

ROS-Dual-CAM

Generated by Doxygen 1.9.4



<b>1 ROS-Dual-CAM</b>	<b>1</b>
<b>2 Namespace Index</b>	<b>3</b>
2.1 Namespace List . . . . .	3
<b>3 Hierarchical Index</b>	<b>5</b>
3.1 Class Hierarchy . . . . .	5
<b>4 Class Index</b>	<b>7</b>
4.1 Class List . . . . .	7
<b>5 File Index</b>	<b>9</b>
5.1 File List . . . . .	9
<b>6 Namespace Documentation</b>	<b>11</b>
6.1 dual-cam Namespace Reference . . . . .	11
6.1.1 Variable Documentation . . . . .	11
6.1.1.1 bridge . . . . .	11
6.1.1.2 cam1 . . . . .	11
6.1.1.3 cam2 . . . . .	12
6.1.1.4 fps . . . . .	12
6.1.1.5 frame1 . . . . .	12
6.1.1.6 frame2 . . . . .	12
6.1.1.7 pub_image . . . . .	12
6.1.1.8 pub_image2 . . . . .	12
6.1.1.9 ret1 . . . . .	12
6.1.1.10 ret2 . . . . .	13
6.1.1.11 start . . . . .	13
6.2 test_local Namespace Reference . . . . .	13
6.2.1 Variable Documentation . . . . .	13
6.2.1.1 config . . . . .	13
6.2.1.2 fps . . . . .	13
6.2.1.3 key . . . . .	14
6.2.1.4 path . . . . .	14
6.2.1.5 real_ttb . . . . .	14
6.2.1.6 start . . . . .	14
6.2.1.7 test_local . . . . .	14
6.2.1.8 val . . . . .	14
6.3 test_real Namespace Reference . . . . .	14
6.3.1 Function Documentation . . . . .	15
6.3.1.1 getImage() . . . . .	16
6.3.1.2 updateLidar() . . . . .	16
6.3.2 Variable Documentation . . . . .	16
6.3.2.1 action . . . . .	16

6.3.2.2 action_high	16
6.3.2.3 action_low	16
6.3.2.4 action_size	16
6.3.2.5 actions	16
6.3.2.6 actor	17
6.3.2.7 algorithm	17
6.3.2.8 algorithms_sel	17
6.3.2.9 bridge	17
6.3.2.10 config	17
6.3.2.11 det	17
6.3.2.12 done	17
6.3.2.13 env	17
6.3.2.14 ep_start_time	18
6.3.2.15 episode_reward	18
6.3.2.16 episode_timing	18
6.3.2.17 episodes	18
6.3.2.18 font	18
6.3.2.19 fps	18
6.3.2.20 lidar_g	18
6.3.2.21 lidar_list	18
6.3.2.22 local_episode	19
6.3.2.23 max_steps	19
6.3.2.24 model	19
6.3.2.25 model_fn	19
6.3.2.26 num_steps	19
6.3.2.27 observation_space	19
6.3.2.28 outfile	19
6.3.2.29 path	19
6.3.2.30 path_results	20
6.3.2.31 pub_cmd_vel	20
6.3.2.32 quit	20
6.3.2.33 real_ttb	20
6.3.2.34 reward	20
6.3.2.35 scan	20
6.3.2.36 start	20
6.3.2.37 state	20
6.3.2.38 sub_image	21
6.3.2.39 sub_scan	21
6.3.2.40 TURTLE	21
6.3.2.41 val_state	21
6.3.2.42 values	21
6.3.2.43 vel_cmd	21

6.3.2.44 x	21
6.3.2.45 z	22
6.4 view_real_results Namespace Reference	22
6.4.1 Function Documentation	22
6.4.1.1 antispoke()	22
6.4.1.2 antispoke2()	23
6.4.1.3 open_test_data()	23
6.4.2 Variable Documentation	23
6.4.2.1 c	23
6.4.2.2 color	23
6.4.2.3 data	23
6.4.2.4 linestyle	23
6.4.2.5 linewidth	23
6.4.2.6 list_dir	24
6.4.2.7 path	24
6.4.2.8 rewards	24
6.4.2.9 size	24
6.4.2.10 STAGE	24
6.4.2.11 stage	24
6.4.2.12 threshold	24
6.4.2.13 threshold_x	24
6.4.2.14 threshold_y	25
6.4.2.15 times	25
6.4.2.16 x	25
6.4.2.17 y	25
<b>7 Class Documentation</b>	<b>27</b>
7.1 test_real.LinearDeepQNetwork Class Reference	27
7.1.1 Constructor & Destructor Documentation	27
7.1.1.1 __init__()	28
7.1.2 Member Function Documentation	28
7.1.2.1 forward()	28
7.1.3 Member Data Documentation	28
7.1.3.1 device	28
7.1.3.2 dropout	28
7.1.3.3 fc1	28
7.1.3.4 fc2	28
7.1.3.5 fc3	29
7.1.3.6 loss	29
7.1.3.7 optimizer	29
7.2 test_local.TestLocal Class Reference	29
7.2.1 Constructor & Destructor Documentation	29

