

Fundamentals of Robotics Project

1.0

Generated by Doxygen 1.10.0

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Namespace Documentation	5
3.1 dataset2Yolo Namespace Reference	5
3.1.1 Function Documentation	5
3.1.1.1 create_annotations()	5
3.1.1.2 create_yaml_file()	5
3.1.2 Variable Documentation	6
3.1.2.1 ASSIGNS	6
3.1.2.2 CATEGORIES	6
3.1.2.3 PROJECT_DIR	6
3.2 detect_area Namespace Reference	6
3.2.1 Variable Documentation	6
3.2.1.1 detectArea	6
3.2.1.2 img	6
3.3 detect_blocks Namespace Reference	7
3.4 params Namespace Reference	7
3.4.1 Variable Documentation	7
3.4.1.1 BASE_LINK_POSITION	7
3.4.1.2 BLOCK_COORD_Z	8
3.4.1.3 CATEGORIES	8
3.4.1.4 IMAGE_SUB_TOPIC	8
3.4.1.5 MIN_LEVEL_CONFIDENCE	8
3.4.1.6 NODE_NAME	8
3.4.1.7 POINTCLOUD_SUB_TOPIC	8
3.4.1.8 PUB_TOPIC	8
3.4.1.9 RY	9
3.4.1.10 TABLE	9
3.4.1.11 ZED_IMG_CROPPED_PATH	9
3.4.1.12 ZED_IMG_PATH	9
3.4.1.13 ZED_POSITION	9
3.5 scale_legos Namespace Reference	9
3.5.1 Function Documentation	10
3.5.1.1 scale_legos()	10
3.5.2 Variable Documentation	10
3.5.2.1 MODELS_PATH	10
3.5.2.2 NEW_SCALE_FACTOR	10
3.5.2.3 OLD_SCALE_FACTOR	10
3.5.2.4 SPAWN_PATH	10

3.6 spawnLego Namespace Reference	11
3.6.1 Function Documentation	11
3.6.1.1 changeModelColor()	11
3.6.1.2 check_sovrapposizioni()	12
3.6.1.3 del_model()	12
3.6.1.4 randNum()	12
3.6.1.5 random_position()	13
3.6.1.6 spawn_model()	13
3.6.2 Variable Documentation	13
3.6.2.1 colorList	13
3.6.2.2 models	13
3.6.2.3 models_path	14
3.6.2.4 toll	14
3.7 training Namespace Reference	14
3.7.1 Variable Documentation	14
3.7.1.1 metrics	14
3.7.1.2 model	14
3.7.1.3 PROJECT_DIR	14
3.7.1.4 results	14
3.7.1.5 split	15
3.8 vision Namespace Reference	15
3.8.1 Function Documentation	15
3.8.1.1 build_pose()	15
3.8.1.2 find_center()	16
3.8.1.3 find_orientation()	16
3.8.1.4 pointCloudCallBack()	16
3.8.1.5 receive_image()	16
3.8.2 Variable Documentation	17
3.8.2.1 image_sub	17
3.8.2.2 loop_rate	17
3.8.2.3 pos_pub	17
3.9 yolo-k-fold-splitter Namespace Reference	17
3.9.1 Variable Documentation	18
3.9.1.1 classes	18
3.9.1.2 cls_idx	18
3.9.1.3 dataset_path	18
3.9.1.4 dataset_yaml	18
3.9.1.5 ds_yamls	18
3.9.1.6 encoding	18
3.9.1.7 exist_ok	18
3.9.1.8 fold_lbl_distrib	18
3.9.1.9 folds	18

3.9.1.10 folds_df	18
3.9.1.11 images	19
3.9.1.12 img_to_path	19
3.9.1.13 indx	19
3.9.1.14 kf	19
3.9.1.15 kfolds	19
3.9.1.16 ksplit	19
3.9.1.17 labels	19
3.9.1.18 labels_df	19
3.9.1.19 lbl_counter	19
3.9.1.20 lbl_to_path	19
3.9.1.21 lines	20
3.9.1.22 parents	20
3.9.1.23 ratio	20
3.9.1.24 save_path	20
3.9.1.25 split_dir	20
3.9.1.26 start	20
3.9.1.27 supported_extensions	20
3.9.1.28 train_totals	20
3.9.1.29 True	20
3.9.1.30 val_totals	20
3.9.1.31 yaml_file	20
4 Class Documentation	21
4.1 Block Class Reference	21
4.1.1 Detailed Description	21
4.1.2 Constructor & Destructor Documentation	21
4.1.2.1 __init__()	21
4.1.3 Member Data Documentation	22
4.1.3.1 category	22
4.1.3.2 category_id	22
4.1.3.3 confidence	22
4.1.3.4 image	22
4.1.3.5 points	22
4.1.3.6 points_count	22
4.1.3.7 xyxy	22
4.2 DetectArea Class Reference	22
4.2.1 Detailed Description	23
4.2.2 Constructor & Destructor Documentation	23
4.2.2.1 __init__()	23
4.2.3 Member Function Documentation	23
4.2.3.1 create_mask()	23

