Final Examination

CSIS 375 – Introduction to Robotic Systems

Dr Daryl Thomas

Haniel Garcia

5 May 2019

1. **Discuss the common misconceptions that we considered in class about what constitutes a robot. Argue that the most important characteristic for determining what constitutes a robot is the extent to which it can make decisions and carry out actions autonomously. Use your argument to write a working definition of autonomous systems.**

Many people have the concept that robots are just machines that need of a human to program them and tell them what to do, when the truth is that an authentic robot should be required to think and make decisions on its own, a real robot should be autonomous. Many people have made speculations about if robots will take over our jobs and if they will lower the markets for us humans. Well the truth is that in fact, robots will increase jobs in the fields of engineering, design and manufacturing, which essentially is fairly significant. As pretty much more processes basically are automated with robotics, humans will definitely be for all intents and purposes free to literally do definitely other work instead of the repetitive tasks that took kind of much longer, or so they for all intents and purposes thought. The savings and productivity that really stem from advances in robotics for the most part boost economic growth in response to an increase in production, basically contrary to popular belief. Evolution for all intents and

purposes is for everyone, demonstrating that in fact, robots will increase jobs in the fields of engineering, design and manufacturing, which particularly is fairly significant. While many of the current jobs will be replaced by robots, robotized algorithms or artificial intelligence, in the future many new and unimagined jobs will appear that do not yet exist.

WHAT CONSTITUES A ROBOT?

A robot can for the most part be a kind of physical electromechanical mechanism with the capacity to, for all intents and purposes, think or resolve, although in reality they literally are particularly limited to executing orders dictated by people in a subtle way.

2. **Explain how the Roboteq motor controller implements the Closed Loop Position mode and the Closed Loop Speed modes of operation. Discuss the relative advantages and disadvantages of each mode, giving examples of how the class used them in the programs that you developed to control the robot's motion.**

CLOSE LOOP SPEED MODE

Close loop speed mode ensures that the motors will run at a precise desired speed. If the speed changes because of changes in load, the controller automatically compensates the power output so that the motor maintains a constant speed. This mode is the traditional closed loop technique where speed is measured with a speed sensor. The speed is compared to the desired speed and a PID control loop adjusts the power output up or down in order to reach and maintain that speed.

Close loop speed mode	
Advantages	Disadvantages
Can go as long as the user wants	Can't keep track of position

Can go as fast as the user wants	Have to manually send the command to stop
	Can't make a turn as accurate as in position mode

CLOSE LOOP POSITION MODE

In the Closed Loop Position mode, the controller can move a motor a precise number of encoder counts, using a predefined acceleration, constant velocity, and deceleration. This mode requires that an encoder be mounted on the motor. The desired position is given in number of counts. Using acceleration, deceleration and top velocity, the controller computes the position at which the motor is expected to be at every one millisecond interval. A PID then computes the power to give to the motor in order to maintain that position. A comparator looks at the desired position and the computed current position and issues a Destination Reached flag.

Closed Loop Position mode	
Advantages	Disadvantages

Keep track of wheel position	We need to reset the counters quite often
Go forward to a certain position	Calibration is needed in order to know how many counts the encoder needs to make a revolution
We are able to know if robot has arrived to destination	
Makes a more accurate turn	
Set speed, acceleration and deceleration rates	
Automatically stops	

Roboteq User Manuals pages 121 and 139

3. **Give a detailed explanation of the term ‘real-time systems’, including their distinguishing features, why they are necessary, where they are typically needed, and their need for interrupts.**

A real-time system could be defined as an informatics system that has the ability of quickly interact with its physical environment and which is able to perform certain functions for its own benefit. Most real-time systems have the facility of performing tasks in a very accurate time interval.

Most of these tasks for the most part are executed immediately in a very precise way, but for this, calibration actually is needed in order to definitely perform the right movements around the very physical environment in a subtle way.

The efficiency of real-time systems consists of two very important factors: the accuracy of the results the system actually is computed and the timing in which the system computes them and outputs them to the other system’s components.

4. **Explain what interrupts are and how they are typically implemented. What problems are the intended to solve? What problems might arise from using them improperly?**

An interrupt's function is to cut the flow of energy from the motor controller to the actual motor.