---

7.2.1.1 <code>__init__()</code> . . . . .	29
7.2.2 Member Function Documentation . . . . .	30
7.2.2.1 <code>image_left_callback()</code> . . . . .	30
7.2.2.2 <code>image_right_callback()</code> . . . . .	30
7.2.2.3 <code>step()</code> . . . . .	30
7.2.3 Member Data Documentation . . . . .	30
7.2.3.1 <code>bridge</code> . . . . .	30
7.2.3.2 <code>defisheye1</code> . . . . .	30
7.2.3.3 <code>defisheye2</code> . . . . .	30
7.2.3.4 <code>image_left</code> . . . . .	31
7.2.3.5 <code>image_right</code> . . . . .	31
7.2.3.6 <code>stitcher</code> . . . . .	31
<b>8 File Documentation</b> . . . . .	<b>33</b>
8.1 <code>dual-cam.py</code> File Reference . . . . .	33
8.2 <code>README.md</code> File Reference . . . . .	33
8.3 <code>test_local.py</code> File Reference . . . . .	33
8.4 <code>test_real.py</code> File Reference . . . . .	34
8.5 <code>view_real_results.py</code> File Reference . . . . .	35
<b>Index</b> . . . . .	<b>37</b>

## Chapter 1

# ROS-Dual-CAM

Running stereo dual cam using OpenCV and ROS noetic.





## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">dual-cam</a>	11
<a href="#">test_local</a>	13
<a href="#">test_real</a>	14
<a href="#">view_real_results</a>	22



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

nn.Module	
test_real.LinearDeepQNetwork . . . . .	<a href="#">27</a>
test_local.TestLocal . . . . .	<a href="#">29</a>



## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">test_real.LinearDeepQNetwork</a> . . . . .	27
<a href="#">test_local.TestLocal</a> . . . . .	29



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">dual-cam.py</a>	33
<a href="#">test_local.py</a>	33
<a href="#">test_real.py</a>	34
<a href="#">view_real_results.py</a>	35





## Chapter 6

# Namespace Documentation

### 6.1 dual-cam Namespace Reference

#### Variables

- `bridge` = `CvBridge()`
- `pub_image` = `rospy.Publisher('/usb_cam/compressed/image_left', CompressedImage, tcp_nodelay=True, queue_size=1)`
- `pub_image2` = `rospy.Publisher('/usb_cam/compressed/image_right', CompressedImage, tcp_nodelay=True, queue_size=1)`
- `cam1` = `cv2.VideoCapture(0, cv2.CAP_V4L)`
- `cam2` = `cv2.VideoCapture(3, cv2.CAP_V4L)`
- `start` = `time.time()`
- `ret1`
- `frame1` = `bridge.cv2_to_compressed_imgmsg(frame1)`
- `ret2`
- `frame2` = `bridge.cv2_to_compressed_imgmsg(frame2)`
- `fps` = `round(1 / (time.time() - start), 1)`

#### 6.1.1 Variable Documentation

##### 6.1.1.1 bridge

```
dual-cam.bridge = CvBridge()
```

##### 6.1.1.2 cam1

```
dual-cam.cam1 = cv2.VideoCapture(0, cv2.CAP_V4L)
```

#### 6.1.1.3 cam2

```
dual-cam.cam2 = cv2.VideoCapture(3, cv2.CAP_V4L)
```

#### 6.1.1.4 fps

```
dual-cam.fps = round(1 / (time.time() - start), 1)
```

#### 6.1.1.5 frame1

```
dual-cam.frame1 = bridge.cv2_to_compressed_imgmsg(frame1)
```

#### 6.1.1.6 frame2

```
dual-cam.frame2 = bridge.cv2_to_compressed_imgmsg(frame2)
```

#### 6.1.1.7 pub\_image

```
dual-cam.pub_image = rospy.Publisher('/usb_cam/compressed/image_left', CompressedImage, tcp_↵  
nodelay=True, queue_size=1)
```

#### 6.1.1.8 pub\_image2

```
dual-cam.pub_image2 = rospy.Publisher('/usb_cam/compressed/image_right', CompressedImage,  
tcp_nodelay=True, queue_size=1)
```

#### 6.1.1.9 ret1

```
dual-cam.ret1
```

#### 6.1.1.10 ret2

```
dual-cam.ret2
```

#### 6.1.1.11 start

```
dual-cam.start = time.time()
```

## 6.2 test\_local Namespace Reference

### Classes

- class [TestLocal](#)

### Variables

- [path](#) = os.path.dirname(os.path.abspath(\_\_file\_\_))
- [config](#) = yaml.load(ymlfile, Loader=yaml.FullLoader)
- [real\\_ttb](#) = rf.RealTtb([config](#), [path](#), output=(640, 640))
- [test\\_local](#) = [TestLocal](#)()
- [key](#) = cv2.waitKey(1)
- [start](#) = time.time()
- [val](#) = test\_local.step()
- [fps](#) = round(1 / (time.time() - [start](#)), 1)

