Haptic Feedback:

Controlling Robots with Touch

New Mexico
Supercomputing Challenge
Final Report
April 4, 2012
Team 74
Los Alamos Middle School

By: Connor Bailey, and Nate Delgado
Mentor: Rob Cunningham

Table of Contents

Executive Summary

Our project investigates the use of haptic technology in conjunction with modern robotics to make it easier for robots to aid people in dangerous situations. For our project we used the Python programming language to create a virtual environment and a virtual robot. We conducted tests in this Python program that determined whether haptic technology would help or hinder robots used in dangerous situations and to what degree they did that. Two tests were created to determine that. The first test focused on the quality of the haptic technology versus the quality of the regular visual feedback of modern robots. The second test determined the speed at which the haptics technology robot would perform as opposed to the visual technology robot. What we discovered from the results of the quality test was that haptic feedback technology greatly increases the ease of use of robots in dangerous situations. All of the tests we conducted showed that the knowledge of temperature being readily available to the user greatly helps the operator avoid the danger zones in this type of situation. We discovered, through doing the speed tests, that haptic technology greatly increases the speed at which the robot can successfully locate the target. Speed can be very important when working with robots in dangerous situations as many times peoples' lives are on the line. Sometimes the faster the robot can complete its task, the greater chance it might have to rescue people and save more lives because of the increased speed of the haptics technology.

Introduction

Haptic feedback technology is a relatively new development that has great potential in many fields. The word 'haptic' refers to the sense of touch. Haptic feedback is feedback received by a user that pertains to the sense of touch. This could mean something as simple as vibration in a cellphone to indicate when a message is received or when a call is coming in. It can also apply to something as complex as a glove worn by a surgeon to remotely do surgery. Haptic feedback technology has been growing in complexity recently and is spreading to many different fields. Some of its best applications would be in any type of remote work. An example would be the surgeon mentioned before. He could be in a separate room from the patient, but the haptic feedback technology would make

it seem like the surgeon was in the same room. The surgeon could be receiving data through the glove about temperature or texture. The surgeon would also sense if the robot arm hit an object.

However, using haptic technology does not necessarily mean that the user must wear a glove. This technology could be put into a simpler control such as a joystick. The joystick could warm up and cool down to indicate temperature. It could show texture, as well as indicating where physical objects were located. For these types of applications of haptic technology in which people use haptics remotely, robots are usually on the other end. They receive and act on data from the user. For our project we chose to focus on haptic technology remotely controlling a virtual robot in a dangerous situation. The dangerous situation could be detonating a bomb or locating an object in a fire. The object that the robot must find is representative of the bomb or of a person stuck in the fire.

Goal

The goal of our project is to answer these questions:

- Which control method for robots in dangerous situations is better, haptic feedback or video feedback?

- What impact does haptic information have on the operator of a robot in a dangerous situation?

The purpose for finding the answers to these questions is to provide a better insight as to what effects haptic technology might have on robotics used in dangerous situations. By finding this information, we can show whether the use of haptic technology as a control mechanism for robots in dangerous situations should be promoted. The results of this project can be brought to other applications of haptic technology as well. For instance, if we find that haptic technology greatly increases the speed of the operation, but decreases the quality, then we could conclude that haptic technology would not be a good choice for the medical field. We could then conclude this because in the medical field haptic technology would be used to help in surgery where quality is more

important than speed. These are the kinds of discoveries we hope to make through this project.

Computational Model

Making a physical, remote, haptic feedback-controlled robot for this project was out of our available price range. So, for this project, we made a model to represent the real robot as closely as possible in a virtual environment. The model we developed was coded in the Python programming language version 3.2.2. In the beginning of the program a virtual world is created. The virtual world consists of four different texture types. The textures are danger, dirt, stone, and water. The virtual world was set up on a coordinate grid. The environment is created in a way that causes the textures to be grouped together. This is done with the intent of making the environment more realistic. (See Appendix A, Figure 1) The purpose of the danger tile is to create a space that the robot cannot pass. The danger tiles are supposed to represent a dangerous zone like a fire. Each square on the coordinate grid represents a tile. Each tile has information set about that tile. The tiles all have a texture, x, y, z, and temperature information stored about them. We have not built in any change in the z variable in our model yet although that is something we would think about doing in the future. The temperature of the tiles in the environment is generated by first setting a temperature to a tile based on the tile's texture. Then the temperatures are averaged across the grid so that tiles near danger tiles go up to represent the movement of heat to surrounding tiles. The x and y variables are set based on the tiles place on the coordinate grid. When the grid is created 10,000 tiles are made. The grid is 100 tiles by 100 tiles large.

After initially having the environment be created with equal parts of all of the textures we realized that there were too many danger tiles rendering it nearly impossible for the robot to move anywhere without hitting a danger tile and dying. Thus, the need to control the number of danger tiles. Originally we had the danger tiles set at 25 percent, but that was too high. Then we changed the amount of danger tiles to 10 percent but that was still too high. We changed the model so that it currently maps the world with only 4 percent danger tiles and equal parts of everything else. (See Appendix A, Figure 2) Once the environment

is created, the test type can be chosen and the data can be collected at the end of the test. Once the environment has been completely generated, the model splits into two different parts. (See Appendix A, Figure 3.) When the program begins, the user can choose from these two parts. The two different parts test speed and quality.

