**ELEN90066 Embedded System Design**

**Final Report:** **Kobuki Obstacle Course Project - On-campus**

SM2/W05/G03

1. **Introduction**

This project is to design and implement a navigation algorithm on Kobuki robot. Following the previous six workshops, we should apply the technique of using the proper sensors on the Kobuki robot and use the software (C or LabVIEW) to construct a Finite State Machine with designed algorithm. Finally, we then implement our design in real Kobuki robot and challenge the obstacle course (shown in figure 1) in final workshop. The goal is to navigate the robot from the start point to the finish point, passing numerous obstacles, a hill terrain, and automatically stop after it get backs to the ground.

|  |
| --- |
| **Diagram  Description automatically generated**  Figure : Obstacle course layout (note that items are not to scale) |
|  |

1. **Background: Sensors**

In this project, we use NI myRIO and the Kobuki robot to implement our navigation algorithm. We use the IMU sensor from NI myRIO and bumpers, odometry, and cliff sensors from Kobuki.

* 1. **Bumpers**

There are three bumper sensors at the front end of the Kobuki, separated at left, center, and right side of Kobuki. When a bumper sensor is pressed, it will return a value 1. In contrast, when the bumper is released, it will return the value 1. In our algorithm, we use the bumpers to detect the obstacle, and further triggers the obstacle avoidance method.

* 1. **Odometry**

There are two encoders installed on both right and left wheels. The odometry record the count value of encoder and transfer into the traversed net distance of each wheel. Using both left and right-side travel distance, we can then obtain the net distance and the net angle that Kobuki has travelled through. Further, in our algorithm, we use it to control the robot drive in a certain distance or rotate in a certain angle.

* 1. **IMU**

There is one IMU sensor on the NI myRIO. In this project, we installed the myRIO at the top center of the Kobuki and use the IMU sensor to receive the linear acceleration value (unit: g) of x axis and y axis. Ideally, when the robot is remained static on the ground, value of z axis should be 1, x and y axis should be 0. In our algorithm, we use the value of x axis acceleration to know the pitch condition of the robot, and further determine if the robot is in a flat, incline or decline plane. For the acceleration of y axis, it is used to determine if the robot is face toward upright.

* 1. **Cliff sensor**

There are three cliff sensors at the bottom of the Kobuki, separated at left, center, and right side. The cliff sensor detects the distance between Kobuki bottom and the ground. Based on the return value’s type, it is separated into two kinds:

* + - * Cliff Sensor Signal:

Return a value in a range: represent the IR Signal value. Smaller means the distance between Kobuki and the ground is larger. Using the cliff sensor signal, algorithm can identify if the Kobuki is tilting.

* + - * Cliff Sensor

Return a true or false. If the distance between Kobuki and ground is bigger than the threshold, it will return a true signal. In contrast, when the distance is smaller than the threshold, it will return false. Using the cliff sensor, the algorithm can identify if the Kobuki traveling to a cliff or not.

We receive the sensor value from remote panel target.vi. The figure below is the monitor panel.

|  |
| --- |
| Figure : Kobuki monitor |

1. **Design procedure:**

The goal of this project is to navigate the Kobuki robot from the start point to the finish point. It contains 4 main regions: Start Point, Obstacle Region, Hill Region, and Finish Point. Furthermore, in the hill region, there are three parts separated, including incline plane, high land, and decline plane. Our algorithm should allow Kobuki pass through these regions without falling from the cliff at both sides of the hill. Besides, all tasks should be finished within 2 minutes 30 seconds. By using the LabVIEW, we construct an algorithm shown in figure below. Next, we dig into each part of our algorithm.

|  |
| --- |
| Figure : Algorithm LabVIEW state Chart |

* 1. **Pause and Start**
     1. **Trigger Mechanism**

Press the B0 button to start or pause the robot.

* + 1. **Strategy**

According to the specification, the Kobuki should remain still at the beginning. After we placed the Kobuki with a certain orientation and started the first movement, B0 button will be pressed to start the Kobuki robot, and then it will go into the RUN state. While the robot is running, the robot should stop at any time the B0 button is pressed until the button is pressed again.

|  |  |
| --- | --- |
| Diagram  Description automatically generated  Figure : Finite State Machine of Pause and Start Region | |
|  |

* 1. **Avoiding Obstacle**
     1. **Trigger Mechanism: B0 button**

As the first state of our machine, once the play button is pressed, it will straightly start avoiding obstacles.

