

face_detection

Generated by Doxygen 1.8.11

Contents

1	Module Index	1
1.1	Modules	1
2	Namespace Index	3
2.1	Namespace List	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Module Documentation	9
5.1	ROS related variables	9
5.1.1	Detailed Description	9
5.1.2	Variable Documentation	9
5.1.2.1	cv_ptr	9
5.1.2.2	face_found_pub_	9
5.1.2.3	image_pub_	9
5.1.2.4	image_sub_	10
5.1.2.5	img_location_	10
5.1.2.6	it_	10
5.1.2.7	move_base_pub_	10
5.1.2.8	nh_	10
5.2	General variables	11
5.2.1	Detailed Description	11
5.2.2	Variable Documentation	11
5.2.2.1	count_	11
5.2.2.2	currentMoveBase	11
5.2.2.3	params	11
5.2.2.4	skip_val	11

6	Namespace Documentation	13
6.1	cv Namespace Reference	13
7	Class Documentation	15
7.1	FaceTracking Class Reference	15
7.1.1	Detailed Description	16
7.1.2	Member Enumeration Documentation	16
7.1.2.1	TrackingState	16
7.1.3	Constructor & Destructor Documentation	17
7.1.3.1	FaceTracking()	17
7.1.3.2	~FaceTracking()	17
7.1.4	Member Function Documentation	17
7.1.4.1	calculateArea(float semi_major_axis, float semi_minor_axis)	17
7.1.4.2	getCloseFaces(std::vector< dlib::rectangle > faces_, int tracking_index)	17
7.1.4.3	getMaxAreaIndex(dlib::cv_image< dlib::bgr_pixel > &img_, std::vector< dlib::rectangle > &input, cv::vector< cv::vector< cv::Point >> &all_landmarks_, cv::vector< cv::vector< cv::Point >> &jaw_line, int &max_area_index)	18
7.1.4.4	getTrackingIndex(dlib::cv_image< dlib::bgr_pixel > img_, std::vector< dlib::rectangle > faces_)	18
7.1.4.5	initialize(std::string path_, std::map< std::__cxx11::string, int > ¶m_)	18
7.1.4.6	moveBase()	19
7.1.4.7	requestDetectedRealTime()	19
7.1.4.8	requestEllipseCenter()	19
7.1.4.9	requestFacesSize()	19
7.1.4.10	RMSE(std::vector< cv::Point > ground_truth, std::vector< cv::Point > fitted_shapes)	19
7.1.4.11	startTracking(cv::Mat &image_, std::vector< dlib::rectangle > &cfaces, cv::vector< cv::vector< cv::Point >> &all_landmarks_, cv::vector< cv::vector< cv::Point >> &jaw_line, int &max_area_index)	20
7.1.4.12	Temp()	20
7.1.4.13	trackLandmark(cv::Mat &input_image_, cv::Mat &output_image_)	20
7.1.5	Member Data Documentation	20
7.1.5.1	cfacesize	21
7.1.5.2	detector	21

7.1.5.3	face_found	21
7.1.5.4	ground_truth	21
7.1.5.5	minEllipse	21
7.1.5.6	move_base	21
7.1.5.7	pose_model	21
7.1.5.8	RANGE_FOR_DETECTED	21
7.1.5.9	RANGE_FOR_TRACKING	21
7.1.5.10	state	21
7.1.5.11	turnLeft	22
7.2	ROSFaceDetection Class Reference	22
7.2.1	Detailed Description	23
7.2.2	Constructor & Destructor Documentation	23
7.2.2.1	ROSFaceDetection()	23
7.2.3	Member Function Documentation	23
7.2.3.1	execute(void)	23
7.2.3.2	faceFound()	23
7.2.3.3	getEllipseCenter()	23
7.2.3.4	imageCallBack(const sensor_msgs::ImageConstPtr &msg)	23
7.2.3.5	moveBase()	24
7.2.4	Member Data Documentation	24
7.2.4.1	faceTracker	24
8	File Documentation	25
8.1	face_detection.h File Reference	25
8.1.1	Detailed Description	25
8.2	face_tracking.h File Reference	26
8.2.1	Detailed Description	26
8.3	face_detection.cpp File Reference	27
8.4	face_detection_node.cpp File Reference	27
8.4.1	Function Documentation	27
8.4.1.1	main(int argc, char **argv)	27
8.5	face_tracking.cpp File Reference	27
	Index	29

