Fundametals of Robotics Project

1.0

Generated by Doxygen 1.10.0

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

Namespace Documentation

3.1 dataset2Yolo Namespace Reference

Functions

- create_annotations ()

    This function creates annotations in .txt format for each image and saves it in /labels.
- create_yaml_file ()

    This function is used to create the yaml file required from YOLO format.

Variables

- list ASSIGNS = ['assign1', 'assign2']

    Name of the folder that contains the given dataset.
- CATEGORIES = json.load(file)

    List of all categories in the dataset.
- PROJECT_DIR = os.getcwd()

    Script working directory.

3.1.1 Function Documentation

3.1.1.1 create_annotations()

```
create_annotations ( )
```

This function creates annotations in .txt format for each image and saves it in /labels.

Moreover this function copy images from the assigned dataset to /images

3.1.1.2 create_yaml_file()

```
create_yaml_file ( )
```

This function is used to create the yaml file required from YOLO format.

3.1.2 Variable Documentation

3.1.2.1 ASSIGNS

```
list ASSIGNS = ['assign1', 'assign2']
```

Name of the folder that contains the given dataset.

3.1.2.2 CATEGORIES

```
CATEGORIES = json.load(file)
```

List of all categories in the dataset.

3.1.2.3 PROJECT_DIR

```
PROJECT_DIR = os.getcwd()
```

Script working directory.

3.2 detect_area Namespace Reference

Classes

- class DetectArea
  Class that detect the area in wich detect blocks.

Variables

- detectArea = DetectArea(input_img = img, output_img_path='detected_area.png')
- img = cv2.imread('zed_image.png')

3.2.1 Variable Documentation

3.2.1.1 detectArea

```
detectArea = DetectArea(input_img = img, output_img_path='detected_area.png')
```

3.2.1.2 img

```
img = cv2.imread('zed_image.png')
```

3.3 detect_blocks Namespace Reference

Classes

- class Block

  Class that rapresent a block.
- class DetectBlocks

  Class that detect blocks.

3.4 params Namespace Reference

Variables

- BASE_LINK_POSITION = np.array([0.5,0.35,1.75])

  Base link position regarding to the origin frame.
- float BLOCK_COORD_Z = 0.875

  Height of the block regarding to the origin frame.
- int CATEGORIES = 11

  Number of categories of blocks.
- str IMAGE_SUB_TOPIC = '/ur5/zed_node/left/image_rect_color'

  ROS topic from where the script get the ZED image.
- float MIN_LEVEL_CONFIDENCE = 0.3

  Level of confidence of the Yolo model to keep the assigned labels.
- str NODE_NAME = 'vision'

  ROS nodes name.
- str POINTCLOUD_SUB_TOPIC = '/ur5/zed_node/point_cloud/cloud_registered'

  ROS topic from where the script get the pointcloud.
- str PUB_TOPIC = 'lego_position'

  ROS topic where to publish positions.
- RY

  Rotation matrix of the ZED camera.
- list TABLE = [[825,549], [1301,552], [1570,913], [658, 921]]

  Area where the vision detect blocks.
- str ZED_IMG_CROPPED_PATH = os.getcwd() + '/cropped_zed_image.png'

  Path where the cropped image is saved (a mask is applied to the photo to reduce confusion)
- str ZED_IMG_PATH = os.getcwd() + '/zed_image.png'

  Path where the original ZED image is saved.
- ZED_POSITION = np.array([-0.9 ,0.24 ,-0.35])

  Zed position regarding to the base link frame.

3.4.1 Variable Documentation

3.4.1.1 BASE_LINK_POSITION

```
BASE_LINK_POSITION = np.array([0.5,0.35,1.75])
```

Base link position regarding to the origin frame.

3.4.1.2 BLOCK_COORD_Z

```
float BLOCK_COORD_Z = 0.875
```

Height of the block regarding to the origin frame.

3.4.1.3 CATEGORIES

```
int CATEGORIES = 11
```

Number of categories of blocks.

3.4.1.4 IMAGE_SUB_TOPIC

```
str IMAGE_SUB_TOPIC = '/ur5/zed_node/left/image_rect_color'
```

ROS topic from where the script get the ZED image.

3.4.1.5 MIN_LEVEL_CONFIDENCE

```
float MIN_LEVEL_CONFIDENCE = 0.3
```

Level of confidence of the Yolo model to keep the assigned labels.

3.4.1.6 NODE_NAME

```
str NODE_NAME = 'vision'
```

ROS nodes name.

3.4.1.7 POINTCLOUD_SUB_TOPIC

```
str POINTCLOUD_SUB_TOPIC = '/ur5/zed_node/point_cloud/cloud_registered'
```

ROS topic from where the script get the pointcloud.

3.4.1.8 PUB_TOPIC

```
str PUB_TOPIC = 'lego_position'
```

ROS topic where to publish positions.

3.4.1.9 RY

```
RY
```

Initial value:
```
00001 =  np.matrix([[ 0.       , -0.49948,  0.86632],
00002                  [-1.       ,  0.      ,  0.      ],
00003                  [-0.       , -0.86632, -0.49948]])
```

Rotation matrix of the ZED camera.

3.4.1.10 TABLE

```
list TABLE = [[825,549], [1301,552], [1570,913], [658, 921]]
```

Area where the vision detect blocks.

3.4.1.11 ZED_IMG_CROPPED_PATH

```
str ZED_IMG_CROPPED_PATH = os.getcwd() + '/cropped_zed_image.png'
```

Path where the cropped image is saved (a mask is applied to the photo to reduce confusion)

3.4.1.12 ZED_IMG_PATH

```
str ZED_IMG_PATH = os.getcwd() + '/zed_image.png'
```

Path where the original ZED image is saved.

3.4.1.13 ZED_POSITION

```
ZED_POSITION = np.array([-0.9 ,0.24 ,-0.35])
```

Zed position regarding to the base link frame.

3.5 scale_legos Namespace Reference

Functions

- scale_legos ()

  This function scale legos modifying their sdf file.

Variables

- str MODELS_PATH = os.getcwd().replace('utils', '') + 'locosim/ros_impedance_controller/worlds/models'

  Path to the world models sdf(s)
- float NEW_SCALE_FACTOR = 0.8

  The new value for the scale of the legos.
- float OLD_SCALE_FACTOR = 0.9

  The value that already is used to scale legos.
- str SPAWN_PATH = os.getcwd().replace('utils', '') + 'spawnLego/models'

  Path to the spawn models sdf(s)

3.5.1 Function Documentation

3.5.1.1 scale_legos()

```
scale_legos ( )
```

This function scale legos modifying their sdf file.

This was done in order to increase the spacing between legos and enhance recognition performance.

3.5.2 Variable Documentation

3.5.2.1 MODELS_PATH

```
str MODELS_PATH = os.getcwd().replace('utils', '') + 'locosim/ros_impedance_controller/worlds/models'
```

Path to the world models sdf(s)

3.5.2.2 NEW_SCALE_FACTOR

```
float NEW_SCALE_FACTOR = 0.8
```

The new value for the scale of the legos.

3.5.2.3 OLD_SCALE_FACTOR

```
float OLD_SCALE_FACTOR = 0.9
```

The value that already is used to scale legos.

3.5.2.4 SPAWN_PATH

```
str SPAWN_PATH = os.getcwd().replace('utils', '') + 'spawnLego/models'
```

Path to the spawn models sdf(s)

3.6 spawnLego Namespace Reference

Functions

- changeModelColor (model_xml, color)

  Changes the color of model.
- check_sovrapposizioni (pos, lego)

  This function check if there is conflict in spawn with other legos.
- del_model (model)

  Removes the model with 'modelName' from the Gazebo scene.
- randNum (min, max)

  Generates a random number.
- random_position ()

  Generates a random position and rotation in the spawning zone.
- spawn_model (model, pos, name=None, ref_frame='world')

  Spawns the model in the given position.