4.2.4 Member Data Documentation	23
4.2.4.1 input_img	23
4.2.4.2 output_img_path	23
4.3 DetectBlocks Class Reference	24
4.3.1 Detailed Description	24
4.3.2 Constructor & Destructor Documentation	24
4.3.2.1 __init__()	24
4.3.3 Member Function Documentation	24
4.3.3.1 find_blocks()	24
4.3.4 Member Data Documentation	25
4.3.4.1 blocks	25
4.3.4.2 model	25
4.3.4.3 zed_img	25
4.3.4.4 zed_img_cropped	25
4.4 frame Struct Reference	25
4.4.1 Detailed Description	25
4.4.2 Member Data Documentation	25
4.4.2.1 rot	25
4.4.2.2 xyz	25
5 File Documentation	27
5.1 planner_pkg/include/planner_pkg/kinematics.h File Reference	27
5.1.1 Detailed Description	28
5.1.2 Function Documentation	28
5.1.2.1 direct_kinematics()	28
5.1.2.2 eul2rotm()	28
5.1.2.3 inverse_kinematics()	29
5.1.2.4 jacobian()	29
5.1.2.5 rotm2eul()	29
5.1.2.6 t10f()	30
5.1.2.7 t21f()	30
5.1.2.8 t32f()	30
5.1.2.9 t43f()	31
5.1.2.10 t54f()	31
5.1.2.11 t65f()	31
5.2 kinematics.h	32
5.3 planner_pkg/include/planner_pkg/movement.h File Reference	32
5.3.1 Detailed Description	33
5.3.2 Function Documentation	33
5.3.2.1 invDiffKinematicControlCompleteQuaternion()	33
5.3.2.2 invDiffKinematicControlSimCompleteQuaternion()	34
5.3.2.3 operator*()	34

5.3.2.4 pd()	35
5.3.2.5 qd()	35
5.3.3 Variable Documentation	36
5.3.3.1 maxT	36
5.4 movement.h	36
5.5 planner_pkg/include/planner_pkg/planner.h File Reference	36
5.5.1 Detailed Description	38
5.5.2 Typedef Documentation	38
5.5.2.1 GripperState	38
5.5.3 Function Documentation	38
5.5.3.1 close_gripper()	38
5.5.3.2 get_gripper_states()	38
5.5.3.3 get_joint_states()	39
5.5.3.4 listen_lego_detection()	39
5.5.3.5 move_to_home()	39
5.5.3.6 open_gripper()	39
5.5.3.7 quat2eul()	40
5.5.3.8 set_joint_states()	40
5.5.3.9 waitJoints()	40
5.5.3.10 waitSec()	41
5.5.3.11 X1_Y1_Z2()	41
5.5.3.12 X1_Y2_Z1()	41
5.5.3.13 X1_Y2_Z2()	41
5.5.3.14 X1_Y2_Z2_CHAMFER()	42
5.5.3.15 X1_Y2_Z2_TWINFILLET()	42
5.5.3.16 X1_Y3_Z2()	42
5.5.3.17 X1_Y3_Z2_FILLET()	42
5.5.3.18 X1_Y4_Z1()	42
5.5.3.19 X1_Y4_Z2()	42
5.5.3.20 X2_Y2_Z2()	42
5.5.3.21 X2_Y2_Z2_FILLET()	43
5.5.4 Variable Documentation	43
5.5.4.1 actual_gripper	43
5.5.4.2 jointState_msg_robot	43
5.5.4.3 loop_frequency	43
5.5.4.4 models_map	43
5.5.4.5 pub_joint_states	43
5.5.4.6 timeStep	44
5.6 planner.h	44
5.7 planner_pkg/src/kinematics.cpp File Reference	45
5.7.1 Detailed Description	45
5.7.2 Function Documentation	46

5.7.2.1 direct_kinematics()	46
5.7.2.2 eul2rotm()	46
5.7.2.3 inverse_kinematics()	46
5.7.2.4 jacobian()	48
5.7.2.5 rotm2eul()	48
5.7.2.6 t10f()	48
5.7.2.7 t21f()	49
5.7.2.8 t32f()	49
5.7.2.9 t43f()	49
5.7.2.10 t54f()	50
5.7.2.11 t65f()	50
5.8 planner_pkg/src/movement.cpp File Reference	50
5.8.1 Detailed Description	51
5.8.2 Function Documentation	51
5.8.2.1 invDiffKinematicControlCompleteQuaternion()	51
5.8.2.2 invDiffKinematicControlSimCompleteQuaternion()	52
5.8.2.3 pd()	52
5.8.2.4 qd()	53
5.9 planner_pkg/src/planner.cpp File Reference	53
5.9.1 Detailed Description	54
5.9.2 Function Documentation	54
5.9.2.1 close_gripper()	54
5.9.2.2 get_gripper_states()	55
5.9.2.3 get_joint_states()	55
5.9.2.4 listen_lego_detection()	55
5.9.2.5 main()	56
5.9.2.6 move_to_home()	56
5.9.2.7 open_gripper()	56
5.9.2.8 quat2eul()	56
5.9.2.9 set_joint_states()	57
5.9.2.10 waitJoints()	57
5.9.2.11 waitSec()	57
5.10 spawnLego/spawnLego.py File Reference	58
5.10.1 Detailed Description	58
5.10.2 Author(s)	58
5.11 utils/dataset_creation/dataset2Yolo.py File Reference	59
5.11.1 Detailed Description	59
5.11.2 Author(s)	59
5.12 utils/dataset_creation/yolo-k-fold-splitter.py File Reference	59
5.13 utils/scale_legos.py File Reference	60
5.13.1 Detailed Description	61
5.13.2 Author(s)	61