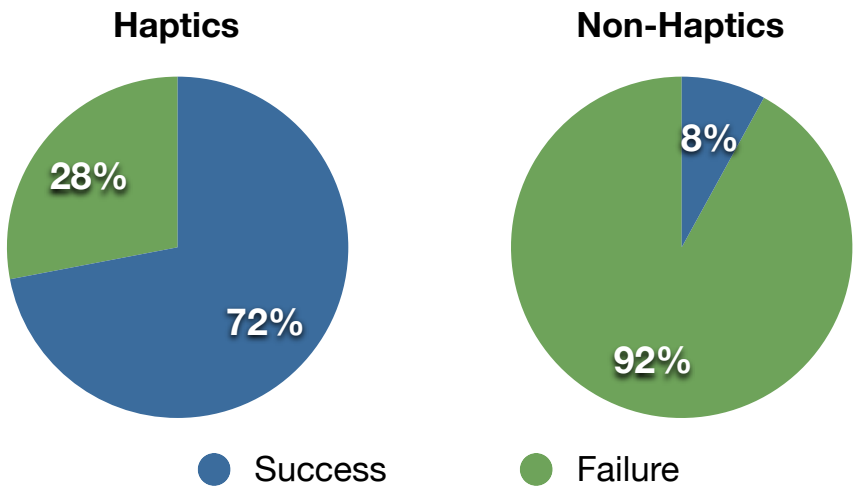
One of the parts determines the effect of haptic feedback on the speed of robotics use. This program tests a haptic technology-enabled robot and a non-haptic technology robot comparing the two robots' speeds to each other. This program works by starting out at a robot's location and a target's location. Their locations are at two opposite corners of the coordinate plane. The robot has to find the target. This is done to represent the behavior of an actual robot. The robot moves around using simple artificial intelligence techniques in which it avoids dangerous places. The artificial intelligence for the haptics robot and the non-haptics robot is the same. The difference between the two robots is their response times. The haptic feedback-enabled robot responds to its environment with a delay relative to the speed of human touch. The non-haptics robot responds at the speed of human sight. These are the independent variables. The results collected for each of the tests that we ran were whether the robot successfully located the target or if it hit a danger tile on the way. If the robot were to hit a dangerous tile that was too hot or some other danger, the robot would die. This would cause a non-successful run. "Time until success" is how long it takes the robot to locate the target. This portion of the program is labelled 'speed' because the point of the program is to determine to what degree haptic feedback technology can either help or hinder robots in dangerous situations. This information gives us the answer to that question.

The second part of our program, the quality part, determines to what degree haptic feedback technology either helps or hinders robots working in dangerous situations. This part of the program works in a different way from the speed portion of the program. This part begins in the same way. It sets the locations of the robot and the target at opposite ends of the coordinate grid. But this program does not use artificial intelligence of any sort; the program runs using human input. At all times, the human receives input that will be the same as the data that an operator would be receiving if he uses a haptic feedback-enabled robot control system. However, unlike an operator of a haptic robot, the user

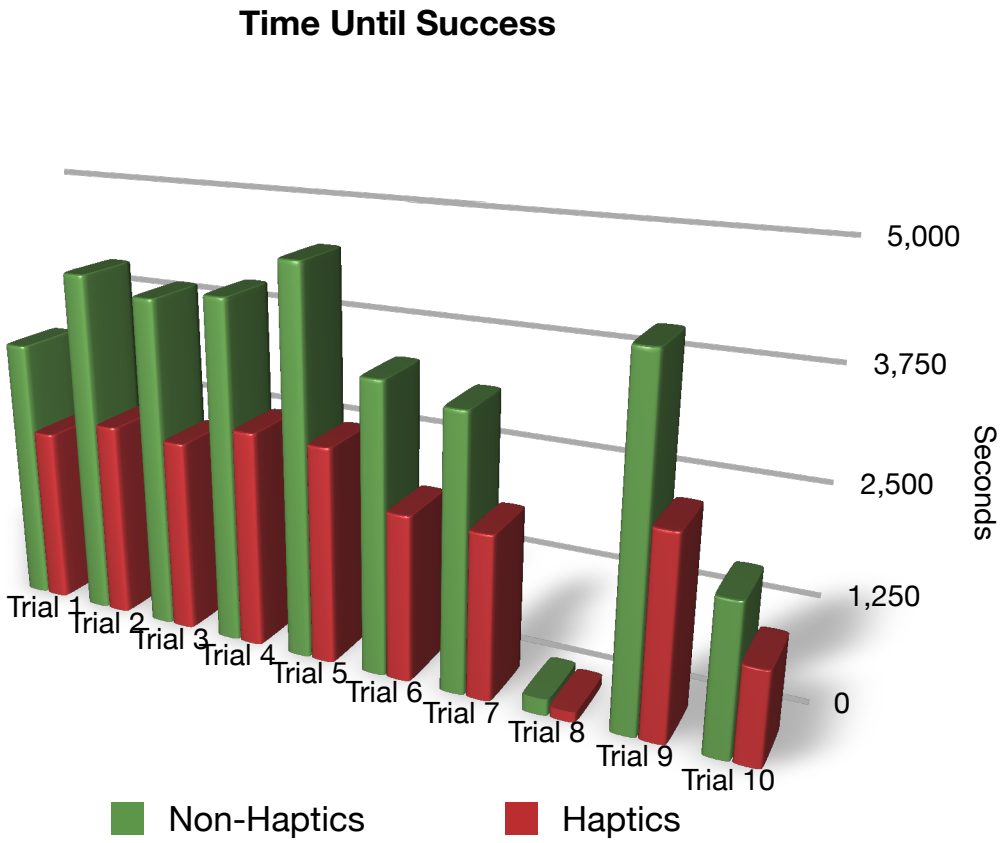
receives the data in a visual text format. The user of the program has the same goal, to reach the target at the other end of the coordinate grid. The data that is collected during the course of running this program is the number of moves. This tells us the quality that the haptic feedback information provides.

Results

For the quality portion of the program, we found that knowing what the temperature of the current tile was greatly improved the results in this type of environment.



A total of 50 trials were conducted with 25 haptics trials and 25 non-haptics trials. Haptics technology was successful 72% of the time while non-haptics technology was successful only 8% of the time. There was an average of 230 moves for the 18 successful haptics trials. The non-haptics took an average of 238 moves. Although this isn't a significant difference in the number of moves, there were only two successes in the non-haptics trials.



