Industrial robots for waste separation task using computer vision

v1.0

Generated by Doxygen 1.8.6

Mon Dec 25 2017 22:50:42

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 ImageConverter Class Reference

A class dedicated to do ROS data management.

`#include <bridge.h>`

Public Member Functions

- ImageConverter ()

  Class constructor.
- ~ImageConverter ()

  Class destructor.
- void lenny_asks_callback (const std_msgs::UInt32 &msg)

  Member that waits for confirmation data to execute code.
- void kinect_color_callback (const sensor_msgs::ImageConstPtr &color_msg)

  Member that subscribes the node to the RGB ros data.
- void kinect_depth_callback (const sensor_msgs::ImageConstPtr &depth_msg)

  Member that subscribes the node to the depth ros data.
- void computer_vision (std::vector< std::vector< double > > &data_to_ros)

  Member that executes the computer vision algorithm.
- void publisher (std::vector< std::vector< double > > &data_to_ros)

  Member that publishes to ros the data found by the computer vision algorithm.
- std::vector< double > selector (std::vector< std::vector< double > > data_to_ros)

  Member that select the data that its going to be published to ros.

Public Attributes

- TransformListenerPtr transform_listener_ptr

  A pointer to manage ROS communication.

3.1.1 Detailed Description

A class dedicated to do ROS data management.

This class manage ROS image-raw data convertion into OpenCV Mat data structure. Also, it stablishes the communication between different ROS nodes to process image data into spatial coordinates

3.1.2 Constructor & Destructor Documentation

3.1.2.1 ImageConverter::ImageConverter ( )

Class constructor.

3.1.2.2 ImageConverter::~ImageConverter ( )

Class destructor.

3.1.3 Member Function Documentation

3.1.3.1 void ImageConverter::computer_vision (std::vector< std::vector< double > > & *data_to_ros*)

Member that executes the computer vision algorithm.

Returns the data_to_ros vector that is a spatial coordinates frame in XYZ and quaternions among other data of interest for the system.

See Also

computer_vision, kinect_depth_callback, kinect_color_callback, lenny_asks_callback

3.1.3.2 void ImageConverter::kinect_color_callback (const sensor_msgs::ImageConstPtr & *color_msg*)

Member that subscribes the node to the RGB ros data.

Subscribe the node to rgb ros data and translates it into BGR openCV data

Parameters

color_msg	as a sensor_msgs::ImageConstPtr that recieves color data from the sensor.

3.1.3.3 void ImageConverter::kinect_depth_callback (const sensor_msgs::ImageConstPtr & *depth_msg*)

Member that subscribes the node to the depth ros data.

Subscribe the node to depth ros data and translates it into depth openCV data

Parameters

depth_msg	as a sensor_msgs::ImageConstPtr that recieves color data from the sensor.

3.1.3.4 void ImageConverter::lenny_asks_callback (const std_msgs::UInt32 & *msg*)

Member that waits for confirmation data to execute code.

Starts the image processing.

Parameters

msg	as a std::msgs::UInt32 message from ROS that is 1 when the robot sends the execution confirmation.

3.1.3.5 void ImageConverter::publisher (std::vector< std::vector< double > > & *data_to_ros*)

Member that publishes to ros the data found by the computer vision algorithm.

Publishes the spatial coordinates data to ros and other information to the system.

Parameters

data_to_ros	as a vector<double> that stores the information of the detection

See Also

selector

3.1.3.6 std::vector< double > ImageConverter::selector (std::vector< std::vector< double > > *data_to_ros*)

Member that select the data that its going to be published to ros.

Select the data to publish

Parameters

data_to_ros	as a vector<vector<double> > that are all the detections found

Returns

A vector of data that actually is going to be published to ros

3.1.4 Member Data Documentation

3.1.4.1 TransformListenerPtr ImageConverter::transform_listener_ptr

A pointer to manage ROS communication.

Pointer that allows ROS communication of space transformations

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

- include/bridge.h
- src/bridge.cpp

3.2 polycolor Class Reference

A class dedicated to the image processing based in openCV.