5.14	utils/training/training.py File Reference	61
5.15	vision/detect_area.py File Reference	61
5.15.1	Detailed Description	61
5.15.2	Author(s)	62
5.16	vision/detect_blocks.py File Reference	62
5.16.1	Detailed Description	62
5.16.2	Author(s)	62
5.17	vision/params.py File Reference	62
5.17.1	Detailed Description	63
5.17.2	Author(s)	63
5.18	vision/vision.py File Reference	63
5.18.1	Detailed Description	64
5.18.2	Author(s)	64
	Index	65

Chapter 1

Class Index

1.1 Class List

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

Block	Class that rapresent a block	21
DetectArea	Class that detect the area in wich detect blocks	22
DetectBlocks	Class that detect blocks	24
frame	Structure to store the position and rotation of the end effector	25

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

planner_pkg/include/planner_pkg/kinematics.h	27
planner_pkg/include/planner_pkg/movement.h	32
planner_pkg/include/planner_pkg/planner.h	36
planner_pkg/src/kinematics.cpp Functions in this file are used to calculate transformation matrices, direct and inverse kinematics, jacobian matrices and trasformation between Euler angles and rotation matrices	45
planner_pkg/src/movement.cpp Functions in this file are used to calculate the trajectory based on quaternions and velocities of the joints	50
planner_pkg/src/planner.cpp Main function and planning of the movement based on the messages received from the vision; also getters and setters for joints are present	53
spawnLego/spawnLego.py Spawn lego in random position and orientation	58
utils/scale_legos.py	60
utils/dataset_creation/dataset2Yolo.py Convert the given dataset to a Yolo format dataset	59
utils/dataset_creation/yolo-k-fold-splitter.py	59
utils/training/training.py	61
vision/detect_area.py Detect the area and crop the ZED image from where the model will recognize blocks	61
vision/detect_blocks.py Detect blocks in the image using YOLOv8 model trained in a custom dataset	62
vision/params.py Parameters used in the vision scripts	62
vision/vision.py Detect blocks from a photos coming from the ZED camera and find position of them	63

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

<i>(xml)</i>	xml of model
<i>(string)</i>	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

<i>pos</i>	(array): positions of other legos
<i>lego</i>	(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

<i>model</i>	(string): name of the model to be deleted
--------------	---

Returns

bool : True if the deletion succeeded

3.6.1.4 randNum()

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

Generates a random number.

Parameters

<i>min</i>	(int): the minimum number
<i>max</i>	(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

<i>model</i>	(string): the name of the lego model
<i>pos</i>	(struct): all the parameters for position and orientation of the lego
<i>name</i>	(string, optional): the name of the model. Defaults to None.
<i>ref_frame</i>	(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:

```
00001 = model.train(
00002     data=PROJECT_DIR + '/split_5_5-Fold_Cross-val/split_' + str(i+1) + '/split_' + str(i+1) +
00003     '_dataset.yaml',
00004     epochs=20,
00005     imgsz=1024,
00006     device=[0,1],
00007     save = True,
00008     workers = 8,
00009     batch = 10,
00010     project = 'robotica',
00011 )
```


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

<i>block</i>	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

<i>y_max_point</i>	list The coordinates of the point of the block with the biggest y
<i>y_min_point</i>	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

<i>y_max_point</i>	list The coordinates of the point of the block with the biggest y
<i>x_min_point</i>	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_distrib` = `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

<i>The</i>	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

<i>th</i>	Joint angles
-----------	--------------

Returns

frame

5.1.2.2 `eul2rotm()`

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

From euler angles to rotation matrix.

Parameters

<i>rpy</i>	Euler angles
------------	--------------

Returns

Matrix3f

5.1.2.3 inverse_kinematics()

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

Compute the inverse kinematics.

Parameters

<i>frame</i>	Current frame of the end effector
--------------	-----------------------------------

Returns

MatrixXf

5.1.2.4 jacobian()

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

Calculate the jacobian matrix.

Parameters

<i>q</i>	Joint angles
----------	--------------

Returns

MatrixXf

5.1.2.5 rotm2eul()

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

From rotation matrix to euler angles.

Parameters

<i>R</i>	Rotation matrix
----------	-----------------

Returns

Vector3f

5.1.2.6 t10f()

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

Functions prototypes.

Functions prototypes.