The speed portion of the program showed that using haptic technology did have quite a big impact on the amount of time it took the robot to reach the target. For the speed portion, a total of 20,000 trials were run. 10,000 trials of Haptics and 10,000 trials with non-haptics. The average time until success for haptics was 1926 seconds, as opposed to the non-haptics trials which took an average of 3250 seconds. This shows a significant decrease in the amount of time it takes until it succeeds if using a haptics-enabled robots. This is a remarkable amount of time that can be saved by using the haptic technology.

Conclusion

In conclusion, we discovered through running the quality part of the model with many different test subjects, that having a good knowledge of the temperature of an area, and having that readily available to the user, greatly increases the success rate of the robot. By simplifying the problem and applying it to the virtual coordinate grid, we were able to isolate this variable in a way that would be totally impossible if we had built a real robot. Through this discovery, we are able to conclude that haptic technology would greatly improve the success rate

of a robot used in this type of dangerous situation. We also discovered, by running the speed part of the model, that haptic technology also greatly increases the speed at which the robot can locate the target successfully. This speed can have a big impact in this field. Robots used in dangerous situations are usually under a time constraint that makes having a faster robot operator combination important. The increased speed comes from the faster response time of a person's sense of touch as opposed to the response time of human sight. In the speed model, we were once again able to isolate the speed variable in a way that we would never be able to do in a real-world experiment.

Future Work

Some parts of this project could be further expanded. We could go into more depth. The model could be expanded so that more factors were involved than what we limited our project to this year. One of these factors could be adding a change in the z variable with parts that control that generation to make it a natural environment. This would make the environment more realistic. Another thing that we could add to the model would be a visualization more advanced than the current mapping that we have built into our model. All of these things we could add to the model to make it more realistic. Another thing we could do is build physical hardware and actual robots with haptic feedback controllers. We did not include that in this project because of the cost of the equipment. If we were somehow able to get further funding, this may be possible for us to do. If we did make physical robots we would trade the variable isolation that we get in making a virtual model for a more realistic test.

Acknowledgements

We would like to thank the following people for the aid they have given us during the course of the development of this project:

- Robert Cunningham – Our mentor who has helped us tremendously and spent countless hours assisting us throughout the entire project.
- Celia Einhorn – Provide feedback

- Jon Brown – Provided feedback
- Dave Bailey – Python expert
- Christian Rittner – Robotics expert
- Hal Bennet – Haptics technology expert
- Donese Mayfield-Bailey – Editor
- Pauline Stephens – Teacher Sponsor

Bibliography

We used the following sources to help us complete the project.

- Python Documentation – http://www.python.org/doc
- Los Alamos National Laboratory Library – http://library.lanl.gov
- ebooks –
    - http://www.Alamos.eblib.com/patron/FullRecord.aspx?p=264364
    - http://www.Alamos.eblib.com/patron/FullRecord.aspx?p=477262
    - http://www.roblesdelatorre.com/gabriel/VH-OA-MC-DG-GR-04.pdf
    - http://www.sensable.com/documents/documents/what_is_haptics.pdf
- Data for speed tests – http://biae.clemson.edu/bpc/bp/Lab/110/reaction.htm
- Python Book – Python Pocket Reference

Glossary

Haptic – of or pertaining to the sense of touch.

Haptic Feedback technology – Technology that relays feedback to a user of or pertaining to the user's sense of touch.

Tactile – Perceptible by touch or apparently tangible.