### 6.2.1 Variable Documentation

#### 6.2.1.1 config

```
test_local.config = yaml.load(ymlfile, Loader=yaml.FullLoader)
```

#### 6.2.1.2 fps

```
test_local.fps = round(1 / (time.time() - start), 1)
```

#### 6.2.1.3 key

```
test_local.key = cv2.waitKey(1)
```

#### 6.2.1.4 path

```
test_local.path = os.path.dirname(os.path.abspath(__file__))
```

#### 6.2.1.5 real\_ttb

```
test_local.real_ttb = rf.RealTtb(config, path, output=(640, 640))
```

#### 6.2.1.6 start

```
test_local.start = time.time()
```

#### 6.2.1.7 test\_local

```
test_local.test_local = TestLocal()
```

#### 6.2.1.8 val

```
test_local.val = test_local.step()
```

## 6.3 test\_real Namespace Reference

### Classes

- class [LinearDeepQNetwork](#)

### Functions

- def [updateLidar](#) (lidar)
- def [getImage](#) (image)

## Variables

- `pub_cmd_vel` = `rospy.Publisher('cmd_vel', Twist, queue_size=1)`
- `lidar_g` = `None`
- `sub_scan` = `rospy.Subscriber('scan', LaserScan, updateLidar)`
- `string TURTLE` = `'003'`
- `bridge` = `CvBridge()`
- `state` = `None`
- `font` = `cv2.FONT_HERSHEY_SIMPLEX`
- `outfile` = `TemporaryFile()`
- `int episodes` = `50`
- `int max_steps` = `50000`
- `list action_low` = `[-1.5, -0.1]`
- `list action_high` = `[1.5, 0.12]`
- `path` = `os.path.dirname(os.path.abspath(__file__))`
- `config` = `yaml.load(ymlfile, Loader=yaml.FullLoader)`
- `env` = `input('Which environment are you running? [1 | 2 | l]:\n')`
- `real_ttb` = `rf.RealTtb(config, path, output=(640, 640))`
- `sub_image` = `rospy.Subscriber('/usb_cam/compressed/image_right', CompressedImage, getImage, queue_size=1)`
- `string path_results` = `path + '/real_results'`
- `algorithms_sel` = `np.array(['1', '2', '3', 'e', 'r'])`
- `string algorithm` = `""`
- `string model` = `'2' else f"dqn_st{env.upper()}_model_5k.pth"`
- `string model_fn` = `f"{path}/{model}"`
- `int action_size` = `5`
- `int observation_space` = `26`
- `actor` = `LinearDeepQNetwork(0, action_size, observation_space)`
- `dictionary det` = `{0: -1.5, 1: -0.75, 2: 0, 3: 0.75, 4: 1.5}`
- `int local_episode` = `0`
- `quit` = `input("Press [Enter] to start the test or press [q] to quit...")`
- `int episode_reward` = `0`
- `lidar_list` = `list()`
- `int num_steps` = `0`
- `ep_start_time` = `time.time()`
- `bool done` = `False`
- `start` = `time.time()`
- `val_state` = `state`
- `actions` = `actor(state)`
- `action` = `torch.argmax(actions).item()`
- `vel_cmd` = `Twist()`
- `x`
- `z`
- `int reward` = `0`
- `scan` = `val_state[0:24]`
- `fps` = `round(1 / (time.time() - start))`
- `episode_timing` = `time.time() - ep_start_time`
- `list values` = `[episode_reward, episode_timing, local_episode, num_steps, real_ttb.pts, lidar_list]`