Parameters

<i>th1</i>	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

<i>th2</i>	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

<i>th3</i>	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

<i>th4</i>	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

<i>th5</i>	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

<i>th6</i>	Angle of the sixth joint
------------	--------------------------

Returns

Matrix4f

5.2 kinematics.h

[Go to the documentation of this file.](#)

```

00001
00011 #ifndef __KINEMATICS_H__
00012 #define __KINEMATICS_H__
00013
00014
00015 #include <Eigen/Dense>
00016 #include <Eigen/Geometry>
00017 #include <cmath>
00018
00019
00020 using namespace std;
00021 using namespace Eigen;
00022
00023
00027 struct frame {
00028     Vector3f xyz;
00029     Matrix3f rot;
00030 };
00031
00032
00037 Matrix4f t10f(float th1);
00038 Matrix4f t21f(float th2);
00039 Matrix4f t32f(float th3);
00040 Matrix4f t43f(float th4);
00041 Matrix4f t54f(float th5);
00042 Matrix4f t65f(float th6);
00043
00044 frame direct_kinematics(VectorXf th);
00045 MatrixXf inverse_kinematics(frame &frame);
00046 MatrixXf jacobian(VectorXf q);
00047
00048 Matrix3f eul2rotm(Vector3f rpy);
00049 Vector3f rotm2eul(Matrix3f R);
00050
00051
00052 #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

Copyright (c) 2024

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

<i>q</i>	The current joint config
<i>xe</i>	The current end-effector position
<i>xd</i>	The desired end-effector position
<i>vd</i>	The desired end-effector linear velocity
<i>omegad</i>	The desired end-effector angular velocity
<i>qe</i>	The current end-effector rotation in quaternion
<i>qd</i>	The desired end-effector rotation in quaternion
<i>Kd</i>	The position gain
<i>Kq</i>	The orientation gain
<i>f</i>	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

<i>xef</i>	Desired end-effector position
<i>phief</i>	Desired end-effector orientation
<i>dt</i>	Time step
<i>jstates</i>	Actual state of the joints
<i>send_j</i>	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

<i>num</i>	Scalar that multiplies
<i>q</i>	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

<i>t</i>	The current time
<i>xef</i>	The desired end-effector position
<i>xe0</i>	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

<i>tb</i>	The current time
<i>q0</i>	The start end-effector quaternion
<i>qf</i>	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.](#)

```
00001
00011 #ifndef __MOVEMENT_H__
00012 #define __MOVEMENT_H__
00013
00014
00015 #include "kinematics.h"
00016
00017 #include "ros/ros.h"
00018 #include <std_msgs/Float64MultiArray.h>
00019 #include <sensor_msgs/JointState.h>
00020 #include <Eigen/Dense>
00021 #include <Eigen/Geometry>
00022
00023 #include <iostream>
00024
00025
00026 using namespace std;
00027 using namespace Eigen;
00028
00029
00031 inline float maxT = 6;
00032
00033
00045 inline Quaternionf operator *(float num, const Quaternionf& q) {
00046     return Quaternionf(q.x() * num, q.y() * num, q.z() * num, q.w() * num);
00047 }
00048
00049 Vector3f pd(float t, Vector3f xef, Vector3f xe0);
00050 Quaternionf qd(float tb, Quaternionf q0, Quaternionf qf);
00051 VectorXf invDiffKinematicControlCompleteQuaternion(VectorXf q, Vector3f xe, Vector3f xd, Vector3f vd,
    Vector3f omegad, Quaternionf qe, Quaternionf qd, Matrix3f Kp, Matrix3f Kq, int f);
00052 void invDiffKinematicControlSimCompleteQuaternion(Vector3f xef, Vector3f phief, double dt, VectorXf
    jstates, void (*send_j)(VectorXf));
00053
00054
00055 #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

<i>amp</i>	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

<i>rate</i>	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

<i>amp</i>	Required gripper length
------------	-------------------------

Returns

void

5.5.3.7 quat2eul()

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

Convert from Quaternion to Euler Angles.

Parameters

<i>q</i>	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

<i>joint_pos</i>	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

<i>waitRot</i>	wait also for z rotation to be aligned
<i>xef</i>	final positon
<i>phief</i>	final rotation

Returns

void

5.5.3.10 waitSec()

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

Wait for the specified time.