Variables

- list colorList = ['Gazebo/Indigo', 'Gazebo/Gold', 'Gazebo/Orange','Gazebo/Red', 'Gazebo/Purple', 'Gazebo/Grass','Gazebo/White', 'Gazebo/Green', 'Gazebo/Yellow', 'Gazebo/Blue', 'Gazebo/Turquoise']

  Colors of the generated legos.
- list models = ["X1-Y1-Z2", "X1-Y2-Z1", "X1-Y2-Z2-CHAMFER", "X1-Y2-Z2-TWINFILLET", "X1-Y2-Z2", "X1-Y3-Z2", "X1-Y4-Z1", "X1-Y4-Z2", "X2-Y2-Z2-FILLET", "X2-Y2-Z2", "X1-Y3-Z2-FILLET"]

  Name of the models.
- str models_path = os.path.dirname(os.path.abspath(__file__)) + "/models"

  Path of the models to add to the scene.
- float toll = 0.039

  Spacing factor for placing pieces.

3.6.1 Function Documentation

3.6.1.1 changeModelColor()

```
changeModelColor (
            model_xml,
            color )
```

Changes the color of model.

Parameters

(xml)	xml of model
(string)	color to apply

Returns

string: color

3.6.1.2 check_sovrapposizioni()

```
check_sovrapposizioni (
            pos,
            lego )
```

This function check if there is conflict in spawn with other legos.

Parameters

pos	(array): positions of other legos
lego	(string): new lego

Returns

bool : True if there is conflict

3.6.1.3 del_model()

```
del_model (
            model )
```

Removes the model with 'modelName' from the Gazebo scene.

Parameters

model	(string): name of the model to be deleted

Returns

bool : True if the deletion succeded

3.6.1.4 randNum()

```
randNum (
            min,
            max )
```

Generates a random number.

Parameters

min	(int): the minimum number
max	(int): the maximum number

Returns

int : the random number generated

3.6.1.5 random_position()

```
random_position ( )
```

Generates a random position and rotation in the spawning zone.

Returns

Pose : the generated position for the brick

3.6.1.6 spawn_model()

```
spawn_model (
              model,
              pos,
              name = None,
              ref_frame = 'world' )
```

Spawns the model in the given position.

Parameters

model	(string): the name of the lego model
pos	(struct): all the parameters for position and orientation of the lego
name	(string, optional): the name of the model. Defaults to None.
ref_frame	(string, optional): the reference frame. Defaults to 'world'

Returns

string : confirmation of the action

3.6.2 Variable Documentation

3.6.2.1 colorList

```
list colorList = ['Gazebo/Indigo', 'Gazebo/Gold', 'Gazebo/Orange','Gazebo/Red', 'Gazebo/Purple',
'Gazebo/Grass','Gazebo/White', 'Gazebo/Green', 'Gazebo/Yellow', 'Gazebo/Blue', 'Gazebo/Turquoise']
```

Colors of the generated legos.

3.6.2.2 models

```
list models = ["X1-Y1-Z2", "X1-Y2-Z1", "X1-Y2-Z2-CHAMFER", "X1-Y2-Z2-TWINFILLET", "X1-Y2-Z2",
"X1-Y3-Z2", "X1-Y4-Z1", "X1-Y4-Z2", "X2-Y2-Z2-FILLET", "X2-Y2-Z2", "X1-Y3-Z2-FILLET"]
```

Name of the models.

3.6.2.3 models_path

```
str models_path = os.path.dirname(os.path.abspath(__file__)) + "/models"
```

Path of the models to add to the scene.

3.6.2.4 toll

```
float toll = 0.039
```

Spacing factor for placing pieces.

3.7 training Namespace Reference

Variables

- list metrics = [ ]
- model = YOLO('yolov8m.pt')
- PROJECT_DIR = os.getcwd()
- results
- split = len([entry for entry in os.listdir(PROJECT_DIR + '/split_5_5-Fold_Cross-val') if os.path.isdir(os.path.join(PROJECT_DIR + '/split_5_5-Fold_Cross-val', entry))])

3.7.1 Variable Documentation

3.7.1.1 metrics

```
list metrics = [ ]
```

3.7.1.2 model

```
model = YOLO('yolov8m.pt')
```

3.7.1.3 PROJECT_DIR

```
PROJECT_DIR = os.getcwd()
```

3.7.1.4 results

```
results
```

Initial value:

```
=  model.train(
        data=PROJECT_DIR + '/split_5_5-Fold_Cross-val/split_' + str(i+1) + '/split_' + str(i+1) +
    '_dataset.yaml',
        epochs=20,
        imgsz=1024,
        device=[0,1],
        save = True,
        workers = 8,
        batch = 10,
        project = 'robotica',
    )
```

3.7.1.5 split

```
split = len([entry for entry in os.listdir(PROJECT_DIR + '/split_5_5-Fold_Cross-val') if os.↩
path.isdir(os.path.join(PROJECT_DIR + '/split_5_5-Fold_Cross-val', entry))])
```

3.8 vision Namespace Reference

Functions

- build_pose (block)

  Find three useful points for compute position and orientation of a block from all the points contained in it, moreover compute the coordinates of the center, find orientation, convert it to quaternions and in the end create the Pose object.
- find_center (y_max_point, y_min_point)

  Find the coordinates of the center of a block.
- find_orientation (y_max_point, x_min_point)

  Find the yaw of a block in Euler angles.
- pointCloudCallBack ()

  This function waits a message from a pointcloud and then reads the points from it and compute the position and the orientation of the blocks in the Gazebo scenario.
- receive_image (data)

  Recive image from ros and save it.

Variables

- image_sub = rospy.Subscriber(IMAGE_SUB_TOPIC, Image, callback=receive_image, queue_size = 1)
- loop_rate = rospy.Rate(1.)
- pos_pub = rospy.Publisher(PUB_TOPIC, legoGroup, queue_size = 11)

3.8.1 Function Documentation

3.8.1.1 build_pose()

```
build_pose (
              block )
```

Find three useful points for compute position and orientation of a block from all the points contained in it, moreover compute the coordinates of the center, find orientation, convert it to quaternions and in the end create the Pose object.

Parameters

block	Block The block object whose position and orientation you want to find

Returns

The pose of a block

3.8.1.2 find_center()

```
find_center (
              y_max_point,
              y_min_point )
```

Find the coordinates of the center of a block.

Parameters

y_max_point	list The coordinates of the point of the block with the biggest y
y_min_point	list The coordinates of the point of the block with the smallest y

Returns

List The coordinates of the center

3.8.1.3 find_orientation()

```
find_orientation (
              y_max_point,
              x_min_point )
```

Find the yaw of a block in Euler angles.

Parameters

y_max_point	list The coordinates of the point of the block with the biggest y
x_min_point	list The coordinates of the point of the block with the smallest x

Returns

Double The yaw angle of the block

3.8.1.4 pointCloudCallBack()

```
pointCloudCallBack ( )
```

This function waits a message from a pointcloud and then reads the points from it and compute the position and the orientation of the blocks in the Gazebo scenario.

In the end it populate the list with all messages for Ros

3.8.1.5 receive_image()

```
receive_image (
              data )
```

Recive image from ros and save it.