Appendices

Appendix A: Computational Model

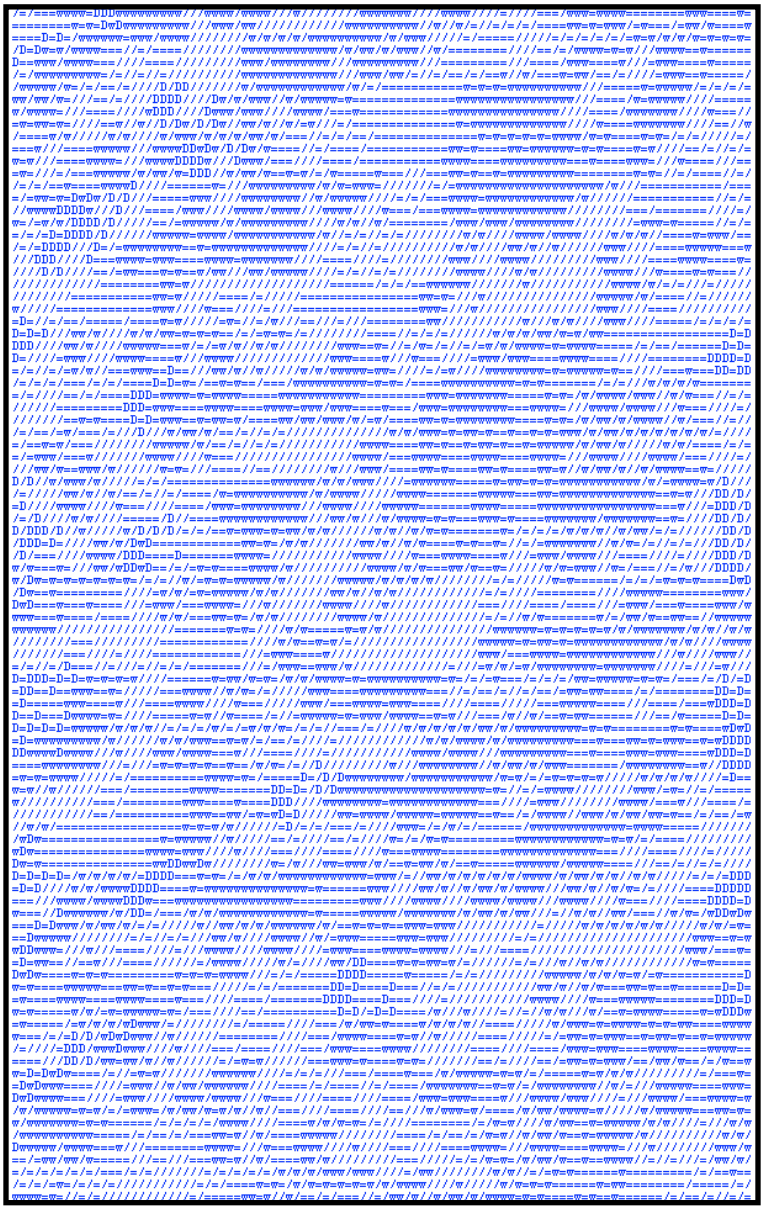
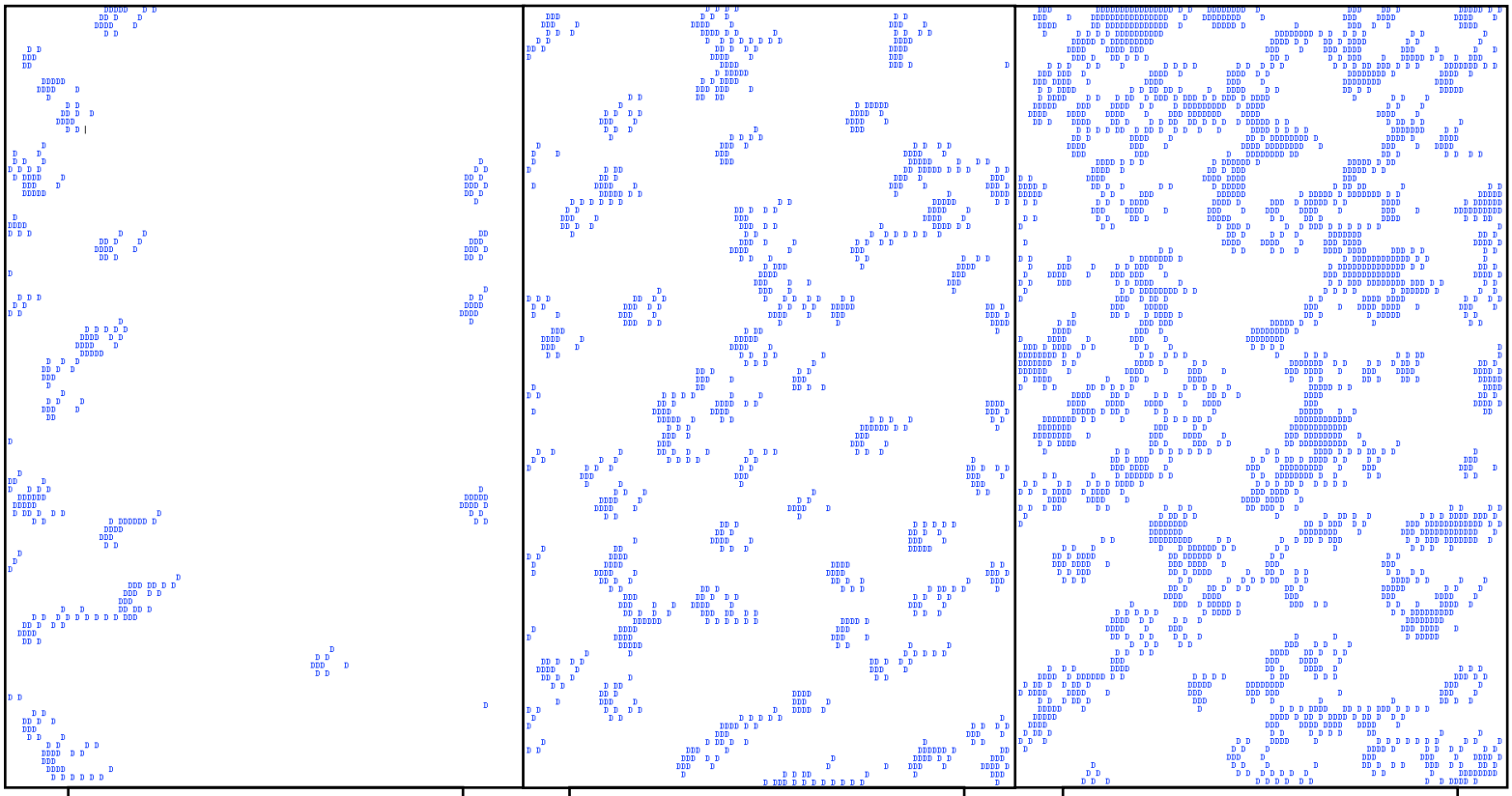


Figure 1:
Figure 1 is an image of an environment map that is generated with the same parameters that we used when we ran all of our tests. Dirt tiles are represented by a "=" mark. Stone is represented by a "/" mark. Water is represented by a "w" and danger is represented by a "D"



Map with the final value 4% danger tile generation turned on.	Map with 10% danger tile generation turned on.	Map with 25% danger tile generation turned on.

Figure 2:

Figure 2 shows three different map type generations. The differences being in the percent of danger tiles being generated. This mapping was done so that it showed only danger tiles unlike the previous mapping. The far left shows a map generated with 4% danger tile generation which is what we finally used for our model. The middle shows 10% danger generation. The far right box shows a 25% danger tile generation which is what the model originally used.