`#include <detection.h>`

Public Member Functions

- polycolor ()

  Class constructor.
- cv::Mat getwhite (cv::Mat hsv, int s, int v)

  Member that gets white detections based on a HSV image.
- cv::Mat getred (cv::Mat h, int r1, int r2)

  Member that gets red detections based on a one channel image.
- cv::Mat getblue (cv::Mat h, int a1, int a2)

  Member that gets blue detections based on a one channel image.
- cv::Mat getgreen (cv::Mat h, int g1, int g2)

  Member that gets green detections based on a one channel image.
- cv::Mat getsaturation (cv::Mat s, int s0, int s1, int s2)

  Member that gets saturation level detections based on a one channel image.
- cv::Mat getvalue (cv::Mat v, int v1, int v2)

  Member that gets value level detections based on a one channel image.
- cv::Mat findcolors (cv::Mat maskwhite, cv::Mat maskred, cv::Mat maskgreen, cv::Mat maskblue, cv::Mat maskvalue)

  Member that finds color bottles except green ones and white.
- cv::Mat findgreen (cv::Mat maskgreen, cv::Mat maskwhite, cv::Mat maskvalue)

  Member that finds green bottles.
- cv::Mat findwhite (cv::Mat maskwhite, cv::Mat maskvalue)

  Member that finds white bottles.
- std::vector< std::vector < cv::Point > > findcontours (cv::Mat proc, int cmin, int cmax, int max_detection)

  Member that finds contours of bottles.
- cv::Mat drawcontors (cv::Mat image, std::vector< std::vector< cv::Point > > contornos, int r, int g, int b)

  Member that draws fitted rectangles to contours of bottles.

3.2.1 Detailed Description

A class dedicated to the image processing based in openCV.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 polycolor::polycolor ( )

Class constructor.

3.2.3 Member Function Documentation

3.2.3.1 cv::Mat polycolor::drawcontors (cv::Mat *image*, std::vector< std::vector< cv::Point > > *contornos*, int *r*, int *g*, int *b*)

Member that draws fitted rectangles to contours of bottles.

This function draws fitted rectangles and ellipses in each contour.

Parameters

contornos	as a vector<vector<cv::Point> > that stores the contour data
r	red intensity of the line to draw (255-r)
g	red intensity of the line to draw
b	red intensity of the line to draw (255-b)

Returns

a cv::Mat RGB image whith the fitted object draw to the original image.

See Also

findcontours

3.2.3.2 cv::Mat polycolor::findcolors (cv::Mat *maskwhite*, cv::Mat *maskred*, cv::Mat *maskgreen*, cv::Mat *maskblue*, cv::Mat *maskvalue*)

Member that finds color bottles except green ones and white.

This function finds color bottles based on different masks for each subtype. The cv::Mat arguments are all binary images.

Parameters

maskwhite	as a cv::Mat white detections
maskred	as a cv::Mat red detections
maskgreen	as a cv::Mat green detections
maskblue	as a cv::Mat blue detections
maskvalue	as a cv:Mat acceptable value levels

Returns

Binary cv::Mat image of HDPE-colored detected bottles

See Also

getvalue, getwhite, getred, getgreen, getblue

3.2.3.3 vector< vector< cv::Point > > polycolor::findcontours (cv::Mat *proc*, int *cmin*, int *cmax*, int *max_detection*)

Member that finds contours of bottles.

This function finds the real contours of the detected objects.

Parameters

proc	as a cv::Mat with the detected objects
cmin	min contours size that is acceptable
cmax	max contours size that is acceptable
max_detection	max number of contours allowed

Returns

A vector of contours detected

See Also

findcolors, findwhite, findgreen, findcolors

3.2.3.4 cv::Mat polycolor::findgreen (cv::Mat *maskgreen,* cv::Mat *maskwhite,* cv::Mat *maskvalue*)

Member that finds green bottles.

This function finds green bottles based on different masks for each subtype. The cv::Mat arguments are all binary images.

