24/1

basic setup was done today of installing inventory on both my desktop and a laptop.

Problems

solutions

laptop

* Initially ubunto was unable to detect my SSDI laptop due to raid being enabled in the BIOS settings. Simply raid the BIOS settings fixed this problem but did require the wiping of the SSD drive meaning there is currently not a Windows installation on a laptop
* secondly inventory was unable to install fully on a laptop due to **acpi conflict. This was fixed by during boot pressing e and adding acpi=off after quiet splash**
* https://askubuntu.com/questions/861743/installation-of-ubuntu-16-04-from-a-usb-drive-freezes

Desktop

• installing inventory went off without a hitch with Ubuntu installer creating a separate partition for itself on the SSD this did however result in corruption of my windows account

to resolve this and new **account was created and basic functionality transferred to** the new account the old account was then deleted

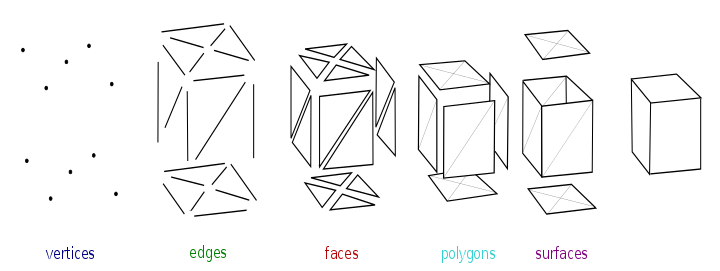
25/1

setup the software drivers for the connect using the following tutorial (link). The point cloud it was then able to be saved to a file. Programmers and also created to take 50 snapshots of the death depth map and average them out using the mean average. Work was also done in looking into how to generate a point cloud from a 3-D model generated by a slicing software. I also found a website that offered our solution to performing operations point cloud inside a relatively convenient package. Downside is that all and c++ and the Peyton wrapper is not currently compatible with inventory only Windows.

Set up klinect software

<https://naman5.wordpress.com/2014/06/24/experimenting-with-kinect-using-opencv-python-and-open-kinect-libfreenect/>

how mesh store data

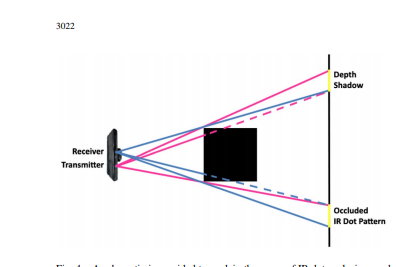


Problems

Solutions

It turns out the regular something called an IR shadow in which a duplicate of an object is created. This is caused by the fact that IR projection matrix projected out into the world the sense debt can be blocked by objects resulting in the camera not being able to see any dots in a certain region resulting in asset value being returned with the current form of image capture using an image it looks like a separate shadow of the object.

The only real solution I found this problem it to remove the shadow from the depth map. The shadow intensity is directly proportional to the coarseness of the sensor to the desired object but even at a significant range of the shadow is still present



It turns out the scanning data received by the Kinect sensor is not a perfect image and has some fluttering and fluctuation on the outer edges of shapes especially those that have complex surfaces such as a ball of wires.

To solve this program was created averages out a pixel intensity a more sophisticated approach is recommended as this could pose significant problems in comparing a virtual model with a realistic model.

Ply to point cloud

https://medium.com/@daviddelaiglesiacastro/3d-point-cloud-generation-from-3d-triangular-mesh-bbb602ecf238

It turns out that generating a point cloud is a relatively simplistic task but is full of potential problems. Sampling of a mesh which is the most popular output format for virtually all 3-D models can result in a point cloud. But a relatively sophisticated algorithm is needed to correctly create the point cloud from the mesh due to the way that meshes store information on the shape and simply asking for the average of all points across the mesh results in a high concentration in areas of high complexity within a shape.

<https://en.wikipedia.org/wiki/File:Mesh_overview.svg>

A few useful links were found in how to convert these meshes into point clouds but again questions about the final results of the point cloud are questionable

Decisions that need made

The decision is made here on whether or not to look into following tutorial guide into how to turn our meshes into point clouds or to whether performer own analysis and generator program that is capable of generating point clouds from decode

26/01

For today I decided to focus on the principle of creating a point cloud from the decode by myself this reduces dependency on external libraries that may have negative influences than not aware of as well as personally being easier to visualise.

I then created a program that is capable of measuring two points to determine what the smallest distance of movement is conducted in the bar test this distance was.

The result was 0.016031219541872845mm on one of the curves at the end of the bar

As well as fixing some minor bugs in the code to do with dealing with situations whereby the printer would be simply moving in the Z axes resulting in invisible line that not present in my 2-D representation of the G code.

My plan is for the sequel to point cloud program is the court will simply take every line and move along said line recording the points present on that line at 1 mm intervals if the line as below as to be smaller than 1 mm it will simply take the start and end point and save them points. This could still need to be implemented and it could prove that a smaller interval sampling is needed when looking along the line.

27/1

today I successfully completed my work on the Jeep record to point cloud program is capable of generating a point cloud that reasonably resembles the object even with relatively small details. I’m not worried about the lack of detail that the point cloud contains as it is unlikely the connect will be able to capture any fine details either. To capture more detail all woman need to do is decrease the steps are taken in the for loop that goes through the G chord lines to create points.