An interrupt is a signal from a device attached to a computer or from a program within the computer that requires the operating system to stop and figure out what to do next. Almost all personal (or larger) computers today are interrupt-driven - that is, they start down the list of computer instructions in one program (perhaps an application such as a word processor) and keep running the instructions until either (A) they can't go any further or (B) an interrupt signal is sensed. After the interrupt signal is sensed, the computer either resumes running the current program or begins running another program. Basically, a single computer can perform only one computer instruction at a time. But, because it can be interrupted, it can take turns in which programs or sets of instructions that it performs. This is known as multitasking. It allows the user to do a number of different things at the same time. The computer simply takes turns managing the programs that the user starts. Of course, the computer operates at speeds that make it

seem as though all of the user's tasks are being performed at the same time. (The computer's operating system is good at using little pauses in operations and user think time to work on other programs.)

https://whatis.techtarget.com/definition/interrupt

5. **Explain the concept of localization in the context of robot navigation. Discuss at least three different methods for keeping track of a robot position as it moves through the environment, giving an example of each.**

Robot localization is the process of determining where a mobile robot is located with respect to its environment. Localization is one of the most fundamental competencies required by an autonomous robot since it's important to the robot to be aware of its own location, this is an essential precursor to making decisions about future actions.

MAPPING

In a typical robot localization scenario, a map of the environment is available and the robot is equipped with sensors that observe the environment as well as monitor its own motion. The localization problem then becomes one of estimating the robot position and orientation within the map using information gathered from these sensors. Robot localization techniques need to be able to deal with noisy observations and generate not only an estimate of the robot location but also a measure of the uncertainty of the location estimate.

In our case, the lidar sensor basically has the capability of generating a map so we could position the robot in this 2D map, which for the most part is quite impressive, however,

due to procrastination and a very little time, we weren't able to implement mapping in out robot system.

POSITION RELATIVE TRACK

In our case, we could've use the motor controller to reply back the position in which the encoder was in relation to the encoder's count by sending the command:

```
?TR [cc]
```

Doing some calibration and calculations, we could determine and calculate the distance the robot would go forward in a determine number of counts for the encoder. This way the robot would go forward and make turns in a more accurate way.

Unfortunately, in the project we developed in the class, we weren't able to get information from the motor controller. We were able to send commands to the motor controller but we were not able to read the actual motor controller's console in order to get the information we needed.

6. **During this semester, the class used both sonar and lidar systems to make distance measurements for obstacle avoidance. Explain the theory of operation of each of the systems you used, then discuss the problems you encountered with each, along with how one might overcome them.**

LIDAR

A LIDAR is a light/based radar sensor suitable for small robotic systems applications. In the robot the class developed, we used it for collision avoidance and for the robot to quickly figure out what's around it. The system can perform a 360-degree scan within a 25-meter range.

The typical scanning frequency of the lidar is 10hz. Under this condition, the resolution will be 0.9°. And the actual scanning frequency can be freely adjusted within the 5 - 15hz range according to the requirements of users. This is a pretty accurate system when it comes to get distances.

The problems we encountered when using this type of sensor was that its reading quality depends on the type of material the light is going to. We found out that any object that its

made out of some kind of light absorbent material will make the lidar return very bad and inaccurate readings, sometimes it wouldn't even return data at all.

SONAR

This kind of sensors measure distances through the use of ultrasonic waves. Ultrasonic waves get reflected going back to their origin point once they hit a target. It consists of a transmitter head and a receiver head align next to each other, the transmitter emits an ultrasonic wave while the receiver listens to the wave to come back after it hits an object. These sensors measure distance taking in consideration the time between the emission and the time the wave was received.

The sonar sensor we used during this semester was a HC-SR04 sensor with a reading angle 30°, a reading distance of 400 cm maximum and a resolution of 0.3 cm.

The limitations we got from using this kind of sensor was that it has a very limited range compared with the lidar, if I remember correctly, the range of the sonar was about 4 meters and he farther the object was the sonar wouldn't be as accurate as if the object was closer to the sensor.

We also detected that the consistency and the material the object is made of it might cause the sonar sensor to return crazy readings, we also discovered that the sonar is able to receive all kind of interferences, even from the light.