Appendix B: Data

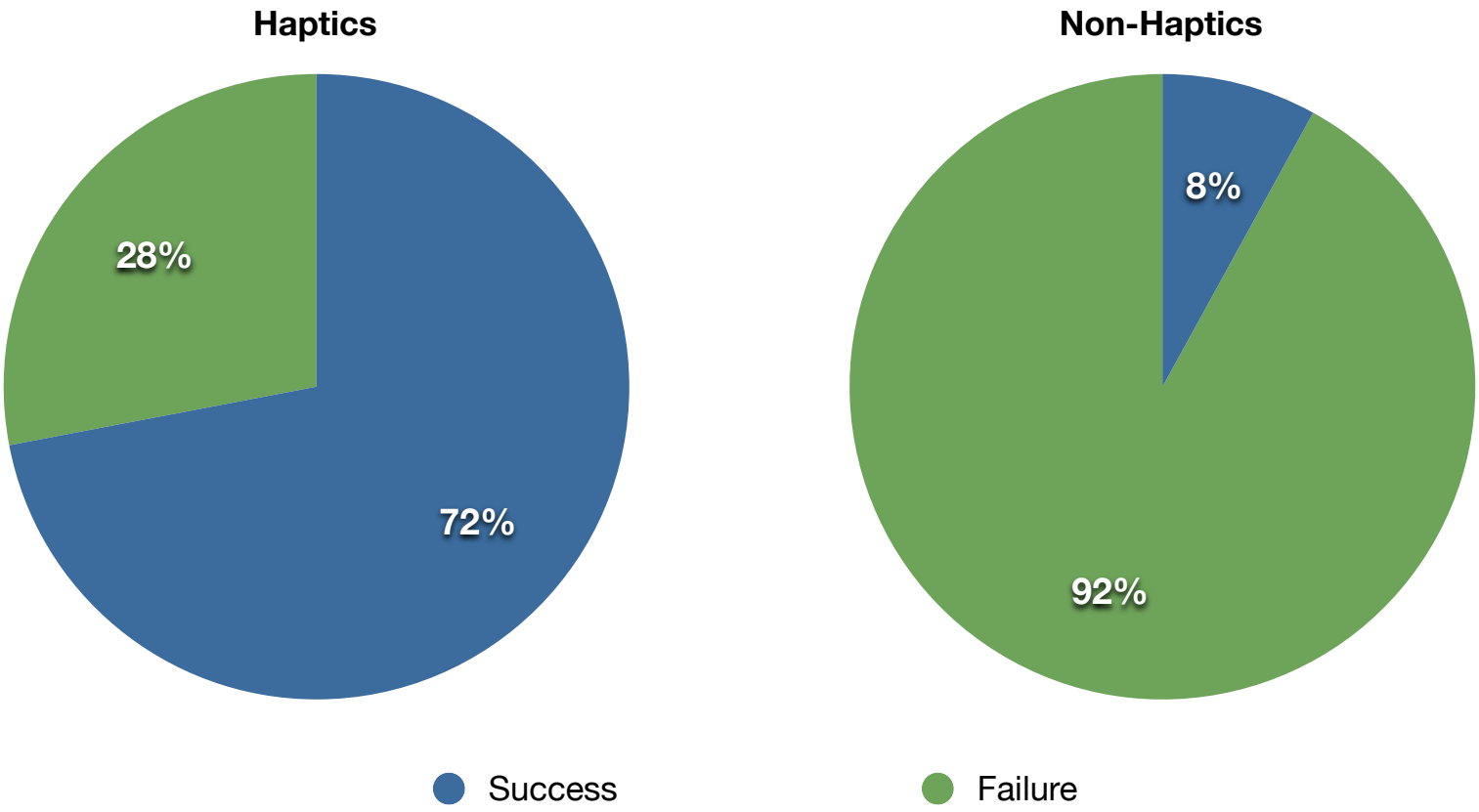


Figure 1

This figure shows the percentage of the times when there was success or failure during the 50 trials that we conducted in the quality test. As you can see, the haptics had a lot more success than the non-haptics.

