Robotic Sensing, Manipulation, and Interaction
COPMGX03
Coursework 4: Surface Mapping assignment

Abdelrahman Ahmed Barghouth
abdelrahman.barghouth.17@ucl.ac.uk

26th December 2017

Section 0: Deriving Force in Newtons:

From the force sensing assignment feedback, I have noticed that I can do the force in Newton derivation better, so in this assignment I am using a new formula and I hope that You would give me feedback on its validity. The formula is illustrated with few comments in (FSR_Newton.m) file. I will not illustrate it here in the report as it is not obviously not the main core of this assignment. However, I would be happy if You asked for more explanation of my method.

Section 1: Investigation, Experiment, and Results:

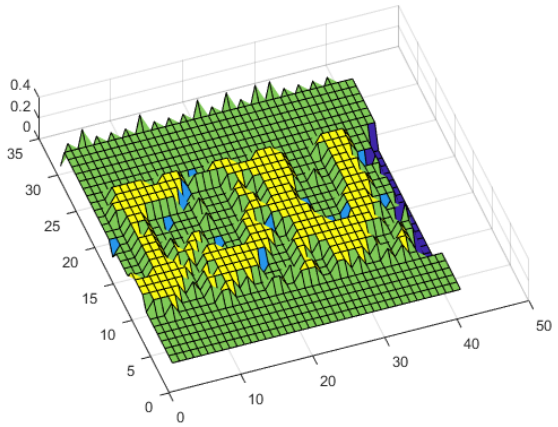
Surface mapping is a very interesting topic as it has several useful applications and also various technical approaches. Even with our touch probe hardware, there are multiple techniques to make a proper surface map. Throughout this investigation chapter, I will discuss just three of them, pointing out each approach pros, cons, variables to be controlled, procedures, usage and of course results. But briefly I will give a hint on the methodology that led me to consider these 3 approaches first.

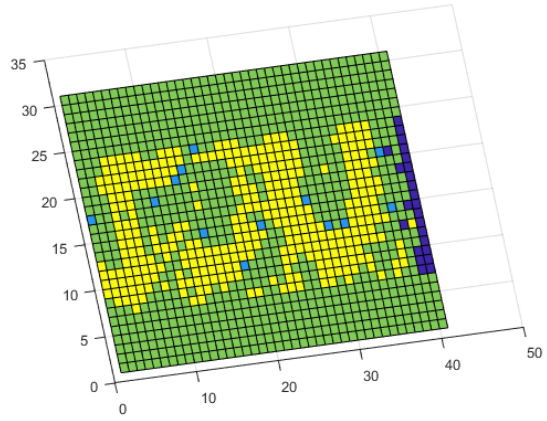
In order to measure any external parameter as for us is surface height, we have to consider our available sensors and what they can allow us to do. In our touch probe, we have FSR and a camera. Using camera will probably give a 2d description of the surface, we are trying to map. However, it is not a depth camera, so we can not get the third dimension which is the height. Hence, camera is not our main soldier in this area, although it can be very effective and have an important role, as I will show in the third approach. Covering the capabilities of the camera, leaves us with the other sensor which is the FSR. The force sensor can detect, contact with other surface, and also pressure. In our application, we theoretically only care about contact computing. That's where the first approach comes out from. I will use the servos, to pass each point to the FSR and ask the FSR connected at the end of the touch probe stick to find out if touches an object or not. Once it detects contact with a surface, I measure the altitude of the probe stick to calculate this point height. So basically the FSR is used here as a limit switch, when it touches a surface, it gives a feedback and so we can detect the surface features. While working in this approach, I have noticed that the FSR gives higher force values when pushing higher surface, within the same probe height. This observation has inspired the second approach. The second approach is like a modified version of the previous system, benefiting from the characteristics of the plastic ground we have, which has tiny difference between ground and its engravings. All I needed to do is to push the touch probe all the way down and calculate the pressure value, the higher value means, higher surface. The output is also refined to be either base or engraving, no other options. Obviously this approach is going to work only on the conditions we have, which is only 2 altitudes in the playground with a very small difference between them. My third approach, which I consider as the most feasible if this project was meant to be used in the market. The approach is trying to cover the flaws of the 2 previous approaches and also add higher certainty to what is the real map for the surface we are working with. The approach is using edge detection from the camera, and then passing edges to the FSR to detect the height of each point. This means that I am putting the FSR at points, that I already know that they are different than the points surrounding. After detecting the edges height, by poking the

surface as was the case with first approach, detecting the heights for non-edge points becomes the new issue to consider. These details and a more in depth coverage for each approach is in the coming paragraphs.