LIDAR VS SONAR

When it comes to choose between a sonar sensor and a lidar, I personally would choose the lidar since it’s the most accurate and its range is better than the sonar.

https://www.slamtec.com/en/Lidar/A3

https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/

7. **During the semester, we used both DC stepper and brushed permanent magnet DC servomotors to drive robot platforms. Compare and contrast their features, including how they are controlled, power requirements, accuracy, and repeatability, Give examples of ideal use cases for each type. Also, explain the types of motor drivers they require and how they operate.**

DC STEPPER

A stepper motor is an electromagnetic motor capable of turning a series of electric pulses into discrete angular displacements, this means it is capable of turn around a certain amount of degrees depending on its control inputs. The stepper motor behaves the same way as an analog-to-digital converter does and it might control itself by electric pulses coming from some digital systems. This kind of motor presents a bit of advantages when it comes to precision and repeatability when it comes to positioning.

DC SERVOMOTORS

Servomotors are usually considered a high-performance version of a stepper motor. This type of motors is used to control the position in the open loop that does not need to use

an encoder, rather than the signal that the number of movement steps to be rotated is specified. This is why the motor needs to be in a known position during startup.

The encoder and controller of a servomotor have added optimized features for the entire system, for all speeds and accuracy. What is happening is the same as stepper motors with closed loop.

STEPPET MOTORS VS SERVOMOTORS

Servomotors are usually considered a high-performance version of a stepper motor. The stepper motors allow a certain control of the position since they have a series of integrated steps. Normally this type of motors are used for control of position in open loop without the need to use an encoder, since the actuation signal specifies the number of movement steps to be rotated. This is why the motor needs to be in a known position during start-up (for example, going to a known position with a limit switch, the first time it is started).

The lack of a closed loop limits the performance of the stepper motors as it can only go to the desired position depending on its capacity. If the capacity of the motor is exceeded, there could be loss of steps and position errors, with the need to restart or reposition the

system. The encoder and controller of a servo motor add optimized performance for the entire system, for all speeds and accuracy.

8. **As you are aware, the autonomous robotic systems we studied this semester are capable of causing harm. Discuss at least five different types of safety systems that one might add to such a system to help prevent them from causing harm.**

EMO BUTTONS

Emergency buttons are able to stop the motor controllers to keep sending signals and power to the actual motors. These kinds of buttons come in handy when a system needs to be shut down immediately.

The disadvantage of using this kind of emergency stop is the need of having someone looking at the button and be ready to press it at all times while the machine is operating.

Something cool about the EMO button in the system we implemented during the class was that even though the button was pressed and the energy was cut from going to the motors, after we release the button, the motor controller would continue with its current job.

FAIL-SAFE

This kind of system is designed to remain in the case of a failure of the system. It is important to know that these kind of systems does not prevent failures but instead it will help to isolate the failure in case it happens.

An example could be the fail-safe system in the elevators, these are designed with brakes outside the elevator that will be held back by the tension of the elevator's cable; if the cable snaps the loss of tension will cause the breaks to be applied.

EXTRA SENSOR

A way to earn extra support and a more accurate system is by implementing the use of extra sensors. These extra sensors will support the main sensor on its readings and in case the main sensor fails the system would still have some extra sensors to still working with.

In our project, we thought about implementing two sensors systems: lidar and sonar systems as sensors to make them both read simultaneously and then being able to get a more accurate distance.

IMPACT ABSORBANTS

These are put around the robot chassis and it will allow the robot to imply and to take less damage taken from an impact cause by a failure. Once the robot has made the impact, its interior among with whatever it is hit, shouldn't be that much damaged.

CONNECTION CHECK

Have a script or a device that will keep on checking the connection status between the controller and the motors and have a fail-safe system that will activate in case the connection is lost, this would come in handy when working with systems remotely at long distances, if the connection is lost, have the system to stop and don’t do anything else, this way we will be reducing the chances of causing any type of harm to anyone around the system.