### 6.3.1 Function Documentation

#### 6.3.1.1 getImage()

```
def test_real.getImage (
    image )
```

#### 6.3.1.2 updateLidar()

```
def test_real.updateLidar (
    lidar )
```

### 6.3.2 Variable Documentation

#### 6.3.2.1 action

```
test_real.action = torch.argmax(actions).item()
```

#### 6.3.2.2 action\_high

```
list test_real.action_high = [1.5, 0.12]
```

#### 6.3.2.3 action\_low

```
list test_real.action_low = [-1.5, -0.1]
```

#### 6.3.2.4 action\_size

```
int test_real.action_size = 5
```

#### 6.3.2.5 actions

```
test_real.actions = actor(state)
```

#### 6.3.2.6 actor

```
test_real.actor = LinearDeepQNetwork(0, action_size, observation_space)
```

#### 6.3.2.7 algorithm

```
test_real.algorithm = ""
```

#### 6.3.2.8 algorithms\_sel

```
test_real.algorithms_sel = np.array(['l', '2', '3', 'e', 'r'])
```

#### 6.3.2.9 bridge

```
test_real.bridge = CvBridge()
```

#### 6.3.2.10 config

```
test_real.config = yaml.load(ymlfile, Loader=yaml.FullLoader)
```

#### 6.3.2.11 det

```
dictionary test_real.det = {0: -1.5, 1: -0.75, 2: 0, 3: 0.75, 4: 1.5}
```

#### 6.3.2.12 done

```
bool test_real.done = False
```

#### 6.3.2.13 env

```
test_real.env = input('Which environment are you running? [1 | 2 | 1]:\n')
```

#### 6.3.2.14 `ep_start_time`

```
test_real.ep_start_time = time.time()
```

#### 6.3.2.15 `episode_reward`

```
int test_real.episode_reward = 0
```

#### 6.3.2.16 `episode_timing`

```
test_real.episode_timing = time.time() - ep_start_time
```

#### 6.3.2.17 `episodes`

```
int test_real.episodes = 50
```

#### 6.3.2.18 `font`

```
test_real.font = cv2.FONT_HERSHEY_SIMPLEX
```

#### 6.3.2.19 `fps`

```
test_real.fps = round(1 / (time.time() - start))
```

#### 6.3.2.20 `lidar_g`

```
test_real.lidar_g = None
```

#### 6.3.2.21 `lidar_list`

```
test_real.lidar_list = list()
```



#### 6.3.2.22 local\_episode

```
int test_real.local_episode = 0
```

#### 6.3.2.23 max\_steps

```
int test_real.max_steps = 50000
```

#### 6.3.2.24 model

```
string test_real.model = '2' else f"dqn_st{env.upper()}_model_5k.pth"
```

#### 6.3.2.25 model\_fn

```
string test_real.model_fn = f"{path}/{model}"
```

#### 6.3.2.26 num\_steps

```
int test_real.num_steps = 0
```

#### 6.3.2.27 observation\_space

```
int test_real.observation_space = 26
```

#### 6.3.2.28 outfile

```
test_real.outfile = TemporaryFile()
```

#### 6.3.2.29 path

```
test_real.path = os.path.dirname(os.path.abspath(__file__))
```

#### 6.3.2.30 path\_results

```
string test_real.path_results = path + '/real_results'
```

#### 6.3.2.31 pub\_cmd\_vel

```
test_real.pub_cmd_vel = rospy.Publisher('cmd_vel', Twist, queue_size=1)
```

#### 6.3.2.32 quit

```
test_real.quit = input("Press [Enter] to start the test or press [q] to quit...")
```

#### 6.3.2.33 real\_ttb

```
test_real.real_ttb = rf.RealTtb(config, path, output=(640, 640))
```

#### 6.3.2.34 reward

```
int test_real.reward = 0
```

#### 6.3.2.35 scan

```
test_real.scan = val_state[0:24]
```

#### 6.3.2.36 start

```
test_real.start = time.time()
```

#### 6.3.2.37 state

```
test_real.state = None
```

#### 6.3.2.38 sub\_image

```
test_real.sub_image = rospy.Subscriber('/usb_cam/compressed/image_right', CompressedImage,  
getImage, queue_size=1)
```

#### 6.3.2.39 sub\_scan

```
test_real.sub_scan = rospy.Subscriber('scan', LaserScan, updateLidar)
```

#### 6.3.2.40 TURTLE

```
string test_real.TURTLE = '003'
```

#### 6.3.2.41 val\_state

```
test_real.val_state = state
```

#### 6.3.2.42 values

```
list test_real.values = [episode_reward, episode_timing, local_episode, num_steps, real_ttb.↵  
pts, lidar_list]
```

#### 6.3.2.43 vel\_cmd

```
test_real.vel_cmd = Twist()
```

#### 6.3.2.44 x

```
test_real.x
```

### 6.3.2.45 z

test\_real.z

## 6.4 view\_real\_results Namespace Reference

### Functions

- def [antispike](#) (old\_list\_x, old\_list\_y)
- def [antispike2](#) (old\_list\_x, old\_list\_y)
- def [open\\_test\\_data](#) (i)

### Variables

- [path](#) = os.path.dirname(os.path.abspath(\_\_file\_\_))
- [list\\_dir](#) = os.listdir([path](#) + '/real\_results/')
- int [threshold\\_x](#) = 10
- int [threshold\\_y](#) = 30
- int [threshold](#) = 10
- int [STAGE](#) = 1
- int [c](#) = 7
- [stage](#) = mpimg.imread([path](#)+'/media/stage\_{}\_real.png'.format([STAGE](#)))
- [data](#) = list()
- dictionary [color](#) = {0: 'firebrick', 1: 'tomato', 2: 'peru', 3: 'gold', 4: 'dodgerblue', 5: 'springgreen', 6: 'indigo', 7: 'deeppink'}
- [size](#) = len([data](#))
- [rewards](#) = list()
- [times](#) = list()
- list [x](#) = []
- list [y](#) = []
- [linestyle](#)
- [linewidth](#)