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

ROS related variables	9
General variables	11

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

cv	13
------------------------------	--------------------

Chapter 3

Class Index

3.1 Class List

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

FaceTracking	
A library to handle face detection tracking and face tracking	15
ROSFaceDetection	
The ROS wrapper class for Hubert's Face Detection application	22

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

face_detection.h	This is the header file of the <code>face_detection/_node.cpp</code> , which was implemented as a ROS wrapper. This ROS node will use the face detection library to perform face detection and face tracking. The tracked face details will then be published to the related topics so that it will grantee a smooth process in the Hubert Brain	25
face_tracking.h	This is the header file of the face_tracking.cpp , which was implemented to detect faces, track the closest face. This method is also called as the First Landmark Tracking, as it will focus only on the first person who enters the screen or if many enters the frame then the closest person (by analysing the area of the bounding box)	26
face_detection.cpp	27
face_detection_node.cpp	27
face_tracking.cpp	27

Chapter 5

Module Documentation

5.1 ROS related variables

Variables

- `ros::NodeHandle` [ROSFaceDetection::nh_](#)
- `image_transport::ImageTransport` [ROSFaceDetection::it_](#)
- `image_transport::Subscriber` [ROSFaceDetection::image_sub_](#)
- `image_transport::Publisher` [ROSFaceDetection::image_pub_](#)
- `ros::Publisher` [ROSFaceDetection::img_location_](#)
- `ros::Publisher` [ROSFaceDetection::move_base_pub_](#)
- `ros::Publisher` [ROSFaceDetection::face_found_pub_](#)
- `cv_bridge::CvImagePtr` [ROSFaceDetection::cv_ptr](#)

5.1.1 Detailed Description

In this group all the variables related to ROS will be define

5.1.2 Variable Documentation

5.1.2.1 `cv_bridge::CvImagePtr` [ROSFaceDetection::cv_ptr](#) [private]

Image pointer

5.1.2.2 `ros::Publisher` [ROSFaceDetection::face_found_pub_](#) [private]

Publish Face detection status (False - Not detected True - Detected)

5.1.2.3 `image_transport::Publisher` [ROSFaceDetection::image_pub_](#) [private]

This is the image publisher(will publish to the system)

5.1.2.4 `image_transport::Subscriber ROSFaceDetection::image_sub_ [private]`

This is the image subscriber(will get subscribe from the camera)

5.1.2.5 `ros::Publisher ROSFaceDetection::img_location_ [private]`

Publish the center points of the ellipse and its height and width Face.msg type data

5.1.2.6 `image_transport::ImageTransport ROSFaceDetection::it_ [private]`

This is used to subscribe to and publish images. In other words, this work as a node handler for images

5.1.2.7 `ros::Publisher ROSFaceDetection::move_base_pub_ [private]`

Publish MoveBase details MoveBse.msg type

5.1.2.8 `ros::NodeHandle ROSFaceDetection::nh_ [private]`

ROS node handler

5.2 General variables

Variables

- `std::map< std::string, int > ROSFaceDetection::params`
- `int ROSFaceDetection::count_ = 0`
- `bool ROSFaceDetection::currentMoveBase = true`
- `int ROSFaceDetection::skip_val`

5.2.1 Detailed Description

In this group all the general variable will be define

5.2.2 Variable Documentation

5.2.2.1 `int ROSFaceDetection::count_ = 0` `[private]`

Counter to track the skip_frame parameter and to execute

5.2.2.2 `bool ROSFaceDetection::currentMoveBase = true` `[private]`

5.2.2.3 `std::map<std::string, int> ROSFaceDetection::params` `[private]`

ROS parameter `std::map<string,int>` use to pass the variables to the library

5.2.2.4 `int ROSFaceDetection::skip_val` `[private]`

Chapter 6

Namespace Documentation

6.1 cv Namespace Reference

Chapter 7

Class Documentation

7.1 FaceTracking Class Reference

A library to handle face detection tracking and face tracking.

```
#include <face_tracking.h>
```