9. **Discuss the moral implications of the development and deployment of autonomous robotic systems. Consider how we might deal with consequences such as lethal actions on the part of robots (either accidental or intentional), the societal ramifications of robots displacing humans in various positions, and the culpability of their designers should these things transpire**

It is very important to understand that autonomous robotic systems are meant to fulfil humans' tasks with the use of different mechanisms such as sensors, actuators and computers.

The rising development of autonomous systems in our daily lives has allowed us to cover needs in our society, this has taking us to take ethics in consideration since human rights must be taken care of. Our dignity may be at risk through the daily evolution of robotics and it is important that we pause and reflect on the consequences of technological changes.

It must be considered that humans' lives are standing in front of an era in which robots are extremely sophisticated allowing a new industrial revolution that will probably affect all of our society.

In recent years technological advances have grown exponentially allowing their connection and influence in society, finding ourselves today in an era where science and technology have transformed the quality of life of human beings, in years ago there was doubt of the possibility of machines coming into existence that are programmed to perform certain functions.

It must be put in mind that deploying robots can negatively influence jobs as many people would be unemployed and having robots in a company could act aggressively if you do not have control over these robots.

A branch where ethics really comes to mind when developing and deploying these autonomous systems is when it comes to military use. Military ethics deals with the use of force, armed conflict in service and in defense of a country as established in the constitution. Military ethics also deals with the ethical behavior of the military, leadership and respectful treatment among professionals of the military. Military ethics is not common ethics, but ethics that establishes the code by which the combatant or soldier is governed. But, what to do when in order to defend the country, military action against people is needed? This question has split society into different opinions, since some of

them want to use it to defend the country even if that means to kill some people in the process which is what the other groups are arguing about.

For the world, technological growth will depend a lot on the ethics of each person that develops an efficient technology, that provides all the rules and laws so that it cannot harm another person through the machines that are implemented, the algorithms must provide security, efficiency, reduce costs, feasible tools for people.

10. Explain, in detail, the program that the class developed to control the robot's motion. Supply commented code to illustrate your discussion.

In order to get the program talking to the lidar, we used the header files on the SDK provided by Slamtec.

```
#include "rplidar.h"

using namespace rp::standalone::rplidar;
```

We made a list of commands we would use in order to get the motor controller to do what we want to do.

Command	Description
"^MMOD X 0_	Set the motor controller in open loop mode
"!C [X] 0_	Reset the counters of X motor to zero
MMOD X 3_	Set the motor controller in position mode
"!PR [X] N_	Tell motor X to go to N position
"!EX_	Activate emergency stop
!MG_	Release emergency stop

We send the commands using the following function:

```
write(port, command, number of bytes to send)
```

We execute the program using the following command:

```
./robotics_s19_v3 [Com Path] [Baudrate]
```

The very first thing the program does is to retrieve the Com Path and the Baudrate and set the respective variables equals to their respective values; in case the user didn't put the parameters needed, the program will shut down and exit.

```
//Initialization of variables
const char * com_path = NULL;
_u32         com_baudrate = 0;

//Retrieve parameters
if (argc > 2) {
	com_path = argv[1];
	com_baudrate = strtoul(argv[2], NULL, 10);
	cout << "RPlidar PATH: " << com_path << endl;
	cout << "RPlidar BAUDRATE: " << com_baudrate << endl;
} else {
	cout << "Invalid initialization of program." << endl;
	cout << "Please initialize as follows:" << endl;
	cout << "robotics_s19 [Com Path] [Baudrate]" << endl;
	return 0;
}
```

Next thing in line is to create an RPlidarDriver object that includes all the functions on the RPlidar header file. If there's not enough memory to create the object, the program will exit. Right after that the program will try to stablish a connection with the lidar using

```
//Set RPlidar Driver
RPlidarDriver * myRplidar =
RPlidarDriver::CreateDriver(DRIVER_TYPE_SERIALPORT);
if (!myRplidar) {
     cout << "Couldn't set RPlidar driver" << endl;
     return 0;
} else {
     cout << "RPlidar driver set" << endl;
}

//Make Connection
if(IS_OK(myRplidar->connect(com_path, com_baudrate))){
     cout << "Connection sucess" << endl;
} else {
     cout << "Couldn't connect to RPlidar" << endl;
     goto finished;
}
```

the Com Path given by the user.