3.8.2 Variable Documentation

3.8.2.1 image_sub

```
image_sub = rospy.Subscriber(IMAGE_SUB_TOPIC, Image, callback=receive_image, queue_size = 1)
```

3.8.2.2 loop_rate

```
loop_rate = rospy.Rate(1.)
```

3.8.2.3 pos_pub

```
pos_pub = rospy.Publisher(PUB_TOPIC, legoGroup, queue_size = 11)
```

3.9 yolo-k-fold-splitter Namespace Reference

Variables

- classes = yaml.safe_load(y)['names']
- cls_idx = sorted(range(0, len(classes)))
- dataset_path = Path('./yolo_dataset')
- str dataset_yaml = split_dir / f'{split}_dataset.yaml'
- list ds_yamls = [ ]
- encoding
- exist_ok
- fold_lbl_distrb = pd.DataFrame(index=folds, columns=cls_idx)
- list folds = [f'split_{n}' for n in range(1, ksplit + 1)]
- folds_df = pd.DataFrame(index=indx, columns=folds)
- list images = [ ]
- str img_to_path = save_path / split / k_split / 'images'
- list indx = [l.stem for l in labels]
- kf = KFold(n_splits=ksplit, shuffle=True, random_state=20)
- kfolds = list(kf.split(labels_df))
- int ksplit = 5
- labels = sorted(dataset_path.rglob("∗labels/∗.txt"))
- labels_df = pd.DataFrame([ ], columns=cls_idx, index=indx)
- lbl_counter = Counter()
- str lbl_to_path = save_path / split / k_split / 'labels'
- lines = lf.readlines()
- parents
- ratio = val_totals / (train_totals + 1E-7)
- save_path = Path(dataset_path / f'split_{ksplit}_{ksplit}-Fold_Cross-val')
- split_dir = save_path / split
- start
- list supported_extensions = ['.jpg', '.jpeg', '.png']
- train_totals = labels_df.iloc[train_indices].sum()
- True
- val_totals = labels_df.iloc[val_indices].sum()
- str yaml_file = './yolo_dataset/data.yaml'

3.9.1 Variable Documentation

3.9.1.1 classes

```
classes = yaml.safe_load(y)['names']
```

3.9.1.2 cls_idx

```
cls_idx = sorted(range(0, len(classes)))
```

3.9.1.3 dataset_path

```
dataset_path = Path('./yolo_dataset')
```

3.9.1.4 dataset_yaml

```
str dataset_yaml = split_dir / f'{split}_dataset.yaml'
```

3.9.1.5 ds_yamls

```
list ds_yamls = []
```

3.9.1.6 encoding

```
encoding
```

3.9.1.7 exist_ok

```
exist_ok
```

3.9.1.8 fold_lbl_distrb

```
fold_lbl_distrb = pd.DataFrame(index=folds, columns=cls_idx)
```

3.9.1.9 folds

```
list folds = [f'split_{n}' for n in range(1, ksplit + 1)]
```

3.9.1.10 folds_df

```
folds_df = pd.DataFrame(index=indx, columns=folds)
```

3.9.1.11 images

```
list images = []
```

3.9.1.12 img_to_path

```
str img_to_path = save_path / split / k_split / 'images'
```

3.9.1.13 indx

```
list indx = [l.stem for l in labels]
```

3.9.1.14 kf

```
kf = KFold(n_splits=ksplit, shuffle=True, random_state=20)
```

3.9.1.15 kfolds

```
kfolds = list(kf.split(labels_df))
```

3.9.1.16 ksplit

```
int ksplit = 5
```

3.9.1.17 labels

```
labels = sorted(dataset_path.rglob("*labels/*.txt"))
```

3.9.1.18 labels_df

```
labels_df = pd.DataFrame([], columns=cls_idx, index=indx)
```

3.9.1.19 lbl_counter

```
lbl_counter = Counter()
```

3.9.1.20 lbl_to_path

```
str lbl_to_path = save_path / split / k_split / 'labels'
```

3.9.1.21 lines

```
lines = lf.readlines()
```

3.9.1.22 parents

```
parents
```

3.9.1.23 ratio

```
ratio = val_totals / (train_totals + 1E-7)
```

3.9.1.24 save_path

```
save_path = Path(dataset_path / f'split_{ksplit}_{ksplit}-Fold_Cross-val')
```

3.9.1.25 split_dir

```
split_dir = save_path / split
```

3.9.1.26 start

```
start
```

3.9.1.27 supported_extensions

```
list supported_extensions = ['.jpg', '.jpeg', '.png']
```

3.9.1.28 train_totals

```
train_totals = labels_df.iloc[train_indices].sum()
```

3.9.1.29 True

```
True
```

3.9.1.30 val_totals

```
val_totals = labels_df.iloc[val_indices].sum()
```

3.9.1.31 yaml_file

```
str yaml_file = './yolo_dataset/data.yaml'
```

Chapter 4

Class Documentation

4.1 Block Class Reference

Class that rapresent a block.

Public Member Functions

- __init__ (self, category, category_id, confidence, xyxy, zed_img_cropped)

Public Attributes

- category
- category_id
- confidence
- image
- points
- points_count
- xyxy

4.1.1 Detailed Description

Class that rapresent a block.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 __init__()

```
__init__ (
              self,
              category,
              category_id,
              confidence,
              xyxy,
              zed_img_cropped )
```

4.1.3 Member Data Documentation

4.1.3.1 category

`category`

4.1.3.2 category_id

`category_id`

4.1.3.3 confidence

`confidence`

4.1.3.4 image

`image`

4.1.3.5 points

`points`

4.1.3.6 points_count

`points_count`

4.1.3.7 xyxy

`xyxy`

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

- vision/detect_blocks.py

4.2 DetectArea Class Reference

Class that detect the area in wich detect blocks.

Public Member Functions

- __init__ (self, input_img, output_img_path)
  Initialization of the class.
- create_mask (self)
  Create a mask to keep in view only the area in wich there are the blocks to avoid error on detection.

Public Attributes

- input_img
- output_img_path

4.2.1 Detailed Description

Class that detect the area in wich detect blocks.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 __init__()

```
__init__ (
        self,
        input_img,
        output_img_path )
```

Initialization of the class.

4.2.3 Member Function Documentation

4.2.3.1 create_mask()

```
create_mask (
        self )
```

Create a mask to keep in view only the area in wich there are the blocks to avoid error on detection.

4.2.4 Member Data Documentation

4.2.4.1 input_img

```
input_img
```

4.2.4.2 output_img_path

```
output_img_path
```

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

- vision/detect_area.py

4.3 DetectBlocks Class Reference

Class that detect blocks.

Public Member Functions

- None __init__ (self, zed_img)

  Initializing the model, detect the area and create a mask to improve the recognition of the blocks.
- find_blocks (self)

  Detect blocks in the image using YOLOv8 model, save the result of the detection in the runs folder and print the result of the detection.

Public Attributes

- blocks
- model
- zed_img
- zed_img_cropped

4.3.1 Detailed Description

Class that detect blocks.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 __init__()

```
None __init__ (
              self,
              zed_img )
```

Initializing the model, detect the area and create a mask to improve the recognition of the blocks.

Parameters

The	image in wich there are the block to recognize

4.3.3 Member Function Documentation

4.3.3.1 find_blocks()

```
find_blocks (
              self )
```

Detect blocks in the image using YOLOv8 model, save the result of the detection in the runs folder and print the result of the detection.

4.3.4 Member Data Documentation

4.3.4.1 blocks

```
blocks
```

4.3.4.2 model

```
model
```

4.3.4.3 zed_img

```
zed_img
```

4.3.4.4 zed_img_cropped

```
zed_img_cropped
```

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

- vision/detect_blocks.py

4.4 frame Struct Reference

Structure to store the position and rotation of the end effector.

```
#include <kinematics.h>
```

Public Attributes

- Matrix3f rot
- Vector3f xyz

4.4.1 Detailed Description

Structure to store the position and rotation of the end effector.

4.4.2 Member Data Documentation

4.4.2.1 rot

```
Matrix3f rot
```

4.4.2.2 xyz

```
Vector3f xyz
```

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

- planner_pkg/include/planner_pkg/kinematics.h

Chapter 5

File Documentation

5.1 planner_pkg/include/planner_pkg/kinematics.h File Reference

```
#include <Eigen/Dense>
#include <Eigen/Geometry>
#include <cmath>
```

Include dependency graph for kinematics.h: This graph shows which files directly or indirectly include this file:

Classes

- struct frame

  Structure to store the position and rotation of the end effector.

Functions

- frame direct_kinematics (VectorXf th)

  Compute the direct kinematics.
- Matrix3f eul2rotm (Vector3f rpy)

  From euler angles to rotation matrix.
- MatrixXf inverse_kinematics (frame &frame)

  Compute the inverse kinematics.
- MatrixXf jacobian (VectorXf q)

  Calculate the jacobian matrix.
- Vector3f rotm2eul (Matrix3f R)

  From rotation matrix to euler angles.
- Matrix4f t10f (float th1)

  Functions prototypes.
- Matrix4f t21f (float th2)

  Create the transformation matrix for the second joint.
- Matrix4f t32f (float th3)

  Create the transformation matrix for the third joint.
- Matrix4f t43f (float th4)

  Create the transformation matrix for the fourth joint.
- Matrix4f t54f (float th5)

  Create the transformation matrix for the fifth joint.
- Matrix4f t65f (float th6)

  Create the transformation matrix for the sixth joint.

