Lab 6: Sonar (Ultrasonic Range Finders) and Camera Sensors

*ECE 564: Fundamentals of Autonomous Robots Lab*

*Team 1: Jacob Cassady, Chase Crutcher, and Olalekan Olakitan Olowo*

*October 19, 2019*

*The group members have worked together and face-to-face at all stages of this project work. The contributions of members to the report and to the codes are equal.*

*(Initials of group members)*

# Introduction

# Lab Parts

## Determining the calibration coefficient and minimum sensing distance of MB1010 sonar sensor

A close up of a map

Description automatically generated

Figure : Linear Regression for Calibration Coefficients

## Rotational tracking using Camera

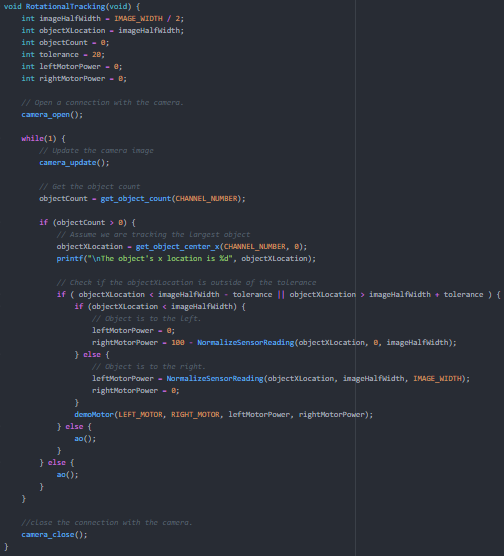


Figure : Rotational Tracking Function

## Translational tracking using Camera

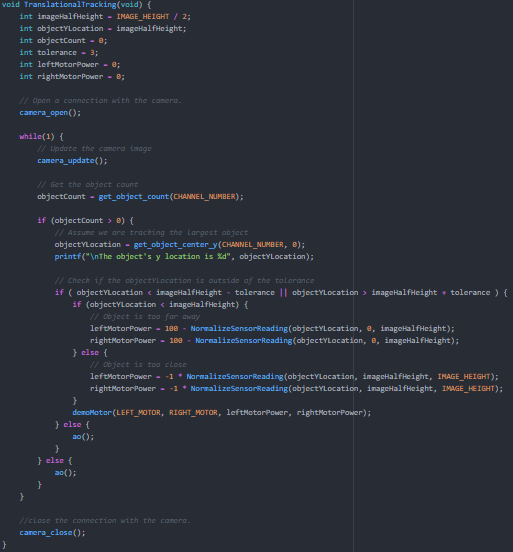


Figure : Translational Tracking Function

## Rotational and Translational tracking using Camera

### Function

void RotationalAndTranslationalTracking(void) {

int imageHalfHeight = IMAGE\_HEIGHT / 2;

int objectYLocation = imageHalfHeight;

int imageHalfWidth = IMAGE\_WIDTH / 2;

int objectXLocation = imageHalfWidth;

int objectCount = 0;

int rotationalTolerance = 20;

int translationalTolerance = 8;

int leftMotorPower = 0;

int rightMotorPower = 0;

// Open a connection with the camera.

camera\_open();

while(1) {

// Update the camera image

camera\_update();

// Get the object count

objectCount = get\_object\_count(CHANNEL\_NUMBER);

if (objectCount > 0) {

// Assume we are tracking the largest object

objectXLocation = get\_object\_center\_x(CHANNEL\_NUMBER, 0);

objectYLocation = get\_object\_center\_y(CHANNEL\_NUMBER, 0);

printf("\nThe object's x location is %d", objectXLocation);

printf("\nThe object's y location is %d", objectYLocation);

// Check if the objectYLocation is outside of the tolerance

if ( objectYLocation < imageHalfHeight - translationalTolerance || objectYLocation > imageHalfHeight + translationalTolerance ) {

if (objectYLocation < imageHalfHeight) {

// Object is too far away

leftMotorPower += 100 - NormalizeSensorReading(objectYLocation, 0, imageHalfHeight);

rightMotorPower += 100 - NormalizeSensorReading(objectYLocation, 0, imageHalfHeight);

} else {

// Object is too close

leftMotorPower += -1 \* NormalizeSensorReading(objectYLocation, imageHalfHeight, IMAGE\_HEIGHT);

rightMotorPower += -1 \* NormalizeSensorReading(objectYLocation, imageHalfHeight, IMAGE\_HEIGHT);

}

}

printf("\nY Power component is %d", leftMotorPower);

leftMotorPower /= 2;

rightMotorPower /= 2;

// Check if the objectXLocation is outside of the tolerance

if ( objectXLocation < imageHalfWidth - rotationalTolerance || objectXLocation > imageHalfWidth + rotationalTolerance ) {

if (objectXLocation < imageHalfWidth) {

// Object is to the left.