### 6.4.1 Function Documentation

#### 6.4.1.1 antispike()

```
def view_real_results.antispike (
    old_list_x,
    old_list_y )
```

#### 6.4.1.2 antispike2()

```
def view_real_results.antispike2 (
    old_list_x,
    old_list_y )
```

#### 6.4.1.3 open\_test\_data()

```
def view_real_results.open_test_data (
    i )
```

### 6.4.2 Variable Documentation

#### 6.4.2.1 c

```
int view_real_results.c = 7
```

#### 6.4.2.2 color

```
view_real_results.color = {0: 'firebrick', 1: 'tomato', 2: 'peru', 3: 'gold', 4: 'dodgerblue',
5: 'springgreen', 6: 'indigo', 7: 'deeppink'}
```

#### 6.4.2.3 data

```
view_real_results.data = list()
```

#### 6.4.2.4 linestyle

```
view_real_results.linestyle
```

#### 6.4.2.5 linewidth

```
view_real_results.linewidth
```

#### 6.4.2.6 list\_dir

```
view_real_results.list_dir = os.listdir(path + '/real_results/')
```

#### 6.4.2.7 path

```
view_real_results.path = os.path.dirname(os.path.abspath(__file__))
```

#### 6.4.2.8 rewards

```
view_real_results.rewards = list()
```

#### 6.4.2.9 size

```
view_real_results.size = len(data)
```

#### 6.4.2.10 STAGE

```
int view_real_results.STAGE = 1
```

#### 6.4.2.11 stage

```
view_real_results.stage = mpimg.imread(path+'media/stage_{}_real.png'.format(STAGE))
```

#### 6.4.2.12 threshold

```
int view_real_results.threshold = 10
```

#### 6.4.2.13 threshold\_x

```
int view_real_results.threshold_x = 10
```

#### 6.4.2.14 threshold\_y

```
int view_real_results.threshold_y = 30
```

#### 6.4.2.15 times

```
view_real_results.times = list()
```

#### 6.4.2.16 x

```
list view_real_results.x = []
```

#### 6.4.2.17 y

```
list view_real_results.y = []
```



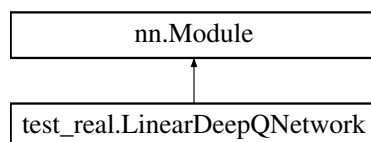


## Chapter 7

# Class Documentation

### 7.1 test\_real.LinearDeepQNetwork Class Reference

Inheritance diagram for test\_real.LinearDeepQNetwork:



#### Public Member Functions

- def `__init__` (self, lr, n\_actions, input\_dims)
- def `forward` (self, state)

#### Public Attributes

- `fc1`
- `fc2`
- `dropout`
- `fc3`
- `optimizer`
- `loss`
- `device`

#### 7.1.1 Constructor & Destructor Documentation

#### 7.1.1.1 `__init__()`

```
def test_real.LinearDeepQNetwork.__init__ (
    self,
    lr,
    n_actions,
    input_dims )
```

### 7.1.2 Member Function Documentation

#### 7.1.2.1 `forward()`

```
def test_real.LinearDeepQNetwork.forward (
    self,
    state )
```

### 7.1.3 Member Data Documentation

#### 7.1.3.1 `device`

```
test_real.LinearDeepQNetwork.device
```

#### 7.1.3.2 `dropout`

```
test_real.LinearDeepQNetwork.dropout
```

#### 7.1.3.3 `fc1`

```
test_real.LinearDeepQNetwork.fc1
```

#### 7.1.3.4 `fc2`

```
test_real.LinearDeepQNetwork.fc2
```

### 7.1.3.5 fc3

`test_real.LinearDeepQNetwork.fc3`

### 7.1.3.6 loss

`test_real.LinearDeepQNetwork.loss`

### 7.1.3.7 optimizer

`test_real.LinearDeepQNetwork.optimizer`

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

- [test\\_real.py](#)

## 7.2 test\_local.TestLocal Class Reference

### Public Member Functions

- `def \_\_init\_\_ (self)`
- `def image\_right\_callback (self, msg)`
- `def image\_left\_callback (self, msg)`
- `def step (self)`

### Public Attributes

- [bridge](#)
- [stitcher](#)
- [image\\_right](#)
- [image\\_left](#)
- [defisheye1](#)
- [defisheye2](#)

### 7.2.1 Constructor & Destructor Documentation

#### 7.2.1.1 `__init__()`

```
def test_local.TestLocal.__init__ (  
    self )
```

## 7.2.2 Member Function Documentation

### 7.2.2.1 image\_left\_callback()

```
def test_local.TestLocal.image_left_callback (
    self,
    msg )
```

### 7.2.2.2 image\_right\_callback()

```
def test_local.TestLocal.image_right_callback (
    self,
    msg )
```

### 7.2.2.3 step()

```
def test_local.TestLocal.step (
    self )
```

## 7.2.3 Member Data Documentation

### 7.2.3.1 bridge

```
test_local.TestLocal.bridge
```

### 7.2.3.2 defishey1

```
test_local.TestLocal.defishey1
```

### 7.2.3.3 defishey2

```
test_local.TestLocal.defishey2
```

#### 7.2.3.4 image\_left

`test_local.TestLocal.image_left`

#### 7.2.3.5 image\_right

`test_local.TestLocal.image_right`

#### 7.2.3.6 stitcher

`test_local.TestLocal.stitcher`

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

- [test\\_local.py](#)