5.1.1 Detailed Description

Author

Soldera Marco (marco.soldera@studenti.unitn.it) - Group Soldera Marco and Morandin Marco

Version

0.1

Date

2024-02-05

Copyright

Copyright (c) 2024

5.1.2 Function Documentation

5.1.2.1 direct_kinematics()

```
frame direct_kinematics (
            VectorXf th )
```

Compute the direct kinematics.

Parameters

th	Joint angles

Returns

frame

5.1.2.2 eul2rotm()

```
Matrix3f eul2rotm (
            Vector3f rpy )
```

From euler angles to rotation matrix.

Parameters

rpy	Euler angles

Returns

Matrix3f

5.1.2.3 inverse_kinematics()

```
MatrixXf inverse_kinematics (
            frame & frame )
```

Compute the inverse kinematics.

Parameters

frame	Current frame of the end effector

Returns

MatrixXf

5.1.2.4 jacobian()

```
MatrixXf jacobian (
            VectorXf q )
```

Calculate the jacobian matrix.

Parameters

q	Joint angles

Returns

MatrixXf

5.1.2.5 rotm2eul()

```
Vector3f rotm2eul (
            Matrix3f R )
```

From rotation matrix to euler angles.

Parameters

R	Rotation matrix

Returns

Vector3f

5.1.2.6 t10f()

```
Matrix4f t10f (
             float th1 )
```

Functions prototypes.

Functions prototypes.

Parameters

th1	Angle of the first joint

Returns

Matrix4f

5.1.2.7 t21f()

```
Matrix4f t21f (
             float th2 )
```

Create the transformation matrix for the second joint.

Parameters

th2	Angle of the second joint

Returns

Matrix4f

5.1.2.8 t32f()

```
Matrix4f t32f (
             float th3 )
```

Create the transformation matrix for the third joint.

Parameters

th3	Angle of the third joint

Returns

Matrix4f

5.1.2.9 t43f()

```
Matrix4f t43f (
            float th4 )
```

Create the transformation matrix for the fourth joint.

Parameters

th4	Angle of the fourth joint

Returns

Matrix4f

5.1.2.10 t54f()

```
Matrix4f t54f (
            float th5 )
```

Create the transformation matrix for the fifth joint.

Parameters

th5	Angle of the fifth joint

Returns

Matrix4f

5.1.2.11 t65f()

```
Matrix4f t65f (
            float th6 )
```

Create the transformation matrix for the sixth joint.

Parameters

th6	Angle of the sixth joint

Returns

Matrix4f

5.2 kinematics.h

Go to the documentation of this file.

```
#ifndef __KINEMATICS_H__
#define __KINEMATICS_H__


#include <Eigen/Dense>
#include <Eigen/Geometry>
#include <cmath>


using namespace std;
using namespace Eigen;


struct frame {
    Vector3f xyz;
    Matrix3f rot;
};


Matrix4f t10f(float th1);
Matrix4f t21f(float th2);
Matrix4f t32f(float th3);
Matrix4f t43f(float th4);
Matrix4f t54f(float th5);
Matrix4f t65f(float th6);

frame direct_kinematics(VectorXf th);
MatrixXf inverse_kinematics(frame &frame);
MatrixXf jacobian(VectorXf q);

Matrix3f eul2rotm(Vector3f rpy);
Vector3f rotm2eul(Matrix3f R);


#endif
```

5.3 planner_pkg/include/planner_pkg/movement.h File Reference

```
#include "kinematics.h"
#include "ros/ros.h"
#include <std_msgs/Float64MultiArray.h>
#include <sensor_msgs/JointState.h>
#include <Eigen/Dense>
#include <Eigen/Geometry>
#include <iostream>
```

Include dependency graph for movement.h: This graph shows which files directly or indirectly include this file:

Functions

- VectorXf invDiffKinematicControlCompleteQuaternion (VectorXf q, Vector3f xe, Vector3f xd, Vector3f vd, Vector3f omegad, Quaternionf qe, Quaternionf qd, Matrix3f Kp, Matrix3f Kq, int f)

    Calculates joint velocities using the jacobian matrix.
- void invDiffKinematicControlSimCompleteQuaternion (Vector3f xef, Vector3f phief, double dt, VectorXf jstates, void(*send_j)(VectorXf))

    Calculates joint configs using quaternions.
- Quaternionf operator* (float num, const Quaternionf &q)

Function prototypes.

- Vector3f pd (float t, Vector3f xef, Vector3f xe0)

  Calculates trajectory for the end-effector position.

- Quaternionf qd (float tb, Quaternionf q0, Quaternionf qf)

  Calculates trajectory for the end-effector orientation with quaternions.

Variables

- float maxT = 6

  Max time for the trajectory.

5.3.1 Detailed Description

Author

Soldera Marco (marco.soldera@studenti.unitn.it) - Group Soldera Marco and Morandin Marco

Version

0.1

Date

2024-02-05

Copyright

5.3.2 Function Documentation

5.3.2.1 invDiffKinematicControlCompleteQuaternion()

```
VectorXf invDiffKinematicControlCompleteQuaternion (
            VectorXf q,
            Vector3f xe,
            Vector3f xd,
            Vector3f vd,
            Vector3f omegad,
            Quaternionf qe,
            Quaternionf qd,
            Matrix3f Kp,
            Matrix3f Kq,
            int f )
```

Calculates joint velocities using the jacobian matrix.

Parameters

q	The current joint config
xe	The current end-effector position
xd	The desired end-effector position
vd	The desired end-effector linear velocity
omegad	The desired end-effector angular velocity
qe	The current end-effector rotation in quaternion
qd	The desired end-effector rotation in quaternion
Kd	The position gain
Kq	The orientation gain
f	counter for debugging

Returns

Vector6f

5.3.2.2 invDiffKinematicControlSimCompleteQuaternion()

```
void invDiffKinematicControlSimCompleteQuaternion (
            Vector3f xef,
            Vector3f phief,
            double dt,
            VectorXf jstates,
            void(*)(VectorXf) send_j )
```

Calculates joint configs using quaternions.

Parameters

xef	Desired end-effector position
phief	Desired end-effector orientation
dt	Time step
jstates	Actual state of the joints
send_j	Function to send joint states

Returns

void

5.3.2.3 operator*()

```
Quaternionf operator* (
            float num,
            const Quaternionf & q )
```

Function prototypes.

Redefinition of multiplication between float and Quaternionf

Parameters

num	Scalar that multiplies
q	Quaternion to be multiplied

Returns

Quaternionf

5.3.2.4 pd()

```
Vector3f pd (
            float tb,
            Vector3f xef,
            Vector3f xe0 )
```

Calculates trajectory for the end-effector position.

Parameters

t	The current time
xef	The desired end-effector position
xe0	The start end-effector position

Returns

Vector3f

5.3.2.5 qd()

```
Quaternionf qd (
            float tb,
            Quaternionf q0,
            Quaternionf qf )
```

Calculates trajectory for the end-effector orientation with quaternions.

Parameters

tb	The current time
q0	The start end-effector quaternion
qf	The desired end-effector quaternion

Returns

Quaternionf

5.3.3 Variable Documentation

5.3.3.1 maxT

```
float maxT = 6
```

Max time for the trajectory.