Parameters

<i>t</i>	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.](#)

```
00001
00011 #ifndef __PLANNER_H__
00012 #define __PLANNER_H__
00013
00014
00015 #include "movement.h"
00016 #include <planner_pkg/legoDetection.h>
00017 #include <planner_pkg/legoGroup.h>
00018
00019 #include <ros/ros.h>
00020 #include <Eigen/Dense>
00021 #include <Eigen/Geometry>
00022
00023 #include <iostream>
00024 #include <vector>
00025 #include <cmath>
00026
00027
00028 using namespace std;
00029 using namespace Eigen;
00030
00031
00035 typedef Matrix<float, 2, 1> GripperState;
00036
00038 float loop_frequency = 1000.0;
00040 double timeStep = 0.001;
00042 GripperState actual_gripper;
00043
00044
00046 ros::Publisher pub_joint_states;
00048 std_msgs::Float64MultiArray jointState_msg_robot;
00049
00050
00055 void listen_lego_detection(ros::Rate rate);
00056 void move_to_home();
00057 Vector3f quat2eul(Quaternionf q);
00058 VectorXf get_joint_states();
00059 void set_joint_states(VectorXf q);
00060 GripperState get_gripper_states();
00061 void open_gripper(float amp);
00062 void close_gripper(float amp);
00063 void waitSec(float t);
00064 void waitJoints(bool waitRot, Vector3f xef, Vector3f phief);
00065
00066
00071 Vector3f X1_Y1_Z2(0.92, 0.27, 0.88);
00072 Vector3f X1_Y2_Z1(0.77, 0.27, 0.88);
00073 Vector3f X1_Y2_Z2(0.62, 0.27, 0.88);
00074
00075 Vector3f X1_Y2_Z2_CHAMFER(0.92, 0.42, 0.88);
00076 Vector3f X1_Y2_Z2_TWINFILLET(0.77, 0.42, 0.88);
00077 Vector3f X2_Y2_Z2(0.62, 0.42, 0.89);
00078
00079 Vector3f X2_Y2_Z2_FILLET(0.92, 0.56, 0.89);
00080 Vector3f X1_Y3_Z2(0.77, 0.56, 0.88);
00081 Vector3f X1_Y3_Z2_FILLET(0.62, 0.56, 0.88);
00082
00083 Vector3f X1_Y4_Z1(0.87, 0.72, 0.88);
00084 Vector3f X1_Y4_Z2(0.65, 0.72, 0.88);
00085
00086 map<std::string, Vector3f> models_map{
00087     {"X1-Y1-Z2", X1_Y1_Z2},
00088     {"X1-Y2-Z1", X1_Y2_Z1},
00089     {"X1-Y2-Z2", X1_Y2_Z2},
00090     {"X1-Y2-Z2-CHAMFER", X1_Y2_Z2_CHAMFER},
00091     {"X1-Y2-Z2-TWINFILLET", X1_Y2_Z2_TWINFILLET},
00092     {"X2-Y2-Z2", X2_Y2_Z2},
00093     {"X2-Y2-Z2-FILLET", X2_Y2_Z2_FILLET},
00094     {"X1-Y3-Z2", X1_Y3_Z2},
```

```

00095     {"X1-Y3-Z2-FILLET", X1_Y3_Z2_FILLET},
00096     {"X1-Y4-Z1", X1_Y4_Z1},
00097     {"X1-Y4-Z2", X1_Y4_Z2}
00098 };
00099
00100
00101 #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

Copyright (c) 2024

5.7.2 Function Documentation

5.7.2.1 `direct_kinematics()`

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

Compute the direct kinematics.

Parameters

<i>th</i>	Joint angles
-----------	--------------

Returns

frame

5.7.2.2 `eul2rotm()`

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

From euler angles to rotation matrix.

Parameters

<i>rpy</i>	Euler angles
------------	--------------

Returns

Matrix3f

5.7.2.3 `inverse_kinematics()`

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


Compute the inverse kinematics.

Parameters

<i>frame</i>	Current frame of the end effector
--------------	-----------------------------------

Returns

MatrixXf

5.7.2.4 jacobian()

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

Calculate the jacobian matrix.

Parameters

<i>q</i>	Joint angles
----------	--------------

Returns

MatrixXf

5.7.2.5 rotm2eul()

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

From rotation matrix to euler angles.

Parameters

<i>R</i>	Rotation matrix
----------	-----------------

Returns

Vector3f

5.7.2.6 t10f()

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

Create the transformation matrix for the first joint.

Functions prototypes.

Parameters

<i>th1</i>	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

<i>th2</i>	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

<i>th3</i>	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

<i>th4</i>	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

<i>th5</i>	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

<i>th6</i>	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

Copyright (c) 2024