Once we have stablished a successful communication bridge between the lidar and the program, it will now try to make a connection to the motor controller using a static Com

```
// Open the port for motor controllers.
int fd;
fd = open("/dev/ttyAMA0", O_RDWR | O_NOCTTY | O_NDELAY);

if (fd == -1) {
     printf("Unable to open /dev/ttyAMA0");
     return 0;
} else {
     printf("Port /dev/ttyAMA0/ succesfully open\n\n");
}

fcntl(fd, F_SETFL, FNDELAY);
```

Path defined within the program's code. In case the program is not able to connect to the motor controller, it will exit.

Once both connections are made successfully, the program will tell the lidar to start its motor and then start the scanner, have in mind that we are not yet retrieving the information from the scanner.

```
if(IS_OK(myRplidar->startMotor()))
      cout << "Starting Motor" << endl
else {
      cout << "Couldn't Start Motor" << endl;
      return 0;
}

sleep(2);

if(IS_OK(myRplidar->startScan(0,1)))
      cout << "Starting Scanner" << endl;
else {
      cout << "Couldn't Start Scanner" << endl;
      return 0;
}
```

We now ask the user to insert a desire distance to go in ft. Since we weren't able to get information from the motor controller to the raspberry pi, we are using a concept on

```
int dist_in_ft;
cout << "What's the distance? (in ft)" << endl;
      cin >> dist_in_ft;

int time_moving;
setCountPositionMode(fd);
time_moving = moveForward_v2(fd, dist_in_ft);
```

timing to set a timer equals to the given distance in ft but in seconds. This is for the program to know when to stop the robot and send the next command.

Every second the program will grab data from the scanner and will allocate the data in an array, it will then take the first three measures and the last three as well and it will check the quality, if the quality is not equals to zero, that means is good data and we can use them in our program, if not, it will just ignore it. We have set a counter to determine how many good elements we got from the array, in the end, the program will average the good elements that got from the scanner.

If the program averages a distance less or equals to 200 cm it fill then send the command to activate the emergency break and won't let the time counter to keep going until the program averages a distance greater than 200.

Once the robot has made it to its destination it will make a 180 degree turn and make its way back to the origin point.

```
rplidar_response_measurement_node_hq_t nodes[8192];
size_t nodeCount =
sizeof(nodes)/sizeof(rplidar_response_measurement_node_hq_t);
u_result result = myRplidar->grabScanDataHq(nodes, nodeCount);

myRplidar->ascendScanData(nodes, nodeCount);

float dist = 0, ang = 0;
int goodReadings = 0;

for(int n = 0; n < 3; n++){
      if(nodes[n].quality != 0){
            dist += nodes[n].dist_mm_q2 / 10.f / (1 << 2);
            ang += nodes[n].angle_z_q14 * 90.f / (1 << 14);
            goodReadings++;
      }
}

for(int m = nodeCount - 4; m < nodeCount; m++){
      if(nodes[m].quality != 0){
            dist += nodes[m].dist_mm_q2 / 10.f / (1 << 2);
            ang += nodes[m].angle_z_q14 * 90.f / (1 << 14);
            goodReadings++;
      }
}

float dist_avg = dist / goodReadings;

//STOP AND WAIT
if(dist_avg < 200 && dist_avg != 0 && dist_avg != NULL){
      emergency_stop(fd);
      emergencyStop_on = true;
      emergency_counter_backup = x;
      x--;
}

//RELEASE EMERGENCY BREAK
if(emergencyStop_on == true && dist_avg >= 200 && dist_avg != 0
&& dist_avg != NULL){
      release_emergency_stop(fd);
      emergencyStop_on = false;
      x = emergency_counter_backup;
}

sleep(1);
```

Once the robot has made it back to its origin point, it will then make a 180 degree turn and it then it will dispose the RPLidar object we created at the beginning of the program.

Once it is done that, it will then exit.

```
cout << "Stopping Scanner" << endl;
myRplidar->stop();

sleep(1);

cout << "Stopping Motors" << endl;
myRplidar->stopMotor();

RPlidarDriver::DisposeDriver(myRplidar);
cout << "RPlidar driver diposed" << endl;
myRplidar = NULL;
```