5.4 movement.h

Go to the documentation of this file.

```
#ifndef __MOVEMENT_H__
#define __MOVEMENT_H__


#include "kinematics.h"

#include "ros/ros.h"
#include <std_msgs/Float64MultiArray.h>
#include <sensor_msgs/JointState.h>
#include <Eigen/Dense>
#include <Eigen/Geometry>

#include <iostream>


using namespace std;
using namespace Eigen;


inline float maxT = 6;


inline Quaternionf operator *(float num, const Quaternionf& q) {
    return Quaternionf(q.x() * num, q.y() * num, q.z() * num, q.w() * num);
}

Vector3f pd(float t, Vector3f xef, Vector3f xe0);
Quaternionf qd(float tb, Quaternionf q0, Quaternionf qf);
VectorXf invDiffKinematicControlCompleteQuaternion(VectorXf q, Vector3f xe, Vector3f xd, Vector3f vd, Vector3f omegad, Quaternionf qe, Quaternionf qd,  Matrix3f Kp, Matrix3f Kq, int f);
void invDiffKinematicControlSimCompleteQuaternion(Vector3f xef, Vector3f phief, double dt, VectorXf jstates, void (*send_j)(VectorXf));


#endif
```

5.5 planner_pkg/include/planner_pkg/planner.h File Reference

```
#include "movement.h"
#include <planner_pkg/legoDetection.h>
#include <planner_pkg/legoGroup.h>
#include <ros/ros.h>
#include <Eigen/Dense>
#include <Eigen/Geometry>
#include <iostream>
#include <vector>
#include <cmath>
```

Include dependency graph for planner.h: This graph shows which files directly or indirectly include this file:

Typedefs

- typedef Matrix< float, 2, 1 > GripperState

  Position of the components of the gripper.

Functions

- void close_gripper (float amp)

  Close the gripper of the robot.
- GripperState get_gripper_states ()

  Read from the topic the actual value of the gripper joint.
- VectorXf get_joint_states ()

  Read from the topic the actual value of the joint.
- void listen_lego_detection (ros::Rate rate)

  Functions prototype.
- void move_to_home ()

  Moves the robot to the home position.
- void open_gripper (float amp)

  Open the gripper of the robot.
- Vector3f quat2eul (Quaternionf q)

  Convert from Quaternion to Euler Angles.
- void set_joint_states (VectorXf q)

  Posts on the topic the vector joint_pos, which contains the value of the angles, that all joints must reach.
- void waitJoints (bool waitRot, Vector3f xef, Vector3f phief)

  Wait for joints to be at the final position.
- void waitSec (float t)

  Wait for the specified time.
- Vector3f X1_Y1_Z2 (0.92, 0.27, 0.88)

  Final positions of bricks based on type.
- Vector3f X1_Y2_Z1 (0.77, 0.27, 0.88)
- Vector3f X1_Y2_Z2 (0.62, 0.27, 0.88)
- Vector3f X1_Y2_Z2_CHAMFER (0.92, 0.42, 0.88)
- Vector3f X1_Y2_Z2_TWINFILLET (0.77, 0.42, 0.88)
- Vector3f X1_Y3_Z2 (0.77, 0.56, 0.88)
- Vector3f X1_Y3_Z2_FILLET (0.62, 0.56, 0.88)
- Vector3f X1_Y4_Z1 (0.87, 0.72, 0.88)
- Vector3f X1_Y4_Z2 (0.65, 0.72, 0.88)
- Vector3f X2_Y2_Z2 (0.62, 0.42, 0.89)
- Vector3f X2_Y2_Z2_FILLET (0.92, 0.56, 0.89)

Variables

- GripperState actual_gripper

  Position of the gripper.
- std_msgs::Float64MultiArray jointState_msg_robot

  Structure of the message to be published with joint positions.
- float loop_frequency = 1000.0

  Loop rate of the node.
- map< std::string, Vector3f > models_map
- ros::Publisher pub_joint_states

  Publisher for the desired joint state.
- double timeStep = 0.001

  Time step.

5.5.1 Detailed Description

Author

Soldera Marco (marco.soldera@studenti.unitn.it) - Group Soldera Marco and Morandin Marco

Version

0.1

Date

2024-02-05

Copyright

Copyright (c) 2024

5.5.2 Typedef Documentation

5.5.2.1 GripperState

```
typedef Matrix<float, 2, 1> GripperState
```

Position of the components of the gripper.

5.5.3 Function Documentation

5.5.3.1 close_gripper()

```
void close_gripper (
            float amp )
```

Close the gripper of the robot.

We public directly on the topic the angles that we want to reach with the fingers of the gripper

Parameters

amp	Required negative gripper length (positive value)

Returns

void

5.5.3.2 get_gripper_states()

```
GripperState get_gripper_states ( )
```

Read from the topic the actual value of the gripper joint.

Returns

GripperState

5.5.3.3 get_joint_states()

```
VectorXf get_joint_states ( )
```

Read from the topic the actual value of the joint.

Returns

Vector6f

5.5.3.4 listen_lego_detection()

```
void listen_lego_detection (
            ros::Rate rate )
```

Functions prototype.

Functions prototype.

Parameters

rate	ros rate

Returns

void

5.5.3.5 move_to_home()

```
void move_to_home ( )
```

Moves the robot to the home position.

Returns

void

5.5.3.6 open_gripper()

```
void open_gripper (
            float amp )
```

Open the gripper of the robot.

We public directly on the topic the angles that we want to reach with the fingers of the gripper

Parameters

amp	Required gripper length

Returns

void

5.5.3.7 quat2eul()

```
Vector3f quat2eul (
            Quaternionf q )
```

Convert from Quaternion to Euler Angles.

Parameters

q	Quaternion to convert

Returns

Vector3f

5.5.3.8 set_joint_states()

```
void set_joint_states (
            VectorXf joint_pos )
```

Posts on the topic the vector joint_pos, which contains the value of the angles, that all joints must reach.

Parameters

joint_pos	Vector that conatins the values of each joint angle position to be published

Returns

void

5.5.3.9 waitJoints()

```
void waitJoints (
            bool waitRot,
            Vector3f xef,
            Vector3f phief )
```

Wait for joints to be at the final position.

Parameters

waitRot	wait also for z rotation to be aligned
xef	final positon
phief	final rotation

Returns

void

5.5.3.10 waitSec()

```
void waitSec (
            float t )
```

Wait for the specified time.

Parameters

t	Time to wait

Returns

void

5.5.3.11 X1_Y1_Z2()

```
Vector3f X1_Y1_Z2 (
            0.  92,
            0.  27,
            0.  88 )
```

Final positions of bricks based on type.

5.5.3.12 X1_Y2_Z1()

```
Vector3f X1_Y2_Z1 (
            0.  77,
            0.  27,
            0.  88 )
```

5.5.3.13 X1_Y2_Z2()

```
Vector3f X1_Y2_Z2 (
            0.  62,
            0.  27,
            0.  88 )
```

5.5.3.14 X1_Y2_Z2_CHAMFER()

```
Vector3f X1_Y2_Z2_CHAMFER (
            0. 92,
            0. 42,
            0. 88 )
```

5.5.3.15 X1_Y2_Z2_TWINFILLET()

```
Vector3f X1_Y2_Z2_TWINFILLET (
            0. 77,
            0. 42,
            0. 88 )
```

5.5.3.16 X1_Y3_Z2()

```
Vector3f X1_Y3_Z2 (
            0. 77,
            0. 56,
            0. 88 )
```

5.5.3.17 X1_Y3_Z2_FILLET()

```
Vector3f X1_Y3_Z2_FILLET (
            0. 62,
            0. 56,
            0. 88 )
```

5.5.3.18 X1_Y4_Z1()

```
Vector3f X1_Y4_Z1 (
            0. 87,
            0. 72,
            0. 88 )
```

5.5.3.19 X1_Y4_Z2()

```
Vector3f X1_Y4_Z2 (
            0. 65,
            0. 72,
            0. 88 )
```

5.5.3.20 X2_Y2_Z2()

```
Vector3f X2_Y2_Z2 (
            0. 62,
            0. 42,
            0. 89 )
```

5.5.3.21 X2_Y2_Z2_FILLET()

```
Vector3f X2_Y2_Z2_FILLET (
            0.  92,
            0.  56,
            0.  89 )
```

5.5.4 Variable Documentation

5.5.4.1 actual_gripper

```
GripperState actual_gripper
```

Position of the gripper.

5.5.4.2 jointState_msg_robot

```
std_msgs::Float64MultiArray jointState_msg_robot
```

Structure of the message to be published with joint positions.

