_ang(1) + jo_ang(2) + jo_ang(3))
      * cos(jo_ang(0)) * sin(jo_ang(4))) + v_2(2) * sin(jo_ang(0)) + v_2(3) * sin(jo_ang(0)) + v_1(2) *
      cos(jo_ang(1) + jo_ang(2)) * cos(jo_ang(0)) + v_1(1) * cos(jo_ang(0)) * cos(jo_ang(1)) + v_2(4) *
      sin(jo_ang(1) + jo_ang(2) + jo_ang(3)) * cos(jo_ang(0)),
00114                     0,
00115                     0,
00116                     0,
00117                     1;
00118
00119     Eigen::MatrixXf jac_mtx_2(6, 1);
00120     jac_mtx_2 <<    -cos(jo_ang(0)) * (v_1(2) * sin(jo_ang(1) + jo_ang(2)) + v_1(1) * sin(jo_ang(1)) +
      v_2(4) * (sin(jo_ang(1) + jo_ang(2)) * sin(jo_ang(3)) - cos(jo_ang(1) + jo_ang(2)) * cos(jo_ang(3))) -
      v_2(4) * sin(jo_ang(4)) * (cos(jo_ang(1) + jo_ang(2)) * sin(jo_ang(3)) + sin(jo_ang(1) + jo_ang(2)) *
      cos(jo_ang(3)))),
00121                     -sin(jo_ang(0)) * (v_1(2) * sin(jo_ang(1) + jo_ang(2)) + v_1(1) * sin(jo_ang(1)) +
      v_2(4) * (sin(jo_ang(1) + jo_ang(2)) * sin(jo_ang(3)) - cos(jo_ang(1) + jo_ang(2)) * cos(jo_ang(3))) -
      v_2(4) * sin(jo_ang(4)) * (cos(jo_ang(1) + jo_ang(2)) * sin(jo_ang(3)) + sin(jo_ang(1) + jo_ang(2)) *
      cos(jo_ang(3)))),
00122                     v_1(2) * cos(jo_ang(1) + jo_ang(2)) - (v_2(4) * sin(jo_ang(1) + jo_ang(2) +
      jo_ang(3) + jo_ang(4))) / 2 + v_1(1) * cos(jo_ang(1)) + (v_2(4) * sin(jo_ang(1) + jo_ang(2) +
      jo_ang(3) - jo_ang(4))) / 2 + v_2(4) * sin(jo_ang(1) + jo_ang(2) + jo_ang(3)),
00123                     sin(jo_ang(0)),
00124                     -cos(jo_ang(0)),
00125                     0;
00126
00127     Eigen::MatrixXf jac_mtx_3(6, 1);
00128     jac_mtx_3 <<    cos(jo_ang(0)) * (v_2(4) * cos(jo_ang(1) + jo_ang(2) + jo_ang(3)) - v_1(2) *
      sin(jo_ang(1) + jo_ang(2)) + v_2(4) * sin(jo_ang(1) + jo_ang(2) + jo_ang(3)) * sin(jo_ang(4))),
00129                     sin(jo_ang(0)) * (v_2(4) * cos(jo_ang(1) + jo_ang(2) + jo_ang(3)) - v_1(2) *
      sin(jo_ang(1) + jo_ang(2)) + v_2(4) * sin(jo_ang(1) + jo_ang(2) + jo_ang(3)) * sin(jo_ang(4))),
00130                     v_1(2) * cos(jo_ang(1) + jo_ang(2)) - (v_2(4) * sin(jo_ang(1) + jo_ang(2) +
      jo_ang(3) + jo_ang(4))) / 2 + (v_2(4) * sin(jo_ang(1) + jo_ang(2) + jo_ang(3) - jo_ang(4))) / 2 +
      v_2(4) * sin(jo_ang(1) + jo_ang(2) + jo_ang(3)),
00131                     sin(jo_ang(0)),
00132                     -cos(jo_ang(0)),
00133                     0;
00134
00135     Eigen::MatrixXf jac_mtx_4(6, 1);
00136     jac_mtx_4 <<    v_2(4) * cos(jo_ang(0)) * (cos(jo_ang(1) + jo_ang(2) + jo_ang(3)) + sin(jo_ang(1) +
      jo_ang(2) + jo_ang(3)) * sin(jo_ang(4))),
00137                     v_2(4) * sin(jo_ang(0)) * (cos(jo_ang(1) + jo_ang(2) + jo_ang(3)) + sin(jo_ang(1)
      + jo_ang(2) + jo_ang(3)) * sin(jo_ang(4))),
00138                     v_2(4) * (sin(jo_ang(1) + jo_ang(2) + jo_ang(3) - jo_ang(4)) / 2 + sin(jo_ang(1) +
      jo_ang(2) + jo_ang(3)) - sin(jo_ang(1) + jo_ang(2) + jo_ang(3) + jo_ang(4)) / 2),
00139                     sin(jo_ang(0)),
00140                     -cos(jo_ang(0)),
00141                     0;
00142
00143     Eigen::MatrixXf jac_mtx_5(6, 1);
00144     jac_mtx_5 <<    -v_2(4) * sin(jo_ang(0)) * sin(jo_ang(4)) - v_2(4) * cos(jo_ang(1) + jo_ang(2) +
      jo_ang(3)) * cos(jo_ang(0)) * cos(jo_ang(4)),
00145                     v_2(4) * cos(jo_ang(0)) * sin(jo_ang(4)) - v_2(4) * cos(jo_ang(1) + jo_ang(2) +
      jo_ang(3)) * cos(jo_ang(4)) * sin(jo_ang(0)),
00146                     -v_2(4) * (sin(jo_ang(1) + jo_ang(2) + jo_ang(3) - jo_ang(4)) / 2 + sin(jo_ang(1)
      + jo_ang(2) + jo_ang(3) + jo_ang(4)) / 2),
00147                     sin(jo_ang(1) + jo_ang(2) + jo_ang(3)) * cos(jo_ang(0)),
00148                     sin(jo_ang(1) + jo_ang(2) + jo_ang(3)) * sin(jo_ang(0)),
00149                     -cos(jo_ang(1) + jo_ang(2) + jo_ang(3));
00150
00151     Eigen::MatrixXf jac_mtx_6(6, 1);
00152     jac_mtx_6 <<    0,
00153                     0,
00154                     0,
00155                     cos(jo_ang(4)) * sin(jo_ang(0)) - cos(jo_ang(1) + jo_ang(2) + jo_ang(3)) *
      cos(jo_ang(0)) * sin(jo_ang(4)),
00156                     -cos(jo_ang(0)) * cos(jo_ang(4)) - cos(jo_ang(1) + jo_ang(2) + jo_ang(3)) *
      sin(jo_ang(0)) * sin(jo_ang(4)),
00157                     -sin(jo_ang(1) + jo_ang(2) + jo_ang(3)) * sin(jo_ang(4));
00158
00159     Eigen::MatrixXf jacob_mtx(6, 6);
```