## Chapter 8

# File Documentation

### 8.1 dual-cam.py File Reference

#### Namespaces

- namespace [dual-cam](#)

#### Variables

- [dual-cam.bridge](#) = CvBridge()
- [dual-cam.pub\\_image](#) = rospy.Publisher('/usb\_cam/compressed/image\_left', CompressedImage, tcp\_↔  
nodelay=True, queue\_size=1)
- [dual-cam.pub\\_image2](#) = rospy.Publisher('/usb\_cam/compressed/image\_right', CompressedImage, tcp\_↔  
nodelay=True, queue\_size=1)
- [dual-cam.cam1](#) = cv2.VideoCapture(0, cv2.CAP\_V4L)
- [dual-cam.cam2](#) = cv2.VideoCapture(3, cv2.CAP\_V4L)
- [dual-cam.start](#) = time.time()
- [dual-cam.ret1](#)
- [dual-cam.frame1](#) = bridge.cv2\_to\_compressed\_imgmsg(frame1)
- [dual-cam.ret2](#)
- [dual-cam.frame2](#) = bridge.cv2\_to\_compressed\_imgmsg(frame2)
- [dual-cam.fps](#) = round(1 / (time.time() - start), 1)

### 8.2 README.md File Reference

### 8.3 test\_local.py File Reference

#### Classes

- class [test\\_local.TestLocal](#)

#### Namespaces

- namespace [test\\_local](#)

## Variables

- `test_local.path` = `os.path.dirname(os.path.abspath(__file__))`
- `test_local.config` = `yaml.load(ymlfile, Loader=yaml.FullLoader)`
- `test_local.real_ttb` = `rf.RealTtb(config, path, output=(640, 640))`
- `test_local.test_local` = `TestLocal()`
- `test_local.key` = `cv2.waitKey(1)`
- `test_local.start` = `time.time()`
- `test_local.val` = `test_local.step()`
- `test_local.fps` = `round(1 / (time.time() - start), 1)`

## 8.4 test\_real.py File Reference

### Classes

- class `test_real.LinearDeepQNetwork`

### Namespaces

- namespace `test_real`

### Functions

- def `test_real.updateLidar` (lidar)
- def `test_real.getImage` (image)

### Variables

- `test_real.pub_cmd_vel` = `rospy.Publisher('cmd_vel', Twist, queue_size=1)`
- `test_real.lidar_g` = `None`
- `test_real.sub_scan` = `rospy.Subscriber('scan', LaserScan, updateLidar)`
- string `test_real.TURTLE` = `'003'`
- `test_real.bridge` = `CvBridge()`
- `test_real.state` = `None`
- `test_real.font` = `cv2.FONT_HERSHEY_SIMPLEX`
- `test_real.outfile` = `TemporaryFile()`
- int `test_real.episodes` = `50`
- int `test_real.max_steps` = `50000`
- list `test_real.action_low` = `[-1.5, -0.1]`
- list `test_real.action_high` = `[1.5, 0.12]`
- `test_real.path` = `os.path.dirname(os.path.abspath(__file__))`
- `test_real.config` = `yaml.load(ymlfile, Loader=yaml.FullLoader)`
- `test_real.env` = `input('Which environment are you running? [1 | 2 | I]:\n')`
- `test_real.real_ttb` = `rf.RealTtb(config, path, output=(640, 640))`
- `test_real.sub_image` = `rospy.Subscriber('/usb_cam/compressed/image_right', CompressedImage, getImage, queue_size=1)`
- string `test_real.path_results` = `path + '/real_results'`
- `test_real.algorithms_sel` = `np.array(['1', '2', '3', 'e', 'r'])`
- string `test_real.algorithm` = `""`
- string `test_real.model` = `'2' else f"dqn_st{env.upper()}_model_5k.pth"`



- string `test_real.model_fn` = f"{path}/{model}"
- int `test_real.action_size` = 5
- int `test_real.observation_space` = 26
- `test_real.actor` = LinearDeepQNetwork(0, action\_size, observation\_space)
- dictionary `test_real.det` = {0: -1.5, 1: -0.75, 2: 0, 3: 0.75, 4: 1.5}
- int `test_real.local_episode` = 0
- `test_real.quit` = input("Press [Enter] to start the test or press [q] to quit...")
- int `test_real.episode_reward` = 0
- `test_real.lidar_list` = list()
- int `test_real.num_steps` = 0
- `test_real.ep_start_time` = time.time()
- bool `test_real.done` = False
- `test_real.start` = time.time()
- `test_real.val_state` = state
- `test_real.actions` = actor(state)
- `test_real.action` = torch.argmax(actions).item()
- `test_real.vel_cmd` = Twist()
- `test_real.x`
- `test_real.z`
- int `test_real.reward` = 0
- `test_real.scan` = val\_state[0:24]
- `test_real.fps` = round(1 / (time.time() - start))
- `test_real.episode_timing` = time.time() - ep\_start\_time
- list `test_real.values` = [episode\_reward, episode\_timing, local\_episode, num\_steps, real\_ttb.pts, lidar\_list]