5.5.4.3 loop_frequency

```
float loop_frequency = 1000.0
```

Loop rate of the node.

5.5.4.4 models_map

```
map<std::string, Vector3f> models_map
```

Initial value:

```
{
    {"X1-Y1-Z2", X1_Y1_Z2},
    {"X1-Y2-Z1", X1_Y2_Z1},
    {"X1-Y2-Z2", X1_Y2_Z2},
    {"X1-Y2-Z2-CHAMFER", X1_Y2_Z2_CHAMFER},
    {"X1-Y2-Z2-TWINFILLET", X1_Y2_Z2_TWINFILLET},
    {"X2-Y2-Z2", X2_Y2_Z2},
    {"X2-Y2-Z2-FILLET",X2_Y2_Z2_FILLET},
    {"X1-Y3-Z2", X1_Y3_Z2},
    {"X1-Y3-Z2-FILLET",X1_Y3_Z2_FILLET},
    {"X1-Y4-Z1", X1_Y4_Z1},
    {"X1-Y4-Z2", X1_Y4_Z2}
}
```

5.5.4.5 pub_joint_states

```
ros::Publisher pub_joint_states
```

Publisher for the desired joint state.

5.5.4.6 timeStep

```
double timeStep = 0.001
```

Time step.

5.6 planner.h

Go to the documentation of this file.

```
#ifndef __PLANNER_H__
#define __PLANNER_H__
#include "movement.h"
#include <planner_pkg/legoDetection.h>
#include <planner_pkg/legoGroup.h>
#include <ros/ros.h>
#include <Eigen/Dense>
#include <Eigen/Geometry>
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;
using namespace Eigen;
typedef Matrix<float, 2, 1> GripperState;
float loop_frequency = 1000.0;
double timeStep = 0.001;
GripperState actual_gripper;
ros::Publisher pub_joint_states;
std_msgs::Float64MultiArray jointState_msg_robot;
void listen_lego_detection(ros::Rate rate);
void move_to_home();
Vector3f quat2eul(Quaternionf q);
VectorXf get_joint_states();
void set_joint_states(VectorXf q);
GripperState get_gripper_states();
void open_gripper(float amp);
void close_gripper(float amp);
void waitSec(float t);
void waitJoints(bool waitRot, Vector3f xef, Vector3f phief);
Vector3f X1_Y1_Z2(0.92, 0.27, 0.88);
Vector3f X1_Y2_Z1(0.77, 0.27, 0.88);
Vector3f X1_Y2_Z2(0.62, 0.27, 0.88);
Vector3f X1_Y2_Z2_CHAMFER(0.92, 0.42, 0.88);
Vector3f X1_Y2_Z2_TWINFILLET(0.77, 0.42, 0.88);
Vector3f X2_Y2_Z2(0.62, 0.42, 0.89);
Vector3f X2_Y2_Z2_FILLET(0.92, 0.56, 0.89);
Vector3f X1_Y3_Z2(0.77, 0.56, 0.88);
Vector3f X1_Y3_Z2_FILLET(0.62, 0.56, 0.88);
Vector3f X1_Y4_Z1(0.87, 0.72, 0.88);
Vector3f X1_Y4_Z2(0.65, 0.72, 0.88);
map<std::string, Vector3f> models_map{
    {"X1-Y1-Z2", X1_Y1_Z2},
    {"X1-Y2-Z1", X1_Y2_Z1},
    {"X1-Y2-Z2", X1_Y2_Z2},
    {"X1-Y2-Z2-CHAMFER", X1_Y2_Z2_CHAMFER},
    {"X1-Y2-Z2-TWINFILLET", X1_Y2_Z2_TWINFILLET},
    {"X2-Y2-Z2", X2_Y2_Z2},
    {"X2-Y2-Z2-FILLET",X2_Y2_Z2_FILLET},
    {"X1-Y3-Z2", X1_Y3_Z2},
```

```
    {"X1-Y3-Z2-FILLET",X1_Y3_Z2_FILLET},
    {"X1-Y4-Z1", X1_Y4_Z1},
    {"X1-Y4-Z2", X1_Y4_Z2}
};


#endif
```

5.7 planner_pkg/src/kinematics.cpp File Reference

Functions in this file are used to calculate transformation matrices, direct and inverse kinematics, jacobian matrices and trasformation between Euler angles and rotation matrices.

`#include "planner_pkg/kinematics.h"`
Include dependency graph for kinematics.cpp:

Functions

- frame direct_kinematics (VectorXf th)
  Compute the direct kinematics.
- Matrix3f eul2rotm (Vector3f rpy)
  From euler angles to rotation matrix.
- MatrixXf inverse_kinematics (frame &frame)
  Compute the inverse kinematics.
- MatrixXf jacobian (VectorXf q)
  Calculate the jacobian matrix.
- Vector3f rotm2eul (Matrix3f R)
  From rotation matrix to euler angles.
- Matrix4f t10f (float th1)
  Create the transformation matrix for the first joint.
- Matrix4f t21f (float th2)
  Create the transformation matrix for the second joint.
- Matrix4f t32f (float th3)
  Create the transformation matrix for the third joint.
- Matrix4f t43f (float th4)
  Create the transformation matrix for the fourth joint.
- Matrix4f t54f (float th5)
  Create the transformation matrix for the fifth joint.
- Matrix4f t65f (float th6)
  Create the transformation matrix for the sixth joint.

5.7.1 Detailed Description

Functions in this file are used to calculate transformation matrices, direct and inverse kinematics, jacobian matrices and trasformation between Euler angles and rotation matrices.

Author

Soldera Marco (marco.soldera@studenti.unitn.it) - Group Soldera Marco and Morandin Marco

Version

0.1

Date

2024-02-05

Copyright

5.7.2 Function Documentation

5.7.2.1 direct_kinematics()

```
frame direct_kinematics (
            VectorXf th )
```

Compute the direct kinematics.

Parameters

th	Joint angles

Returns

frame

5.7.2.2 eul2rotm()

```
Matrix3f eul2rotm (
            Vector3f rpy )
```

From euler angles to rotation matrix.

Parameters

rpy	Euler angles

Returns

Matrix3f

5.7.2.3 inverse_kinematics()

```
MatrixXf inverse_kinematics (
            frame & frame )
```

Compute the inverse kinematics.

Parameters

frame	Current frame of the end effector

Returns

MatrixXf

5.7.2.4 jacobian()

```
MatrixXf jacobian (
            VectorXf q )
```

Calculate the jacobian matrix.

Parameters

q	Joint angles

Returns

MatrixXf

5.7.2.5 rotm2eul()

```
Vector3f rotm2eul (
            Matrix3f R )
```

From rotation matrix to euler angles.

Parameters

R	Rotation matrix

Returns

Vector3f

5.7.2.6 t10f()

```
Matrix4f t10f (
            float th1 )
```

Create the transformation matrix for the first joint.

Functions prototypes.

Parameters

th1	Angle of the first joint

Returns

Matrix4f

5.7.2.7 t21f()

```
Matrix4f t21f (
            float th2 )
```

Create the transformation matrix for the second joint.

Parameters

th2	Angle of the second joint

Returns

Matrix4f

5.7.2.8 t32f()

```
Matrix4f t32f (
            float th3 )
```

Create the transformation matrix for the third joint.

Parameters

th3	Angle of the third joint

Returns

Matrix4f

5.7.2.9 t43f()

```
Matrix4f t43f (
            float th4 )
```

Create the transformation matrix for the fourth joint.

Parameters

th4	Angle of the fourth joint

Returns

Matrix4f

5.7.2.10 t54f()

```
Matrix4f t54f (
            float th5 )
```

Create the transformation matrix for the fifth joint.

Parameters

th5	Angle of the fifth joint

Returns

Matrix4f

5.7.2.11 t65f()

```
Matrix4f t65f (
            float th6 )
```

Create the transformation matrix for the sixth joint.