* + 1. **Strategy: Running in Clockwise Arc**

In the begin of avoiding obstacle state, the Kobuki will run straight to the right to make sure the Kobuki is always running against the right side. After any of the bumper triggers, which means the Kobuki touched an object, the Kobuki first backs, then turn left in a certain angle and start running in a clockwise arc by using different motor speeds until the next bumper trigger. In this case, the radius of the running arc is only based on the speed difference of motors.

* + 1. **Exit Mechanism: Any Cliff**

Considering the obstacle avoidance method we used, the robot will stick to the right side until it reaches the incline plane without a wall at the right side. Therefore, the robot will reach the right side of incline plane, which is a cliff. However, based on the condition of the final test site, the orientation of the Kobuki when it reaches the cliff is non-deterministic. Hence, all Center Cliff, Right Cliff and Left Cliff sensors should be use. Moreover, due to the threshold of the pure cliff sensors, it may fail if the cliff is not sufficiently high, so that “Cliff sensor signals” for all center, left and right, which returns a value are used instead. Once any cliff sensor signal value is in a certain range, which means the robot detects a cliff, then the program moves into the incline state.

* + 1. **Testing procedure: Trial Error**

According to our algorithm, there are few things to be concerned. Firstly, we want the robot keep running against the right-hand side of a traversable path such that the radius of the running arc should keep small. Secondly, the robot should correctly detect the cliff even though the cliff is not high. Last, the robot should run as fast as possible since there is a time limit. Therefore, if the robot accidently hits the left side obstacle with a high probability, we should decrease the tuning angle and arc radius. If the sensor cannot detect the cliff, then the threshold should be tightened.

* + 1. **Final Implement Detail [motor speed in the format of (Left, Right)]**

Driving straight speed value: (250, 250)

Driving backward distance: 25mm

Driving backward speed value: (-100, -100)

Turning left angle: 30 degrees

Turning left speed value: (-100, 100)

Running in an arc speed value: (450, 150)

Cliff sensor signal threshold value: 800

|  |
| --- |
| Diagram  Description automatically generated  Figure : Finite State Machine of Obstacle Avoidance Region |
|  |

* 1. **Incline Region**
     1. **Trigger Mechanism: Any Cliff**

As long as any of three cliff sensors is triggered, in our situation the sensor signal value should be in a certain range, the robot starts to climb hill.

* + 1. **Strategy**

After the cliff sensor is triggered, first, we drive the Kobuki backward to leave the cliff. The purpose of this movement is to prevent the wheel drop happens and tilt up the front of the robot. Then the Kobuki start to roughly navigate the direction upward using X and Y accelerators and run in a straight line. Although the Kobuki is running straight ahead after direction navigation, it may still meet a cliff due to the accuracy of accelerators. To deal with this, if right or left cliff sensor is triggered, the Kobuki will drive backward to leave the cliff and turn to the opposite direction with small angle and keep moving.

* + 1. **Exit mechanism: Center Cliff Signal**

The transition from the incline state to the flat state is based on the center cliff signal since the Kobuki will only tilt up a little bit at the end of the uphill.

* + 1. **Testing procedure: Trial Error**

There are only two main parameters should be adjusted. One is the range of the accelerator while navigating the direction and the other is turning angle while detecting a cliff. Due to the errors in accelerator values, it is not appropriate to set the value to a specific number, so that the value should be in a certain range. But there is still a trade-off. On the one hand, if the range is too small, the robot may fail to adjust the direction. On the other hand, if the range is too big, there will be no point in adjusting the direction.

Following that problem, if the navigation is perfect, the Kobuki will never meet any cliff and no turning is needed. But if the navigation is bad, we may need a big turning angle. Otherwise, it will turn many times and waste the time.