## 8.5 view\_real\_results.py File Reference

### Namespaces

- namespace `view_real_results`

### Functions

- def `view_real_results.antispikes` (old\_list\_x, old\_list\_y)
- def `view_real_results.antispikes2` (old\_list\_x, old\_list\_y)
- def `view_real_results.open_test_data` (i)

### Variables

- `view_real_results.path` = os.path.dirname(os.path.abspath(\_\_file\_\_))
- `view_real_results.list_dir` = os.listdir(path + '/real\_results/')
- int `view_real_results.threshold_x` = 10
- int `view_real_results.threshold_y` = 30
- int `view_real_results.threshold` = 10
- int `view_real_results.STAGE` = 1
- int `view_real_results.c` = 7
- `view_real_results.stage` = mpimg.imread(path + '/media/stage\_{}\_real.png'.format(STAGE))
- `view_real_results.data` = list()
- dictionary `view_real_results.color` = {0: 'firebrick', 1: 'tomato', 2: 'peru', 3: 'gold', 4: 'dodgerblue', 5: 'spring-green', 6: 'indigo', 7: 'deeppink'}
- `view_real_results.size` = len(data)
- `view_real_results.rewards` = list()
- `view_real_results.times` = list()
- list `view_real_results.x` = []
- list `view_real_results.y` = []
- `view_real_results.linestyle`
- `view_real_results.linewidth`



# Index

- `__init__`
  - `test_local.TestLocal`, [29](#)
  - `test_real.LinearDeepQNetwork`, [27](#)
- `action`
  - `test_real`, [16](#)
- `action_high`
  - `test_real`, [16](#)
- `action_low`
  - `test_real`, [16](#)
- `action_size`
  - `test_real`, [16](#)
- `actions`
  - `test_real`, [16](#)
- `actor`
  - `test_real`, [16](#)
- `algorithm`
  - `test_real`, [17](#)
- `algorithms_sel`
  - `test_real`, [17](#)
- `antispike`
  - `view_real_results`, [22](#)
- `antispike2`
  - `view_real_results`, [22](#)
- `bridge`
  - `dual-cam`, [11](#)
  - `test_local.TestLocal`, [30](#)
  - `test_real`, [17](#)
- `c`
  - `view_real_results`, [23](#)
- `cam1`
  - `dual-cam`, [11](#)
- `cam2`
  - `dual-cam`, [11](#)
- `color`
  - `view_real_results`, [23](#)
- `config`
  - `test_local`, [13](#)
  - `test_real`, [17](#)
- `data`
  - `view_real_results`, [23](#)
- `defisheye1`
  - `test_local.TestLocal`, [30](#)
- `defisheye2`
  - `test_local.TestLocal`, [30](#)
- `det`
  - `test_real`, [17](#)
- `device`
  - `test_real.LinearDeepQNetwork`, [28](#)
- `done`
  - `test_real`, [17](#)
- `dropout`
  - `test_real.LinearDeepQNetwork`, [28](#)
- `dual-cam`, [11](#)
  - `bridge`, [11](#)
  - `cam1`, [11](#)
  - `cam2`, [11](#)
  - `fps`, [12](#)
  - `frame1`, [12](#)
  - `frame2`, [12](#)
  - `pub_image`, [12](#)
  - `pub_image2`, [12](#)
  - `ret1`, [12](#)
  - `ret2`, [12](#)
  - `start`, [13](#)
- `dual-cam.py`, [33](#)
- `env`
  - `test_real`, [17](#)
- `ep_start_time`
  - `test_real`, [17](#)
- `episode_reward`
  - `test_real`, [18](#)
- `episode_timing`
  - `test_real`, [18](#)
- `episodes`
  - `test_real`, [18](#)
- `fc1`
  - `test_real.LinearDeepQNetwork`, [28](#)
- `fc2`
  - `test_real.LinearDeepQNetwork`, [28](#)
- `fc3`
  - `test_real.LinearDeepQNetwork`, [28](#)
- `font`
  - `test_real`, [18](#)
- `forward`
  - `test_real.LinearDeepQNetwork`, [28](#)
- `fps`
  - `dual-cam`, [12](#)
  - `test_local`, [13](#)
  - `test_real`, [18](#)
- `frame1`
  - `dual-cam`, [12](#)
- `frame2`
  - `dual-cam`, [12](#)