Parameters

th6	Angle of the sixth joint

Returns

Matrix4f

5.8 planner_pkg/src/movement.cpp File Reference

Functions in this file are used to calculate the trajectory based on quaternions and velocities of the joints.

```
#include "planner_pkg/movement.h"
```

Include dependency graph for movement.cpp:

Functions

- VectorXf invDiffKinematicControlCompleteQuaternion (VectorXf q, Vector3f xe, Vector3f xd, Vector3f vd, Vector3f omegad, Quaternionf qe, Quaternionf qd, Matrix3f Kp, Matrix3f Kq, int f)

  Calculates joint velocities using the jacobian matrix.
- void invDiffKinematicControlSimCompleteQuaternion (Vector3f xef, Vector3f phief, double dt, VectorXf jstates, void(∗send_j)(VectorXf))

  Calculates joint configs using quaternions.
- Vector3f pd (float tb, Vector3f xef, Vector3f xe0)

  Calculates trajectory for the end-effector position.
- Quaternionf qd (float tb, Quaternionf q0, Quaternionf qf)

  Calculates trajectory for the end-effector orientation with quaternions.

5.8.1 Detailed Description

Functions in this file are used to calculate the trajectory based on quaternions and velocities of the joints.

Author

Soldera Marco (`marco.soldera@studenti.unitn.it`) - Group Soldera Marco and Morandin Marco

Version

0.1

Date

2024-02-05

Copyright

5.8.2 Function Documentation

5.8.2.1 invDiffKinematicControlCompleteQuaternion()

```
VectorXf invDiffKinematicControlCompleteQuaternion (
            VectorXf q,
            Vector3f xe,
            Vector3f xd,
            Vector3f vd,
            Vector3f omegad,
            Quaternionf qe,
            Quaternionf qd,
            Matrix3f Kp,
            Matrix3f Kq,
            int f )
```

Calculates joint velocities using the jacobian matrix.

Parameters

q	The current joint config
xe	The current end-effector position
xd	The desired end-effector position
vd	The desired end-effector linear velocity
omegad	The desired end-effector angular velocity
qe	The current end-effector rotation in quaternion
qd	The desired end-effector rotation in quaternion
Kd	The position gain
Kq	The orientation gain
f	counter for debugging

Returns

Vector6f

5.8.2.2 invDiffKinematicControlSimCompleteQuaternion()

```
void invDiffKinematicControlSimCompleteQuaternion (
            Vector3f xef,
            Vector3f phief,
            double dt,
            VectorXf jstates,
            void(*)(VectorXf) send_j )
```

Calculates joint configs using quaternions.

Parameters

xef	Desired end-effector position
phief	Desired end-effector orientation
dt	Time step
jstates	Actual state of the joints
send_j	Function to send joint states

Returns

void

5.8.2.3 pd()

```
Vector3f pd (
            float tb,
            Vector3f xef,
            Vector3f xe0 )
```

Calculates trajectory for the end-effector position.

Parameters

t	The current time
xef	The desired end-effector position
xe0	The start end-effector position

Returns

Vector3f

5.8.2.4 qd()

```
Quaternionf qd (
            float tb,
            Quaternionf q0,
            Quaternionf qf )
```

Calculates trajectory for the end-effector orientation with quaternions.

Parameters

tb	The current time
q0	The start end-effector quaternion
qf	The desired end-effector quaternion

Returns

Quaternionf

5.9 planner_pkg/src/planner.cpp File Reference

Main function and planning of the movement based on the messages received from the vision; also getters and setters for joints are present.

`#include "planner_pkg/planner.h"`
Include dependency graph for planner.cpp:

Functions

- void close_gripper (float amp)

  Close the gripper of the robot.
- GripperState get_gripper_states ()

  Read from the topic the actual value of the gripper joint.
- VectorXf get_joint_states ()

  Read from the topic the actual value of the joint.
- void listen_lego_detection (ros::Rate rate)

Listen to the /lego_position topic if messages arrives from the vision node; for each lego detected it sends the robot to the lego position knowing which type of lego it is.

- int main (int argc, char **argv)
- void move_to_home ()

  Moves the robot to the home position.
- void open_gripper (float amp)

  Open the gripper of the robot.
- Vector3f quat2eul (Quaternionf q)

  Convert from Quaternion to Euler Angles.
- void set_joint_states (VectorXf joint_pos)

  Posts on the topic the vector joint_pos, which contains the value of the angles, that all joints must reach.
- void waitJoints (bool waitRot, Vector3f xef, Vector3f phief)

  Wait for joints to be at the final position.
- void waitSec (float t)

  Wait for the specified time.

5.9.1 Detailed Description

Main function and planning of the movement based on the messages received from the vision; also getters and setters for joints are present.

Author

Soldera Marco (marco.soldera@studenti.unitn.it) - Group Soldera Marco and Morandin Marco

Version

0.1

Date

2024-02-05

Copyright

Copyright (c) 2024

5.9.2 Function Documentation

5.9.2.1 close_gripper()

```
void close_gripper (
            float amp )
```

Close the gripper of the robot.

We public directly on the topic the angles that we want to reach with the fingers of the gripper

Parameters

amp	Required negative gripper length (positive value)

Returns

void

5.9.2.2 get_gripper_states()

```
GripperState get_gripper_states ( )
```

Read from the topic the actual value of the gripper joint.

Returns

GripperState

5.9.2.3 get_joint_states()

```
VectorXf get_joint_states ( )
```

Read from the topic the actual value of the joint.

Returns

Vector6f

5.9.2.4 listen_lego_detection()

```
void listen_lego_detection (
            ros::Rate rate )
```

Listen to the /lego_position topic if messages arrives from the vision node; for each lego detected it sends the robot to the lego position knowing which type of lego it is.

Functions prototype.

Parameters

rate	ros rate

Returns

void

5.9.2.5 main()

```
int main (
            int argc,
            char ** argv )
```

5.9.2.6 move_to_home()

```
void move_to_home ( )
```

Moves the robot to the home position.

Returns

void

5.9.2.7 open_gripper()

```
void open_gripper (
            float amp )
```

Open the gripper of the robot.

We public directly on the topic the angles that we want to reach with the fingers of the gripper

Parameters

amp	Required gripper length

Returns

void

5.9.2.8 quat2eul()

```
Vector3f quat2eul (
            Quaternionf q )
```

Convert from Quaternion to Euler Angles.

Parameters

q	Quaternion to convert

Returns

Vector3f

5.9.2.9 set_joint_states()

```
void set_joint_states (
            VectorXf joint_pos )
```

Posts on the topic the vector joint_pos, which contains the value of the angles, that all joints must reach.

Parameters

joint_pos	Vector that conatins the values of each joint angle position to be published

Returns

void

5.9.2.10 waitJoints()

```
void waitJoints (
            bool waitRot,
            Vector3f xef,
            Vector3f phief )
```

Wait for joints to be at the final position.

Parameters

waitRot	wait also for z rotation to be aligned
xef	final positon
phief	final rotation

Returns

void

5.9.2.11 waitSec()

```
void waitSec (
            float t )
```

Wait for the specified time.

Parameters

t	Time to wait

Returns

void

5.10 spawnLego/spawnLego.py File Reference

Spawn lego in random position and orientation.

Namespaces

- namespace spawnLego

Functions

- changeModelColor (model_xml, color)

    Changes the color of model.
- check_sovrapposizioni (pos, lego)

    This function check if there is conflict in spawn with other legos.
- del_model (model)

    Removes the model with 'modelName' from the Gazebo scene.
- randNum (min, max)

    Generates a random number.
- random_position ()

    Generates a random position and rotation in the spawning zone.
- spawn_model (model, pos, name=None, ref_frame='world')

    Spawns the model in the given position.

Variables

- list colorList = ['Gazebo/Indigo', 'Gazebo/Gold', 'Gazebo/Orange','Gazebo/Red', 'Gazebo/Purple', 'Gazebo/Grass','Gazebo/White', 'Gazebo/Green', 'Gazebo/Yellow', 'Gazebo/Blue', 'Gazebo/Turquoise']

    Colors of the generated legos.
- list models = ["X1-Y1-Z2", "X1-Y2-Z1", "X1-Y2-Z2-CHAMFER", "X1-Y2-Z2-TWINFILLET", "X1-Y2-Z2", "X1-Y3-Z2", "X1-Y4-Z1", "X1-Y4-Z2", "X2-Y2-Z2-FILLET", "X2-Y2-Z2", "X1-Y3-Z2-FILLET"]

    Name of the models.
- str models_path = os.path.dirname(os.path.abspath(__file__)) + "/models"

    Path of the models to add to the scene.
- float toll = 0.039

    Spacing factor for placing pieces.