Parameters

maskgreen	as a cv::Mat green detections
maskwhite	as a cv::Mat white detections
maskvalue	as a cv:Mat acceptable value levels

Returns

Binary cv::Mat image of PET-colored detected bottles

See Also

getgreen, getvalue

3.2.3.5 cv::Mat polycolor::findwhite (cv::Mat *maskwhite,* cv::Mat *maskvalue*)

Member that finds white bottles.

This function finds white bottles based on different masks for each subtype. The cv::Mat arguments are all binary images.

Parameters

maskwhite	as a cv::Mat white detections
maskvalue	as a cv:Mat acceptable value levels

Returns

Binary cv::Mat image of HDPE detected bottles

See Also

getwhite, getvalue

3.2.3.6 cv::Mat polycolor::getblue (cv::Mat *h*, int *a1*, int *a2*)

Member that gets blue detections based on a one channel image.

This function finds red detections based on one-channel image (hue)

Parameters

h	as a cv::Mat. One-channel image. Hue channel
a1	min hue level
a2	max hue level

Returns

Binary cv::Mat image of detected pixels

3.2.3.7 cv::Mat polycolor::getgreen (cv::Mat *h*, int *g1*, int *g2*)

Member that gets green detections based on a one channel image.

This function finds green detections based on one-channel image (hue)

Parameters

h	as a cv::Mat. One-channel image. Hue channel
g1	min hue level
g2	max hue level

Returns

Binary cv::Mat image of detected pixels

3.2.3.8 cv::Mat polycolor::getred (cv::Mat *h*, int *r1*, int *r2*)

Member that gets red detections based on a one channel image.

This function finds red detections based on one-channel image (hue)

Parameters

h	as a cv::Mat. One-channel image. Hue channel
r1	min hue level
r2	max hue level

Returns

Binary cv::Mat image of detected pixels

3.2.3.9 cv::Mat polycolor::getsaturation (cv::Mat *s*, int *s0*, int *s1*, int *s2*)

Member that gets saturation level detections based on a one channel image.

This function finds saturation detections based on one-channel image (saturation)

Parameters

s	as a cv::Mat. One-channel image. Saturation channel
s0	min saturation level
s1	max saturation level
s2	min saturation level to establish other limit (s0=s2 if s2 is not necessary)

Returns

Binary cv::Mat image of detected pixels

3.2.3.10 cv::Mat polycolor::getvalue (cv::Mat *v*, int *v1*, int *v2*)

Member that gets value level detections based on a one channel image.

This function finds value detections based on one-channel image (value)

Parameters

v	as a cv::Mat. One-channel image. Value channel
v1	min saturation level
v2	max saturation level

Returns

Binary cv::Mat image of detected pixels

3.2.3.11 cv::Mat polycolor::getwhite (cv::Mat *hsv*, int *s*, int *v*)

Member that gets white detections based on a HSV image.

This function finds white detections based on saturation and value levels

Parameters

hsv	as a cv::Mat. The image in HSV color space.
s	max saturation
v	min value

Returns

Binary cv::Mat image of detected pixels

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

- include/detection.h
- src/polycolor.cpp

3.3 space Class Reference

A class dedicated to pose estimation.

`#include <perception.h>`

Public Member Functions

- space ()

  Class constructor.
- std::vector< cv::Point > getpoints (std::vector< std::vector< cv::Point > > contornos)

  Member that finds the object centroid.
- std::vector< cv::Point > scale (std::vector< cv::Point > Points, int originalcols, int originalrows, int tinycols, int tinyrows)

  Member that finds the corresponding Point in the depth image of the data found in the rgb image.
- std::vector< std::vector < double > > find_depth (std::vector< cv::Point > Points_scaled, std::vector< cv::Point > Points_real, cv::Mat depth)

  Member that finds the depth of and object using kinect's depth data.
- std::vector< std::vector < double > > xyz_coord (std::vector< std::vector< double > > Points, double fx, double fy, double cx, double cy)

  Member that takes the mixed-coordinate frame and translates it to a real coordinate system.
- std::vector< std::vector < double > > getgeometrydata (std::vector< std::vector< double > > kinect_xyz, std::vector< std::vector< cv::Point > > contornos, int bottle_type, double fx, double fy, double z)

  Member that gathers all the information of the detected object.
- std::vector< double > getquaternion (double roll, double pitch, double yaw)

  Member that do a transformation in the way the orientation of an object is described.
- std::vector< std::vector < double > > push_data (std::vector< std::vector< double > > color, std::vector< std::vector< double > > green, std::vector< std::vector< double > > white)

  Member that gathers all the information in a vector ready to be send through ROS.
- void pusher (std::vector< std::vector< double > > &A, std::vector< std::vector< double > > &data_to_ros)

  Member that actually push the different data in a single vector.