* + 1. **Final Implement Detail**

Drive back distance at the beginning: 100 mm

Drive back speed value at the beginning: (-75, -75)

Navigation speed value: (-50, 50)

Navigation accelerator range: -0.05<y<0.05 && x>0

Drive incline speed value: (250, 250)

Turning angle detecting cliff: 15 degrees

Center cliff signal value threshold: 1200

|  |
| --- |
| Diagram  Description automatically generated  Figure : Finite State Machine of Incline Region |
|  |

* 1. **Flat State (High Land)**
     1. **Trigger Mechanism: Cliff Sensor Signal**

Due to the upturned condition of the Kobuki at the end of the uphill, the cliff sensor signal can be used to detect the height difference of the junction.

* + 1. **Strategy**

Because of the navigation part in the incline state, the orientation of the Kobuki after fully landing at the flat is still roughly facing ahead, such that the Kobuki just needs to keep moving and simply avoid cliff using the same strategy in incline state.

* + 1. **Exit Mechanism: Accelerator and Safety Distance**

The transition to the decline state is mainly based on the accelerator. The negative X accelerator value means the Kobuki starts move downward. However, due to the structure of the Kobuki, when it moves from the uphill to the flat, it may have a sudden fall. That situation will strongly affect the accelerator and cause sensor values to be confused. Therefore, we add a safety distance to eliminate the effect that X accelerator suddenly drops to negative at the beginning of the flat state. Hence, after moving a certain distance, the X accelerator can correctly indicate the orientation of the Kobuki.

* + 1. **Testing Procedure: Trial Error**

The most important parameters in this state are the threshold of the center cliff signal while entering to this state and the threshold of the X accelerator while exiting this state. While determining center cliff signal value, we compare it with the normal value when the Kobuki is placed horizontally. If the cliff signal value is smaller than that, we assume it detects a terrain transformation from uphill to flat.

The X accelerator threshold is determined by the same way. If the accelerator is smaller than the normal value, we assume the Kobuki starts to face downward.

|  |
| --- |
| Diagram, schematic  Description automatically generated  Figure : State Machine of High Land Region |
|  |

* + 1. **Final Implement Details**

Center cliff signal threshold value: 1200

Drive speed: (150, 150)

Drive back distance: 50

Turning angle: 15 degrees

Safety distance: 150 mm

X accelerator threshold value: -0.2

* 1. **Decline**
     1. **Trigger Mechanism: X accelerator and Safety Distance**

As discussed in the flat state, if the Kobuki is not just entering to the flat state and is facing downward, it is sufficient to say the Kobuki is in the decline state.

* + 1. **Strategy**

The strategy is as the same as the one in the flat state. Just need to adjust direction if a cliff is encountered.

* + 1. **Exit Mechanism: X accelerator**

The X accelerator is used to make sure the Kobuki finishes the downhill and on the ground. If the accelerator value is in the same range as it is in placed horizontally, we assume it should stop.

* + 1. **Testing Procedure: Trial Error**

As the same as in the flat state.

* + 1. **Final Implement Details**

X accelerator value threshold while entering: -0.2

Safety distance: 150

Drive speed values: (200, 200)

Drive back distance: 50

Turning angle: 15 degrees

|  |
| --- |
| Diagram, schematic  Description automatically generated  Figure : Finite Sate Machine of Decline Region |
|  |

We rewrite our Finite State Machine with lectures’ notation in figure below. It contains entire algorithm.

|  |
| --- |
| Figure : Complete Finite State Machine |
|  |

1. **Validation procedure (this could include reliability or reachability analyses)**
   1. **Starting Region:**

At the beginning, after we put the robot and press the B0 button, it should turn into the first driving straight forward movement.

Then, in the arena, after we choose the orientation and start the robot, it will eventually hit an obstacle or wall and trigger the bumpers. Let the bumper event be .

The exit mechanism of this region is the first . Therefore,

**Extreme case: Fail to reach the right-side wall**

Key Factor: Obstacle configuration

There exists the obstacle configuration that block the way from starting point to the right-side wall. If an obstacle is in the middle of the way, the Kobuki will circle around the obstacle forever. Such a situation means that the robot will be stuck in the obstacle avoidance region forever.

which also means:

The possibility of extreme case appearing is entirely depends on the obstacle configuration. Although we could decide the starting direction, there exist the chance that robot would never reach to the right side of the arena. This is one of the major flaws of this algorithm.