The first approach is a simple blind scanning for the surface. I call it blind as it does not use any visual sensors like camera. For this I edited my GUI from the previous assignment to enable me choose either blind approach or use camera approach. While I switch between 1st and 2nd approaches, which are both blind, by changing the code due to space limitations in my GUI. For the first approach I let the probe start from (0,0) position in the x-y dimension and then start poking the surface. I let the probe stick with the FSR attached at its end to go step by step down and sense the contact with any surface. After touching a surface, I take the height of the probe, and compare it with the previously calculated height of the probe when touching the basic playground, to detect the height of this point in surface. Although it might seem like a straight forward task, but there was some variables that has to be taken in consideration in order to get reasonable accuracy. First of all the vibration in servo motors. When moving the probe, each servo has its own noise and so it vibrates which affects the accuracy of the probe position. For solving this, I tried to put delays in my motion, in order to decrement the consecutive motion and give each motor some time to saturate and stabilize. This methodology enhanced my probe accuracy, and the only problem was when the probe is touching surface, it tries at first to get out of its dedicated slots in the hardware, and this caused the x motor also to not be very accurate. For this I attached, a small piece of paper in the empty slot which allows the servo responsible for z-axis motion to vibrate. I also attached my x-axis gear track with the ground of the hardware, using a white plastic piece like the one with wires. These enhanced my vibration issues, and allowed the probe to have a proper motion and sensitivity. On the other side, the FSR itself was not sensitive enough due to noise that can happen, from motor vibrations, that has decreased but still exists, it is also non sensitive as some readings are not accurate enough and this caused a wrong prediction of the surface. In order to avoid this, I took 10 samples of the FSR readings with a time delay between each of them, to ensure that readings are distinctive from each other. I calculate the average of these 10 samples, and then based on this calculation I take a decision whether it is in contact or not. Moreover, the wiring especially of the FSR sometimes affects its sensitivity, as the sensor rotates depending on the probe position. To make constraint for this unwanted orientations, I attached the wiring of the FSR to the horizontal support that carries the z-axis servo.

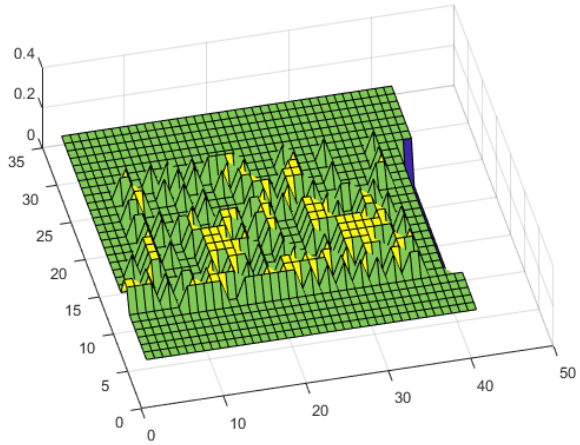
For this 1st approach, I have developed a callback function with 4 inputs and surface plot as an output. Two of the inputs are publishers for xy position, z position, in order to control the servo motors. The third input is accuracy, which is the step needed for the probe to move with. The accuracy is 0.1-1 cm range. This accuracy is chosen based on the surface we are trying to map. If this surface like ours has tiny details, I tend to decrease this accuracy to 0.1, or 0.2. While it is possible in some other surfaces, I will need large steps to save time as the surface is not very detailed. The final input is the Newton subscriber to get the force readings from the sensor. In order to run this approach, I use a button in my GUI to call the callback function I developed and then the probe starts moving over the surface and plot a point by point in a mesh grid. This approach is very simple to implement, can control speed of the probe step depending on the surface features, it is also using limited resources which is just the FSR and the servo motors. On the other hand, this approach is time consuming, as it takes average of 1.15 hours to map a 3x4 cm ground, it also depends on the accuracy of the servo motors to locate the probe in the appropriate position, although the servos are oscillating and not very accurate, the third drawback of this approach is that it depends only on the FSR readings, and take it for granted and do not use any other methods to double check this readings. Obviously this approach has some cons, however it gave a reasonable output for the surface as shown in below figure.