3.3.1 Detailed Description

A class dedicated to pose estimation.

This class is focused in pose estimation, with a contour detected this class allows the user to estimate its 3D pose using the depth data from a kinect sensor and the rgb image.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 space::space ( )

Class constructor.

3.3.3 Member Function Documentation

3.3.3.1 vector< vector< double > > space::find_depth (std::vector< cv::Point > *Points_scaled,* std::vector< cv::Point > *Points_real,* cv::Mat *depth*)

Member that finds the depth of and object using kinect's depth data.

This funtion uses a point of an image (pixel location) to find its corresponding depth data in other image. Then it gather the pixel location and the depth data in a mixed-coordinate frame, where P(x,y) is the position of the pixel in the image and z is the real depth of the object.

Parameters

Points_scaled	a vector of cv::Point elements that are the location of the pixels of interest.
Points_real	a vector of cv::Point with the non-scaled (x,y) points.
depth	a cv::Mat that stores the depth data.

Returns

a vector<vector<double> > That stores the 3D point P(x,y,z) which x,y are in pixel coordinates and z in real ones.

See Also

getpoints, scale

3.3.3.2 vector< vector< double > > space::getgeometrydata (std::vector< std::vector< double > > *kinect_xyz*, std::vector< std::vector< cv::Point > > *contornos*, int *bottle_type*, double *fx*, double *fy*, double *z*)

Member that gathers all the information of the detected object.

This function gathers the pose estimation information and other geometry data preparing it to send in a structured ROS message. It sends (X,Y,Z) location,(x,y,z,w) rotation,width,length and bottle type.

Parameters

kinect_xyz	as a vector<vector<double> > that stores the real coordinates in (x,y,z) of the centroid of the object.
contornos	as a vector<vector<cv::Point> > which are the contours of the objects.
bottle_type	a int that is 1 when colored-HDPE, 2 when colored-PET and 3 when HDPE.
fx	focal length in x-axis expressed in pixels
fy	focal lenght in y-axis expressed in pixels
cx	Point in x-axis of the origin of the image
cy	Point in y-axis of the origin of the image

Returns

a vector<vector<double> > with all the characteristics and the information of the detected objects.

See Also

getpoints, find_depth, push_data

3.3.3.3 vector< cv::Point > space::getpoints (std::vector< std::vector< cv::Point > > *contornos*)

Member that finds the object centroid.

This function calculates moments to estimate the centroid (in pixels) of the object detected

Parameters

contornos	a vector<vector<cv::Point> > its a vector containing a group

Returns

A vector<cv::Point> that store the centroids of the objects detected

3.3.3.4 vector< double > space::getquaternion (double *roll*, double *pitch*, double *yaw*)

Member that do a transformation in the way the orientation of an object is described.

This function convert euler angles (roll, pitch and yaw) in quaternion.

Parameters

roll	roll
pitch	pitch
yaw	yaw

Returns

a vector<double> which contains (x,y,z,w)

See Also

getgeometrydata

3.3.3.5 vector< vector< double > > space::push_data (std::vector< std::vector< double > > *color*, std::vector< std::vector< double > > *green*, std::vector< std::vector< double > > *white*)

Member that gathers all the information in a vector ready to be send through ROS.

This function gathers all the information of each type of bottle to create a vector with all the available data.

Parameters

color	is the getgeometrydata of color bottles
green	is the getgeometrydata of green bottles
white	is the getgeometrydata of white bottles

Returns

a vector<vector<double> > with all the data of the detected objects

See Also

getgeometrydata, pusher

3.3.3.6 void space::pusher (std::vector< std::vector< double > > & *A*, std::vector< std::vector< double > > & *data_to_ros*)

Member that actually push the different data in a single vector.