* 1. **Obstacle Avoidance Region:**

After the first bumpers event, the program will exit from the starting region and the obstacle avoidance region will be triggered. Then, it will drive backward to avoid hugging the obstacle or wall. Instead, for all bumper event, the algorithm requires Kobuki to drive backward. Let the driving backward event be

Then, the robot should perform a forward curve driving movement to avoid the obstacle until it hit the next obstacle or cliff. Let cliff event be

Ideally, the robot will strictly stick to the right side of the arena. It will eventually reach the first cliff at the right side of the incline plane.

Next, it will turn into next region: Incline Region.

**Extreme case: Fail to maintain right-hand rule**

Key Factor: radius of turn and the adjusting angle

At the obstacle avoidance region, our algorithm controls the robot to drive back with distance equals 25 and turn left 30 degrees after it detect an obstacle. Lastly, moving forward in a curve with left wheel speed equals 450 and right wheel speed equals to 150. By trial-and-error method, we selected the best result of the radius of curve and the left turning angle in our experiment. However, there exist an extreme case that:

The case is demonstrated in the figure below. When the robot passes through a narrow space between two obstacles, there exist possibility that the robot triggers the left bumper sensor and start to stick to the left object. In ideal scenario, if the space between two obstacles is small enough, the robot will move from the location with notation one to five and back to the right obstacle. However, in extreme case, when the space is large enough for Kobuki to pass through, but not large enough to avoid first left bumper event, it is possible that robot move from location 1 to five and keeps circling around the left object.

|  |
| --- |
| Figure : Extreme Case 1 |
|  |

By trial-and-error method, we tested the performance of robot with different radius of curve and adjusting angle. It should lower the possibility of this extreme case happening. As the matter of fact, our robot should stick to the right closely, which should not lead to this extreme case in most of the time.

* 1. **Incline Region**

After the first cliff event, the program turns into incline region. The first movement is drive backward. As the matter of fact, every any cliff event followed by a driving backward movement .

Then, the robot should adjust the orientation toward upright. With a proper threshold range of y axis acceleration, the robot will rotate in the proper angle and turn to next movement: driving straight forward. Let the movement be .

Even if the orientation is not exactly upward, the algorithm provides a safety guard that prevent the robot fell out from the side. When cliff sensor returns true, the robot will drive back, turn opposite side 15 degrees and back to the driving straight forward movement. Let the right cliff sensor event be and left cliff sensor event be .

The driving backward movement is to prevent any wheel drop followed by rotating movement, which will lead to the tilting of the front end of the Kobuki. Tilting of Kobuki might accidently trigger the high land region entry mechanism. At the end of the incline plane, the center cliff sensor will sense flat plane of high land region. This event will make the program turn into High Land Region.

To prevent the center cliff sensor accidently triggered at right or left side of the ramp, we choose a small adjusting angle after the right or left cliff event happens to maintain the robot moving direction upward. The center cliff sensor will be accidently triggered only if the ramp has extremely large width and length.

**Extreme case: Cliff Detection Failure**

Key Factor: Cliff without enough height

There is one extreme case that might cause the Kobuki fail to recognize the incline plane. The key point is about the threshold setting of cliff sensor signal value. Our algorithm keeps the Kobuki robot moving at the very right side of the obstacle area. When it drives to the edge between ground and incline plane, there exist a case that might lead to the failure of detecting cliff. The case is shown in figure below, it demonstrates a case when Kobuki almost turn to right 90 degrees at the begin of incline plane. This might lead to a case in figure?, the distance h could be too small for cliff sensor to detect the cliff. By trial-and-error, we decided to set the threshold value of cliff sensor signal with 1200. However, in extreme case, it is possible that h is too small and the cliff sensor signal fail to reach the threshold. Therefore, it will fell off from the cliff and never enter the incline region.

|  |  |
| --- | --- |
| A picture containing text, clipart  Description automatically generated  Figure : Extreme Case 2 | A picture containing shape  Description automatically generated  Figure : Extreme Case 3 |
|  |  |

* 1. **Flat Region:**

After the center cliff signal triggered the transition from the incline region to the high land region, the program maintains the straightforward movement. The safety mechanism in this region is the same with the incline region. From the testing experience, we knew that Kobuki will heavily fell from the incline plane to the flat plane. To prevent such an event disturbing the IMU sensor and accidently triggering the exit mechanism of high land region, we set a buffer distance threshold. The exit mechanism will only be triggered when the travel distance of high land region is larger than the threshold.