```
00160     jacob_mtx.setZero();
00161     jacob_mtx << jac_mtx_1, jac_mtx_2, jac_mtx_3, jac_mtx_4, jac_mtx_5, jac_mtx_6;
00162     return jacob_mtx;
00163 }
00164
00171 Eigen::Matrix4f joint1Transf(float j1_angle) {
00172
00173     Eigen::VectorXf j1_1(6);
00174     j1_1 << 0, -0.425, -0.3922, 0, 0, 0;
00175
00176     Eigen::VectorXf j1_2(6);
00177     j1_2 << 0.1625, 0, 0, 0.1333, 0.0997, 0.0996 + 0.14;
00178
00179     Eigen::Matrix4f j1_matrix;
00180     j1_matrix << cos(j1_angle), -sin(j1_angle), 0, 0, sin(j1_angle), cos(j1_angle), 0, 0, 0, 0, 1,
      j1_2(0), 0, 0, 0, 1;
00181     return j1_matrix;
00182 }
00183
00190 Eigen::Matrix4f joint2Transf(float j2_angle) {
00191
00192     Eigen::VectorXf j2_1(6);
00193     j2_1 << 0, -0.425, -0.3922, 0, 0, 0;
00194
00195     Eigen::VectorXf j2_2(6);
00196     j2_2 << 0.1625, 0, 0, 0.1333, 0.0997, 0.0996 + 0.14;
00197
00198     Eigen::Matrix4f j2_matrix;
00199     j2_matrix << cos(j2_angle), -sin(j2_angle), 0, 0, 0, 0, -1, 0, sin(j2_angle), cos(j2_angle), 0, 0,
      0, 0, 0, 1;
00200     return j2_matrix;
00201 }
00202
00209 Eigen::Matrix4f joint3Transf(float j3_angle) {
00210
00211     Eigen::VectorXf j3_1(6);
00212     j3_1 << 0, -0.425, -0.3922, 0, 0, 0;
00213
00214     Eigen::VectorXf j3_2(6);
00215     j3_2 << 0.1625, 0, 0, 0.1333, 0.0997, 0.0996 + 0.14;
00216
00217     Eigen::Matrix4f j3_matrix;
00218     j3_matrix << cos(j3_angle), -sin(j3_angle), 0, j3_1(1), sin(j3_angle), cos(j3_angle), 0, 0, 0, 0,
      1, j3_2(2), 0, 0, 0, 1;
00219     return j3_matrix;
00220 }
00221
00228 Eigen::Matrix4f joint4Transf(float j4_angle) {
00229
00230     Eigen::VectorXf j4_1(6);
00231     j4_1 << 0, -0.425, -0.3922, 0, 0, 0;
00232
00233     Eigen::VectorXf j4_2(6);
00234     j4_2 << 0.1625, 0, 0, 0.1333, 0.0997, 0.0996 + 0.14;
00235
00236     Eigen::Matrix4f j4_matrix;
00237     j4_matrix << cos(j4_angle), -sin(j4_angle), 0, j4_1(2), sin(j4_angle), cos(j4_angle), 0, 0, 0, 0,
      1, j4_2(3), 0, 0, 0, 1;
00238     return j4_matrix;
00239 }
00240
00247 Eigen::Matrix4f joint5Transf(float j5_angle) {
00248
00249     Eigen::VectorXf j5_1(6);
00250     j5_1 << 0, -0.425, -0.3922, 0, 0, 0;
00251
00252     Eigen::VectorXf j5_2(6);
00253     j5_2 << 0.1625, 0, 0, 0.1333, 0.0997, 0.0996 + 0.14;
00254
00255     Eigen::Matrix4f j5_matrix;
00256     j5_matrix << cos(j5_angle), -sin(j5_angle), 0, 0, 0, 0, -1, -j5_2(4), sin(j5_angle), cos(j5_angle),
      0, 0, 0, 0, 0, 1;
00257     return j5_matrix;
00258 }
00259
00266 Eigen::Matrix4f joint6Transf(float j6_angle) {
00267
00268     Eigen::VectorXf j6_1(6);
00269     j6_1 << 0, -0.425, -0.3922, 0, 0, 0;
00270
00271     Eigen::VectorXf j6_2(6);
00272     j6_2 << 0.1625, 0, 0, 0.1333, 0.0997, 0.0996 + 0.14;
00273
00274     Eigen::Matrix4f j6_matrix;
00275     j6_matrix << cos(j6_angle), -sin(j6_angle), 0, 0, 0, 0, 1, j6_2(5), -sin(j6_angle), -cos(j6_angle),
      0, 0, 0, 0, 0, 1;
00276     return j6_matrix;
```

```
00277 }
00278
00285 end_effector directKinematic(Eigen::VectorXf j_angles) {
00286
00287     Eigen::Matrix4f temp_mtx;
00288     temp_mtx = joint1Transf(j_angles(0)) * joint2Transf(j_angles(1)) * joint3Transf(j_angles(2)) *
      joint4Transf(j_angles(3)) * joint5Transf(j_angles(4)) * joint6Transf(j_angles(5));
00289
00290     end_effector retMatrix;
00291     retMatrix.orient = temp_mtx.block(0, 0, 3, 3);
00292     retMatrix.posit = temp_mtx.block(0, 3, 3, 1);
00293     return retMatrix;
00294 }
00295
00296 // Find the joint motion as a function of the "desired" end-effector motion and configuration
00303 Eigen::MatrixXf inverseKinematic(end_effector &arm) {
00304
00305     Eigen::VectorXf v_1d(6);
00306     v_1d << 0, -0.425, -0.3922, 0, 0, 0;
00307
00308     Eigen::VectorXf v_2d(6);
00309     v_2d << 0.1625, 0, 0, 0.1333, 0.0997, 0.0996 + 0.14;
00310
00311     Eigen::Matrix4f temp_mtx;
00312     temp_mtx.setZero();
00313     temp_mtx.block(0, 3, 3, 1) = arm.posit;
00314     temp_mtx.block(0, 0, 3, 3) = arm.orient;
00315     temp_mtx(3, 3) = 1;
00316
00317     // finding th1
00318
00319     Eigen::Vector4f j1_v;
00320     j1_v = temp_mtx * Eigen::Vector4f(0, 0, -v_2d(5), 1);
00321     float jo1_1 = real(atan2(j1_v(1), j1_v(0)) + acos(v_2d(3) / hypot(j1_v(1), j1_v(0)))) + M_PI / 2;
00322     float jo1_2 = real(atan2(j1_v(1), j1_v(0)) - acos(v_2d(3) / hypot(j1_v(1), j1_v(0)))) + M_PI / 2;
00323
00324     // finding th5
00325
00326     float jo5_1 = +real(acos((arm.posit(0) * sin(jo1_1) - arm.posit(1) * cos(jo1_1) - v_2d(3)) /
      v_2d(5)));
00327     float jo5_2 = -real(acos((arm.posit(0) * sin(jo1_1) - arm.posit(1) * cos(jo1_1) - v_2d(3)) /
      v_2d(5)));
00328     float jo5_3 = +real(acos((arm.posit(0) * sin(jo1_2) - arm.posit(1) * cos(jo1_2) - v_2d(3)) /
      v_2d(5)));
00329     float jo5_4 = -real(acos((arm.posit(0) * sin(jo1_2) - arm.posit(1) * cos(jo1_2) - v_2d(3)) /
      v_2d(5)));
00330
00331     // finding th6
00332
00333     Eigen::Matrix4f inv_mtx;
00334     inv_mtx = temp_mtx.inverse();
00335
00336     Eigen::Vector3f x_vect;
00337     x_vect = inv_mtx.block(0, 0, 3, 1);
00338
00339     Eigen::Vector3f y_vect;
00340     y_vect = inv_mtx.block(0, 1, 3, 1);
00341
00342     float jo6_1 = real(atan2(((-x_vect(1) * sin(jo1_1) + y_vect(1) * cos(jo1_1))) / sin(jo5_1),
      ((x_vect(0) * sin(jo1_1) - y_vect(0) * cos(jo1_1))) / sin(jo5_1)));
00343     float jo6_2 = real(atan2(((-x_vect(1) * sin(jo1_1) + y_vect(1) * cos(jo1_1))) / sin(jo5_2),
      ((x_vect(0) * sin(jo1_1) - y_vect(0) * cos(jo1_1))) / sin(jo5_2)));
00344     float jo6_3 = real(atan2(((-x_vect(1) * sin(jo1_2) + y_vect(1) * cos(jo1_2))) / sin(jo5_3),
      ((x_vect(0) * sin(jo1_2) - y_vect(0) * cos(jo1_2))) / sin(jo5_3)));
00345     float jo6_4 = real(atan2(((-x_vect(1) * sin(jo1_2) + y_vect(1) * cos(jo1_2))) / sin(jo5_4),
      ((x_vect(0) * sin(jo1_2) - y_vect(0) * cos(jo1_2))) / sin(jo5_4)));
00346
00347     Eigen::Matrix4f transf_mtx;
00348     transf_mtx = joint1Transf(jo1_1).inverse() * temp_mtx * joint6Transf(jo6_1).inverse() *
      joint5Transf(jo5_1).inverse();
00349
00350     Eigen::Vector3f v4_1;
00351     v4_1 = transf_mtx.block(0, 3, 3, 1);
00352
00353     float f4_1;
00354     f4_1 = hypot(v4_1(0), v4_1(2));
00355
00356     transf_mtx = joint1Transf(jo1_1).inverse() * temp_mtx * joint6Transf(jo6_2).inverse() *
      joint5Transf(jo5_2).inverse();
00357
00358     Eigen::Vector3f v4_2;
00359     v4_2 = transf_mtx.block(0, 3, 3, 1);
00360
00361     float f4_2;
00362     f4_2 = hypot(v4_2(0), v4_2(2));
00363
00364     transf_mtx = joint1Transf(jo1_2).inverse() * temp_mtx * joint6Transf(jo6_3).inverse() *
```