Suggested solution

Problems

The program that creates the point cloud from decode at present does not take into account situations with a printhead is simply just moving to a new location and is not actually depositing material. Meaning there are situations when lines are created across the model that are not actually trying to be printed in real life.

Improve the filtering, G chord processing part of my application to remove all G code that does not involve the depositing of material

There also exists the creation of unusual points around the model in the visualisation of the point cloud.

Create a program that looks for points that are completely separate from the rest of the main body

or that are not close to a significant quantity of points

28/1

continuing on from yesterday’s worker program was created the isolated individual groups of points this truth proved to be a very difficult task as firstly the program initially performed a surgeon said a certain radius but defining the exact radius prove problematic. So instead I’m using the Matlab plot function to plot all the points so I can look at the grid and approximate the position of the problematic points.

A bug was also discovered in the point generation part of the code whereby the two points defined by the program to search between with previous strategy beating to round down the X1 point roundup next to point proved to be an ineffective solution as it resulted in negative numbers as there were certain lines in the G chord that they quickly became negative so going slightly to the left of the X1 point resulted in huge negative values.

Further modifications to the code were also conducted with the removal of the random points that were caused by a domain while loop in the code that looks along the line to generate points going slightly past the end point due to the fact that while loop only checks for the condition at the start of the loop and any changes to make the variable does not result in an early exit.

A simple filter was also implemented with the same basic principles of the cold start problem whereby the printhead needed moved into the correct position. This solution now also makes the calls are problem redundant and the code are necessary.

At this time it is considered a good idea to re-design the code into a more manageable format as the current format is full of redundant cord and a bit of spaghetti code which is prove problematic in the future.

At present the code seems to provide a very good conversion from G code into a point cloud in two dimensions work on the three-dimensional point cloud still needs to be developed.

Problem

Solution

Negative positional values were given in the point cloud that I generated

instead of using a round down strategy for X1 roundup strategy for X2 of while loop was instead used allowed for a simple iteration whereby X1 is increased gradually until it becomes greater than x2 this resulted in no more negative values

The move commands that were present in the G chord would

cause the program to include them as part of the model when in fact they were simply the printhead moving to a new location and not actually depositing filament.

A new filter was added that allowed the removal of all move commands while simultaneously preserving the model as they could not simply just be removed but required that the endpoint of of simple move command still be processed at the start point of a filament deposit.

Random points being generated from the conversion of the G chord to the point cloud

A slight modification into how one one of the four loops operate and slight redesign as explained above

1/02

Work was done in looking into how to visualise a 3-D point cloud that we are generating. Initial the Matlab plot function does offer a 3-D virtualisation of points with rotation but sadly has too much overhead and resulted in a virtually unusable plot or snapshot of a single angle with possible any attempts to rotate the image resulted in severe lag and jumping around of the image. Instead the open3d library was used allowed for the generation of 3-D graphics and easily handles the requirements for displaying a large quantity of point clouds. I also began rewriting the code that creates the point cloud as I feel in its current form it is very confusing full of unused functions and hard to follow. Sadly though I discovered that the new version that implemented produces a different set of results and further debugging is needed on the new version. As point seem to be missed in the new version and other points and to randomly end added. Additionally the new version of the G chord to point cloud now is capable of turning the entire G chord into a three-dimensional point cloud where the old version could only handle to 2d

Problem

Solution

The code was currently unreadable

the court is rewriting using classes and functions to make it more legible with encapsulation of variables

the code only generated for a point cloud one layer at a time

additional chord was added the alleged the processing of 3d

there was no 3-D virtualisation of the code

a library was found offered 3-D virtualisation

The Matlab function was unable to display a large quantity of points efficiently

a more efficient libraries found offered 3-D virtualization in this case the open 3-D library

The new version of the code did not provide the same results as the old version with the old version providing a more correct point cloud.

02/02

Today all bugs were fixed in the G chord to point cloud converter with a note providing a much more satisfying result is still the question of the weird missing points in the smaller sections where vertical lines are present but this seems to mitigate it with increased sample size perhaps dynamic sample sizing is needed but at this time I am sufficiently happy with the detail and quality of the point cloud being generated.

Work was also done in setting up multiple Kinect sensors to run and take multiple images of an object or some unusual reason only one USB port on a computer will allow a connection to our Kinect sensor that being the one on the front panel none of the back ports would allow for connection to one of the sensor. Some could also added that allowed for virtualisation of the point cloud generated by the Xbox sensor into the same point cloud virtual stagnation that we are using for our point cloud that is made up from the G code. From this it is clear that some level of sanitisation is needed from the Xbox is point cloud as a large volume of point is created making a hard to actually identify where the modellers. The image quality received from the Kinect sensor is okay but I believe that the idea of averaging out multiple frames is a bad idea due to fact it allows for random variations that been picked up by the sensor to be amplified and make them to appear like actual object instead of just random noise.

Also a general cleanup was performed to reorganise folders with early versions of scripts and test parts the script moved into a separate folder away from the main folders to allow for a more organized look to the folder structure