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

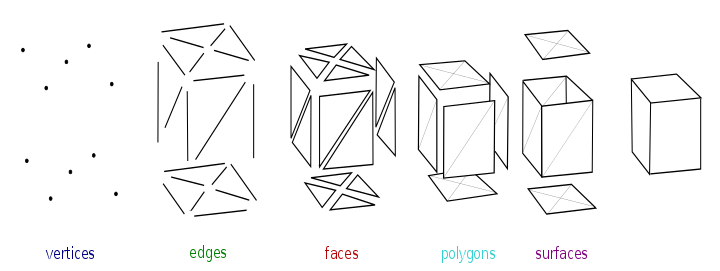
25/1

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

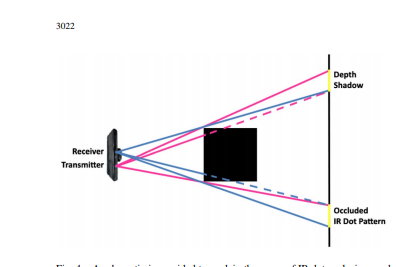


Problems

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.

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.