```
      joint5Transf(jo5_3).inverse();
00365
00366     Eigen::Vector3f v4_3;
00367     v4_3 = transf_mtx.block(0, 3, 3, 1);
00368
00369     float f4_3;
00370     f4_3 = hypot(v4_3(0), v4_3(2));
00371
00372     transf_mtx = joint1Transf(jo1_2).inverse() * temp_mtx * joint6Transf(jo6_4).inverse() *
      joint5Transf(jo5_4).inverse();
00373
00374     Eigen::Vector3f v4_4;
00375     v4_4 = transf_mtx.block(0, 3, 3, 1);
00376
00377     float f4_4;
00378     f4_4 = hypot(v4_4(0), v4_4(2));
00379
00380     // find th3
00381
00382     float jo3_1;
00383     if ((pow(f4_1, 2) - pow(v_1d(1), 2) - pow(v_1d(2), 2)) / (2 * v_1d(1) * v_1d(2)) > 1) { jo3_1 = 0;
      }
00384     else if ((pow(f4_1, 2) - pow(v_1d(1), 2) - pow(v_1d(2), 2)) / (2 * v_1d(1) * v_1d(2)) < -1) {
      jo3_1 = M_PI; }
00385     else { jo3_1 = acos((pow(f4_1, 2) - pow(v_1d(1), 2) - pow(v_1d(2), 2)) / (2 * v_1d(1) * v_1d(2)));
      }
00386     float jo3_5 = -jo3_1;
00387
00388     float jo3_2;
00389     if ((pow(f4_2, 2) - pow(v_1d(1), 2) - pow(v_1d(2), 2)) / (2 * v_1d(1) * v_1d(2)) > 1) { jo3_2 = 0;
      }
00390     else if ((pow(f4_2, 2) - pow(v_1d(1), 2) - pow(v_1d(2), 2)) / (2 * v_1d(1) * v_1d(2)) < -1) {
      jo3_2 = M_PI; }
00391     else { jo3_2 = acos((pow(f4_2, 2) - pow(v_1d(1), 2) - pow(v_1d(2), 2)) / (2 * v_1d(1) * v_1d(2)));
      }
00392     float jo3_6 = -jo3_2;
00393
00394     float jo3_3;
00395     if ((pow(f4_3, 2) - pow(v_1d(1), 2) - pow(v_1d(2), 2)) / (2 * v_1d(1) * v_1d(2)) > 1) { jo3_3 = 0;
      }
00396     else if ((pow(f4_3, 2) - pow(v_1d(1), 2) - pow(v_1d(2), 2)) / (2 * v_1d(1) * v_1d(2)) < -1) {
      jo3_3 = M_PI; }
00397     else { jo3_3 = acos((pow(f4_3, 2) - pow(v_1d(1), 2) - pow(v_1d(2), 2)) / (2 * v_1d(1) * v_1d(2)));
      }
00398     float jo3_7 = -jo3_3;
00399
00400     float jo3_4;
00401     if ((pow(f4_4, 2) - pow(v_1d(1), 2) - pow(v_1d(2), 2)) / (2 * v_1d(1) * v_1d(2)) > 1) { jo3_4 = 0;
      }
00402     else if ((pow(f4_4, 2) - pow(v_1d(1), 2) - pow(v_1d(2), 2)) / (2 * v_1d(1) * v_1d(2)) < -1) {
      jo3_4 = M_PI; }
00403     else { jo3_4 = acos((pow(f4_4, 2) - pow(v_1d(1), 2) - pow(v_1d(2), 2)) / (2 * v_1d(1) * v_1d(2)));
      }
00404     float jo3_8 = -jo3_4;
00405
00406     // find th2
00407
00408     float jo2_1 = atan2(-v4_1(2), -v4_1(0)) - asin((-v_1d(2) * sin(jo3_1)) / f4_1);
00409     float jo2_2 = atan2(-v4_2(2), -v4_2(0)) - asin((-v_1d(2) * sin(jo3_2)) / f4_2);
00410     float jo2_3 = atan2(-v4_3(2), -v4_3(0)) - asin((-v_1d(2) * sin(jo3_3)) / f4_3);
00411     float jo2_4 = atan2(-v4_4(2), -v4_4(0)) - asin((-v_1d(2) * sin(jo3_4)) / f4_4);
00412     float jo2_5 = atan2(-v4_1(2), -v4_1(0)) - asin((v_1d(2) * sin(jo3_1)) / f4_1);
00413     float jo2_6 = atan2(-v4_2(2), -v4_2(0)) - asin((v_1d(2) * sin(jo3_2)) / f4_2);
00414     float jo2_7 = atan2(-v4_3(2), -v4_3(0)) - asin((v_1d(2) * sin(jo3_3)) / f4_3);
00415     float jo2_8 = atan2(-v4_4(2), -v4_4(0)) - asin((v_1d(2) * sin(jo3_4)) / f4_4);
00416
00417     // find th4
00418
00419     Eigen::Matrix4f j4_mtx;
00420     j4_mtx = joint3Transf(jo3_1).inverse() * joint2Transf(jo2_1).inverse() *
      joint1Transf(jo1_1).inverse() * temp_mtx * joint6Transf(jo6_1).inverse() *
      joint5Transf(jo5_1).inverse();
00421
00422     Eigen::Vector3f x_4v;
00423     x_4v = j4_mtx.block(0, 0, 3, 1);
00424     float jo4_1 = atan2(x_4v(1), x_4v(0));
00425
00426     j4_mtx = joint3Transf(jo3_2).inverse() * joint2Transf(jo2_2).inverse() *
      joint1Transf(jo1_1).inverse() * temp_mtx * joint6Transf(jo6_2).inverse() *
      joint5Transf(jo5_2).inverse();
00427     x_4v = j4_mtx.block(0, 0, 3, 1);
00428     float jo4_2 = atan2(x_4v(1), x_4v(0));
00429
00430     j4_mtx = joint3Transf(jo3_3).inverse() * joint2Transf(jo2_3).inverse() *
      joint1Transf(jo1_2).inverse() * temp_mtx * joint6Transf(jo6_3).inverse() *
      joint5Transf(jo5_3).inverse();
00431     x_4v = j4_mtx.block(0, 0, 3, 1);
```