Figure 2

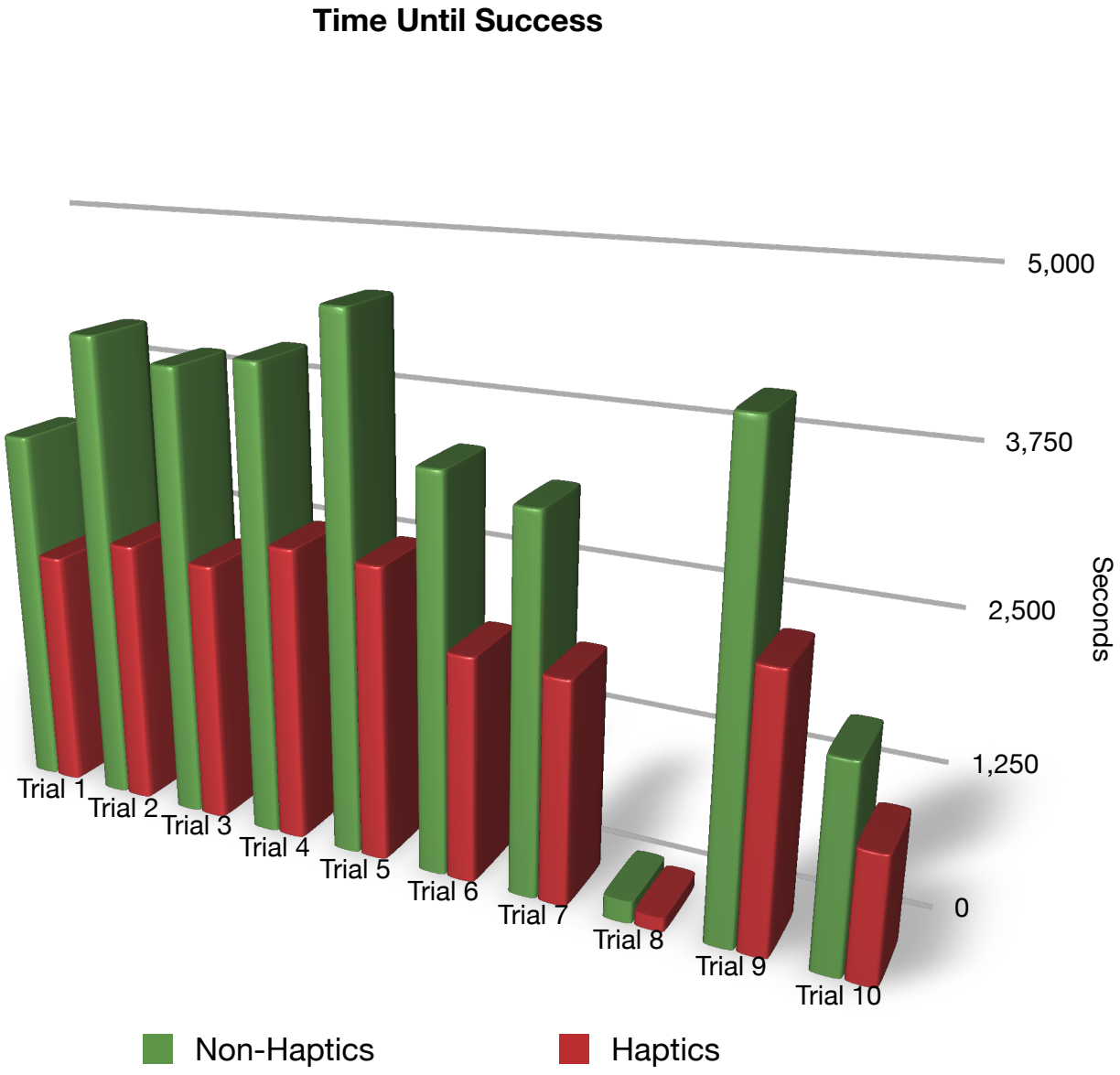


Figure 2 Shows the time until success of non-haptics and haptics tests. As can be seen in the data, the haptics tests are almost always much faster than the non-haptics runs. The trials being compared to each other are of the robot running on the same environment.

