HOMEWORK 2 - GROUP 01

Generated by Doxygen 1.9.4

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

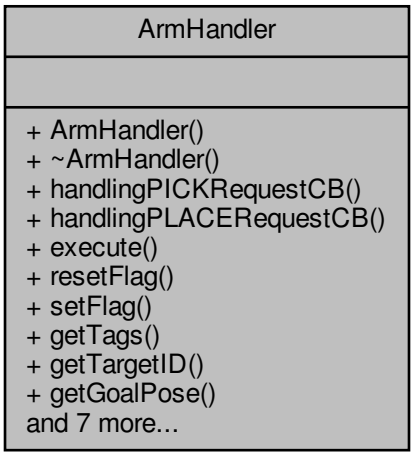
Class Documentation

3.1 ArmHandler Class Reference

Class that provides the services and functions used by arm_handler node. In particular, this class provides executing instruction for the main node, poses for the Tiago's arm, and manage the use ofcollision objects and the gripper through the following remaining classes. State (flag) defition: state 0: the service is ready, and Tiago's arm is in home or inert position state 1: the picking service has been called and Tiago is going to grasp the object state 2: Tiago has grabbed the object and has its arm in a safe position to travel state 3: the placing service has been called and Tiago is going to place the object state 4: Tiago has placed the object and has its arm in a safe position to travel state -1: Tiago encountered problems during the motion.

`#include <ArmHandler_lib.h>`

Collaboration diagram for ArmHandler:



Public Member Functions

- ArmHandler ()
- ~ArmHandler ()
- bool handlingPICKRequestCB (object_manipulation::arm_handler::Request &req, object_manipulation::arm_handler::Response &res)

  Callback for the arm hndler Pick service.
- bool handlingPLACERequestCB (object_manipulation::arm_handler::Request &req, object_manipulation::arm_handler::Response &res)

  Callback for the arm hndler Place service.
- int execute ()

  Specify to the main function which manimulation action start with Moveit!
- void resetFlag ()

  reset flag
- void setFlag (int v)

  set custom flag's value
- std::vector< myTAG > getTags ()

  Get the Tags object.
- int getTargetID ()

  Get the Target I D object.
- geometry_msgs::PoseStamped getGoalPose ()

  Get the Goal Pose object.
- geometry_msgs::PoseStamped getApproachPose ()

  Get the Approach Pose object.
- geometry_msgs::PoseStamped getGraspPose ()

  Get the Grasp Pose object.
- geometry_msgs::PoseStamped getTravelPose ()

  Get the Pose to travel.
- geometry_msgs::PoseStamped getPlacingPose ()

  Get the Pose to place the object.
- bool moveToGoal (moveit::planning_interface::MoveGroupInterface &move_group, geometry_msgs::PoseStamped goal)

  Function used to move the wanted group to the wanted pose.
- bool graspObject ()

  Funcion used to grasp the target object.
- bool releaseObject ()

  Function to relesae the object.

3.1.1 Detailed Description

Class that provides the services and functions used by arm_handler node. In particular, this class provides executing instruction for the main node, poses for the Tiago's arm, and manage the use ofcollision objects and the gripper through the following remaining classes. State (flag) defition: state 0: the service is ready, and Tiago's arm is in home or inert position state 1: the picking service has been called and Tiago is going to grasp the object state 2: Tiago has grabbed the object and has its arm in a safe position to travel state 3: the placing service has been called and Tiago is going to place the object state 4: Tiago has placed the object and has its arm in a safe position to travel state -1: Tiago encountered problems during the motion.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 ArmHandler()

```
ArmHandler::ArmHandler ( )  [inline]
```

3.1.2.2 ~ArmHandler()

```
ArmHandler::~ArmHandler ( )  [inline]
```

3.1.3 Member Function Documentation

3.1.3.1 execute()

```
int ArmHandler::execute ( )  [inline]
```

Specify to the main function which manimulation action start with Moveit!

Returns

true

false

3.1.3.2 getApproachPose()

```
geometry_msgs::PoseStamped ArmHandler::getApproachPose ( )
```

Get the Approach Pose object.

Returns

geometry_msgs::PoseStamped

3.1.3.3 getGoalPose()

```
geometry_msgs::PoseStamped ArmHandler::getGoalPose ( )  [inline]
```

Get the Goal Pose object.

Returns

geometry_msgs::PoseStamped

3.1.3.4 getGraspPose()

```
geometry_msgs::PoseStamped ArmHandler::getGraspPose ( )
```

Get the Grasp Pose object.

Returns

geometry_msgs::PoseStamped

3.1.3.5 getPlacingPose()

```
geometry_msgs::PoseStamped ArmHandler::getPlacingPose ( )
```

Get the Pose to place the object.

Returns

geometry_msgs::PoseStamped

3.1.3.6 getTags()

```
std::vector< myTAG > ArmHandler::getTags ( )  [inline]
```

Get the Tags object.

Returns

std::vector<myTAG>

3.1.3.7 getTargetID()

```
int ArmHandler::getTargetID ( )  [inline]
```

Get the Target I D object.

Returns

int

3.1.3.8 getTravelPose()

```
geometry_msgs::PoseStamped ArmHandler::getTravelPose ( )
```

Get the Pose to travel.

Returns

geometry_msgs::PoseStamped

3.1.3.9 graspObject()

```
bool ArmHandler::graspObject ( )
```

Funcion used to grasp the target object.

Returns

true

false

3.1.3.10 handlingPICKRequestCB()

```
bool ArmHandler::handlingPICKRequestCB (
            object_manipulation::arm_handler::Request & req,
            object_manipulation::arm_handler::Response & res )
```

Callback for the arm hndler Pick service.