```
00432     float jo4_3 = atan2(x_4v(1), x_4v(0));
00433
00434     j4_mtx = joint3Transf(jo3_4).inverse() * joint2Transf(jo2_4).inverse() *
      joint1Transf(jo1_2).inverse() * temp_mtx * joint6Transf(jo6_4).inverse() *
      joint5Transf(jo5_4).inverse();
00435     x_4v = j4_mtx.block(0, 0, 3, 1);
00436     float jo4_4 = atan2(x_4v(1), x_4v(0));
00437
00438     j4_mtx = joint3Transf(jo3_5).inverse() * joint2Transf(jo2_5).inverse() *
      joint1Transf(jo1_1).inverse() * temp_mtx * joint6Transf(jo6_1).inverse() *
      joint5Transf(jo5_1).inverse();
00439     x_4v = j4_mtx.block(0, 0, 3, 1);
00440     float jo4_5 = atan2(x_4v(1), x_4v(0));
00441
00442     j4_mtx = joint3Transf(jo3_6).inverse() * joint2Transf(jo2_6).inverse() *
      joint1Transf(jo1_1).inverse() * temp_mtx * joint6Transf(jo6_2).inverse() *
      joint5Transf(jo5_2).inverse();
00443     x_4v = j4_mtx.block(0, 0, 3, 1);
00444     float jo4_6 = atan2(x_4v(1), x_4v(0));
00445
00446     j4_mtx = joint3Transf(jo3_7).inverse() * joint2Transf(jo2_7).inverse() *
      joint1Transf(jo1_2).inverse() * temp_mtx * joint6Transf(jo6_3).inverse() *
      joint5Transf(jo5_3).inverse();
00447     x_4v = j4_mtx.block(0, 0, 3, 1);
00448     float jo4_7 = atan2(x_4v(1), x_4v(0));
00449
00450     j4_mtx = joint3Transf(jo3_8).inverse() * joint2Transf(jo2_8).inverse() *
      joint1Transf(jo1_2).inverse() * temp_mtx * joint6Transf(jo6_4).inverse() *
      joint5Transf(jo5_4).inverse();
00451     x_4v = j4_mtx.block(0, 0, 3, 1);
00452     float jo4_8 = atan2(x_4v(1), x_4v(0));
00453
00454     Eigen::MatrixXf result_mtx(8, 6);
00455     result_mtx <<   jo1_1, jo2_1, jo3_1, jo4_1, jo5_1, jo6_1, jo1_1, jo2_2, jo3_2, jo4_2, jo5_2, jo6_2,
      jo1_2, jo2_3, jo3_3, jo4_3, jo5_3, jo6_3, jo1_2, jo2_4, jo3_4, jo4_4, jo5_4, jo6_4,
00456                     jo1_1, jo2_5, jo3_5, jo4_5, jo5_1, jo6_1, jo1_1, jo2_6, jo3_6, jo4_6, jo5_2,
      jo6_2, jo1_2, jo2_7, jo3_7, jo4_7, jo5_3, jo6_3, jo1_2, jo2_8, jo3_8, jo4_8, jo5_4, jo6_4;
00457     return result_mtx;
00458 }
00459
00460 #endif
```

8.3 movement.cpp File Reference

script used to move the ur5

```
#include <iostream>
#include <cmath>
#include <ctime>
#include <complex>
#include <Eigen/Dense>
#include ¨ros/ros.h¨
#include ¨kinetics.h¨
#include ¨motion/legoTask.h¨
#include ¨motion/eventResult.h¨
#include <std_msgs/Int32.h>
#include <std_msgs/Float64MultiArray.h>
#include <ros_impedance_controller/generic_float.h>
```

Classes

- struct ExecutingTask

  Structure used to coordinate messages between planner and movement.

Macros

- #define node_name ¨moviment_module¨

  Ros node name.
- #define sub_task_commander ¨/planner/taskCommander¨

  topic where 'moviment_module' is subscribed
- #define pub_task_resulter ¨/motion/taskResulter¨

  topic that 'moviment_module' publishes envetResult messages
- #define pub_joint_commander ¨/ur5/joint_group_pos_controller/command¨

  publisher which talks to locosim joint
- #define client_gripper_commander ¨/move_gripper¨

  It is a topic where a client could command real robot gripper [Only in real robot, no simulation].
- #define **queque_size** 10

  buffer size
- #define **loop_wait_rate** 1000

  1 msec of waiting
- #define **joint_number** 9

  number of ur5 joints
- #define **position_gain** 5

  parameter used to scale matrices
- #define **orientation_gain** 30

  parameter used to scale matrices
- #define **no_command** 0

  no command
- #define **command_test** 1

  test
- #define **command_wait** 2

  wait
- #define **command_move** 3

  move
- #define **command_grasp** 4

  grasp
- #define **command_ungrasp** 5

  ungrasp
- #define **command_def_pos** 6

  define position
- #define **command_fast_catch** 7

  fast catch
- #define **command_catch** 8

  normal catch
- #define **command_handshake** 9

  command to begin the handshake sequence
- #define **command_auth_key** 10

  command to request the authorization code from the movement module
- #define **result_error** -1

  constant used to identify an execution error
- #define **result_unknown** 0

  constant used to identify an unknown result
- #define **result_completed** 1

  constant used to identify an execution success

- #define **z_above_object** 0.7

  z default value to place the arm above an object
- #define **default_joint_state_vector** -0.32, -0.78, -2.56, -1.63, -1.57, 3.49

  default values for the joints
- #define **default_target_position** -0.3, -0.6, 0.4

  x, y, z homing
- #define **half_point** 0.0, -0.4, 0.6

  x, y, z --> used to avoid arm unknown position
- #define **default_dt** 0.001

  default time variation
- #define **default_max_traj_time** 4

  max trajectory time: max time that robot can make a trajectory
- #define **sleep_time** 50

  default time used to wait a command
- #define **damping_exponent** -5

  exponent for the damping coefficient
- #define **max_joint_speed** 1.5

  max joint speed
- #define **min_joint_speed** -1.5

  min joint speed
- #define **max_diameter_ext** 130

  max fingers space
- #define **min_diameter_ext** 22

  min fingers space
- #define **key_max_resolution** 1000000

  max key size
- #define **movement_secret_key** 73560

  never transmit this key!
- #define **movement_preshared_key** 92341

  used to make the first transmission
- #define **planner_preshared_key** 65433

  used to read the first receive

Functions

- Eigen::VectorXf **joint_state_vector** (6)
- Eigen::VectorXf **gripper_state_vector** (3)
- void **taskCommanderCallback** (const motion::legoTask::ConstPtr &msg_taskCommand)

  reception and analysis of a received request. If it find an ack request in the end, it will publish eventResult message
- void **pubTaskResulter** (int risultato)
- Eigen::Vector3f **getTrajectory** (double time, Eigen::Vector3f begin_position, Eigen::Vector3f final_position)

  it returns the Trajectory vector using time and positions (start and end positions)
- Eigen::VectorXf **getJointSpeeds** (Eigen::VectorXf joint_st, Eigen::Vector3f curr_position, Eigen::Vector3f destin_position, Eigen::Vector3f velocity, Eigen::Matrix3f curr_orientation, Eigen::Vector3f final_orientation)

  it returns the joints speed
- void **updateJointStates** ()

  update joint state
- float **gripper2joints** (float diameter)

  returns the converted diameter of fingers
- void **nullCommandExecute** ()

used when motion has no command to execute

- void **waitCommandExecute** (int wait_time)

  used to wait a command from planner
- void **moveProcedure** (Eigen::Vector3f v_position, Eigen::Vector3f v_orientation, float dt)

  used to move the arm
- void **graspObject** (bool catchIt)

  used to grasp objects (open-close fingers)
- void **moveDefaultPosition** ()

  move the arm to a default position
- void **fastCatchProcedure** ()

  used when the command received is command_fast_catch: goes to the object in a fast way
- void **catchProcedure** ()

  used when the command received is command_catch: goes to the object using a full protocol
- ros::Time **getTimeNow** ()

  return the current time
- ros::Duration **getInterval** (ros::Time start_t)

  return the interval between the start time and the current time
- void **handShake** ()

  the procedure to pass the security keys
- int **randomNumber** (int max_n)

  to generate random numbers
- void **AuthKeySend** ()
- bool **verifyAuthKey** (int auth_key)

  routine that checks if planner have sent the correct authorization code
- int **generateNextRandom** ()

  generates the next authorization code
- void **showKeys** ()

  shows the movement and planner keys
- int **main** (int argc, char **argv)

Variables

- bool **verbose_flag** = false

  flag used to enable deep logging
- bool **verbose_security_flag** = true

  used to show security logs
- bool **security_flag** = false

  indicates the authentication system is active
- bool **bypass_main_ack_once** = false
- int **movement_comm_key**

  used to check auth keys
- int **planner_comm_key**

  used to create auth keys to send
- int **movement_auth_key**

  stores the authorization key
- int **next_random_key**

  stores the next random key
- ros::Publisher pub_joint_commander_handle

  Movement publisher.
- ros::Publisher pub_task_resulter_handle

Movement publisher.

- ros::Subscriber sub_task_commander_handle

  movement subscriber

- ros::ServiceClient client_gripper_commander_handle

  ServiceClient object.

- motion::legoTask task_command

  Communication message that planner send to movement.

- motion::eventResult evento

  Ack that movement uses to respond.

- int risultato_var

  Used to store the success of the task.

- ExecutingTask planner_eseguendo

  Structure to coordinate messages between planner and movement.

8.3.1 Detailed Description

script used to move the ur5

Authors

Filippo Conti, Mattia Meneghin e Nicola Gianuzzi

Version

0.1

Date

2023-06-12

Copyright

Copyright (c) 2023

8.3.2 Macro Definition Documentation

8.3.2.1 client_gripper_commander