Appendix C: Code

```
#  © Connor Bailey and Nate Delgado

import random

field_size = 100
elements = field_size ** 2
tiles = [[0,0,0,0,0] for it in range(10000)] # tiles = [x,y,z,temp,texture]
ratio = [0,0]
delay = [0,0]
moves = 0
regenseed = 0
regen = False
sets = 0
setup = [0,0]
setup[0] = random.choice([303,304,404,403])
setup[1] = 9999
texts = ['danger', 'water', 'dirt', 'stone']
temps = [1750, 66, 55, 47]

def neighbors( xy):
   n = [ 0, 0, 0, 0]
   x = xy[0]
   y = xy[1]
   n[0] = tile( xr( x), y)
   n[1] = tile( xl( x), y)
   n[2] = tile( x, ya( y))
   n[3] = tile( x, yb( y))
   return n

def map():
   s = ''
   y = 99
   x = 0
   while y >= 0:
       while x < field_size:
           m = tiles[tile ( x, y)][4]
           if m == 'danger':
               m = 'D'
           elif m == 'stone':
               m = ' '
           elif m == 'dirt':
               m = ' '
           elif m == 'water':
               m = ' '
           s = s + str(m)
           x += 1
       x = 0
       y -= 1
       print (s)
       s = ''

randints = 0
dangers = 0
max = 4
```

```
def get_rand():
   global randints, dangers, max
   n = random.randint(0,3)
   randints += 1
   if n == 0:
       dangers += 1
   if dangers / randints > max / 100:
       randints -= 1
       dangers -= 1
       n = get_rand()
   return n

def dump_tiles():
   i = 0
   while i < elements:
       list = [tiles[i+j] for j in range(field_size)]
       print (list)
       print ("\n")
       i += field_size

def xr( x):
   r = x+1
   if r % field_size == 0:
       r = -1
   return r

def xl( x):
   r = x-1
   if (r < 0) or (r % field_size == field_size -1):
       r = -1
   return r

def ya( y):
   r = y + 1
   if r == field_size :
       r = -1
   return r

def yb( y):
   r = y - 1
   if r < 0 :
       r = -1
   return r

def tile( x, y):
   if x < 0 or y < 0:
       return -1
   return (field_size * y) + x

def seed_config():
  print('SCC 2012 (Nate Delgado + Connor Bailey)')
  in1 = int(input('Run with seed? '))
  seed = in1
  print('Running with seed "' + str(seed) + '"...')
  random.seed(seed)
  print('Beggining The Configuation - ')
```

```
def xy_config():
  i = 0
  for i in range(10000):
      tiles[i][0] = i % 100
      tiles[i][1] = i // 100
      tiles[i][2] = 1
      i += 1
  print('20% done...')

def temp_seed_config():
  x = 0
  y = 0
  while y < field_size:
      while x < field_size:
          curtext = tiles[(100 * y) + x][4]
          if curtext == 'danger':
              tiles[(100 * y) + x][3] = random.randint(1300,2200)
          if curtext == 'water':
              tiles[(100 * y) + x][3] = random.randint(64,68)
          if curtext == 'dirt':
              tiles[(100 * y) + x][3] = random.randint(50,55)
          if curtext == 'stone':
              tiles[(100 * y) + x][3] = random.randint(43,50)
          x += 1
      x = 0
      y += 1
  print('100% done...')

def temp_avg_config():
   x = 0
   y = 0
   while y < field_size:
       while x < field_size:
           directions = neighbors( (x, y))
           curtemp = tiles[(100 * y) + (x)][3]
           sum = 0
           num = 0
           for i in range(4):
               if directions[i] >= 0:
                   sum += tiles[directions[i]][3]
                   num += 1
           avg = round (sum / num, 2)
           tiles[(100 * y) + (x)][3] = avg
           x += 1
       x = 0
       y += 1
   print("Loading...")

def texture_seed_config():
  x = 0
  y = 0
  while y != 100:
      while x != 100:
          tiles[(100 * y) + x][4] = texts[get_rand()]
          x += 4
      x = 0
      y += 4
  print('40% done...')
```

```
def texture_cross_config():
  it = 0
  itr = 0
  x = 0
  y = 0
  while itr != len(texts):
      curtext = texts[itr]
      while y < field_size:
          while x < field_size:
              while it != 4:
                  directions = [(100 * (y)) + ((x) + 1), (100 * (y)) + ((x) -
1), (100 * ((y) + 1)) + (x), (100 * ((y) - 1)) + (x)]
                  if x != 0 and x != 99 and y != 99 and y != 0:
                      if tiles[(100 * y) + (x)][4] == curtext:
                          tiles[(directions[it])][4] = curtext
                  it += 1
              x += 4
              it = 0
          y += 4
          x = 0
      itr += 1
      y = 0
  print('60% done...')

def texture_fill_config():
  neartexts = [0,0,0,0]
  i = 0
  it = 0
  lol = 0
  itr = 0
  x = 0
  y = 0
  ugh = 0
  while lol != 11:
      while i != 10000:
          if tiles[i][4] == 0:
              while it != 4:
                  directions = [(100 * (tiles[i][1])) + ((tiles[i][0]) + 1),
(100 * (tiles[i][1])) + ((tiles[i][0]) - 1), (100 * ((tiles[i][1]) + 1)) +
(tiles[i][0]), (100 * ((tiles[i][1]) - 1)) + (tiles[i][0])]
                  if (directions[it] // 100) == 100 or (directions[it] //
100) == -1  or (directions[it] % 100) == 100 or (directions[it] % 100) == -1:
                      neartexts[it] = 0
                  else:
                      neartexts[it] = tiles[(directions[it])][4]
                  if neartexts[it] == 1:
                      neartexts == get_rand()
                  it += 1
                  while tiles[i][4] == 0 and ugh < 50:
                      tiles[i][4] = random.choice(neartexts)
                      ugh += 1
                  ugh = 0
              it = 0
          i += 1
      lol += 1
      i = 0
  print('80% done...')
```

```
def cmd_config():
      print('Welcome to ConfigDebuggerCLI!')
      print('Hit "Enter" to skip me...')
      while 1:
          in1 = input('>>> ')
          if in1 == '':
              print('Bye! Have fun executing the rest of the program. ;)')
              logger()
          elif in1 == 'bt':
              config.biome_test()
          elif in1 == 'tt':
              in1 = int(input('Tile #: '))
              print(tiles[in1])
          elif in1 == 'all':
              print(tiles)
          elif in1 == 'ct':
              x = int(input('X: '))
              y = int(input('Y: '))
              print((100 * y) + x)
              print(tiles[(100 * y) + x])
          elif in1 == 'rt':
              i = 0
              while i != 10000:
                  if tiles[i][4] == 0:
                      print((tiles[i][1] * 100) + tiles[i][0])
                  i += 1

              print('Done')
          else:
              print('Sorry, I did not understand that.')

def logger():
  print('************************************')
  print('Running Logger')
  checked = False
  while checked == False:
      choice = input('Do you want to continue these tests ("n" to setup a
test): ')
      if choice == '' or choice == 'y':
          checked = True
      if choice == 'n':
          in1 = input('Test type or debug (q,s, debug, regen, reratio): ')
          if in1 == 'debug':
              cmd_config()
          if in1 == 'q':
              in2 = input('Haptics: ')
              if in2 == 'y':
                  checked = True
                  qh()
              if in2 == 'n':
                  checked = True
                  qnh()
          if in1 == 's':
              in2 = input('Haptics: ')
              if in2 == 'y' or in1 == '':
                  checked = True
                  speed_tests()
```

```