Parameters

req	Tags (objects) identifyed by ID and pose, target object's ID
res	

Returns

true

false

3.1.3.11 handlingPLACERequestCB()

```
bool ArmHandler::handlingPLACERequestCB (
            object_manipulation::arm_handler::Request & req,
            object_manipulation::arm_handler::Response & res )
```

Callback for the arm hndler Place service.

Parameters

req	
res	

Returns

true

false

3.1.3.12 moveToGoal()

```
bool ArmHandler::moveToGoal (
            moveit::planning_interface::MoveGroupInterface & move_group,
            geometry_msgs::PoseStamped goal )
```

Function used to move the wanted group to the wanted pose.

Parameters

move_group	
goal	

Returns

true

false

3.1.3.13 releaseObject()

```
bool ArmHandler::releaseObject ( )
```

Function to relesae the object.

Returns

true

false

3.1.3.14 resetFlag()

```
void ArmHandler::resetFlag ( )  [inline]
```

reset flag

3.1.3.15 setFlag()

```
void ArmHandler::setFlag (
            int v )  [inline]
```

set custom flag's value

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

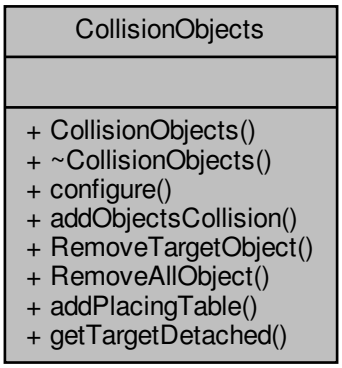
- ArmHandler_lib.h

3.2 CollisionObjects Class Reference

Class used to handle the moveit collision objects, in the specific case of the objects located in the ias_lab room simulation, where objects are identified throught an april tag. An additional objetcs representing the table has been added.

```
#include <CollisionObjects_lib.h>
```

Collaboration diagram for CollisionObjects:



Public Member Functions

- CollisionObjects ()
- ~CollisionObjects ()
- void configure (std::vector< myTAG > &tags_, moveit::planning_interface::PlanningSceneInterface &planning_scene_interface_, int target_ID)

  Set the parameters needed for the creation of the collision objects.
- bool addObjectsCollision (moveit_visual_tools::MoveItVisualTools &visual_tools)

  Method used to add collision objets to the scene, according to the tag identification.
- void RemoveTargetObject (moveit_visual_tools::MoveItVisualTools visual_tools)

  Method used to remove collision objets to the scene, according to the tag identification.
- void RemoveAllObject (moveit_visual_tools::MoveItVisualTools visual_tools)

  remove all collision objects of the scene; exception for target object
- std::vector< moveit_msgs::CollisionObject > addPlacingTable (moveit_visual_tools::MoveItVisualTools &visual_tools, double distance)

  Method used to add collision object of the coloured cylindrical table.
- void getTargetDetached (moveit_visual_tools::MoveItVisualTools visual_tools)

  get detached the target object on rviz

3.2.1 Detailed Description

Class used to handle the moveit collision objects, in the specific case of the objects located in the ias_lab room simulation, where objects are identified throught an april tag. An additional objetcs representing the table has been added.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 CollisionObjects()

```
CollisionObjects::CollisionObjects ( )  [inline]
```

3.2.2.2 ~CollisionObjects()

```
CollisionObjects::~CollisionObjects ( )  [inline]
```

3.2.3 Member Function Documentation

3.2.3.1 addObjectsCollision()

```
bool CollisionObjects::addObjectsCollision (
            moveit_visual_tools::MoveItVisualTools & visual_tools )
```

Method used to add collision objets to the scene, according to the tag identification.

Returns

true

false

3.2.3.2 addPlacingTable()

```
std::vector< moveit_msgs::CollisionObject > CollisionObjects::addPlacingTable (
            moveit_visual_tools::MoveItVisualTools & visual_tools,
            double distance )
```

Method used to add collision object of the coloured cylindrical table.

Returns

std::vector <moveit_msgs::CollisionObject>

3.2.3.3 configure()

```
void CollisionObjects::configure (
            std::vector< myTAG > & tags_,
            moveit::planning_interface::PlanningSceneInterface & planning_scene_interface_,
            int target_ID )
```

Set the parameters needed for the creation of the collision objects.

Parameters

*tags_*	
*planning_scene_↩ interface_*	
target_ID	

3.2.3.4 getTargetDetached()

```
void CollisionObjects::getTargetDetached (
            moveit_visual_tools::MoveItVisualTools visual_tools )
```

get detached the target object on rviz

Returns

void

3.2.3.5 RemoveAllObject()

```
void CollisionObjects::RemoveAllObject (
            moveit_visual_tools::MoveItVisualTools visual_tools )
```

remove all collision objects of the scene; exception for target object

Returns

void

3.2.3.6 RemoveTargetObject()

```
void CollisionObjects::RemoveTargetObject (
            moveit_visual_tools::MoveItVisualTools visual_tools )
```

Method used to remove collision objets to the scene, according to the tag identification.

Parameters

visual_tools	

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

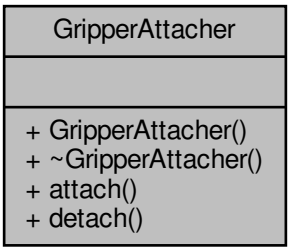
- CollisionObjects_lib.h

3.3 GripperAttacher Class Reference

Class used to manage the gazebo_ros_link_attacher package in order to attach objects to the Tiago gripper. The gripper can grasp only one object at time, so you need to detach the object before attach another one.

```
#include <GripperAttacher_lib.h>
```

Collaboration diagram for GripperAttacher:



Public Member Functions

- GripperAttacher ()
- ~GripperAttacher ()
- bool attach (std::string model_name, std::string link_name)

  Attach the object to the Tiago gripper (left finger in particular)
- bool detach ()

  Detach the current attached object.