```
#define client_gripper_commander ¨/move_gripper¨
```

It is a topic where a client could command real robot gripper [Only in real robot, no simulation].

8.3.2.2 node_name

```
#define node_name ¨moviment_module¨
```

Ros node name.

8.3.2.3 pub_joint_commander

```
#define pub_joint_commander ¨/ur5/joint_group_pos_controller/command¨
```

publisher which talks to locosim joint

8.3.2.4 pub_task_resulter

```
#define pub_task_resulter ¨/motion/taskResulter¨
```

topic that 'moviment_module' publishes envetResult messages

8.3.2.5 sub_task_commander

```
#define sub_task_commander ¨/planner/taskCommander¨
```

topic where 'moviment_module' is subscribed

8.3.3 Variable Documentation

8.3.3.1 client_gripper_commander_handle

```
ros::ServiceClient client_gripper_commander_handle
```

ServiceClient object.

Note

Another way to communicate instead of topics

8.3.3.2 evento

```
motion::eventResult evento
```

Ack that movement uses to respond.

8.3.3.3 planner_eseguendo

```
ExecutingTask planner_eseguendo
```

Structure to coordinate messages between planner and movement.

8.3.3.4 pub_joint_commander_handle

```
ros::Publisher pub_joint_commander_handle
```

Movement publisher.

8.3.3.5 pub_task_resulter_handle

```
ros::Publisher pub_task_resulter_handle
```

Movement publisher.

8.3.3.6 risultato_var

```
int risultato_var
```

Used to store the success of the task.

8.3.3.7 sub_task_commander_handle

```
ros::Subscriber sub_task_commander_handle
```

movement subscriber

8.3.3.8 task_command

```
motion::legoTask task_command
```

Communication message that planner send to movement.

8.4 planner.cpp File Reference

Planner module in charge of communicate with the vision package.