Public Types

- enum [TrackingState](#) { [Initialize](#), [Tracking](#), [Updating](#) }

Public Member Functions

- [FaceTracking](#) ()
A Constructor for the class.
- [~FaceTracking](#) ()
A Destructor.
- void [initialize](#) (std::string path_, std::map< std::__cxx11::string, int > ¶m_)
Initialize the library This is the function which initialize the library and this must be called otherwise, the other function cannot run this library. The ROS parameter which are defined in launch files will be passed into the library via this function.
- void [trackLandmark](#) (cv::Mat &input_image_, cv::Mat &output_image_)
Face detection/tracking function This is the main trigger function. Takes input image and returns the detailed tracked image.
- cv::RotatedRect [requestEllipseCenter](#) ()
Center bound of ellipse of the current tracking face will be returned, upon the request from the ROS node.
- bool [requestDetectedRealTime](#) ()
Status of the face detection, This value is a boolean variable which return true if a face is present in the current frame.
- int [requestFacesSize](#) ()
Number of close faces.
- std::map< std::__cxx11::string, bool > [moveBase](#) ()
Status of the base frame This function will return whether or not to rotate the base of the robot. Furthermore it will also return which way the robot should turn Here move is true when the camera needs to rotate and the direction tells whether to turn left or right.
- int [Temp](#) ()

Private Member Functions

- float [calculateArea](#) (float semi_major_axis, float semi_minor_axis)
calculate the area of the ellipse
- float [RMSE](#) (std::vector< cv::Point > [ground_truth](#), std::vector< cv::Point > fitted_shapes)
calculate Root Mean Square Error of the current face and the ground truth here the ground truth will be the values of the previously tracked face
- int [getTrackingIndex](#) (dlib::cv_image< dlib::bgr_pixel > img_, std::vector< dlib::rectangle > faces_)
return tracking index of the current tracking face of the dlib face rectangle vector
- std::vector< dlib::rectangle > [getCloseFaces](#) (std::vector< dlib::rectangle > faces_, int tracking_index)
Choose whether the face is in the correct range. We have provided some parameters which could decide the range of tracking.
- void [getMaxAreaIndex](#) (dlib::cv_image< dlib::bgr_pixel > &img_, std::vector< dlib::rectangle > &input, cv::vector< cv::vector< cv::Point >> &all_landmarks_, cv::vector< cv::vector< cv::Point >> &jaw_line, int &max_area_index)
calculate the area of the ellipse and to find which face has the largest area (closest to the camera)
- void [startTracking](#) (cv::Mat &image_, std::vector< dlib::rectangle > &cfaces, cv::vector< cv::vector< cv::Point >> &all_landmarks_, cv::vector< cv::vector< cv::Point >> &jaw_line, int &max_area_index)
calculate the area of the ellipse and to find which face has the largest area (closest to the camera)

Private Attributes

- int [cfacesize](#) = 0
- int [RANGE_FOR_DETECTED](#)
- int [RANGE_FOR_TRACKING](#)
- bool [face_found](#)
- bool [move_base](#)
- bool [turnLeft](#)
- [TrackingState](#) state = [Initialize](#)
- cv::RotatedRect [minEllipse](#)
- std::vector< cv::Point > [ground_truth](#)
- dlib::frontal_face_detector [detector](#)
- dlib::shape_predictor [pose_model](#)

7.1.1 Detailed Description

A library to handle face detection tracking and face tracking.

Perform face detection and face tracking.

7.1.2 Member Enumeration Documentation

7.1.2.1 enum [FaceTracking::TrackingState](#)

This is an enum class to check the current tracking state

Enumerator

- Initialize*** Initial state
- Tracking*** Tracking state
- Updating*** Update ground truth

7.1.3 Constructor & Destructor Documentation

7.1.3.1 FaceTracking::FaceTracking ()

A Constructor for the class.

Will be using to initialize initial values of the user variables

7.1.3.2 FaceTracking::~FaceTracking () [inline]

A Destructor.

This is an empty destructor created for future use (can remove)

7.1.4 Member Function Documentation

7.1.4.1 float FaceTracking::calculateArea (float *semi_major_axis*, float *semi_minor_axis*) [private]

calculate the area of the ellipse

Parameters

in	<i>semi_major_axis</i>	ellipse width
in	<i>semi_minor_axis</i>	ellipse height

Returns

return the area of the ellipse (float)

7.1.4.2 std::vector< dlib::rectangle > FaceTracking::getCloseFaces (std::vector< dlib::rectangle > *faces_*, int *tracking_index*) [private]

Choose whether the face is in the correct range. We have provided some parameters which could decide the range of tracking.