3.3.1 Detailed Description

Class used to manage the gazebo_ros_link_attacher package in order to attach objects to the Tiago gripper. The gripper can grasp only one object at time, so you need to detach the object before attach another one.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 GripperAttacher()

```
GripperAttacher::GripperAttacher ( )  [inline]
```

3.3.2.2 ~GripperAttacher()

```
GripperAttacher::~GripperAttacher ( )  [inline]
```

3.3.3 Member Function Documentation

3.3.3.1 attach()

```
bool GripperAttacher::attach (
            std::string model_name,
            std::string link_name )
```

Attach the object to the Tiago gripper (left finger in particular)

Parameters

model_name	name of the model to be attached
link_name	name of the link to be attached

Returns

true

false

3.3.3.2 detach()

```
bool GripperAttacher::detach ( )
```

Detach the current attached object.

Returns

true

false

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

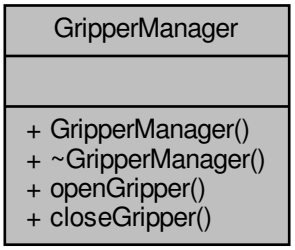
- GripperAttacher_lib.h

3.4 GripperManager Class Reference

Class used to handle the gripper. In particular, this class, provides two methods, one for open the gripper, and another to close it.

`#include <GripperManager_lib.h>`

Collaboration diagram for GripperManager:



Public Member Functions

- GripperManager ()
- ~GripperManager ()
- void openGripper ()

  Function that open the gripper.
- void closeGripper ()

  Function that close the gripper.

3.4.1 Detailed Description

Class used to handle the gripper. In particular, this class, provides two methods, one for open the gripper, and another to close it.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 GripperManager()

`GripperManager::GripperManager ( ) [inline]`

3.4.2.2 ~GripperManager()

```
GripperManager::~GripperManager ( )  [inline]
```

3.4.3 Member Function Documentation

3.4.3.1 closeGripper()

```
void GripperManager::closeGripper ( )
```

Function that close the gripper.

3.4.3.2 openGripper()

```
void GripperManager::openGripper ( )
```

Function that open the gripper.

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

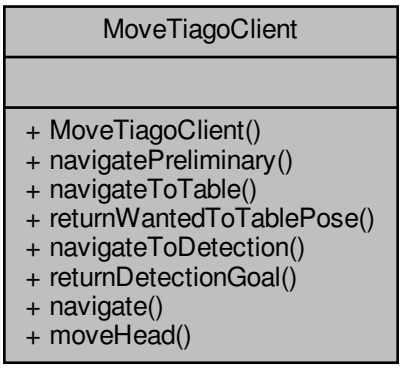
- GripperManager_lib.h

3.5 MoveTiagoClient Class Reference

Class used to implement all the motions up to the table + head motions. It's a client for obstacle_finder_server and head_controller.

```
#include <MoveTiago_lib.h>
```

Collaboration diagram for MoveTiagoClient:



Public Member Functions

- MoveTiagoClient ()
- bool navigatePreliminary ()

  Take TIAGo to preliminary position in which the table can be easily reached.
- bool navigateToTable (int target)

  Take TIAGo to the table according to picking object.
- object_manipulation::MoveTiagoGoal returnWantedToTablePose (int target)

  Return wanted pose of TIAGo to pick an object.
- std::vector< std::pair< double, double > > navigateToDetection ()

  Take TIAGo to a position where it can detect placing tables, detect the tables position.
- object_manipulation::MoveTiagoGoal returnDetectionGoal ()

  Just return detection position (useful when we go back after placing an object)
- bool navigate (double x, double y, double theta)

  Take TIAGo to a custom position.
- bool moveHead (float x_cam, float y_cam, float z_cam)

  Create a ROS action client to move TIAGo's head.

3.5.1 Detailed Description

Class used to implement all the motions up to the table + head motions. It's a client for obstacle_finder_server and head_controller.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 MoveTiagoClient()

```
MoveTiagoClient::MoveTiagoClient ( )  [inline]
```

3.5.3 Member Function Documentation

3.5.3.1 moveHead()

```
bool MoveTiagoClient::moveHead (
            float x_cam,
            float y_cam,
            float z_cam )
```

Create a ROS action client to move TIAGo's head.

Parameters

x_cam	
y_cam	
z_cam	

Returns

true

false

3.5.3.2 navigate()

```
bool MoveTiagoClient::navigate (
            double x,
            double y,
            double theta )
```

Take TIAGo to a custom position.

Parameters

x	
y	
theta	

Returns

true

false

3.5.3.3 navigatePreliminary()

```
bool MoveTiagoClient::navigatePreliminary ( )
```

Take TIAGo to preliminary position in which the table can be easily reached.

Returns

true

false

3.5.3.4 navigateToDetection()

```
std::vector< std::pair< double, double > > MoveTiagoClient::navigateToDetection ( )
```

Take TIAGo to a position where it can detect placing tables, detect the tables position.

Returns

std::vector<std::pair<double, double>> Position of tables

3.5.3.5 navigateToTable()

```
bool MoveTiagoClient::navigateToTable (
            int target )
```

Take TIAGo to the table according to picking object.

Parameters

target	

Returns

true

false

3.5.3.6 returnDetectionGoal()

```
object_manipulation::MoveTiagoGoal MoveTiagoClient::returnDetectionGoal ( )
```

Just return detection position (useful when we go back after placing an object)

Returns

object_manipulation::MoveTiagoGoal detection_goal

3.5.3.7 returnWantedToTablePose()

```
object_manipulation::MoveTiagoGoal MoveTiagoClient::returnWantedToTablePose (
            int target )
```

Return wanted pose of TIAGo to pick an object.