```
#include <iostream>
#include <cmath>
#include <ctime>
#include <complex>
#include <Eigen/Dense>
#include ¨ros/ros.h¨
#include ¨kinetics.h¨
#include ¨motion/legoFound.h¨
#include ¨motion/legoTask.h¨
#include ¨motion/eventResult.h¨
```

Classes

- struct WaitingTask

  used to coordinate messages between planner and movement

Macros

- #define **node_name** ¨planner_module¨

  Node name.
- #define **sub_detect_commander** ¨/vision/detectCommander¨

  topic where 'planner_module' is subscribed
- #define **pub_detect_resulter** ¨/planner/detectResulter¨

  topic that 'planner_module' publishes envetResult messages
- #define **pub_task_commander** ¨/planner/taskCommander¨

  topic that 'planner_module' publishes commands
- #define **sub_task_resulter** ¨/motion/taskResulter¨

  topic that 'planner_module' subscribes to take task from motion
- #define **no_command** 0

  no command
- #define **command_detect** 1

  detect command
- #define **command_quit** 2

  quit command
- #define **command_test** 1

  test
- #define **command_wait** 2

  wait
- #define **command_move** 3

  move
- #define **command_grasp** 4

  grasp
- #define **command_ungrasp** 5

  ungrasp
- #define **command_def_pos** 6

  define position
- #define **command_fast_catch** 7

  fast catch
- #define **command_catch** 8

  normal catch
- #define **command_handshake** 9

  handshake command
- #define **command_auth_key** 10

  authentication key command
- #define **result_error** -1

  constant used to identify an execution error
- #define **result_unknown** 0

  constant used to identify an unknown result
- #define **result_completed** 1

  constant used to identify an execution success
- #define **queque_size** 10

  buffer size
- #define **loop_wait_rate** 100

  10 msec of waiting
- #define **lego_coord_z** 0.835

  z lego coordinate as the table height

- #define default_sim2bas_matrix 1.000, 0.000, 0.000, 0.500, 0.000, -1.000, 0.000, 0.350, 0.000, 0.000, -1.000, 1.750, 0.000, 0.000, 0.000, 1.000

  base matrix for functions
- #define default_cam2sim_matrix 0.866, 0.000, 0.500, -0.400, 0.000, 1.000, 0.000, 0.530, -0.500, 0.000, 0.↩ 866, 1.400, 0.000, 0.000, 0.000, 1.000

  base matrix for functions
- #define **class_00_relocation** 0.42, -0.000, 0.82

  X1-Y1-Z2 x=0.913 y=0.288 z=0.869.
- #define **class_01_relocation** 0.42, -0.111, 0.82

  X1-Y2-Z1.
- #define **class_02_relocation** 0.42, -0.227, 0.82

  X1-Y2-Z2.
- #define **class_03_relocation** 0.42, -0.355, 0.82

  X1-Y2-Z2-CHAMFER x=0.913 y=0.693 //OK.
- #define **class_04_relocation** 0.3, -0.000, 0.82

  X1-Y2-Z2-TWINFILLET.
- #define **class_05_relocation** 0.3, -0.111, 0.82

  X1-Y3-Z2.
- #define **class_06_relocation** 0.3, -0.227, 0.82

  X1-Y3-Z2-FILLET.
- #define **class_07_relocation** 0.3, -0.355, 0.82

  X1-Y4-Z1.
- #define **class_08_relocation** 0.16, -0.000, 0.82

  X1-Y4-Z2 x=0.613 y=0.331.
- #define **class_09_relocation** 0.16, -0.227, 0.82

  X2-Y2-Z2.
- #define **class_10_relocation** 0.16, -0.355, 0.82

  X2-Y2-Z2-FILLET x=0.616 y=0.693.
- #define **class_00_orient** 0.0, 0.0, 0.0

  orientation of lego class 0
- #define **class_01_orient** 0.0, 0.0, 0.0

  orientation of lego class 1
- #define **class_02_orient** 0.0, 0.0, 0.0

  orientation of lego class 2
- #define **class_03_orient** 0.0, 0.0, 0.0

  orientation of lego class 3
- #define **class_04_orient** 0.0, 0.0, 0.0

  orientation of lego class 4
- #define **class_05_orient** 0.0, 0.0, 0.0

  orientation of lego class 5
- #define **class_06_orient** 0.0, 0.0, 0.0

  orientation of lego class 6
- #define **class_07_orient** 0.0, 0.0, 0.0

  orientation of lego class 7
- #define **class_08_orient** 0.0, 0.0, 0.0

  orientation of lego class 8
- #define **class_09_orient** 0.0, 0.0, 0.0

  orientation of lego class 9
- #define **class_10_orient** 0.0, 0.0, 0.0

  orientation of lego class 10
- #define **class_00_real_grasp** 60

diameter of end effector to grasp lego class 0 in real world

- #define **class_01_real_grasp** 60

  diameter of end effector to grasp lego class 1 in real world
- #define **class_02_real_grasp** 60

  diameter of end effector to grasp lego class 2 in real world
- #define **class_03_real_grasp** 60

  diameter of end effector to grasp lego class 3 in real world
- #define **class_04_real_grasp** 60

  diameter of end effector to grasp lego class 4 in real world
- #define **class_05_real_grasp** 60

  diameter of end effector to grasp lego class 5 in real world
- #define **class_06_real_grasp** 60

  diameter of end effector to grasp lego class 6 in real world
- #define **class_07_real_grasp** 60

  diameter of end effector to grasp lego class 7 in real world
- #define **class_08_real_grasp** 60

  diameter of end effector to grasp lego class 8 in real world
- #define **class_09_real_grasp** 60

  diameter of end effector to grasp lego class 9 in real world
- #define **class_10_real_grasp** 60

  diameter of end effector to grasp lego class 10 in real world
- #define **class_00_real_ungrasp** 100

  diameter of end effector to ungrasp lego class 0 in real world
- #define **class_01_real_ungrasp** 100

  diameter of end effector to ungrasp lego class 1 in real world
- #define **class_02_real_ungrasp** 100

  diameter of end effector to ungrasp lego class 2 in real world
- #define **class_03_real_ungrasp** 100

  diameter of end effector to ungrasp lego class 3 in real world
- #define **class_04_real_ungrasp** 100

  diameter of end effector to ungrasp lego class 4 in real world
- #define **class_05_real_ungrasp** 100

  diameter of end effector to ungrasp lego class 5 in real world
- #define **class_06_real_ungrasp** 100

  diameter of end effector to ungrasp lego class 6 in real world
- #define **class_07_real_ungrasp** 100

  diameter of end effector to ungrasp lego class 7 in real world
- #define **class_08_real_ungrasp** 100

  diameter of end effector to ungrasp lego class 8 in real world
- #define **class_09_real_ungrasp** 100

  diameter of end effector to ungrasp lego class 9 in real world
- #define **class_10_real_ungrasp** 100

  diameter of end effector to ungrasp lego class 10 in real world
- #define **class_00_virt_grasp** 38

  diameter of end effector to grasp lego class 0 in the simulation
- #define **class_01_virt_grasp** 40

  diameter of end effector to grasp lego class 1 in the simulation
- #define **class_02_virt_grasp** 40

  diameter of end effector to grasp lego class 2 in the simulation
- #define **class_03_virt_grasp** 40

  diameter of end effector to grasp lego class 3 in the simulation

- #define **class_04_virt_grasp** 40

  diameter of end effector to grasp lego class 4 in the simulation
- #define **class_05_virt_grasp** 40

  diameter of end effector to grasp lego class 5 in the simulation
- #define **class_06_virt_grasp** 40

  diameter of end effector to grasp lego class 6 in the simulation
- #define **class_07_virt_grasp** 40

  diameter of end effector to grasp lego class 7 in the simulation
- #define **class_08_virt_grasp** 40

  diameter of end effector to grasp lego class 8 in the simulation
- #define **class_09_virt_grasp** 43

  diameter of end effector to grasp lego class 9 in the simulation
- #define **class_10_virt_grasp** 43

  diameter of end effector to grasp lego class 10 in the simulation
- #define **class_00_virt_ungrasp** 100

  diameter of end effector to ungrasp lego class 0 in the simulation
- #define **class_01_virt_ungrasp** 100

  diameter of end effector to ungrasp lego class 1 in the simulation
- #define **class_02_virt_ungrasp** 100

  diameter of end effector to ungrasp lego class 2 in the simulation
- #define **class_03_virt_ungrasp** 100

  diameter of end effector to ungrasp lego class 3 in the simulation
- #define **class_04_virt_ungrasp** 100

  diameter of end effector to ungrasp lego class 4 in the simulation
- #define **class_05_virt_ungrasp** 100

  diameter of end effector to ungrasp lego class 5 in the simulation
- #define **class_06_virt_ungrasp** 100

  diameter of end effector to ungrasp lego class 6 in the simulation
- #define **class_07_virt_ungrasp** 100

  diameter of end effector to ungrasp lego class 7 in the simulation
- #define **class_08_virt_ungrasp** 100

  diameter of end effector to ungrasp lego class 8 in the simulation
- #define **class_09_virt_ungrasp** 100

  diameter of end effector to ungrasp lego class 9 in the simulation
- #define **class_10_virt_ungrasp** 100

  diameter of end effector to ungrasp lego class 10 in the simulation
- #define **default_ungrasp_diam** 100

  default value for ungrasp diameter
- #define **param0** ¨-s¨

  command line parameter
- #define **param1** ¨-secureOn¨

  command line parameter
- #define **key_max_resolution** 1000000

  max key size
- #define **planner_secret_key** 35680

  never transmit this key!
- #define **planner_preshared_key** 65433

  used to make the first transmission
- #define **movement_preshared_key** 92341

  used to read the first receive

Functions

- void **readParams** (int argc, char ∗∗argv)

  parameters functions
- void **defaultFunction** ()

  function executed if there are no parameters inserted
- void **secureEnable** ()

  function used to enable security
- void **unknownUsage** ()

  function executed in error case
- bool **handShake** ()

  start the handshake process
- int **randomNumber** (int max_n)

  to generate random numbers
- void **subDetectCommanderCallback** (const motion::legoFound::ConstPtr &msg_detect)

  reception and analysis of a received request.
- void **pubDetectResulter** (int risultato)

  publish the result
- void **pubTaskCommander** (bool s_ack)

  publish the task
- void **subTaskResulterCallback** (const motion::eventResult::ConstPtr &msg_event)

  read ack sent by movement
- Eigen::Vector3f **camera2SimulationR** (Eigen::Vector3f simul_lego_pos)

  adapt camera point values in virtual world for robot
- Eigen::Vector3f **camera2RealR** (Eigen::Vector3f camera_lego_pos)

  adapt camera point values in real world for robot
- void **ungraspCommand** ()

  Enlarge robot fingers.
- void **homingCommand** ()

  move the arm to default to avoid camera interferences
- void **catchCommand** (Eigen::Vector3f position)

  send the complete catch command to the moviment module
- void **selectClass** (int lego_cl)

  In relation to lego received from vision, load right lego parameters.
- ros::Time **getTimeNow** ()

  Returns the current time.
- ros::Duration **getInterval** (ros::Time start_t)

  Returns the difference between the currentTime and the start time.
- int **generateAuthKey** ()

  generates an encrypted authorization key
- int **getNextKey** (int extra_d)

  used to obtain a random key used to generate authorization key received from movement
- void **showKeys** ()

  shows the movement and planner keys
- int **main** (int argc, char ∗∗argv)

Variables

- bool **verbose_flag** = false

  to enable deep logging
- bool **verbose_security_flag** = true

  used to show security logs
- bool **security_flag** = false

  indicates the authentication system is active
- int **planner_comm_key**

  used to check auth keys
- int **movement_comm_key**

  used to create auth keys to send
- int **movement_auth_key**

  stores the authorization key
- int **next_random_key**

  stores the next random key
- ros::Publisher **pub_task_commander_handle**

  publisher
- ros::Publisher **pub_detect_resulter_handle**

  publisher
- ros::Subscriber **sub_detect_commander_handle**

  subscriber
- ros::Subscriber **sub_task_resulter_handle**

  subscriber
- bool **is_real_robot** = false

  false --> simulation; true --> real robot
- bool **keep_orientation** = false

  false --> load classes orientation from lego classes; true --> keep vision orientation
- bool **quit_planner** = false

  false --> planner active

- bool **request_motion_ack** = true

  request an ack from movement
- bool **is_vision_msg_received** = false

  if planner receives a command from vision, this var becomes true
- WaitingTask movement_task

  struct used to coordinate msgs between planner and movement
- WaitingTask **vision_task**
- motion::legoFound **msg_lego_detect**

  message used by vision to talk to planner
- motion::legoTask **msg_taskCommand**

  message used by vision to talk to planner

- motion::eventResult **msg_evento_task**
- motion::eventResult **msg_evento_detect**

  message used by vision to talk to planner
- objectPositionOrientation **lego_dest**

  lego destination

8.4.1 Detailed Description

Planner module in charge of communicate with the vision package.

Authors

Filippo Conti, Mattia Meneghin e Nicola Gianuzzi

Version

0.1

Date

2023-06-12

Copyright

Copyright (c) 2023

8.4.2 Macro Definition Documentation

8.4.2.1 default_cam2sim_matrix

```
#define default_cam2sim_matrix 0.866, 0.000, 0.500, -0.400, 0.000, 1.000, 0.000, 0.530, -0.↩
500, 0.000, 0.866, 1.400, 0.000, 0.000, 0.000, 1.000
```

base matrix for functions

8.4.2.2 default_sim2bas_matrix

```
#define default_sim2bas_matrix 1.000, 0.000, 0.000, 0.500, 0.000, -1.000, 0.000, 0.350, 0.000,
0.000, -1.000, 1.750, 0.000, 0.000, 0.000, 1.000
```

base matrix for functions

8.4.3 Variable Documentation

8.4.3.1 movement_task

