Exam 4 Documentation

Group Nr:

Names: Patrick Hoffmann, Sebastian Horngacher, Martin Schnöller

Examination 1:

Bug1 and Dist Bug:

Input: clientID for the API, the handles for the range sensors and the name of the goal.

Output: “FINISHED” if the goal is reached, else returns FALSE

DetectCorner:

Input: ClientID of the API, the handles of the range sensors, the index of a reference ray and a reference distance.

Output: true if there is a corner, else false

Remark:

The function checks the distance of the range sensors around the reference ray, if one of these distances is greater than the reference distance we know there is a corner.

FollowBoundry

Input: ClientID of the API, the range sensorHandles and rightSide to see if the bounty is on the right side.

Output: the current range to the wall, if a leaving Condition accrues “follow boundary end”, if a corner is detected “drive around corner”

isChairInFront:

Input: clientId of the api and range sensor handles:

Output: true if chair in front else false

Remark:

We check the average distances of the rays that are in front of the robot, if the avg distance is greater then a reference value, we know it is not a corner and has to be a chair.

wallOrient:

Input: the ClientID of the Api, the handles of the range sensors, the index of the ray which hit the wall first, and if the uBot is in followBoundry mode

Remark:

The function takes two rays and sees them as vectors, we calculate the vector that connects the two rays -> this new vector has the same angle as the wall.

The Other Functions use Basic knowledge or information given in the slides, so we won’t mention them here.

Examination 2:

calcCenter/ getBottom:

Input: Both functions take contours of a object as input

Outubut: calcCenter returns the masscenter of the object, getBottom returns the bottom extreme point of the object

Remark: both functions use cv2.momments to calculate the results

getContures:

Input: a image in cv2 format, colour boundaries

Output: all contours of objects in the given colour

Remark:

We use a mask to eliminate all colours that are not in the boundaries. With the new image cv2 can calculate the contours.

FindAllBlobs:

Input: clientId of the Api, the camera handles and the homography matrix

Output: Bottom most point of all the blobs in the scene

Remark:

Calculates the contours for all colours and uses the information to get the bottom most point.

AStar class:

We adapted a A Star algorithm we found in the internet to reach the goal in the scene.   
The main problem was that the coordinates we got were not the perfectly cantered, there for the min distance the A star algo needs to each obstacle was greater than the distance possible with our coordinates -> the robot was in a endless loop trying to find a path

When we lowered the min Distance to robot was to big and would ran over some obstacles. The solution was that we subtracted an offset from the coordinates to centre them.

Examination 3:

getArmPos:

Input: height of the point to reach and distance away from the robot

Output: angles for first, second and third arm joint to reach a position.

Remark:

Uses cos sentence to get the angles, the third arm Part Is always in the same angle relative to the ground.