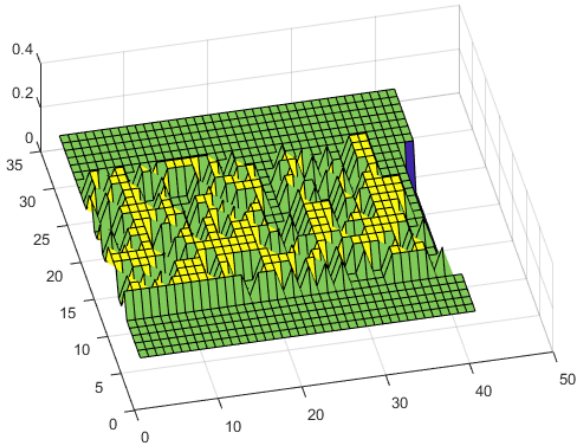




There are some points that are not very correct as for example the ‘C’ is just resembled by two non-complete columns. This happens because the probe is sometimes sliding at the surface edges due to the silicon bubble placed over it. In my ground the ‘C’ is very curvy and so the probe could not detect its edges, and could only detect its centers. There also some errors as was in the area between ‘L’ and ‘C’. However, We can see that the map is almost clear and shows the details of the surface despite the errors. In a nutshell, this approach has proven to be reliable and its output is a good indication of the surface characteristics. This approach is easy to implement, using few resources, but it has some errors that can lead to false results in more complicated grounds. But the most clear con was the time consumption of this approach. Hence, I tried to solve this issue using the 2nd approach.

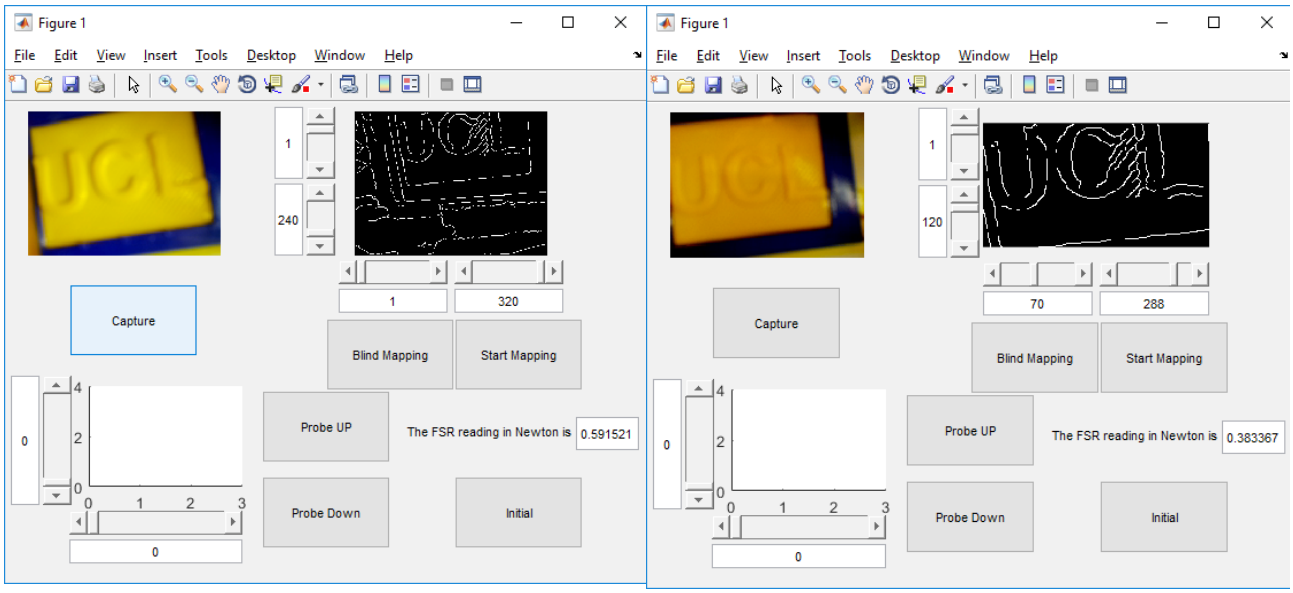
The 2nd approach is also a blind mapping, which means without using camera. The major difference between the 1st and 2nd approach is that the probe is not detecting contact or not, otherwise it is computing the amount of pressure exerted when pushing its maximum down. This approach is much faster as it saves time in reading sensor at each height and take a decision whether there is a contact or not. Alternatively, it pushes the probe all the way down and compute the force, if the force is high it means that the surface was high and so a high resistance from the surface is exerted to the probe stick. Using this criteria I could finish scanning the surface in a very small time which is 17 minutes, compared with 75 minutes taken by the first approach. Other than this, the 2 approaches are almost similar. However, the results were not as accurate as was for the first approach as shown in figure below.