```
WaitingTask movement_task
```

struct used to coordinate msgs between planner and movement

8.5 recogniseArea.py File Reference

Classes

- class recogniseArea.RecogniseArea

  Defines custom area.

Variables

- **recogniseArea.file_path** = Path(__file__).resolve()

  path of file
- **recogniseArea.root_path** = file_path.parents[0]

  directory which contains recogniseArea.py
- bool **recogniseArea.is_real_robot** = False

  flag used to indicate if execute the robot in simulation or real world
- **recogniseArea.roi** = RecogniseArea(img_path=sys.argv[1], output_path=sys.argv[2])

  roi Image

8.5.1 Detailed Description

Authors

Filippo Conti, Mattia Meneghin e Nicola Gianuzzi

8.6 recogniseLego.py File Reference

Classes

- class recogniseLego.RecogniseLego

  Class that uses custom trained weights and detect lego blocks with YOLOv5.
- class recogniseLego.Lego

  class that represent legos

Variables

- int **recogniseLego.resize_width** = 75

  new width of image
- **recogniseLego.file_path** = Path(__file__).resolve()

  path of file
- **recogniseLego.root_path** = file_path.parents[0]

  directory which contains recogniseLego.py
- **recogniseLego.pkg_vision_path** = os.path.abspath(os.path.join(root_path, ¨..¨))

  vision folder
- **recogniseLego.zed_roi_image** = os.path.join(pkg_vision_path, ¨../../src/vision/images/img_Area.png¨)

  path of image that will contain legos that should be recognise
- float **recogniseLego.min_level_confidence** = 0.75

  minimum confidence

- **recogniseLego.weights_path** = os.path.join(pkg_vision_path, ¨../../install/include/vision/include/best.pt¨)

  weights path
- **recogniseLego.pkg_yolo_path** = os.path.join(pkg_vision_path, ¨include/yolo5¨)

  yolo path
- **recogniseLego.lego_model** = torch.hub.load('ultralytics/yolov5', 'custom', weights_path)

  lego model contained in yolo folder
- list recogniseLego.lego_models_list

  list of lego classes
- **recogniseLego.legoDetect** = LegoDetect(img_origin_path=sys.argv[1])

8.6.1 Detailed Description

Authors

Filippo Conti, Mattia Meneghin e Nicola Gianuzzi

8.6.2 Variable Documentation

8.6.2.1 lego_models_list

```
list recogniseLego.lego_models_list
```

Initial value:

```
00001 =  [  'X1-Y1-Z2',
00002                'X1-Y2-Z1',
00003                'X1-Y2-Z2',
00004                'X1-Y2-Z2-CHAMFER',
00005                'X1-Y2-Z2-TWINFILLET',
00006                'X1-Y3-Z2',
00007                'X1-Y3-Z2-FILLET',
00008                'X1-Y4-Z1',
00009                'X1-Y4-Z2',
00010                'X2-Y2-Z2',
00011                'X2-Y2-Z2-FILLET']
```

list of lego classes

8.7 spawnLego.cpp File Reference

Script used to spawn lego in the gazebo world.

```
#include <iostream>
#include ¨ros/ros.h¨
#include <cmath>
#include <ctime>
#include <complex>
#include <std_msgs/Int32.h>
#include <geometry_msgs/Pose.h>
#include <gazebo_msgs/SpawnModel.h>
#include <gazebo_msgs/SpawnModelRequest.h>
#include <gazebo_msgs/SpawnModelResponse.h>
#include <gazebo_msgs//DeleteModel.h>
#include <fstream>
#include <sstream>
```

Classes

- struct spawnedLego

Macros

- #define **node_name** ¨spawn_module¨
- #define **cli_spawner_commander** ¨gazebo/spawn_sdf_model¨
- #define **cli_delete_commander** ¨gazebo/delete_model¨
- #define **sub_spawner_commander** ¨/vision/spawnerCommander¨
- #define **pub_spawner_ack** ¨/vision/spawnerAck¨
- #define **models_path** ¨/home/utente/Robotics_ICE23_UNITN/catkin_ws/src/environment/legos/¨
- #define **default_model_file** ¨/model.sdf¨
- #define **default_reference_frame** ¨world¨
- #define **type_numbers_of_legos** 11
- #define **lego_colours_number** 7
- #define **table_altitude** 0.90
- #define **max_vector** 10
- #define **min_vector** -10
- #define **max_rotation** 2 ∗ M_PI
- #define **min_rotation** 0
- #define **table_max_x** 0.35
- #define **table_min_x** 0.05
- #define **table_max_y** 0.75
- #define **table_min_y** 0.25
- #define **queque_size** 10
- #define **param0** ¨-a1¨
- #define **param1** ¨-assignment1¨
- #define **param2** ¨-a2¨
- #define **param3** ¨-assignment2¨
- #define **param4** ¨-a3¨
- #define **param5** ¨-assignment3¨
- #define **param6** ¨-a4¨
- #define **param7** ¨-assignment4¨
- #define **param8** ¨-s1¨
- #define **param9** ¨-special1¨
- #define **param10** ¨-s2¨
- #define **param11** ¨-special2¨
- #define **max_lego_spawn** 500
- #define **max_lego_in_table** 11
- #define **loop_wait_rate** 0.75
- #define **max_random_position_iterations** 100
- #define **default_min_d_between** 0.1

Functions

- void readParams (int argc, char **argv)

  Check which params are selected.
- void **defaultFunction** ()

  default function (if argc < 1)
- void **unknownUsage** ()

  Functuion executed in case parameters are not valid.
- bool spawnLego (bool random_pos, bool random_or, bool keep_base_down, int lego_class=-1, int colour_index=-1, double min_dis=0)

  Spawn a lego model.
- string getModelXML (string model_name)

  Gets the lego XML file.
- void calculateRandomOr (bool keep_base_d)

  Calculates a random orientation for legos.
- bool calculateRandomPose (double max_x, double min_x, double max_y, double min_y, double min_distance=0)

  Calculates a random lego pose.
- string setColour (string l_xml, int color_index=-1)

  Sets a color in lego model.
- bool **cliSpawnObjectRequest** ()

  Sends a spawn request to Gazebo spawn service.
- double randomInInterval (double max_d, double min_d=0)

  Calculates a random number.
- int randomNumber (int max_n)

  Calculates a random number.
- void **assignment1** ()

  Execute the first assignment.
- void **assignment2** ()

  Execute the second assignment.
- void **assignment3** ()

  Execute the third assignment.
- void **assignment4** ()

  Execute the fourth assignment.
- void **special1** ()

  Execute the special 1.
- void **special2** ()

  Execute the special 2.
- void **deleteAllLego** ()

  Delete all spawned legos.
- bool **cliDeleteObjectRequest** ()

  Sends a delete request to Gazebo spawn service.
- void subSpawnCommanderCallback (const std_msgs::Int32::ConstPtr &msg)

  testing function used to link to another module
- void pubSpawnerAck (bool result)

  testing function used to respond to another module
- void **waitCommand** ()

  sleep function used to wait a command
- double objectDistance (double obj1_x=0, double obj1_y=0, double obj1_z=0, double obj2_x=0, double obj2_y=0, double obj2_z=0)

  Find the distance between two legos.
- bool checkCollitions (double obj_x, double obj_y, double obj_z, double min_distance=0)

  Checks if there are collitions between legos.
- int **main** (int argc, char **argv)

Variables

- bool **verbose_flag** = true
- ros::ServiceClient **cli_spawner_commander_handle**
- ros::ServiceClient **cli_delete_commander_handle**
- ros::Publisher **pub_spawner_ack_handle**
- ros::Subscriber **sub_spawner_commander_handle**
- gazebo_msgs::SpawnModel **msg_spawn_object**
- geometry_msgs::Pose **pose_object**
- gazebo_msgs::DeleteModel **msg_delete**
- string **lego_names** [ ] = {¨X1-Y1-Z2¨, ¨X1-Y2-Z1¨, ¨X1-Y2-Z2¨, ¨X1-Y2-Z2-CHAMFER¨, ¨X1-Y2-Z2-TWINFILLET¨, ¨X1-Y3-Z2¨, ¨X1-Y3-Z2-FILLET¨, ¨X1-Y4-Z1¨, ¨X1-Y4-Z2¨, ¨X2-Y2-Z2¨, ¨X2-Y2-Z2-FILLET¨}
- string **lego_colours** [ ] = {¨Gazebo/Indigo¨, ¨Gazebo/Orange¨, ¨Gazebo/Red¨, ¨Gazebo/Purple¨, ¨Gazebo/SkyBlue¨, ¨Gazebo/DarkYellow¨, ¨Gazebo/Green¨ }
- spawnedLego **spawn_pool** [max_lego_spawn]
- int **spawn_counter** = 0

8.7.1 Detailed Description

Script used to spawn lego in the gazebo world.

Authors

Filippo Conti, Nicola Gianuzzi, Mattia Meneghin

Version

0.1

Date

2023-06-12

Copyright

Copyright (c) 2023

8.7.2 Function Documentation

8.7.2.1 calculateRandomOr()