Ideally, the robot travel through the high land in straight line will eventually drive into decline plane.

**Extreme case:**

|  |
| --- |
| Diagram  Description automatically generated  Figure : Hill Shape |
|  |

In the workshop, we noticed that the width of the flat region is slightly bigger than both ramps. Therefore, in extreme case, if the robot drive into top-left corner of the flat region, the robot will not avoid the cliff at corner. It might drop the left wheel and cross to the decline plane. Or, if the length of d is too big, the robot will even trigger the left cliff safety mechanism until it turns toward right side of the flat region. With such a condition, the robot will never enter the decline region. However, in practice, we won’t face this problem with a small length d.

* 1. **Decline and Stop:**

After the robot drive into the decline plane, then the program turns into decline region based on the receive value of accelerometer of x axis. The safety mechanism is the same with flat region and incline region. The robot will drive downward and reach the ground eventually. Finally, the robot will stop.

**Extreme case: Misidentified stop criteria**

Key Factor: speed difference between two regions

When the robot turns into decline plane from the high land, we design to increase the driving speed. However, if the speed is increased and the increasing amount is too large, the x axis acceleration value is possible to turn into the range that our algorithm considers “back to ground”. This issue will happen if the speed difference is too large between high land section and decline plane section. Therefore, by trial-and-error, we decide a set of speed that does not have large difference between flat region and decline region.

* 1. **Timeliness:**

**Key Factor: The configuration is too hard at the right side**

The major defect of our algorithm is about time limit. Regarding our algorithm is tightly sticking to the right side of obstacles area, the obstacle configuration of right side is crucial for our running time. If the configuration of right side is rugged and complicated, our robot will not finish the task in time.

1. **The outcome of the run in the final workshop**
   1. **Startup**

Our Kobuki successfully kept still at the beginning and then started to run after B0 button was pressed.

* 1. **Obstacle Avoidance**

After adjusting directions and encountering obstacles and walls many times, our Kobuki finally reached the hill and detected a cliff to start climb the hill.

* 1. **Ascending Hill**

After detecting a cliff, our Kobuki successfully navigated the orientation facing forward and avoided driving off the side of the hill and finally reached the flat top.

* 1. **Flat Top**

After passing the changing point between the ascending hill and the flat top, our Kobuki successfully detected the changing height of the path when the head was slightly tilted up. And finally moved into the descending hill without driving off the edge.

* 1. **Descending Hill**

When driving downward, our Kobuki changed the direction a little bit due to the error of navigating the orientation and faced to the edge. However, our Kobuki successfully detected the cliff and rearranged itself and eventually reached the ground floor.

* 1. **Stop**

After fully reaching the ground floor, our Kobuki instantaneously stop and passed the whole test using 2 minutes and 29 seconds.

1. **Discussion section**
   1. **Improvement**
      1. **Obstacle avoidance**

Considering our strategy, we assumed that the Kobuki is always in the avoiding state and keep running in an arc after the first time the bumper is triggered, until it detects any cliff. This costs a lot of time to reach the ascending hill and may not follow the orientation requirements. To be more precise, we should record the direction of the Kobuki according to the ground orientation and return to the ground orientation to drive forward.

* + 1. **Time Consuming**

The time limit in the simplified test case is 2 minutes and 30 seconds. Our Kobuki almost ran out of time. To improve this, one thing we can do is to improve our obstacle avoidance strategy as discussed above. And other thing is increasing the driving speed. However, since we use bumper to detect any obstacles, the Kobuki may accidentally tilt up when hitting an obstacle with high speed. This may the following transitions to fail. We need to somehow balance the speed and successful rate.