PRELIMINARILY RESULTS

When setting up the code to find the normal vectors I first found the vectors to be very noisy. Vectors over a series of frames were found to be pointing in different directions for the same flat surface. This equated to a very wide spread of vectors when centralized, seen in Figure 1.

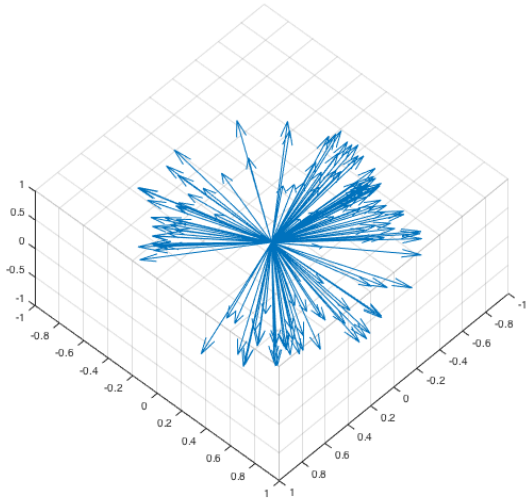

Figure 1: Noisy normal vectors centralized of a box

The filter introduced was a rolling average filter. This summed each individual element over 15 frames and then divided the sum by 15. Every new frame was added to the sum with the oldest subtracted. Adding a loop to take in depth values at the start of the program allowed the camera to settle and give more accurate values. Taking all the depth values for the data at the start of the program was also added to stop this same problem, as the values taken were in the same loop that did the calculations. And because the calculation loop didn't run in real time the new depth values at the start of the next loop would be slightly wrong. With the new filter and loop structures the vector noise reduced greatly. Figure 2 with a similar object, as in Figure 1, shows multicolured box (point cloud) and filtered vectors in green.

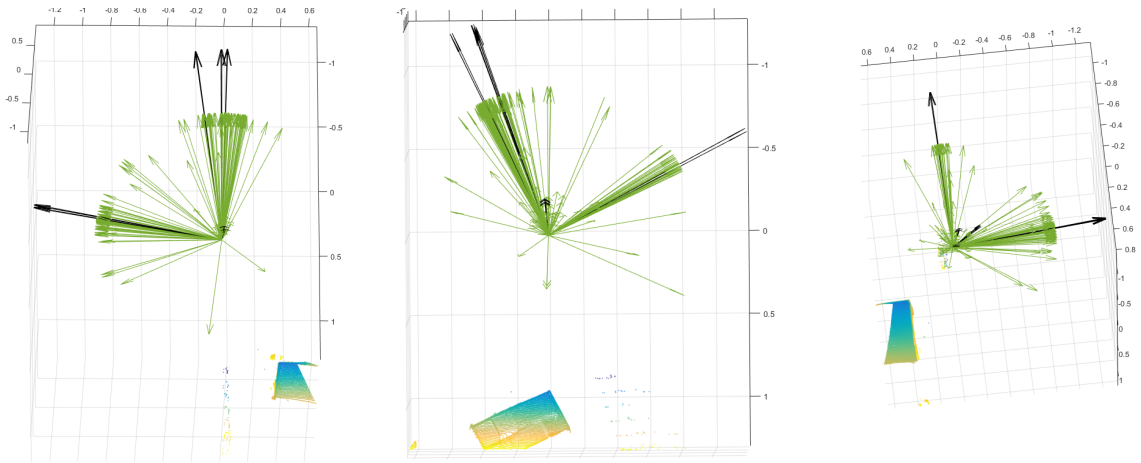

Figure 2: Normal vectors to point cloud, left left view, middle top view, right right view

With the frames taken set to 31 all filters with buffer of 15 frames, and 5 frames computed the following times were recorded.

Time to save 31 frames 3.129910 seconds.
Time to filter 31 frames is 9.848604 seconds.
Time to compute 5 frames 3.949866 seconds.

This result shows that the system would not be able to run real time on my laptop, which was expected, and in fact could be a huge improvement on the summer projects method which took 1 hour to process 120 frames. My program still has more calculations to be added but is vast improvement.

Iterating over all the vectors, the vectors with lots of close matches were found and in Figure 3 can be seen as the black vectors. These give an approximation for all the other normal vectors. This approximation calculation hasn't been finished yet but has given the following readings in Table 1.

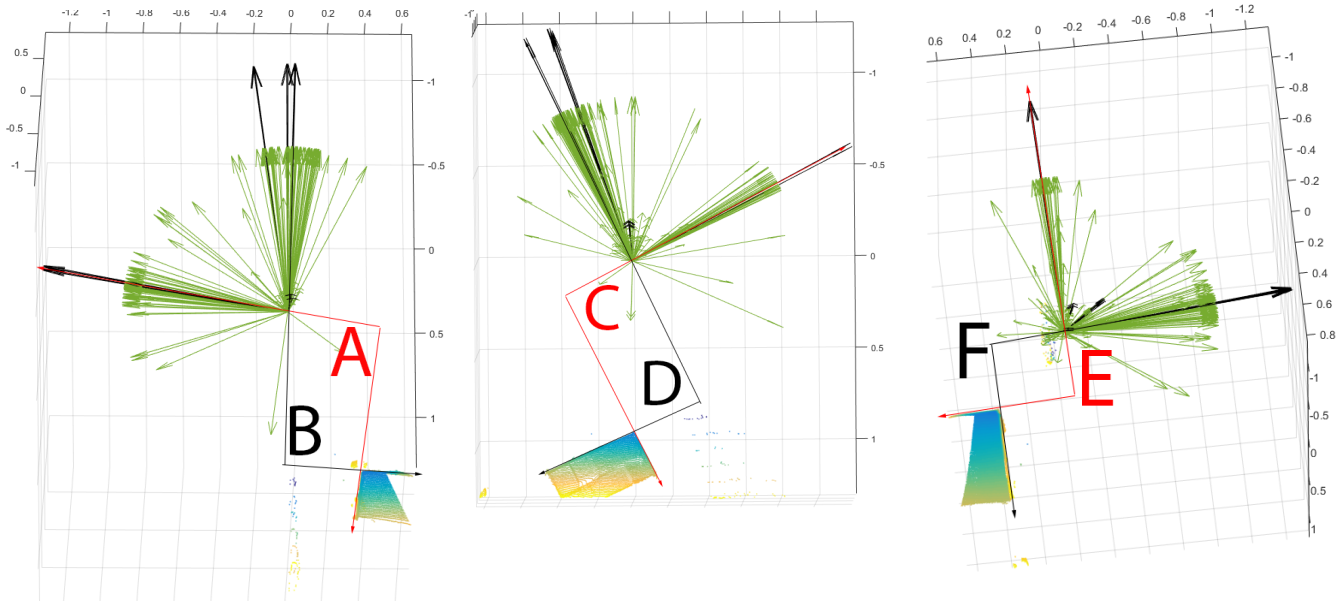

Figure 3: Normal vectors from object, left left view, middle top view, right right view

Angle	A	B	C	D	E	F
	92.2	91.8	91.3	88.6	90.0	92.7

Table 1: Normal vectors from given shape

At present the limitations of the program are if you place an object only with curved faces. This will through the program off and give wrong results.