rightMotorPower += 100 - NormalizeSensorReading(objectXLocation, 0, imageHalfWidth) - 50;

} else {

// Object is to the right.

leftMotorPower += NormalizeSensorReading(objectXLocation, imageHalfWidth, IMAGE\_WIDTH) - 50;

}

}

// Move motor appropriately.

if (leftMotorPower == 0 && rightMotorPower == 0) {

ao();

} else {

printf("\nleftMotorPower before bind %d", leftMotorPower);

printf("\nrightMotorPower before bind %d", rightMotorPower);

leftMotorPower = bind(leftMotorPower, -100, 100);

rightMotorPower = bind(rightMotorPower, -100, 100);

printf("\nleftMotorPower after bind %d", leftMotorPower);

printf("\nrightMotorPower after bind %d", rightMotorPower);

demoMotor(LEFT\_MOTOR, RIGHT\_MOTOR, leftMotorPower, rightMotorPower);

}

} else {

ao();

}

}

//close the connection with the camera.

camera\_close();

}

## Obstacle avoidance while tracking a target

### Thread Driver Function

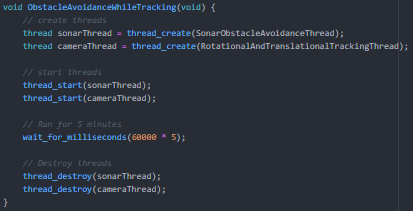


Figure : Thread Driver Function

### Avoidance Thread Function



Figure 5 : Sonar Obstacle Avoidance Thread Function

### Rotational and Translational Tracking Thread

void RotationalAndTranslationalTrackingThread(void) {

int imageHalfHeight = IMAGE\_HEIGHT / 2;

int objectYLocation = imageHalfHeight;

int imageHalfWidth = IMAGE\_WIDTH / 2;

int objectXLocation = imageHalfWidth;

int objectCount = 0;

int rotationalTolerance = 20;

int translationalTolerance = 8;

int leftMotorPower = 0;

int rightMotorPower = 0;

// Open a connection with the camera.

camera\_open();

while(1) {

while(!gv\_sonarEvent) {

// Update the camera image

camera\_update();

// Get the object count

objectCount = get\_object\_count(CHANNEL\_NUMBER);

if (objectCount > 0) {

// Assume we are tracking the largest object

objectXLocation = get\_object\_center\_x(CHANNEL\_NUMBER, 0);

objectYLocation = get\_object\_center\_y(CHANNEL\_NUMBER, 0);

gv\_lastX = objectXLocation;

gv\_lastY = objectYLocation;

// Check if the objectYLocation is outside of the tolerance

if ( objectYLocation < imageHalfHeight - translationalTolerance || objectYLocation > imageHalfHeight + translationalTolerance ) {

if (objectYLocation < imageHalfHeight) {

// Object is too far away

leftMotorPower += 100 - NormalizeSensorReading(objectYLocation, 0, imageHalfHeight);

rightMotorPower += 100 - NormalizeSensorReading(objectYLocation, 0, imageHalfHeight);

} else {

// Object is too close

leftMotorPower += -1 \* NormalizeSensorReading(objectYLocation, imageHalfHeight, IMAGE\_HEIGHT);

rightMotorPower += -1 \* NormalizeSensorReading(objectYLocation, imageHalfHeight, IMAGE\_HEIGHT);

}

}

leftMotorPower /= 2;

rightMotorPower /= 2;

// Check if the objectXLocation is outside of the tolerance

if ( objectXLocation < imageHalfWidth - rotationalTolerance || objectXLocation > imageHalfWidth + rotationalTolerance ) {

if (objectXLocation < imageHalfWidth) {

// Object is to the left.

rightMotorPower += 100 - NormalizeSensorReading(objectXLocation, 0, imageHalfWidth) - 50;

} else {

// Object is to the right.

leftMotorPower += NormalizeSensorReading(objectXLocation, imageHalfWidth, IMAGE\_WIDTH) - 50;

}

}

// Move motor appropriately.

if (leftMotorPower == 0 && rightMotorPower == 0) {

ao();

} else {

leftMotorPower = bind(leftMotorPower, -100, 100);

rightMotorPower = bind(rightMotorPower, -100, 100);

demoMotor(LEFT\_MOTOR, RIGHT\_MOTOR, leftMotorPower, rightMotorPower);

}

} else {

ao();

if(gv\_lastX < IMAGE\_WIDTH / 2) {

// Pan to the left looking for the object.

goDemobotMav(LEFT\_MOTOR, RIGHT\_MOTOR, 600, 0, 1000);

} else {

// Pan to the right looking for the object.

goDemobotMav(LEFT\_MOTOR, RIGHT\_MOTOR, 600, 1000, 0);

}

}

}

wait\_for\_milliseconds(100);

}

//close the connection with the camera.

camera\_close();

}

## Extra Credit

# Conclusion

# Suggestions