Parameters

target	

Returns

object_manipulation::MoveTiagoGoal Goal to reach

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

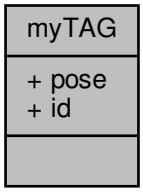
- MoveTiago_lib.h

3.6 myTAG Struct Reference

Structure of apriltag.

`#include <CollisionObjects_lib.h>`

Collaboration diagram for myTAG:



Public Attributes

- geometry_msgs::PoseWithCovarianceStamped pose
- std::vector< int > id

3.6.1 Detailed Description

Structure of apriltag.

April tag data structure.

3.6.2 Member Data Documentation

3.6.2.1 id

```
std::vector< int > myTAG::id
```

3.6.2.2 pose

```
geometry_msgs::PoseWithCovarianceStamped myTAG::pose
```

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

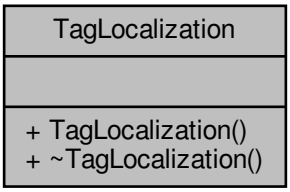
- CollisionObjects_lib.h
- TagLocalization_lib.h

3.7 TagLocalization Class Reference

Class that implements TagLocalization Service SERVER.

```
#include <TagLocalization_lib.h>
```

Collaboration diagram for TagLocalization:



Public Member Functions

- TagLocalization (ros::NodeHandle node, std::string name)
- ~TagLocalization ()

3.7.1 Detailed Description

Class that implements TagLocalization Service SERVER.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 TagLocalization()

```
TagLocalization::TagLocalization (
            ros::NodeHandle node,
            std::string name )  [inline]
```

3.7.2.2 ~TagLocalization()

```
TagLocalization::~TagLocalization ( )  [inline]
```

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

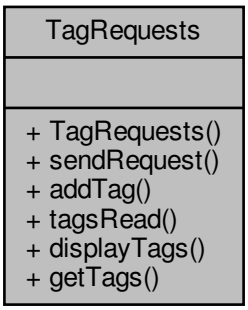
- TagLocalization_lib.h

3.8 TagRequests Class Reference

Class used to implement TagRequests (Service client, basically)

```
#include <TagLocalization_lib.h>
```

Collaboration diagram for TagRequests:



Public Member Functions

- TagRequests (ros::NodeHandle node)
- void sendRequest ()

  Send to tag localizer node a service request to read tags.
- void addTag (myTAG newTAG)

  Add tag to detected_tags vector.
- std::vector< myTAG > tagsRead ()

  Return detected tags.
- void displayTags (std::vector< myTAG > tags)

  Print out all the information about the detected tags.
- std::vector< myTAG > getTags ()

  Perform both sendRequest() and tagsRead() in one call.

3.8.1 Detailed Description

Class used to implement TagRequests (Service client, basically)

3.8.2 Constructor & Destructor Documentation

3.8.2.1 TagRequests()

```
TagRequests::TagRequests (
            ros::NodeHandle node )  [inline]
```

3.8.3 Member Function Documentation

3.8.3.1 addTag()

```
void TagRequests::addTag (
            myTAG newTAG )
```

Add tag to detected_tags vector.

Parameters

newTAG	

3.8.3.2 displayTags()

```
void TagRequests::displayTags (
            std::vector< myTAG > tags )
```

Print out all the information about the detected tags.

3.8.3.3 getTags()

```
std::vector< myTAG > TagRequests::getTags ( )
```

Perform both sendRequest() and tagsRead() in one call.

Returns

std::vector< myTAG >

3.8.3.4 sendRequest()

```
void TagRequests::sendRequest ( )
```

Send to tag localizer node a service request to read tags.

3.8.3.5 tagsRead()

```
std::vector< myTAG > TagRequests::tagsRead ( )
```

Return detected tags.

Returns

std::vector< myTAG >

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

- TagLocalization_lib.h

Chapter 4

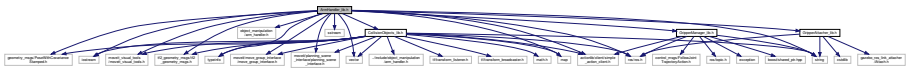
File Documentation

4.1 ArmHandler_lib.h File Reference

Class that provides the services and functions used by arm_handler node. In particular, this class provides executing instruction for the main node, poses for the Tiago's arm, and manage the use ofcollision objects and the gripper through the following remaining classes.

```
#include <ros/ros.h>
#include <geometry_msgs/PoseWithCovarianceStamped.h>
#include <iostream>
#include <actionlib/client/simple_action_client.h>
#include <object_manipulation/arm_handler.h>
#include <moveit_visual_tools/moveit_visual_tools.h>
#include <tf2_geometry_msgs/tf2_geometry_msgs.h>
#include <typeinfo>
#include <sstream>
#include <moveit/move_group_interface/move_group_interface.h>
#include <moveit/planning_scene_interface/planning_scene_interface.h>
#include <string>
#include <vector>
#include <map>
#include "GripperManager_lib.h"
#include "GripperAttacher_lib.h"
#include "CollisionObjects_lib.h"
```

Include dependency graph for ArmHandler_lib.h:



Classes

- class ArmHandler

  Class that provides the services and functions used by arm_handler node. In particular, this class provides executing instruction for the main node, poses for the Tiago's arm, and manage the use ofcollision objects and the gripper through the following remaining classes. State (flag) defition: state 0: the service is ready, and Tiago's arm is in home or inert position state 1: the picking service has been called and Tiago is going to grasp the object state 2: Tiago has grabbed the object and has its arm in a safe position to travel state 3: the placing service has been called and Tiago is going to place the object state 4: Tiago has placed the object and has its arm in a safe position to travel state -1: Tiago encountered problems during the motion.