- getImage
  - test\_real, 15
- image\_left
  - test\_local.TestLocal, 30
- image\_left\_callback
  - test\_local.TestLocal, 30
- image\_right
  - test\_local.TestLocal, 31
- image\_right\_callback
  - test\_local.TestLocal, 30
- key
  - test\_local, 13
- lidar\_g
  - test\_real, 18
- lidar\_list
  - test\_real, 18
- linestyle
  - view\_real\_results, 23
- linewidth
  - view\_real\_results, 23
- list\_dir
  - view\_real\_results, 23
- local\_episode
  - test\_real, 18
- loss
  - test\_real.LinearDeepQNetwork, 29
- max\_steps
  - test\_real, 19
- model
  - test\_real, 19
- model\_fn
  - test\_real, 19
- num\_steps
  - test\_real, 19
- observation\_space
  - test\_real, 19
- open\_test\_data
  - view\_real\_results, 23
- optimizer
  - test\_real.LinearDeepQNetwork, 29
- outfile
  - test\_real, 19
- path
  - test\_local, 14
  - test\_real, 19
  - view\_real\_results, 24
- path\_results
  - test\_real, 19
- pub\_cmd\_vel
  - test\_real, 20
- pub\_image
  - dual-cam, 12
- pub\_image2
  - dual-cam, 12
- quit
  - test\_real, 20
- README.md, 33
- real\_ttb
  - test\_local, 14
  - test\_real, 20
- ret1
  - dual-cam, 12
- ret2
  - dual-cam, 12
- reward
  - test\_real, 20
- rewards
  - view\_real\_results, 24
- scan
  - test\_real, 20
- size
  - view\_real\_results, 24
- STAGE
  - view\_real\_results, 24
- stage
  - view\_real\_results, 24
- start
  - dual-cam, 13
  - test\_local, 14
  - test\_real, 20
- state
  - test\_real, 20
- step
  - test\_local.TestLocal, 30
- stitcher
  - test\_local.TestLocal, 31
- sub\_image
  - test\_real, 20
- sub\_scan
  - test\_real, 21
- test\_local, 13
  - config, 13
  - fps, 13
  - key, 13
  - path, 14
  - real\_ttb, 14
  - start, 14
  - test\_local, 14
  - val, 14
- test\_local.py, 33
- test\_local.TestLocal, 29
  - \_\_init\_\_, 29
  - bridge, 30
  - defisheye1, 30
  - defisheye2, 30
  - image\_left, 30
  - image\_left\_callback, 30
  - image\_right, 31

- image\_right\_callback, 30
- step, 30
- stitcher, 31
- test\_real, 14
  - action, 16
  - action\_high, 16
  - action\_low, 16
  - action\_size, 16
  - actions, 16
  - actor, 16
  - algorithm, 17
  - algorithms\_sel, 17
  - bridge, 17
  - config, 17
  - det, 17
  - done, 17
  - env, 17
  - ep\_start\_time, 17
  - episode\_reward, 18
  - episode\_timing, 18
  - episodes, 18
  - font, 18
  - fps, 18
  - getImage, 15
  - lidar\_g, 18
  - lidar\_list, 18
  - local\_episode, 18
  - max\_steps, 19
  - model, 19
  - model\_fn, 19
  - num\_steps, 19
  - observation\_space, 19
  - outfile, 19
  - path, 19
  - path\_results, 19
  - pub\_cmd\_vel, 20
  - quit, 20
  - real\_ttb, 20
  - reward, 20
  - scan, 20
  - start, 20
  - state, 20
  - sub\_image, 20
  - sub\_scan, 21
  - TURTLE, 21
  - updateLidar, 16
  - val\_state, 21
  - values, 21
  - vel\_cmd, 21
  - x, 21
  - z, 21
- test\_real.LinearDeepQNetwork, 27
  - \_\_init\_\_, 27
  - device, 28
  - dropout, 28
  - fc1, 28
  - fc2, 28
  - fc3, 28
  - forward, 28
  - loss, 29
  - optimizer, 29
- test\_real.py, 34
- threshold
  - view\_real\_results, 24
- threshold\_x
  - view\_real\_results, 24
- threshold\_y
  - view\_real\_results, 24
- times
  - view\_real\_results, 25
- TURTLE
  - test\_real, 21
- updateLidar
  - test\_real, 16
- val
  - test\_local, 14
- val\_state
  - test\_real, 21
- values
  - test\_real, 21
- vel\_cmd
  - test\_real, 21
- view\_real\_results, 22
  - antispikes, 22
  - antispikes2, 22
  - c, 23
  - color, 23
  - data, 23
  - linestyle, 23
  - linewidth, 23
  - list\_dir, 23
  - open\_test\_data, 23
  - path, 24
  - rewards, 24
  - size, 24
  - STAGE, 24
  - stage, 24
  - threshold, 24
  - threshold\_x, 24
  - threshold\_y, 24
  - times, 25
  - x, 25
  - y, 25
- view\_real\_results.py, 35
- x
  - test\_real, 21
  - view\_real\_results, 25
- y
  - view\_real\_results, 25
- z
  - test\_real, 21