5.10.1 Detailed Description

Spawn lego in random position and orientation.

5.10.2 Author(s)

- Created by Marco Soldera

5.11 utils/dataset_creation/dataset2Yolo.py File Reference

Convert the given dataset to a Yolo format dataset.

Namespaces

- namespace dataset2Yolo

Functions

- create_annotations ()

    This function creates annotations in .txt format for each image and saves it in /labels.
- create_yaml_file ()

    This function is used to create the yaml file required from YOLO format.

Variables

- list ASSIGNS = ['assign1', 'assign2']

    Name of the folder that contains the given dataset.
- CATEGORIES = json.load(file)

    List of all categories in the dataset.
- PROJECT_DIR = os.getcwd()

    Script working directory.

5.11.1 Detailed Description

Convert the given dataset to a Yolo format dataset.

5.11.2 Author(s)

- Created by Marco Morandin

5.12 utils/dataset_creation/yolo-k-fold-splitter.py File Reference

Namespaces

- namespace yolo-k-fold-splitter

Variables

- classes = yaml.safe_load(y)['names']
- cls_idx = sorted(range(0, len(classes)))
- dataset_path = Path('./yolo_dataset')
- str dataset_yaml = split_dir / f'{split}_dataset.yaml'
- list ds_yamls = [ ]
- encoding
- exist_ok
- fold_lbl_distrb = pd.DataFrame(index=folds, columns=cls_idx)
- list folds = [f'split_{n}' for n in range(1, ksplit + 1)]
- folds_df = pd.DataFrame(index=indx, columns=folds)
- list images = [ ]
- str img_to_path = save_path / split / k_split / 'images'
- list indx = [l.stem for l in labels]
- kf = KFold(n_splits=ksplit, shuffle=True, random_state=20)
- kfolds = list(kf.split(labels_df))
- int ksplit = 5
- labels = sorted(dataset_path.rglob("∗labels/∗.txt"))
- labels_df = pd.DataFrame([ ], columns=cls_idx, index=indx)
- lbl_counter = Counter()
- str lbl_to_path = save_path / split / k_split / 'labels'
- lines = lf.readlines()
- parents
- ratio = val_totals / (train_totals + 1E-7)
- save_path = Path(dataset_path / f'split_{ksplit}_{ksplit}-Fold_Cross-val')
- split_dir = save_path / split
- start
- list supported_extensions = ['.jpg', '.jpeg', '.png']
- train_totals = labels_df.iloc[train_indices].sum()
- True
- val_totals = labels_df.iloc[val_indices].sum()
- str yaml_file = './yolo_dataset/data.yaml'

5.13 utils/scale_legos.py File Reference

Namespaces

- namespace scale_legos

Functions

- scale_legos ()

  This function scale legos modifying their sdf file.

Variables

- str MODELS_PATH = os.getcwd().replace('utils', '') + 'locosim/ros_impedance_controller/worlds/models'

  Path to the world models sdf(s)
- float NEW_SCALE_FACTOR = 0.8

  The new value for the scale of the legos.
- float OLD_SCALE_FACTOR = 0.9

  The value that already is used to scale legos.
- str SPAWN_PATH = os.getcwd().replace('utils', '') + 'spawnLego/models'

  Path to the spawn models sdf(s)

5.13.1 Detailed Description

5.13.2 Author(s)

- Created by Marco Morandin

5.14 utils/training/training.py File Reference

Namespaces

- namespace training

Variables

- list metrics = [ ]
- model = YOLO('yolov8m.pt')
- PROJECT_DIR = os.getcwd()
- results
- split = len([entry for entry in os.listdir(PROJECT_DIR + '/split_5_5-Fold_Cross-val') if os.path.isdir(os.path.↩ join(PROJECT_DIR + '/split_5_5-Fold_Cross-val', entry))])

5.15 vision/detect_area.py File Reference

Detect the area and crop the ZED image from where the model will recognize blocks.

Classes

- class DetectArea

  Class that detect the area in wich detect blocks.

Namespaces

- namespace detect_area

Variables

- detectArea = DetectArea(input_img = img, output_img_path='detected_area.png')
- img = cv2.imread('zed_image.png')

5.15.1 Detailed Description

Detect the area and crop the ZED image from where the model will recognize blocks.

5.15.2 Author(s)

- Created by Marco Morandin

5.16 vision/detect_blocks.py File Reference

Detect blocks in the image using YOLOv8 model trained in a custom dataset.

Classes

- class Block

    Class that rapresent a block.
- class DetectBlocks

    Class that detect blocks.

Namespaces

- namespace detect_blocks

5.16.1 Detailed Description

Detect blocks in the image using YOLOv8 model trained in a custom dataset.

There is the block class that rapresent a block with his characteristics

5.16.2 Author(s)

- Created by Marco Morandin

5.17 vision/params.py File Reference

Parameters used in the vision scripts.

Namespaces

- namespace params

Variables

- BASE_LINK_POSITION = np.array([0.5,0.35,1.75])

  Base link position regarding to the origin frame.
- float BLOCK_COORD_Z = 0.875

  Height of the block regarding to the origin frame.
- int CATEGORIES = 11

  Number of categories of blocks.
- str IMAGE_SUB_TOPIC = '/ur5/zed_node/left/image_rect_color'

  ROS topic from where the script get the ZED image.
- float MIN_LEVEL_CONFIDENCE = 0.3

  Level of confidence of the Yolo model to keep the assigned labels.
- str NODE_NAME = 'vision'

  ROS nodes name.
- str POINTCLOUD_SUB_TOPIC = '/ur5/zed_node/point_cloud/cloud_registered'

  ROS topic from where the script get the pointcloud.
- str PUB_TOPIC = 'lego_position'

  ROS topic where to publish positions.
- RY

  Rotation matrix of the ZED camera.
- list TABLE = [[825,549], [1301,552], [1570,913], [658, 921]]

  Area where the vision detect blocks.
- str ZED_IMG_CROPPED_PATH = os.getcwd() + '/cropped_zed_image.png'

  Path where the cropped image is saved (a mask is applied to the photo to reduce confusion)
- str ZED_IMG_PATH = os.getcwd() + '/zed_image.png'

  Path where the original ZED image is saved.
- ZED_POSITION = np.array([-0.9 ,0.24 ,-0.35])

  Zed position regarding to the base link frame.

5.17.1 Detailed Description

Parameters used in the vision scripts.

5.17.2 Author(s)

- Created by Marco Morandin

5.18 vision/vision.py File Reference

Detect blocks from a photos caming from the ZED camera and find position of them.

Namespaces

- namespace vision

Functions

- build_pose (block)

  Find three useful points for compute position and orientation of a block from all the points contained in it, moreover compute the coordinates of the center, find orientation, convert it to quaternions and in the end create the Pose object.
- find_center (y_max_point, y_min_point)

  Find the coordinates of the center of a block.
- find_orientation (y_max_point, x_min_point)

  Find the yaw of a block in Euler angles.
- pointCloudCallBack ()

  This function waits a message from a pointcloud and then reads the points from it and compute the position and the orientation of the blocks in the Gazebo scenario.
- receive_image (data)

  Recive image from ros and save it.

Variables

- image_sub = rospy.Subscriber(IMAGE_SUB_TOPIC, Image, callback=receive_image, queue_size = 1)
- loop_rate = rospy.Rate(1.)
- pos_pub = rospy.Publisher(PUB_TOPIC, legoGroup, queue_size = 11)

5.18.1 Detailed Description

Detect blocks from a photos caming from the ZED camera and find position of them.

5.18.2 Author(s)

- Created by Marco Morandin

Index