4.1.1 Detailed Description

Class that provides the services and functions used by arm_handler node. In particular, this class provides executing instruction for the main node, poses for the Tiago's arm, and manage the use ofcollision objects and the gripper through the following remaining classes.

Author

group 01

Date

2022-01-21

4.2 ArmHandler_lib.h

Go to the documentation of this file.

```
1
13 // ROS headers
14 #include <ros/ros.h>
15 #include <geometry_msgs/PoseWithCovarianceStamped.h>
16 #include <iostream>
17 #include <actionlib/client/simple_action_client.h>
18 #include <object_manipulation/arm_handler.h>
19 #include <moveit_visual_tools/moveit_visual_tools.h>
20 #include <tf2_geometry_msgs/tf2_geometry_msgs.h>
21 #include <typeinfo>
22 #include <sstream>
23
24 // MoveIt! headers
25 #include <moveit/move_group_interface/move_group_interface.h>
26 #include <moveit/planning_scene_interface/planning_scene_interface.h>
27 #include <tf2_geometry_msgs/tf2_geometry_msgs.h>
28
29 // Std C++ headers
30 #include <string>
31 #include <vector>
32 #include <map>
33
34
35 // Our librearies implementation
36 #include "GripperManager_lib.h"
37 #include "GripperAttacher_lib.h"
38 #include "CollisionObjects_lib.h"
39
40
41
55 class ArmHandler
56 {
57
58 public:
59
60     ArmHandler() {};
61
62     ~ArmHandler() {};
63
64
73     bool handlingPICKRequestCB(object_manipulation::arm_handler::Request &req,
       object_manipulation::arm_handler::Response &res);
74
83     bool handlingPLACERequestCB(object_manipulation::arm_handler::Request &req,
       object_manipulation::arm_handler::Response &res);
84
91     inline int execute() { return flag; }
92
96     inline void resetFlag() { flag = 0; }
97
101     inline void setFlag(int v) { flag = v; }
102
108     inline std::vector<myTAG> getTags() { return detected_pose; }
109
115     inline int getTargetID() { return target_id; }
116
```

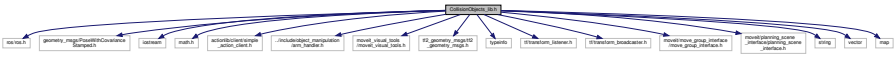
```
122     inline geometry_msgs::PoseStamped getGoalPose() { return goal_pose; }
123
129     geometry_msgs::PoseStamped getApproachPose();
130
136     geometry_msgs::PoseStamped getGraspPose();
137
143     geometry_msgs::PoseStamped getTravelPose();
144
150     geometry_msgs::PoseStamped getPlacingPose();
151
160     bool moveToGoal(moveit::planning_interface::MoveGroupInterface& move_group,
       geometry_msgs::PoseStamped goal);
161
168     bool graspObject();
169
176     bool releaseObject();
177
178
179 private:
180
187     geometry_msgs::PoseStamped ToPoseStamped(geometry_msgs::PoseWithCovarianceStamped input);
188
189
191     geometry_msgs::PoseStamped goal_pose , preliminary_goal_pose, grasp_goal_pose, travel_pose,
       placing_pose;
192
194     int target_id;
195
198     int flag = 0;
199
201     std::vector<myTAG> detected_pose;
202
204     tf2::Vector3 col1, col2, col3;
205
207     GripperManager gripper;
208     GripperAttacher attacher;
209
210 };
211
```

4.3 CollisionObjects_lib.h File Reference

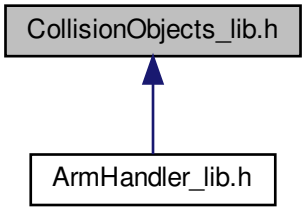
Class that manage the collision objects, such as construction, initialization in the world and the deletion.

```
#include <ros/ros.h>
#include <geometry_msgs/PoseWithCovarianceStamped.h>
#include <iostream>
#include <math.h>
#include <actionlib/client/simple_action_client.h>
#include "../include/object_manipulation/arm_handler.h"
#include <moveit_visual_tools/moveit_visual_tools.h>
#include <tf2_geometry_msgs/tf2_geometry_msgs.h>
#include <typeinfo>
#include <tf/transform_listener.h>
#include <tf/transform_broadcaster.h>
#include <moveit/move_group_interface/move_group_interface.h>
#include <moveit/planning_scene_interface/planning_scene_interface.h>
#include <string>
#include <vector>
#include <map>
```

Include dependency graph for CollisionObjects_lib.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct myTAG

  Structure of apriltag.
- class CollisionObjects

  Class used to handle the moveit collision objects, in the specific case of the objects located in the ias_lab room simulation, where objects are identified throught an april tag. An additional objetcs representing the table has been added.

4.3.1 Detailed Description

Class that manage the collision objects, such as construction, initialization in the world and the deletion.