Parameters

in	<i>faces_</i>	dlib face rectangle vector
in	<i>tracking_index</i>	current tracking index

Returns

return dlib face rectangle vector with the new close face

Remarks

- Range of tracking : Upto what extend should we keep tracking this person
- Range of detection : Upto what extend should we consider detecting a person
- tracking index

7.1.4.3 `void FaceTracking::getMaxAreaIndex (dlib::cv_image< dlib::bgr_pixel > &img_, std::vector< dlib::rectangle > &input, cv::vector< cv::vector< cv::Point >> &all_landmarks_, cv::vector< cv::vector< cv::Point >> &jaw_line, int &max_area_index) [private]`

calculate the area of the ellipse and to find which face has the largest area (closest to the camera)

Parameters

in	<i>img_</i>	input image ni the dlib format
in	<i>input</i>	dlib face rectangle vector
out	<i>all_landmarks_</i> —	get the points of all faces detected by the dlib face detector
out	<i>jaw_line</i>	get the jaw line points (from the 68 points) of all faces detected by the dlib face detector
out	<i>max_area_index</i>	find the face with the maximum area

Remarks

- The jaw line points will be used to form an ellipse, which will then be use to calculate the area

7.1.4.4 `int FaceTracking::getTrackingIndex (dlib::cv_image< dlib::bgr_pixel > img_, std::vector< dlib::rectangle > faces_) [private]`

return tracking index of the current tracking face of the dlib face rectangle vector

Parameters

in	<i>img_</i> —	input image
in	<i>faces_</i> —	dlib face rectangle vector

Returns

index of the current tracking face

7.1.4.5 `void FaceTracking::initialize (std::string path_, std::map< std::__cxx11::string, int > ¶m_)`

Initialize the library This is the function which initialize the library and this must be called otherwise, the other function cannot run this library. The ROS parameter which are defined in launch files will be passed into the library via this function.

Parameters

in	<i>path</i> ↔ —	path to the dlib pose model
in	<i>param</i> ↔ —	ROS params

7.1.4.6 `std::map< std::__cxx11::string, bool > FaceTracking::moveBase ()`

Status of the base frame This function will return whether to whether or not to rotate the base of the robot. Furthermore it will also return which way the robot should turn Here move is true when the camera needs to rotate and the direction tells whether to turn left or right.

Returns

return the move base command (move,direction)

7.1.4.7 `bool FaceTracking::requestDetectedRealTime ()`

Status of the face detection, This value is a boolean variable which return true if a face is present in the current frame.

Returns

return the status of the face detection

7.1.4.8 `cv::RotatedRect FaceTracking::requestEllipseCenter ()`

Center bound of ellipse of the current tracking face will be returned, upon the request from the ROS node.

Returns

return the bound of ellipse of the current tracking face

7.1.4.9 `int FaceTracking::requestFacesSize ()`

Number of close faces.

Returns

return the number of close faces detected

7.1.4.10 `float FaceTracking::RMSE (std::vector< cv::Point > ground_truth, std::vector< cv::Point > fitted_shapes)`
[private]

calculate Root Mean Square Error of the current face and the ground truth here the ground truth will be the values of the previously tracked face

Parameters

in	<i>ground_truth</i>	ground truth face (vector point of the ellipse)
in	<i>fitted_shapes</i>	currently tracked face (vector point of the ellipse)

Returns

return the results from the calculation (float)

```
7.1.4.11 void FaceTracking::startTracking ( cv::Mat & image_, std::vector< dlib::rectangle > & cfaces, cv::vector<
cv::vector< cv::Point >> & all_landmarks_, cv::vector< cv::vector< cv::Point >> & jaw_line, int & max_area_index
) [private]
```

calculate the area of the ellipse and to find which face has the largest area (closest to the camera)

Parameters

in	<i>img_</i>	input image in cv::Mat format
in	<i>cfaces</i>	dlib face rectangle vector of the close face
in	<i>all_landmarks_↵</i> —	get the points of all faces detected by the dlib face detector
in	<i>jaw_line</i>	get the jaw line points (from the 68 points) of all faces detected by the dlib face detector
in	<i>max_area_index</i>	find the face with the maximum area

Remarks

- The input image will be over written with the new information about the faces
- This function also perform the tracking part in this library

```
7.1.4.12 int FaceTracking::Temp ( )
```

```
7.1.4.13 void FaceTracking::trackLandmark ( cv::Mat & input_image_, cv::Mat & output_image_ )
```

Face detection/tracking function This is the main trigger function. Takes input image and returns the detailed tracked image.

