Final report

Team Gamma

Ajda Frankovič, Martin Preradović, Niki Bizjak

1 Introduction

2 Methods

In this section, the algorithms and methods we have used in our system are presented.

2.1 Camera pixel to world position

Object detectors must be able to compute world positions of detected objects. To do that, `rgb` camera and depth images are used. Because the robot is moving, and the environment is changing very fast, we must make sure that the received colour and depth image are synchronized with one another. Even a small delay between images can cause computed world positions to be inaccurate.

Figure 1: Pinhole camera model

When object is detected on `rgb` colour image, the pixel position on image plane is known. Using camera calibration matrix, we can obtain information about focal length and image format. Focal length is the distance from the centre of camera coordinate system to image plane. The image format gives us information about camera sensor width and height.

Using this information, the ray from the centre of camera coordinate system to the detected object pixel on image plane can be created. Using depth image information, we can obtain the distance from camera centre to the

detected object in space. Using the computed ray and distance to the object, we can compute its position in three-dimensional space in camera frame.

Using frame transformations, we can then compute the detected object position in world coordinate system using transformation matrix.

In our system, the face and ring positions are computed this way. The cylinder detection works with point clouds, where the points are already computed in the world frame.

2.2 Faces

In this section, we present algorithms that we have used to detect faces, compute their orientation in space and classify them.

2.2.1 Face detection

Face detection was done using Viola-Jones face detection algorithm. It was chosen because it can be run in real-time and it has a high detection rate.

2.2.2 Face orientation computation

After the face has been detected in image, the position in world coordinate system is computed. The approaching point is then calculated using static map information. Approaching point is a point directly in front of the face that the robot must visit.

To compute approaching point, we must first get the face orientation. To do that, the static map information is used. The calculated face position in world coordinates is first converted to map coordinates. Then, the corresponding pixel in map is calculated.

Because of depth sensor inaccuracy, the computed map pixel coordinate doesn't always lie on the wall. In figure 2, computed face pixel is coloured red.

Using a simple breadth first search in face pixel position proximity, the closest wall pixel is determined. In figure 2, this pixel is also coloured red.

After wall pixel has been found, the Hough line finding algorithm is applied to an area around wall pixel. The detected lines in figure 2 are rep-
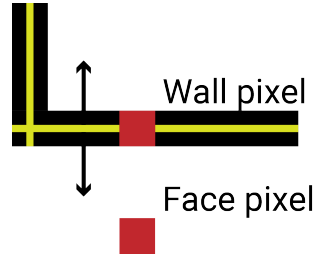
Figure 2: Face orientation computation

resented with green colour. If more than one line is found, the line which is closest to the wall pixel is selected. After the line is found, there are two possible orientations as represented in figure 2. The vector that is pointing in the direction of the robot is chosen to be the face orientation.

A similar algorithm is used to detect ring orientation.

2.2.3 Face classification

After faces are detected and localized in the world, the faces are identified.

2.3 Colour classification

The robot must be able to detect ring and cylinder colours. There are six possible classes: black, white, red, green, blue and yellow.

In homework 2, we tested different colour spaces and classification models and concluded that the k-nearest neighbours algorithm worked best. The colour space with the highest accuracy in homework was `HSV` colour model, but in the Gazebo simulator, `RGB` space worked better.

So the colour classifier in the final task uses the k-nearest neighbours algorithm and takes in an input vector in $(red, green, blue)$ format and returns a colour label.

Because of the non-even lighting in the Gazebo simulator (and probably real world too), the classifier sometimes returns incorrect label. The problem was solved using the robustification process as described in the 2.6 section. Colour classifier is run multiple times on different object detections and the most frequent colour is chosen as the object colour.

2.4 Rings

In this section, the algorithm for ring detection is presented.

2.4.1 Ring detection

2.4.2 Ring color detection

2.5 Cylinder detection

2.5.1 Removal of planes

2.5.2 Cylinder detection

2.5.3 Approaching point computation

2.6 Robustification

The sensor data that the robot receives is not always accurate. When object is detected, the depth sensor might have computed the distance inaccurately or the robot was not localized. The detectors might also return false positives and so on.

To combat this, a robustification is performed on detected objects. The object detectors are sending detections to robustifier, which collects data and determines if the detection is a true positive or not.

For detection to be considered true positive, the following must hold:

1. Detected object should be detected at least `threshold` times

2. There must be at least `minimum distance` meters between this detection and every other previous detection.

If two detections are close enough (the distance is less than `minimum distance`), they are considered the same object. The world position and approaching point position of those two detections are averaged. So the object position is closer to its actual position in space.