Author

group 01

Date

2022-01-12

4.4 CollisionObjects_lib.h

Go to the documentation of this file.

```
1
11 // ROS headers
12 #include <ros/ros.h>
13 #include <geometry_msgs/PoseWithCovarianceStamped.h>
14 #include <iostream>
15 #include <math.h>
16 #include <actionlib/client/simple_action_client.h>
17 #include "../include/object_manipulation/arm_handler.h"
18 #include <moveit_visual_tools/moveit_visual_tools.h>
19 #include <tf2_geometry_msgs/tf2_geometry_msgs.h>
20 #include <typeinfo>
21 #include <tf/transform_listener.h>
22 #include <tf/transform_broadcaster.h>
23
24 // MoveIt! headers
```

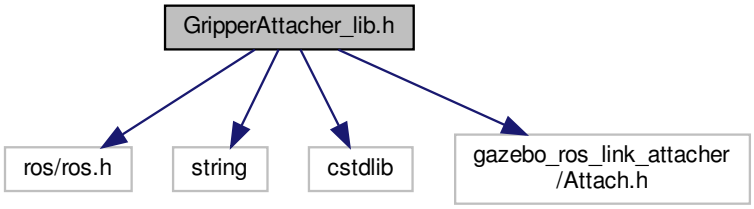
```
25 #include <moveit/move_group_interface/move_group_interface.h>
26 #include <moveit/planning_scene_interface/planning_scene_interface.h>
27 #include <tf2_geometry_msgs/tf2_geometry_msgs.h>
28
29 // Std C++ headers
30 #include <string>
31 #include <vector>
32 #include <map>
33
38 typedef struct {
39   geometry_msgs::PoseWithCovarianceStamped pose;
40   std::vector<int> id;
41 } myTAG;
42
43
50 class CollisionObjects
51 {
52
53 public:
54
55
56     CollisionObjects() {};
57
58     ~CollisionObjects() {};
59
67     void configure(std::vector<myTAG>& tags_,
68                    moveit::planning_interface::PlanningSceneInterface& planning_scene_interface_,
69                    int target_ID);
70
77     bool addObjectsCollision(moveit_visual_tools::MoveItVisualTools& visual_tools);
78
84     void RemoveTargetObject(moveit_visual_tools::MoveItVisualTools visual_tools);
85
91     void RemoveAllObject(moveit_visual_tools::MoveItVisualTools visual_tools);
92
93
99     std::vector <moveit_msgs::CollisionObject> addPlacingTable(moveit_visual_tools::MoveItVisualTools&
       visual_tools, double distance);
100
101
107     void getTargetDetached(moveit_visual_tools::MoveItVisualTools visual_tools);
108
109
110 private:
111
117     moveit_msgs::CollisionObject addBlueHexagon(myTAG BHexPose);
118
124     moveit_msgs::CollisionObject addGreenTriangle(myTAG GTrianglePose );
125
131     moveit_msgs::CollisionObject addRedCube(myTAG RCubePose);
132
138     moveit_msgs::CollisionObject addGoldObstacle(myTAG GHexPose);
139
145     moveit_msgs::CollisionObject addTable();
146
152     void getTargetAttached(moveit_visual_tools::MoveItVisualTools visual_tools);
153
154
156     std::vector<myTAG> tags;
157
159     moveit::planning_interface::PlanningSceneInterface planning_scene_interface;
160
162     int target_id;
163 };
164
165
```

4.5 GripperAttacher_lib.h File Reference

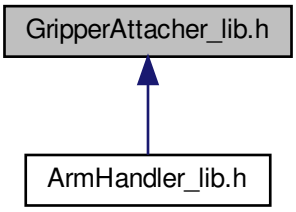
Class that uses the service provided by gazebo_ros_link_attacher to attach the object tothe gripper in Gazebo.

```
#include "ros/ros.h"
#include <string>
#include <cstdlib>
```

```
#include "gazebo_ros_link_attacher/Attach.h"
```
Include dependency graph for GripperAttacher_lib.h:



This graph shows which files directly or indirectly include this file:



Classes

- class GripperAttacher

  Class used to manage the gazebo_ros_link_attacher package in order to attach objects to the Tiago gripper. The gripper can grasp only one object at time, so you need to detach the object before attach another one.

4.5.1 Detailed Description

Class that uses the service provided by gazebo_ros_link_attacher to attach the object tothe gripper in Gazebo.

Author

group 01

Date

2022-01-15

4.6 GripperAttacher_lib.h

Go to the documentation of this file.

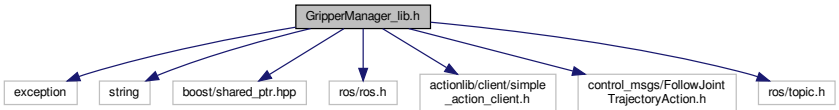
```
1
11 #include "ros/ros.h"
12 #include <string>
13 #include <cstdlib>
14 #include "gazebo_ros_link_attacher/Attach.h"
15
16
23 class GripperAttacher
24 {
25
26 public:
27     GripperAttacher()
28     {
29         gripper_in_use = false;
30     };
31
32     ~GripperAttacher() {};
33
42     bool attach(std::string model_name, std::string link_name);
43
50     bool detach();
51
52 private:
53
54     ros::NodeHandle n;
55     std::string model, link;
56     bool gripper_in_use;
57
58 };
59
60
```

4.7 GripperManager_lib.h File Reference

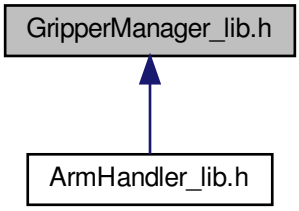
Class that performs the two fundamental gripper operation: open and close the gripper.

```
#include <exception>
#include <string>
#include <boost/shared_ptr.hpp>
#include <ros/ros.h>
#include <actionlib/client/simple_action_client.h>
#include <control_msgs/FollowJointTrajectoryAction.h>
#include <ros/topic.h>
```

Include dependency graph for GripperManager_lib.h:



This graph shows which files directly or indirectly include this file:



Classes

- class GripperManager

  Class used to handle the gripper. In particular, this class, provides two methods, one for open the gripper, and another to close it.