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

<i>q</i>	The current joint config
<i>xe</i>	The current end-effector position
<i>xd</i>	The desired end-effector position
<i>vd</i>	The desired end-effector linear velocity
<i>omegad</i>	The desired end-effector angular velocity
<i>qe</i>	The current end-effector rotation in quaternion
<i>qd</i>	The desired end-effector rotation in quaternion
<i>Kd</i>	The position gain
<i>Kq</i>	The orientation gain
<i>f</i>	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

<i>xef</i>	Desired end-effector position
<i>phief</i>	Desired end-effector orientation
<i>dt</i>	Time step
<i>jstates</i>	Actual state of the joints
<i>send_j</i>	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

<i>t</i>	The current time
<i>xef</i>	The desired end-effector position
<i>xe0</i>	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

<i>tb</i>	The current time
<i>q0</i>	The start end-effector quaternion
<i>qf</i>	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

<i>amp</i>	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

<i>rate</i>	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

<i>amp</i>	Required gripper length
------------	-------------------------

Returns

void

5.9.2.8 quat2eul()

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

Convert from Quaternion to Euler Angles.

Parameters

<i>q</i>	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

<i>joint_pos</i>	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

<i>waitRot</i>	wait also for z rotation to be aligned
<i>xef</i>	final positon
<i>phief</i>	final rotation

Returns

void

5.9.2.11 waitSec()

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

Wait for the specified time.

Parameters

<i>t</i>	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 coming 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 coming from the ZED camera and find position of them.

5.18.2 Author(s)

- Created by Marco Morandin

Index

- `__init__`
 - Block, [21](#)
 - DetectArea, [23](#)
 - DetectBlocks, [24](#)
- `actual_gripper`
 - planner.h, [43](#)
- `ASSIGNS`
 - dataset2Yolo, [6](#)
- `BASE_LINK_POSITION`
 - params, [7](#)
- `Block`, [21](#)
 - `__init__`, [21](#)
 - category, [22](#)
 - category_id, [22](#)
 - confidence, [22](#)
 - image, [22](#)
 - points, [22](#)
 - points_count, [22](#)
 - xyxy, [22](#)
- `BLOCK_COORD_Z`
 - params, [7](#)
- `blocks`
 - DetectBlocks, [25](#)
- `build_pose`
 - vision, [15](#)
- `CATEGORIES`
 - dataset2Yolo, [6](#)
 - params, [8](#)
- `category`
 - Block, [22](#)
- `category_id`
 - Block, [22](#)
- `changeModelColor`
 - spawnLego, [11](#)
- `check_sovrapposizioni`
 - spawnLego, [11](#)
- `classes`
 - yolo-k-fold-splitter, [18](#)
- `close_gripper`
 - planner.cpp, [54](#)
 - planner.h, [38](#)
- `cls_idx`
 - yolo-k-fold-splitter, [18](#)
- `colorList`
 - spawnLego, [13](#)
- `confidence`
 - Block, [22](#)
- `create_annotations`
 - dataset2Yolo, [5](#)
- `create_mask`
 - DetectArea, [23](#)
- `create_yaml_file`
 - dataset2Yolo, [5](#)
- `dataset2Yolo`, [5](#)
 - ASSIGNS, [6](#)
 - CATEGORIES, [6](#)
 - create_annotations, [5](#)
 - create_yaml_file, [5](#)
 - PROJECT_DIR, [6](#)
- `dataset_path`
 - yolo-k-fold-splitter, [18](#)
- `dataset_yaml`
 - yolo-k-fold-splitter, [18](#)
- `del_model`
 - spawnLego, [12](#)
- `detect_area`, [6](#)
 - DetectArea, [6](#)
 - img, [6](#)
- `detect_blocks`, [7](#)
- `DetectArea`, [22](#)
 - `__init__`, [23](#)
 - create_mask, [23](#)
 - input_img, [23](#)
 - output_img_path, [23](#)
- `detectArea`
 - detect_area, [6](#)
- `DetectBlocks`, [24](#)
 - `__init__`, [24](#)
 - blocks, [25](#)
 - find_blocks, [24](#)
 - model, [25](#)
 - zed_img, [25](#)
 - zed_img_cropped, [25](#)
- `direct_kinematics`
 - kinematics.cpp, [46](#)
 - kinematics.h, [28](#)
- `ds_yamls`
 - yolo-k-fold-splitter, [18](#)
- `encoding`
 - yolo-k-fold-splitter, [18](#)
- `eul2rotm`
 - kinematics.cpp, [46](#)
 - kinematics.h, [28](#)
- `exist_ok`
 - yolo-k-fold-splitter, [18](#)