Parameters

in	<i>input_↵</i> <i>image_</i>	input image
----	---------------------------------	-------------

Returns

output image

7.1.5 Member Data Documentation

7.1.5.1 `int FaceTracking::cfacesize = 0` [private]

Number of close faces detected

7.1.5.2 `dlib::frontal_face_detector FaceTracking::detector` [private]

Dlib - face landmark detection

7.1.5.3 `bool FaceTracking::face_found` [private]

Detection status

7.1.5.4 `std::vector<cv::Point> FaceTracking::ground_truth` [private]

Ground truth

7.1.5.5 `cv::RotatedRect FaceTracking::minEllipse` [private]

Tracked face ellipse

7.1.5.6 `bool FaceTracking::move_base` [private]

Base status

7.1.5.7 `dlib::shape_predictor FaceTracking::pose_model` [private]

Dlib - pose model for face landmark detection

7.1.5.8 `int FaceTracking::RANGE_FOR_DETECTED` [private]

Detection range

7.1.5.9 `int FaceTracking::RANGE_FOR_TRACKING` [private]

Tracking range

7.1.5.10 `TrackingState FaceTracking::state = Initialize` [private]

Tracking status

7.1.5.11 bool FaceTracking::turnLeft [private]

Base direction value (true if left, false otherwise)

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

- [face_tracking.h](#)
- [face_tracking.cpp](#)

7.2 ROSFaceDetection Class Reference

The ROS wrapper class for Hubert's Face Detection application.

```
#include <face_detection.h>
```

Public Member Functions

- [ROSFaceDetection](#) ()
A Constructor for the class.
- void [execute](#) (void)
The executor node.

Public Attributes

- [FaceTracking](#) [faceTracker](#)

Private Member Functions

- void [imageCallBack](#) (const sensor_msgs::ImageConstPtr &msg)
This callback will grab the image data from the camera.
- void [getEllipseCenter](#) ()
This callback will publish the detected face details.
- void [moveBase](#) ()
This callback will publish move base status.
- void [faceFound](#) ()
This callback will publish the current status of face detection.

Private Attributes

- ros::NodeHandle [nh_](#)
- image_transport::ImageTransport [it_](#)
- image_transport::Subscriber [image_sub_](#)
- image_transport::Publisher [image_pub_](#)
- ros::Publisher [img_location_](#)
- ros::Publisher [move_base_pub_](#)
- ros::Publisher [face_found_pub_](#)
- cv_bridge::CvImagePtr [cv_ptr](#)
- std::map< std::string, int > [params](#)
- int [count_](#) = 0
- bool [currentMoveBase](#) = true
- int [skip_val](#)

7.2.1 Detailed Description

The ROS wrapper class for Hubert's Face Detection application.

Perform face detection and face tracking with the help of an external library [FaceTracking](#) under ROS environment

7.2.2 Constructor & Destructor Documentation

7.2.2.1 ROSFaceDetection::ROSFaceDetection ()

A Constructor for the class.

Will be using to initialize initial values of the user variables and the ROS parameters, i.e Subscribers, Publishers and the lanuch file params

7.2.3 Member Function Documentation

7.2.3.1 void ROSFaceDetection::execute (void)

The executor node.

This node will be call from the ROS NODE main and will start the main ROS thread for publishers and subscribers. Node will run with 10 Hz rate.

7.2.3.2 void ROSFaceDetection::faceFound () [private]

This callback will publish the current status of face detection.

In this callback, the status of the tracking will be published, via the topic `"/face_detection/face_found"`. The function will be checked in every iteration

Remarks

- detected → True
- not detected → False

7.2.3.3 void ROSFaceDetection::getEllipseCenter () [private]

This callback will publish the detected face details.

In this callback, center points of the ellipse center along with its height and the width of the bounding box of the detected face will be published to the topic, `"/face_detection/img_location"`. The data will be in the type of `FaceDetection::Face`

7.2.3.4 void ROSFaceDetection::imageCallBack (const sensor_msgs::ImageConstPtr & msg) [private]

This callback will grab the image data from the camera.