            if in2 == 'n':
                checked = True
                speed_tests()
        if in1 == 'regen':

            print('Regenerating environment...')
            regen(False,0)
        if in1 == 'reratio':
            #print('Previous ratio: ' + str(ratio[0]) + ' successes, ' +
str(ratio[1]) + ' failures!')
            ratio = 0
        if in1 == 'clear':
            ratio = 0
            moves = 0
            sets = 0
            delay = 0
            regenseed = 0
        else:
            checked = False
    else:
        checked = False

def qh():
  print('************************************')
  print('The coordinates for the goal tile are X: '+ str(tiles[setup[1]][0])
+ ' Y: ' + str(tiles[setup[1]][1]))
  ftest = 0
  i = 0
  coords = [0,0]
  moves = 0
  coords[0] = tiles[setup[0]][0]
  coords[1] = tiles[setup[0]][0]
  while ftest != setup[1]:
      print('X: ' + str(coords[0]) + ' Y: ' + str(coords[1]))
      tnum = (coords[1]) * 100 + coords[0]
      print('Type: ' + str(tiles[tnum][4]))
      print('Temp: ' + str(tiles[tnum][3]))
      while i == 0:
          in1 = input('WASD: ')
          if in1 == 'w':
              coords[1] += 1
              i = 2
          if in1 == 'a':
              coords[0] -= 1
              i = 2
          if in1 == 's':
              coords[1] -= 1
              i = 2
          if in1 == 'd':
              coords[0] += 1
              i = 2
          else:
              pass
          ftest = coords[1] * 100 + coords[0]
      if tiles[tnum][4] == 'danger':
          print('Oh no!!! You hit danger')
          ratio[1] += 1
```

```
          print('There have been ' + str(ratio[0]) + ' successes, and ' +
str(ratio[1]) + ' failures')
      print('************************************')
      i = 0
      moves += 1
  print('Wow you won!! Be sure to record the value below!')
  print('It took ' + str(moves) + ' moves to finish this test.')

def qnh():
  print('The coordinates for the goal tile are X: '+ str(tiles[setup[1]][0])
+ ' Y: ' + str(tiles[setup[1]][1]))
  ftest = 0
  i = 0
  coords = [0,0]
  moves = 0
  coords[0] = tiles[setup[0]][0]
  coords[1] = tiles[setup[0]][0]
  while ftest != setup[1]:
      print('X: ' + str(coords[0]) + ' Y: ' + str(coords[1]))
      tnum = (coords[1]) * 100 + coords[0]
      print('Type: ' + str(tiles[tnum][4]))
      while i == 0:
          in1 = input('WASD: ')
          if in1 == 'w':
              coords[1] += 1
              i = 2
          if in1 == 'a':
              coords[0] -= 1
              i = 2
          if in1 == 's':
              coords[1] -= 1
              i = 2
          if in1 == 'd':
              coords[0] += 1
              i = 2
          else:
              pass
          ftest = coords[1] * 100 + coords[0]
      if tiles[tnum][4] == 'danger':
          print('Oh no!!! You hit danger')
          ratio[1] += 1
          print('There have been ' + str(ratio[0]) + ' successes, and ' +
str(ratio[1]) + ' failures')
          logger()
      print('************************************')
      i = 0
      moves += 1
  print('Wow you won! Be sure to record the value below!')
  print('It took ' + str(moves) + ' moves to finish this test.')

def speed_tests(regenseed = 0):
  delay = [0.0,0.0]
  sets = 0
  shratio = [0,0]
  snhratio = [0,0]
  def sh(delay,shratio):
      pretile = [0,0,0,3000,0]
      curtile = [0,0,0,0,0]
```

```
    ftest = 0
    i = 0
    coords = [0,0]
    coords[0] = tiles[setup[0]][0]
    coords[1] = tiles[setup[0]][0]
    while ftest <= 9999 and curtile[4] != 'danger':
        if coords[1] == 99 and coords[0] == 99:
            break
        curtile = tiles[(coords[1] * 100) + coords[0]]
        if curtile[3] > pretile[3] + (pretile[3] * .75):
            choice = random.randint(0,1)
            if choice == 0:
                coords[1] -= 1
                if coords[1] == 100:
                   coords[1] -= 1
            if choice == 1:
                coords[0] += 1
                if coords[0] == 100:
                   coords[0] -= 1
        else:
            choice = random.randint(0,1)
            if choice == 0:
                coords[1] += 1
                if coords[1] == 100:
                    coords[1] -= 1
            if choice == 1:
                coords[0] += 1
                if coords[0] == 100:
                    coords[0] -= 1
        ftest = coords[1] * 100 + coords[0]
        #print('X: ' + str(curtile[0]) + ' Y: ' + str(curtile[1]))
        pretile = curtile
        delay[0] += .12
    if curtile[4] == 'danger':
        shratio[1] += 1
    else:
        shratio[0] += 1

def snh(delay,snhratio):
    sets = 0
    curtile = [0,0,0,0,0]
    ftest = 0
    i = 0
    coords = [0,0]
    coords[0] = tiles[setup[0]][0]
    coords[1] = tiles[setup[0]][0]
    while ftest <= 9999 and curtile[4] != 'danger':
        if coords[1] == 99 and coords[0] == 99:
            break
        curtile = tiles[(coords[1] * 100) + coords[0]]
        choice = random.randint(0,1)
        if choice == 0:
            coords[1] += 1
            if coords[1] == 100:
                coords[1] -= 1
        if choice == 1:
            coords[0] += 1
            if coords[0] == 100:
```

```
                    coords[0] -= 1
            ftest = coords[1] * 100 + coords[0]
            delay[1] += .19
        if curtile[4] == 'danger':
            snhratio[1] += 1
        else:
            snhratio[0] += 1

  in1 = 999
  print('Now crunching large numbers quickly! (Speed Tests: May take a
while.)')
  while sets <= in1:
      sh(delay = delay,shratio = shratio)
      snh(delay = delay,snhratio = snhratio)
      sets += 1
  print('% % % % % % % % **** Results! **** % % % % % % % %')
  print('The sum of the Speed with Haptics Tests were: ' + str(delay[0]))
  print('The sum of the Speed without Haptics tests were: ' + str(delay[1]))
  print('SPEED WITHOUT HAPTICS HAVE GIVEN THIS RESULT: There have been ' +
str(snhratio[0]) + ' successes, and ' + str(snhratio[1]) + ' failures')
  print('SPEED WITH HAPTICS HAVE GIVEN THIS RESULT: There have been ' +
str(shratio[0]) + ' successes, and ' + str(shratio[1]) + ' failures')
  print('Now regenerating to do more tests.')
  print('\n\n\n\n')
  regenseed += 1
  delay = [0.0,0.0]
  regen(regen = True, regenseed = regenseed)

def regen(regen,regenseed):
   if regen == True:
       regenseed += 1
       random.seed(regenseed)
       xy_config()
       texture_seed_config()
       texture_cross_config()
       texture_fill_config()
       temp_seed_config()
       temp_avg_config()
       temp_avg_config()
       temp_avg_config()
       temp_avg_config()
       temp_avg_config()
       temp_avg_config()
       print ("Configuration Complete")
       runlogger = False
   else:
       seed_config()
       xy_config()
       texture_seed_config()
       texture_cross_config()
       texture_fill_config()
       temp_seed_config()
       temp_avg_config()
       temp_avg_config()
       temp_avg_config()
       temp_avg_config()
       temp_avg_config()
       temp_avg_config()
```

```
        print ("Configuration Complete")
        runlogger = True
    if runlogger == True:
        logger()
    else:
        speed_tests(regenseed)

regen(regen = False, regenseed = 0)
```