Typedefs

- typedef actionlib::SimpleActionClient< control_msgs::FollowJointTrajectoryAction > arm_control_client
- typedef boost::shared_ptr< arm_control_client > arm_control_client_Ptr

4.7.1 Detailed Description

Class that performs the two fundamental gripper operation: open and close the gripper.

Author

group 01

Date

2022-01-15

4.7.2 Typedef Documentation

4.7.2.1 arm_control_client

```
typedef actionlib::SimpleActionClient<control_msgs::FollowJointTrajectoryAction> arm_control_client
```

4.7.2.2 arm_control_client_Ptr

```
typedef boost::shared_ptr< arm_control_client> arm_control_client_Ptr
```

4.8 GripperManager_lib.h

Go to the documentation of this file.

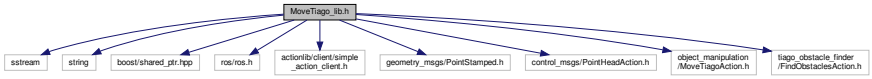
```
1
11 // C++ standard headers
12 #include <exception>
13 #include <string>
14
15 // Boost headers
16 #include <boost/shared_ptr.hpp>
17
18 // ROS headers
19 #include <ros/ros.h>
20 #include <actionlib/client/simple_action_client.h>
21 #include <control_msgs/FollowJointTrajectoryAction.h>
22 #include <ros/topic.h>
23
24 // Action interface type for moving  joints, provided as a typedef for convenience
25 typedef actionlib::SimpleActionClient<control_msgs::FollowJointTrajectoryAction> arm_control_client;
26 typedef boost::shared_ptr< arm_control_client>  arm_control_client_Ptr;
27
28
34 class GripperManager
35 {
36
37 public:
38
39     GripperManager()
40     {
41         createGripperClient(gripper_client, "gripper_controller");
42     }
43
44     ~GripperManager() {};
45
50     void openGripper();
51
52
57     void closeGripper();
58
59
60 private:
61
68     void createGripperClient(arm_control_client_Ptr& action_client, const std::string
        arm_controller_name);
69
70
78     void waypointsArmGoal(control_msgs::FollowJointTrajectoryGoal& goal, double l, double r);
79
80
86     void closeGripperGoal(control_msgs::FollowJointTrajectoryGoal& goal);
87
88
94     void openGripperGoal(control_msgs::FollowJointTrajectoryGoal& goal);
95
96     arm_control_client_Ptr gripper_client;
97
98 };
99
100
101
102
103
104
105
```

4.9 MoveTiago_lib.h File Reference

Class that performs Tiago's navigation inside the room using the HW1 action server, as well as perform the little movements of Tiago's head needed to correctly detect all the april tag in the table scene.

```
#include <sstream>
#include <string>
#include <boost/shared_ptr.hpp>
#include <ros/ros.h>
#include <actionlib/client/simple_action_client.h>
#include <geometry_msgs/PointStamped.h>
#include <control_msgs/PointHeadAction.h>
#include <object_manipulation/MoveTiagoAction.h>
#include <tiago_obstacle_finder/FindObstaclesAction.h>
```

Include dependency graph for MoveTiago_lib.h:



Classes

- class MoveTiagoClient

  Class used to implement all the motions up to the table + head motions. It's a client for obstacle_finder_server and head_controller.

Typedefs

- typedef actionlib::SimpleActionClient< control_msgs::PointHeadAction > PointHeadClient
- typedef boost::shared_ptr< PointHeadClient > PointHeadClientPtr

4.9.1 Detailed Description

Class that performs Tiago's navigation inside the room using the HW1 action server, as well as perform the little movements of Tiago's head needed to correctly detect all the april tag in the table scene.

Author

group 01

Date

2022-01-02

4.9.2 Typedef Documentation

4.9.2.1 PointHeadClient

```
typedef actionlib::SimpleActionClient<control_msgs::PointHeadAction> PointHeadClient
```

4.9.2.2 PointHeadClientPtr

```
typedef boost::shared_ptr<PointHeadClient> PointHeadClientPtr
```

4.10 MoveTiago_lib.h

Go to the documentation of this file.

```
1
12 #ifndef MOVE_TIAGO_MY_LIBRARY_H
13 #define MOVE_TIAGO_MY_LIBRARY_H
14
15 // c++ packages
16 #include <sstream>
17 #include <string>
18
19 // Boost headers
20 #include <boost/shared_ptr.hpp>
21
22 // ROS headers
23 #include <ros/ros.h>
24 #include <actionlib/client/simple_action_client.h>
25 #include <geometry_msgs/PointStamped.h>
26 #include <control_msgs/PointHeadAction.h>
27
28 // Action for moving the robot to the table
29 #include <object_manipulation/MoveTiagoAction.h>
30 #include <tiago_obstacle_finder/FindObstaclesAction.h>
31
32
33
34
35 typedef actionlib::SimpleActionClient<control_msgs::PointHeadAction> PointHeadClient;
36 typedef boost::shared_ptr<PointHeadClient> PointHeadClientPtr;
37 static const std::string cameraFrame = "/xtion_rgb_optical_frame";
38
39
45 class MoveTiagoClient
46 {
47
48 public:
49
50   MoveTiagoClient() : ac("obstacle_finder", true) {
51
52     // Position in which the table can be easily reached
53     prelim_goal.pos_x = 8.15;
54     prelim_goal.pos_y = -1.3; //-0.9;
55     prelim_goal.angle_theta = -1.57;
56
57     // Position to pick blue
58     blue_goal.pos_x = 8.0;
59     blue_goal.pos_y = -2.08;
60     blue_goal.angle_theta = -1.58;
61
62     // Position to pick green
63     green_goal.pos_x = 7.6;
64     green_goal.pos_y = -3.9;
65     green_goal.angle_theta = +1.57;
66
67     // Position to pick red
68     red_goal.pos_x = 7.5;
69     red_goal.pos_y = -1.9;
70     red_goal.angle_theta = -1.58;
71
72     // Position where detect placing tables
73     detection_goal.pos_x = 11;
74     detection_goal.pos_y = -3.0;
75     detection_goal.angle_theta =  +1.57;
76
77
78     ROS_INFO("Waiting for obstacle_finder server to start.");
79     ac.waitForServer();
80     ROS_INFO("Server started");
81
82   }
83
90   bool navigatePreliminary();
91
99   bool navigateToTable(int target);
100
```