Parameters

<i>msg</i>	sensor_msgs::ImageConstPtr type message
------------	---

In this callback the ROS image data will be converted to OpeCV image(cv::Mat) in order to perform the further processing

Remarks

- The image msg will be converted via cv_bridge http://docs.ros.org/jade/api/cv_bridge/html/c++/cv_bridge_8h.html
- The tracked image, with the details will be published

7.2.3.5 void ROSFaceDetection::moveBase () [private]

This callback will publish move base status.

In this callback, the status of the MoveBase wil be published. This will tell its subscribers to whether to move the base or not and to what direction. It will be publish to the topic face_detection/move_base. The data will be in the type of face_detection::MoveBase

7.2.4 Member Data Documentation**7.2.4.1 FaceTracking ROSFaceDetection::faceTracker**

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

- [face_detection.h](#)
- [face_detection.cpp](#)

Chapter 8

File Documentation

8.1 face_detection.h File Reference

This is the header file of the face_detection/_node.cpp, which was implemented as a ROS wrapper. This ROS node will use the face detection library to perform face detection and face tracking. The tracked face details will then be published to the related topics so that it will grantee a smooth process in the Hubert Brain.

```
#include "ros/ros.h"
#include "ros/package.h"
#include <geometry_msgs/Vector3Stamped.h>
#include <geometry_msgs/PointStamped.h>
#include <std_msgs/Int32.h>
#include <std_msgs/String.h>
#include <std_msgs/Bool.h>
#include "face_tracking.h"
#include <sensor_msgs/Image.h>
#include <sensor_msgs/image_encodings.h>
#include <cv_bridge/cv_bridge.h>
#include <image_transport/image_transport.h>
#include <opencv2/imgproc/imgproc.hpp>
#include "face_detection/MoveBase.h"
#include "face_detection/Face.h"
```

Classes

- class [ROSFaceDetection](#)

The ROS wrapper class for Hubert's Face Detection application.

8.1.1 Detailed Description

This is the header file of the face_detection/_node.cpp, which was implemented as a ROS wrapper. This ROS node will use the face detection library to perform face detection and face tracking. The tracked face details will then be published to the related topics so that it will grantee a smooth process in the Hubert Brain.

Author

Fredrik Lagerstedt
Zhanyu Tuo
Terje Stenstrom
Nipun C. Gammanage

Date

Initial release - October/2019

8.2 face_tracking.h File Reference

This is the header file of the [face_tracking.cpp](#), which was implemented to detect faces, track the closest face. This method is also called as the First Landmark Tracking, as it will focus only on the first person who enters the screen or if many enters the frame then the closest person (by analysing the area of the bounding box)

```
#include <opencv/cv.h>
#include <opencv2/opencv.hpp>
#include <dlib/opencv.h>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <dlib/image_processing/frontal_face_detector.h>
#include <dlib/image_processing/render_face_detections.h>
#include <dlib/image_processing.h>
#include <dlib/gui_widgets.h>
#include <cstdio>
#include <ctime>
#include <iostream>
#include <string>
```

Classes

- class [FaceTracking](#)
A library to handle face detection tracking and face tracking.

Namespaces

- [cv](#)

8.2.1 Detailed Description

This is the header file of the [face_tracking.cpp](#), which was implemented to detect faces, track the closest face. This method is also called as the First Landmark Tracking, as it will focus only on the first person who enters the screen or if many enters the frame then the closest person (by analysing the area of the bounding box)

Author

Fredrik Lagerstedt
Zhanyu Tuo
Terje Stenstrom
Nipun C. Gammanage

Date

Initial release - October/2019

8.3 face_detection.cpp File Reference

```
#include "face_detection.h"
```

8.4 face_detection_node.cpp File Reference

```
#include "face_detection.h"
```

Functions

- int [main](#) (int argc, char **argv)