This function pushes the data of the different bottles into a vector that stores the information to be send to ROS

Parameters

A	is the getgeometrydata vector
data_to_ros	is the vector where the information is being gathered.

3.3.3.7 vector< cv::Point > space::scale (std::vector< cv::Point > *Points*, int *originalcols*, int *originalrows*, int *tinycols*, int *tinyrows*)

Member that finds the corresponding Point in the depth image of the data found in the rgb image.

This function matches the pixels of the rgb image centroids to the depth image pixels so the data can be compared between the both images.

Parameters

Points	a vector<cv::Points> that stores the original points of the rbg image that contains the centroids of the objects
originalcols	a integer number of columns of the rgb image
originalrows	a integer number of rows of the rgb image
tinycols	a integer number of columns of the depth image
tinyrows	a integer number of rows of the depth image

Returns

a vector<cv::Point> that stores the pixels of the centroids matched to the depth image

See Also

getpoints

3.3.3.8 vector< vector< double > > space::xyz_coord (std::vector< std::vector< double > > *Points*, double *fx*, double *fy*, double *cx*, double *cy*)

Member that takes the mixed-coordinate frame and translates it to a real coordinate system.

This function takes the mixed-coordinate frame and convert it to a real coordinate frame with the kinect sensor as the reference point

Parameters

Points	a vector<vector<double> > of (x,y,z) points in a mixed-coordinate system
fx	focal length in x-axis expressed in pixels
fy	focal lenght in y-axis expressed in pixels
cx	Point in x-axis of the origin of the image
cy	Point in y-axis of the origin of the image

Returns

a vector<vector<double > > That stores the 3D point (x,y,z) in the real space coordinates frame whith the sensor as reference point.

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

- include/perception.h
- src/space.cpp

3.4 test Class Reference

A class that is used when the framework is tested.

`#include <test.h>`

Public Member Functions

- test ()

  A class constructor.
- std::vector< double > get_data (int cols, int rows, std::vector< std::vector< cv::Point > > contornos, int band)

  Member that organize the data to be printed to a file.
- std::vector< double > get_data_to_print (std::vector< double > color, std::vector< double > green, std::vector< double > white, int image, int band)

  Member that gathers the information of all bottle types and print the data to a file.

3.4.1 Detailed Description

A class that is used when the framework is tested.

This class is used when the computer vision algorithm is tested in a loop mode to analize a library of images.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 test::test ( )

A class constructor.

Class constructor

3.4.3 Member Function Documentation

3.4.3.1 vector< double > test::get_data (int *cols*, int *rows*, std::vector< std::vector< cv::Point > > *contornos*, int *band*)

Member that organize the data to be printed to a file.

3.4.3.2 vector< double > test::get_data_to_print (std::vector< double > *color*, std::vector< double > *green*, std::vector< double > *white*, int *image*, int *band*)

Member that gathers the information of all bottle types and print the data to a file.

This function gathers the information of all bottle types into a single data line making it easy to compare with a database.

Parameters

color	the get_data of color bottles (colored HDPE)
green	the get_data of green bottles (colored PET)

white	the get_data of white bottles (HDPE)
band	a int that is used as a flag. 1 to detection analisis 2 to pose estimation analisis.

Returns

a vector<double> whith the line of data that will be saved in a file

See Also

get_data

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

- include/test.h
- src/test.cpp

Chapter 4

File Documentation

4.1 include/bridge.h File Reference

```
#include <ros/ros.h>
#include <image_transport/image_transport.h>
#include <cv_bridge/cv_bridge.h>
#include <sensor_msgs/image_encodings.h>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <vision/bottle_data.h>
#include <std_msgs/UInt32.h>
#include <tf/transform_listener.h>
#include <tf_conversions/tf_eigen.h>
#include <geometry_msgs/TransformStamped.h>
#include <geometry_msgs/PoseStamped.h>
#include <fstream>
#include <string>
#include <ctime>
```

Classes

- class ImageConverter

  A class dedicated to do ROS data management.