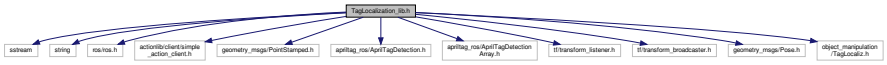
```
107   object_manipulation::MoveTiagoGoal returnWantedToTablePose(int target);
108
114   std::vector<std::pair<double, double» navigateToDetection();
115
121   object_manipulation::MoveTiagoGoal returnDetectionGoal();
122
132   bool navigate(double x, double y, double theta);
133
143   bool moveHead(float x_cam, float y_cam, float z_cam);
144
145
146 private:
147
149   PointHeadClientPtr pointHeadClient;
150   actionlib::SimpleActionClient<object_manipulation::MoveTiagoAction> ac;
151
153   object_manipulation::MoveTiagoGoal blue_goal, green_goal, red_goal, prelim_goal, detection_goal;
154
155 };
156
157
158 #endif
```

4.11 TagLocalization_lib.h File Reference

Core of the tag_localization node, performing all thetag detection functionalities.

```
#include <sstream>
#include <string>
#include <ros/ros.h>
#include <actionlib/client/simple_action_client.h>
#include <geometry_msgs/PointStamped.h>
#include <apriltag_ros/AprilTagDetection.h>
#include <apriltag_ros/AprilTagDetectionArray.h>
#include <tf/transform_listener.h>
#include <tf/transform_broadcaster.h>
#include <geometry_msgs/Pose.h>
#include <object_manipulation/TagLocaliz.h>
```

Include dependency graph for TagLocalization_lib.h:



Classes

- struct myTAG

  Structure of apriltag.
- class TagLocalization

  Class that implements TagLocalization Service SERVER.
- class TagRequests

  Class used to implement TagRequests (Service client, basically)

4.11.1 Detailed Description

Core of the tag_localization node, performing all thetag detection functionalities.

Author

group 01

Date

2021-12-28

4.12 TagLocalization_lib.h

Go to the documentation of this file.

```
1
11 #ifndef TAG_LOCALIZATION_MY_LIBRARY_H
12 #define TAG_LOCALIZATION_MY_LIBRARY_H
13
14 // c++ packages
15 #include <sstream>
16 #include <string>
17
18 // ROS headers
19 #include <ros/ros.h>
20 #include <actionlib/client/simple_action_client.h>
21 #include <geometry_msgs/PointStamped.h>
22 #include <apriltag_ros/AprilTagDetection.h>
23 #include <apriltag_ros/AprilTagDetectionArray.h>
24 #include <tf/transform_listener.h>
25 #include <tf/transform_broadcaster.h>
26 #include <geometry_msgs/Pose.h>
27
28 // Tag localization requests
29 #include <object_manipulation/TagLocaliz.h>
30
31
32
33
34 /*****************************************/
35 /***               TYPEDEF             ***/
36 /*****************************************/
37
38
43 typedef struct {
44   geometry_msgs::PoseWithCovarianceStamped pose;
45   std::vector<int> id;
46 } myTAG;
47
48
49
50
51 /*****************************************/
52 /***        TagLocalization CLASS      ***/
53 /*****************************************/
54
59 class TagLocalization{
60
61 public:
62   TagLocalization(ros::NodeHandle node, std::string name){
63
64     // the node becomes a service advertiser to return apriltag poses
65     tag_server = node.advertiseService(name, &TagLocalization::tagRequestCB, this);
66     // subscribe to tag_detections topic
67     tag_sub = node.subscribe("/tag_detections",1000, &TagLocalization::tagReceivedCB, this);
68   }
69
70   ~TagLocalization() {}
71
72 private:
73
82   bool tagRequestCB(object_manipulation::TagLocaliz::Request &req,
       object_manipulation::TagLocaliz::Response &res);
83
```

```
89   void tagReceivedCB(const apriltag_ros::AprilTagDetectionArray::ConstPtr &msg);
90
91
93   ros::ServiceServer tag_server;
94   ros::Subscriber tag_sub;
95
97   int tag_to_detect;
98
100   geometry_msgs::PoseWithCovarianceStamped tag_to_detect_pose;
101
103   bool is_tag_detected = false;
104
106   std::vector< myTAG > found_tags;
107
108 };
109
110
111
112
113 /*****************************************/
114 /***          TagRequests CLASS         ***/
115 /*****************************************/
116
121 class TagRequests {
122
123 public:
124
125   TagRequests(ros::NodeHandle node){
126     tag_client = node.serviceClient<object_manipulation::TagLocaliz>("/tag_localiz_srv");
127   }
128
133   void sendRequest();
134
140   void addTag(myTAG newTAG);
141
147   std::vector< myTAG > tagsRead();
148
153   void displayTags(std::vector< myTAG > tags);
154
160   std::vector< myTAG > getTags();
161
162
163 private:
164
166   ros::ServiceClient tag_client;
167
169   std::vector< myTAG > detected_tags;
170
171 };
172
173 #endif
```

Index