8.4.1 Function Documentation

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

8.5 face_tracking.cpp File Reference

```
#include "face_tracking.h"
```


Index

- ~FaceTracking
 - FaceTracking, [17](#)
- calculateArea
 - FaceTracking, [17](#)
- cfacesize
 - FaceTracking, [20](#)
- count_
 - General variables, [11](#)
- currentMoveBase
 - General variables, [11](#)
- cv, [13](#)
- cv_ptr
 - ROS related variables, [9](#)
- detector
 - FaceTracking, [21](#)
- execute
 - ROSFaceDetection, [23](#)
- face_detection.cpp, [27](#)
- face_detection.h, [25](#)
- face_detection_node.cpp, [27](#)
 - main, [27](#)
- face_found
 - FaceTracking, [21](#)
- face_found_pub_
 - ROS related variables, [9](#)
- face_tracking.cpp, [27](#)
- face_tracking.h, [26](#)
- faceFound
 - ROSFaceDetection, [23](#)
- faceTracker
 - ROSFaceDetection, [24](#)
- FaceTracking, [15](#)
 - ~FaceTracking, [17](#)
 - calculateArea, [17](#)
 - cfacesize, [20](#)
 - detector, [21](#)
 - face_found, [21](#)
 - FaceTracking, [17](#)
 - getCloseFaces, [17](#)
 - getMaxAreaIndex, [18](#)
 - getTrackingIndex, [18](#)
 - ground_truth, [21](#)
 - Initialize, [16](#)
 - initialize, [18](#)
 - minEllipse, [21](#)
 - move_base, [21](#)
 - moveBase, [19](#)
 - pose_model, [21](#)
 - RANGE_FOR_DETECTED, [21](#)
 - RANGE_FOR_TRACKING, [21](#)
 - RMSE, [19](#)
 - requestDetectedRealTime, [19](#)
 - requestEllipseCenter, [19](#)
 - requestFacesSize, [19](#)
 - startTracking, [20](#)
 - state, [21](#)
 - Temp, [20](#)
 - trackLandmark, [20](#)
 - Tracking, [16](#)
 - TrackingState, [16](#)
 - turnLeft, [21](#)
 - Updating, [16](#)
- General variables, [11](#)
 - count_, [11](#)
 - currentMoveBase, [11](#)
 - params, [11](#)
 - skip_val, [11](#)
- getCloseFaces
 - FaceTracking, [17](#)
- getEllipseCenter
 - ROSFaceDetection, [23](#)
- getMaxAreaIndex
 - FaceTracking, [18](#)
- getTrackingIndex
 - FaceTracking, [18](#)
- ground_truth
 - FaceTracking, [21](#)
- image_pub_
 - ROS related variables, [9](#)
- image_sub_
 - ROS related variables, [9](#)
- imageCallBack
 - ROSFaceDetection, [23](#)
- img_location_
 - ROS related variables, [10](#)
- Initialize
 - FaceTracking, [16](#)
- initialize
 - FaceTracking, [18](#)
- it_
 - ROS related variables, [10](#)
- main
 - face_detection_node.cpp, [27](#)

- minEllipse
 - FaceTracking, 21
- move_base
 - FaceTracking, 21
- move_base_pub_
 - ROS related variables, 10
- moveBase
 - FaceTracking, 19
 - ROSFaceDetection, 24
- nh_
 - ROS related variables, 10
- params
 - General variables, 11
- pose_model
 - FaceTracking, 21
- RANGE_FOR_DETECTED
 - FaceTracking, 21
- RANGE_FOR_TRACKING
 - FaceTracking, 21
- RMSE
 - FaceTracking, 19
- ROS related variables, 9
 - cv_ptr, 9
 - face_found_pub_, 9
 - image_pub_, 9
 - image_sub_, 9
 - img_location_, 10
 - it_, 10
 - move_base_pub_, 10
 - nh_, 10
- ROSFaceDetection, 22
 - execute, 23
 - faceFound, 23
 - faceTracker, 24
 - getEllipseCenter, 23
 - imageCallBack, 23
 - moveBase, 24
 - ROSFaceDetection, 23
- requestDetectedRealTime
 - FaceTracking, 19
- requestEllipseCenter
 - FaceTracking, 19
- requestFacesSize
 - FaceTracking, 19
- skip_val
 - General variables, 11
- startTracking
 - FaceTracking, 20
- state
 - FaceTracking, 21
- Temp
 - FaceTracking, 20
- trackLandmark
 - FaceTracking, 20
- Tracking
 - FaceTracking, 16
- TrackingState
 - FaceTracking, 16
- turnLeft
 - FaceTracking, 21
- Updating
 - FaceTracking, 16