- find_blocks
 - DetectBlocks, 24
- find_center
 - vision, 15
- find_orientation
 - vision, 16
- fold_lbl_distrib
 - yolo-k-fold-splitter, 18
- folds
 - yolo-k-fold-splitter, 18
- folds_df
 - yolo-k-fold-splitter, 18
- frame, 25
 - rot, 25
 - xyz, 25
- get_gripper_states
 - planner.cpp, 55
 - planner.h, 38
- get_joint_states
 - planner.cpp, 55
 - planner.h, 39
- GripperState
 - planner.h, 38
- image
 - Block, 22
- image_sub
 - vision, 17
- IMAGE_SUB_TOPIC
 - params, 8
- images
 - yolo-k-fold-splitter, 18
- img
 - detect_area, 6
- img_to_path
 - yolo-k-fold-splitter, 19
- indx
 - yolo-k-fold-splitter, 19
- input_img
 - DetectArea, 23
- invDiffKinematicControlCompleteQuaternion
 - movement.cpp, 51
 - movement.h, 33
- invDiffKinematicControlSimCompleteQuaternion
 - movement.cpp, 52
 - movement.h, 34
- inverse_kinematics
 - kinematics.cpp, 46
 - kinematics.h, 29
- jacobian
 - kinematics.cpp, 48
 - kinematics.h, 29
- jointState_msg_robot
 - planner.h, 43
- kf
 - yolo-k-fold-splitter, 19
- kfolds
 - yolo-k-fold-splitter, 19
- kinematics.cpp
 - direct_kinematics, 46
 - eul2rotm, 46
 - inverse_kinematics, 46
 - jacobian, 48
 - rotm2eul, 48
 - t10f, 48
 - t21f, 49
 - t32f, 49
 - t43f, 49
 - t54f, 50
 - t65f, 50
- kinematics.h
 - direct_kinematics, 28
 - eul2rotm, 28
 - inverse_kinematics, 29
 - jacobian, 29
 - rotm2eul, 29
 - t10f, 30
 - t21f, 30
 - t32f, 30
 - t43f, 31
 - t54f, 31
 - t65f, 31
- ksplit
 - yolo-k-fold-splitter, 19
- labels
 - yolo-k-fold-splitter, 19
- labels_df
 - yolo-k-fold-splitter, 19
- lbl_counter
 - yolo-k-fold-splitter, 19
- lbl_to_path
 - yolo-k-fold-splitter, 19
- lines
 - yolo-k-fold-splitter, 19
- listen_lego_detection
 - planner.cpp, 55
 - planner.h, 39
- loop_frequency
 - planner.h, 43
- loop_rate
 - vision, 17
- main
 - planner.cpp, 55
- maxT
 - movement.h, 36
- metrics
 - training, 14
- MIN_LEVEL_CONFIDENCE
 - params, 8
- model
 - DetectBlocks, 25
 - training, 14
- models

- spawnLego, [13](#)
- models_map
 - planner.h, [43](#)
- MODELS_PATH
 - scale_legos, [10](#)
- models_path
 - spawnLego, [13](#)
- move_to_home
 - planner.cpp, [56](#)
 - planner.h, [39](#)
- movement.cpp
 - invDiffKinematicControlCompleteQuaternion, [51](#)
 - invDiffKinematicControlSimCompleteQuaternion, [52](#)
 - pd, [52](#)
 - qd, [53](#)
- movement.h
 - invDiffKinematicControlCompleteQuaternion, [33](#)
 - invDiffKinematicControlSimCompleteQuaternion, [34](#)
 - maxT, [36](#)
 - operator*, [34](#)
 - pd, [35](#)
 - qd, [35](#)
- NEW_SCALE_FACTOR
 - scale_legos, [10](#)
- NODE_NAME
 - params, [8](#)
- OLD_SCALE_FACTOR
 - scale_legos, [10](#)
- open_gripper
 - planner.cpp, [56](#)
 - planner.h, [39](#)
- operator*
 - movement.h, [34](#)
- output_img_path
 - DetectArea, [23](#)
- params, [7](#)
 - BASE_LINK_POSITION, [7](#)
 - BLOCK_COORD_Z, [7](#)
 - CATEGORIES, [8](#)
 - IMAGE_SUB_TOPIC, [8](#)
 - MIN_LEVEL_CONFIDENCE, [8](#)
 - NODE_NAME, [8](#)
 - POINTCLOUD_SUB_TOPIC, [8](#)
 - PUB_TOPIC, [8](#)
 - RY, [8](#)
 - TABLE, [9](#)
 - ZED_IMG_CROPPED_PATH, [9](#)
 - ZED_IMG_PATH, [9](#)
 - ZED_POSITION, [9](#)
- parents
 - yolo-k-fold-splitter, [20](#)
- pd
 - movement.cpp, [52](#)
 - movement.h, [35](#)
- planner.cpp
 - close_gripper, [54](#)
 - get_gripper_states, [55](#)
 - get_joint_states, [55](#)
 - listen_lego_detection, [55](#)
 - main, [55](#)
 - move_to_home, [56](#)
 - open_gripper, [56](#)
 - quat2eul, [56](#)
 - set_joint_states, [56](#)
 - waitJoints, [57](#)
 - waitSec, [57](#)
- planner.h
 - actual_gripper, [43](#)
 - close_gripper, [38](#)
 - get_gripper_states, [38](#)
 - get_joint_states, [39](#)
 - GripperState, [38](#)
 - jointState_msg_robot, [43](#)
 - listen_lego_detection, [39](#)
 - loop_frequency, [43](#)
 - models_map, [43](#)
 - move_to_home, [39](#)
 - open_gripper, [39](#)
 - pub_joint_states, [43](#)
 - quat2eul, [40](#)
 - set_joint_states, [40](#)
 - timeStep, [43](#)
 - waitJoints, [40](#)
 - waitSec, [41](#)
 - X1_Y1_Z2, [41](#)
 - X1_Y2_Z1, [41](#)
 - X1_Y2_Z2, [41](#)