The resulting plot does not show enough details, although the letters can be barely recognized. The result on the left is using a force of 2.8 Newton as threshold, while the one on the right is 2.4 Newton. We can see that result on the right has detected more parts in the 'L', 'U' letters, but more points were detected as engravings although they are not. From my perspective, this approach can reach better results by trying to find the appropriate threshold. But I can also see that it is hard to reach accuracy of the 1st approach. Finally, this approach is almost the same like the first but it is faster, and less accurate, so it can be used in applications where time is of essence and map needs only to get basic characteristics of the surface and not tiny details. For example if needed to detect if there is an obstacle in the environment or not and the approximate position of this obstacle.

Now for the third approach, I used a camera in order to have an initial prediction of the surface needed to be mapped. In order to use the camera, I have added some buttons and sliders in my GUI as shown in figures below. I added 3 new buttons, 4 sliders, and one image axis. One button is called 'Blind Mapping', which is responsible for running the previous 2 approaches.



In order to take capture of the probe playground, I removed the z-axis support and placed the camera in its place to have a clear view. Each figure from the above has 2 axes for image show. The image axes on the left is the raw rgb image taken by the camera. After the user adjusts the camera and the playground using the servo sliders, the user should click on the 'Capture' button to get the image and transform it into greyscale image and detect edges using canny edge detection algorithm implemented in MATLAB. The image quality is a very important factor in this approach, so I tried to put some extra lighting to the playground as shown in the figure on the right. After capturing the image, the lighting is not essential any more, so the figure on the left appears with less lighting (not intended). The only issue that I could not figure out is that the image coming from camera has the playground totally centered, while the edge detected image in the right side axes, appears to be slightly going above. At first I thought it was because of the vibration that happens when I press capture that the camera moves slightly, but I tried to press gently and the same results came out. After edge detection, the processed image is displayed on the axes on the right. After that the user can start editing the area of interest using the sliders. The cropping of the image is shown between the figure on the right and the figure on the left with respect to the image axes on the right in each figure. After cropping the area of interest, the user press 'Start Mapping' button.

The mapping of this approach is taking some techniques from the first approach, which is poking the surface and detecting its height. However, this approach is not poking each point. The probe is touching the edges only. The edges are the white points which is over 200 in the greyscale image. In

order to achieve that, we have first to convert the image into coordinates to figure out which positions, the probe should poke. The button calls a callback function, which initially reads the edge detected image and put it inside an image matrix starting from (1,1) to (width_cropped, height_cropped), instead of dealing with a whole bunch of numbers (width_start, width_end, height_start, height_end). Taking the new matrix image and use nested for loop to get access to each pixel and detect if this pixel has a value higher than 200 or not. If this pixel has a high value this means that this pixel is an edge and so we need to relocate the probe to calculate its height. Using simple interpolation, the (x, y) coordinates can be derived and then the probe starts poking as was the case in the first approach. If this position is already detected by previous edge pixels, the algorithm skips this point. After the detection of the height of all pixels that resembles edges, the prediction of the other points in the mesh grid starts. Another nested loop to access each point in the grid, is implemented. For each pixel, it takes the average of the heights in its neighbourhood. Although it is not the best way to detect the actual height for this point, but it worked well for our small playground. Unfortunately while trying to enhance results, the z-axis servo stopped working and I could not provide a figure to show results for this approach. I am really sorry for that, I will try to fix the motor and my code on it, or You can try it if You want and see the difference in results. To finalize this approach, I will talk about results that I have seen and could not prove its existence. This approach is using much more resources than the previous 2 approaches and needs some preparation for the image to be useful, but it uses 2 sensors to double check the features of the surface. Furthermore, it is not time consuming as it takes around 43 minutes, which is much faster than first approach, slower than second, but no doubt that the difference in accuracy is observable.