```
void calculateRandomOr (
            bool keep_base_d )
```

Calculates a random orientation for legos.

Parameters

keep_base_d	if false, the function calculates a random orientation

Note

if kee_base_d=true, lego base will be down, so orientation will be (x=0, y=0, z=0, w=rand(max_rotation, min↩ _rotation))

8.7.2.2 calculateRandomPose()

```
bool calculateRandomPose (
            double max_x,
            double min_x,
            double max_y,
            double min_y,
            double min_distance = 0 )
```

Calculates a random lego pose.

Parameters

max_x	is max x pose lego could have
min_x	is min x pose lego could have
max_y	is max y pose lego could have
min_y	is min y pose lego could have
min_distance	is the min distance between legos

8.7.2.3 checkCollitions()

```
bool checkCollitions (
            double obj_x,
            double obj_y,
            double obj_z,
            double min_distance = 0 )
```

Checks if there are collitions between legos.

Parameters

obj_x	x pose coordinate
obj_y	y pose coordinate
obj_z	z pose coordinate
min_distance	

8.7.2.4 getModelXML()

```
string getModelXML (
            string model_name )
```

Gets the lego XML file.

Parameters

model_name	is the name of xml file

8.7.2.5 objectDistance()

```
double objectDistance (
            double obj1_x = 0,
            double obj1_y = 0,
            double obj1_z = 0,
            double obj2_x = 0,
            double obj2_y = 0,
            double obj2_z = 0 )
```

Find the distance between two legos.

Parameters

obj1_x	x coordinate of first object
obj1_y	y coordinate of first object
obj1_z	z coordinate of first object
obj2_x	x coordinate of second object
obj2_y	y coordinate of second object
obj2_z	z coordinate of second object

Returns

Distance between two points

8.7.2.6 pubSpawnerAck()

```
void pubSpawnerAck (
            bool result )
```

testing function used to respond to another module

Parameters

result	int used to send the result of response

8.7.2.7 randomInInterval()

```
double randomInInterval (
            double max_d,
            double min_d = 0 )
```

Calculates a random number.

Parameters

max_d	max double number
min_d	min double number

Returns

a random double value

8.7.2.8 randomNumber()

```
int randomNumber (
            int max_n )
```

Calculates a random number.

Parameters

max_n	max integer number

Returns

a random integer number

8.7.2.9 readParams()

```
void readParams (
            int argc,
            char ** argv )
```

Check which params are selected.

Parameters

argc	is the number of params inserted
argv	is a list of all parameters inserted

Note

parameters could be aX or sX

8.7.2.10 setColour()

```
string setColour (
            string l_xml,
            int color_index = -1 )
```

Sets a color in lego model.

Parameters

l_xml	is lego xml file
color_index	index for color

8.7.2.11 spawnLego()

```
bool spawnLego (
            bool random_pos,
            bool random_or,
            bool keep_base_down,
            int lego_class = -1,
            int colour_index = -1,
            double min_dis = 0 )
```

Spawn a lego model.

Parameters

random_pos	if true, the function calculates a random position
random_or	if true, the function calculates a random orientation
keep_base_down	if true, the lego will be spawned on its natural base
lego_class	integer used to select lego class [0, 11)
colour_index	integer used to select the colour of lego spawned
min_dis	double used to indicate the minimal distance between legos

8.7.2.12 subSpawnCommanderCallback()

```
void subSpawnCommanderCallback (
            const std_msgs::Int32::ConstPtr & msg )
```

testing function used to link to another module

Parameters

msg	received by an external module

8.8 vision.py File Reference

Namespaces

- namespace vision

  Main vision script.

Functions

- vision.subReceiveImage (data)

  Receives the images from zed camera and it will be converted to cv2 image.
- vision.pointCloudCallBack (msg)

  Receives the msg from ZED camera and use it to find the point cloud.
- vision.detectResulterCallback (ack_ready)

  Checks if the motion planner is ready to receive the position of the lego.
- **vision.pubPlannerCommander** ()

  Sends the lego position to planner.
- **vision.pubPlannerQuitter** ()

  Send quit command to planner.

Variables

- str **vision.node_name** = ¨vision¨

  Node name
- str **vision.sub_detect_resulter** = ¨/planner/detectResulter¨

  Subscriber to planner.
- str **vision.pub_detect_commander** = ¨/vision/detectCommander¨

  Vision Publisher.
- str **vision.sub_receive_image** = ¨/ur5/zed_node/left_raw/image_raw_color¨

  Subscriber to Zed Camera image.
- str **vision.sub_receive_pointcloud** = ¨/ur5/zed_node/point_cloud/cloud_registered¨

  Subscriver to Zed Camera pointcloud
- int **vision.is_real_robot** = 0

  flag for simulation or real world
- bool **vision.allow_receive_image** = True

  flag used to receive image
- bool **vision.allow_receive_pointcloud** = False

  flag used to receive lego pointcloud
- bool **vision.vision_ready** = False

  flag used to start vision
- list **vision.lego_position_array** = [ ]

  contains all legos poses
- list **vision.lego_list** = [ ]

  contains all legos detected
- **vision.matrixVirtual** = numpy.matrix([[0.000, -0.499, 0.866], [-1.000, 0.000, 0.000], [0.000, -0.866, -0.499]])

  virtual matrix used to transform camera coordinates into robot coordinates
- **vision.vectorVirtual** = numpy.array([-0.900, 0.240, -0.350])

  virtual vector used to transform camera coordinates into robot coordinates
- int **vision.no_command** = 0

not used

- int **vision.command_detect** = 1

  used to define command for motion

- int **vision.command_quit** = 2

  used to quit planner in case of exception

- int **vision.default_queque_size** = 10

  default queue size for subscribers

- **vision.base_offset** = numpy.array([0.500, 0.350, 1.750])

  used to transform camera coordinates into robot coordinates

- float **vision.table_coord_z** = 0.860 + 0.100

  z table coordinate

- **vision.file_path** = Path(__file__).resolve()

  path of file

- **vision.root_path** = file_path.parents[0]

  directory which contains vision.py

- **vision.pkg_vision_path** = os.path.abspath(os.path.join(root_path, ¨..¨))

  vision folder

- **vision.zed_image_file** = os.path.join(pkg_vision_path, ¨../../src/vision/images/img_lego.png¨)

  path where img_lego is located

- **vision.anonymous**
- **vision.pos_pub** = rospy.Publisher(pub_detect_commander, legoFound, queue_size = default_queque_size)

  publisher for legoFound message

- **vision.ack_sub** = rospy.Subscriber(sub_detect_resulter, eventResult, detectResulterCallback, queue_size = default_queque_size)

  subscriber used to receive ack from planner

- **vision.image_sub** = rospy.Subscriber(sub_receive_image, Image, subReceiveImage, queue_size = default_queque_size)

  subscriber used to receive the image from camera

- **vision.pointcloud_sub** = rospy.Subscriber(sub_receive_pointcloud, PointCloud2, pointCloudCallBack, queue_size = default_queque_size)

  subscriber used to receive the pointcloud

- **vision.bridge** = CvBridge()

  cv2 bridge

8.8.1 Detailed Description

Authors

Filippo Conti, Mattia Meneghin e Nicola Gianuzzi

Index