 - X1_Y2_Z2_CHAMFER, [41](#)
 - X1_Y2_Z2_TWINFILLET, [42](#)
 - X1_Y3_Z2, [42](#)
 - X1_Y3_Z2_FILLET, [42](#)
 - X1_Y4_Z1, [42](#)
 - X1_Y4_Z2, [42](#)
 - X2_Y2_Z2, [42](#)
 - X2_Y2_Z2_FILLET, [42](#)
- planner_pkg/include/planner_pkg/kinematics.h, [27](#), [32](#)
- planner_pkg/include/planner_pkg/movement.h, [32](#), [36](#)
- planner_pkg/include/planner_pkg/planner.h, [36](#), [44](#)
- planner_pkg/src/kinematics.cpp, [45](#)
- planner_pkg/src/movement.cpp, [50](#)
- planner_pkg/src/planner.cpp, [53](#)
- POINTCLOUD_SUB_TOPIC
 - params, [8](#)
- pointCloudCallback
 - vision, [16](#)
- points
 - Block, [22](#)
- points_count
 - Block, [22](#)
- pos_pub
 - vision, [17](#)
- PROJECT_DIR

- dataset2Yolo, 6
- training, 14
- pub_joint_states
 - planner.h, 43
- PUB_TOPIC
 - params, 8
- qd
 - movement.cpp, 53
 - movement.h, 35
- quat2eul
 - planner.cpp, 56
 - planner.h, 40
- randNum
 - spawnLego, 12
- random_position
 - spawnLego, 12
- ratio
 - yolo-k-fold-splitter, 20
- receive_image
 - vision, 16
- results
 - training, 14
- rot
 - frame, 25
- rotm2eul
 - kinematics.cpp, 48
 - kinematics.h, 29
- RY
 - params, 8
- save_path
 - yolo-k-fold-splitter, 20
- scale_legos, 9
 - MODELS_PATH, 10
 - NEW_SCALE_FACTOR, 10
 - OLD_SCALE_FACTOR, 10
 - scale_legos, 10
 - SPAWN_PATH, 10
- set_joint_states
 - planner.cpp, 56
 - planner.h, 40
- spawn_model
 - spawnLego, 13
- SPAWN_PATH
 - scale_legos, 10
- spawnLego, 11
 - changeModelColor, 11
 - check_sovrapposizioni, 11
 - colorList, 13
 - del_model, 12
 - models, 13
 - models_path, 13
 - randNum, 12
 - random_position, 12
 - spawn_model, 13
 - toll, 14
- spawnLego/spawnLego.py, 58
- split
 - training, 14
- split_dir
 - yolo-k-fold-splitter, 20
- start
 - yolo-k-fold-splitter, 20
- supported_extensions
 - yolo-k-fold-splitter, 20
- t10f
 - kinematics.cpp, 48
 - kinematics.h, 30
- t21f
 - kinematics.cpp, 49
 - kinematics.h, 30
- t32f
 - kinematics.cpp, 49
 - kinematics.h, 30
- t43f
 - kinematics.cpp, 49
 - kinematics.h, 31
- t54f
 - kinematics.cpp, 50
 - kinematics.h, 31
- t65f
 - kinematics.cpp, 50
 - kinematics.h, 31
- TABLE
 - params, 9
- timeStep
 - planner.h, 43
- toll
 - spawnLego, 14
- train_totals
 - yolo-k-fold-splitter, 20
- training, 14
 - metrics, 14
 - model, 14
 - PROJECT_DIR, 14
 - results, 14
 - split, 14
- True
 - yolo-k-fold-splitter, 20
- utils/dataset_creation/dataset2Yolo.py, 59
- utils/dataset_creation/yolo-k-fold-splitter.py, 59
- utils/scale_legos.py, 60
- utils/training/training.py, 61
- val_totals
 - yolo-k-fold-splitter, 20
- vision, 15
 - build_pose, 15
 - find_center, 15
 - find_orientation, 16
 - image_sub, 17
 - loop_rate, 17
 - pointCloudCallBack, 16
 - pos_pub, 17

- receive_image, 16
- vision/detect_area.py, 61
- vision/detect_blocks.py, 62
- vision/params.py, 62
- vision/vision.py, 63
- waitJoints
 - planner.cpp, 57
 - planner.h, 40
- waitSec
 - planner.cpp, 57
 - planner.h, 41
- X1_Y1_Z2
 - planner.h, 41
- X1_Y2_Z1
 - planner.h, 41
- X1_Y2_Z2
 - planner.h, 41
- X1_Y2_Z2_CHAMFER
 - planner.h, 41
- X1_Y2_Z2_TWINFILLET
 - planner.h, 42
- X1_Y3_Z2
 - planner.h, 42
- X1_Y3_Z2_FILLET
 - planner.h, 42
- X1_Y4_Z1
 - planner.h, 42
- X1_Y4_Z2
 - planner.h, 42
- X2_Y2_Z2
 - planner.h, 42
- X2_Y2_Z2_FILLET
 - planner.h, 42
- xyxy
 - Block, 22
- xyz
 - frame, 25
- yaml_file
 - yolo-k-fold-splitter, 20
- yolo-k-fold-splitter, 17
 - classes, 18
 - cls_idx, 18
 - dataset_path, 18
 - dataset_yaml, 18
 - ds_yamls, 18
 - encoding, 18
 - exist_ok, 18
 - fold_lbl_distrb, 18
 - fold, 18
 - fold_df, 18
 - images, 18
 - img_to_path, 19
 - indx, 19
 - kf, 19
 - kfolds, 19
 - ksplit, 19
 - labels, 19
 - labels_df, 19
 - lbl_counter, 19
 - lbl_to_path, 19
 - lines, 19
 - parents, 20
 - ratio, 20
 - save_path, 20
 - split_dir, 20
 - start, 20
 - supported_extensions, 20
 - train_totals, 20
 - True, 20
 - val_totals, 20
 - yaml_file, 20
- zed_img
 - DetectBlocks, 25
- zed_img_cropped
 - DetectBlocks, 25
- ZED_IMG_CROPPED_PATH
 - params, 9
- ZED_IMG_PATH
 - params, 9
- ZED_POSITION
 - params, 9