Typedefs

- typedef boost::shared_ptr
  < tf::TransformListener > TransformListenerPtr

  A pointer to manage ROS communication.

4.1.1 Typedef Documentation

4.1.1.1 typedef boost::shared_ptr<tf::TransformListener> TransformListenerPtr

A pointer to manage ROS communication.

Pointer that allows ROS communication of space transformations

4.2 include/detection.h File Reference

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
#include <math.h>
```

Classes

- class polycolor

  A class dedicated to the image processing based in openCV.

4.3 include/perception.h File Reference

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
#include <math.h>
```

Classes

- class space

  A class dedicated to pose estimation.

4.4 include/test.h File Reference

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
#include <math.h>
```

Classes

- class test

  A class that is used when the framework is tested.

4.5 src/bridge.cpp File Reference

```
#include "../include/bridge.h"
#include "../include/perception.h"
#include "../include/detection.h"
#include <boost/assign/std/vector.hpp>
```

Variables

- int k_roll =1

  This variable controls the communication through ROS.
- std::ofstream myfile

  This ofstream creates a file to store data to evaluate the framework.

4.5.1 Variable Documentation

4.5.1.1 int k_roll =1

This variable controls the communication through ROS.

k_roll is a int variable that manages the communication between the robot and the computer vision system through ROS. It is 1 when the system is ready 0 when the robot is executing some other task.

4.5.1.2 std::ofstream myfile

This ofstream creates a file to store data to evaluate the framework.

It is a file where registers are stored to analize the performance of the framework.

4.6 src/detector.cpp File Reference

```
#include "../include/perception.h"
#include "../include/detection.h"
```

Functions

- int main (int argc, char ∗∗argv)

  This function executes the computer vision code when it is used stand-alone mode.

Variables

- double fx =1.0466649972613395e+03

  int fx focal length in x-axis expressed in pixels
- double fy =1.0466649972613395e+03

  int fy focal lenght in y-axis expressed in pixels

- double cx =640.0

  int cx Point in x-axis of the origin of the image
- double cy =512.0

  int cy Point in y-axis of the origin of the image

4.6.1 Function Documentation

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

This function executes the computer vision code when it is used stand-alone mode.

This function executes the computer vision code when it is used stand-alone mode

4.6.2 Variable Documentation

4.6.2.1 double cx =640.0

int cx Point in x-axis of the origin of the image

Camera matrix's Point in x-axis of the origin of the image

4.6.2.2 double cy =512.0

int cy Point in y-axis of the origin of the image

Camera matrix's Point in y-axis of the origin of the image

4.6.2.3 double fx =1.0466649972613395e+03

int fx focal length in x-axis expressed in pixels

Camera matrix's focal length in x-axis expressed in pixels

4.6.2.4 double fy =1.0466649972613395e+03

int fy focal lenght in y-axis expressed in pixels

Camera matrix's focal length in y-axis expressed in pixels

4.7 src/image_converter.cpp File Reference

```
#include "../include/bridge.h"
#include "../include/perception.h"
#include "../include/detection.h"
```

Functions

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

  This function launches the ros node that perform the image processing task.

4.7.1 Function Documentation

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

This function launches the ros node that perform the image processing task.

This main launch a ros node that manage the ROS-OpenCV communication and conversions, then it executes the computer vision algorithm and send the data back to ROS

4.8 src/kinect_to_robot.cpp File Reference

```
#include <ros/ros.h>
#include <tf2_ros/transform_broadcaster.h>
#include <tf2/LinearMath/Quaternion.h>
```

Functions

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

  This function launches the ros node that manage spatial static transformations.

4.8.1 Function Documentation

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

This function launches the ros node that manage spatial static transformations.

This main launch a ros node that manage spatial transformation between the camera's frame and the robot's one. This node perform a "spatial-geometric communication between both systems through the framework.

4.9 src/polycolor.cpp File Reference

```
#include "../include/detection.h"
```

4.10 src/space.cpp File Reference

```
#include "../include/perception.h"
```

4.11 src/test.cpp File Reference

```
#include "../include/test.h"
```