DEPTH-MAP FUSION IN THE PRESENCE OF VIBRATIONS

Alexandre Veuthey, EPFL

Supervisor: Marjan Shahpaski, IVRL

ABSTRACT

Depth-map fusion is the process of fusing (averaging) multiple similar observations of a scene in order to decrease the noise that is present in the individual depth maps. Consequently, it also renders a more detailed scene.

In this project, we would like to investigate if having vibrations during the depth scanning can be used for improving the quality of the measured depth. The project will focus on the selection and implementation of suitable depth fusion algorithm(s). The depth maps will be acquired by a Kinect or a structured-light system. Finally, the output of this system will be compared to that of a static depth scanning system.

Keywords— Depth-map, fusion, registration, noise removal, Kinect, SIFT, ICP

1. INTRODUCTION

Several algorithms that solve simultaneous localization and mapping (SLAM) are currently available. These algorithms work in either an online [1], [2] or offline manner. However, an important assumption that these algorithms make is small movement between two different viewpoints. This assumption is equivalent in the online approach to having a high framerate, and in the offline approach to having access to a good estimate for the transform between two viewpoints.

Various reasons made us choose an offline approach. Firstly, the Kinect device has a limited resolution in both depth and color cameras, meaning that the depth estimation of the scenes will be imprecise. Secondly, a structured-light system cannot output data at a fast enough rate. However, its depth estimation is much more precise and will allow reconstruction of a more detailed scene. Lastly, as occlusion in scenes may mask data from a first viewpoint, we need the following viewpoints to form a large baseline with the first, thus breaking the small-movement assumption.

One particular algorithm, Iterative Closest Point (ICP), first proposed in [3], is a quasi-standard for registration and is widely used in current applications [1]. The simplified method is the following: transform a *source* point cloud iteratively until it matches a *reference* point cloud, thus minimizing the euclidean distance between the two point clouds (defined point-to-point or point-to-plane). However, the method relies on finding corresponding points between them, which is traditionally done by using the closest 3D point. This might throw ICP into a local minimum in some cases, particularly when noise is present. ICP alone is thus not powerful enough for viewpoints with large baseline.

The need for better correspondences between the point clouds led us to use keypoints obtained with the help of the Scale-Invariant Feature Transform (SIFT) method. As the name suggests, the algorithm detects features in objects in a scale-invariant manner, meaning in our case that a change of point of view does not impact the feature recognition. In terms of required processing in the context of a structured-light system capture, this only means taking one more RGB image with the projector turned off. This image is then fed into SIFT to establish correspondences, whose 3D locations in the point clouds are then used to estimate a coarse transform, used as an initial step for a more detailed ICP processing.

Section 2 presents the data we worked with, along with a general description of how it was captured. Section 3 details the steps that form our solution. Section 4 shows examples of depth-map fusion while section 5 is a discussion about the examples. Finally, section 6 proposes several applications for our solution.

2. DATA PRESENTATION

Two different ways of capturing depth information were used during this project: a Microsoft Kinect V1 for Windows, and a structured-light system comprised of a synchronized Canon Rebel T1i camera and a Acer H6502BD projector with Phase-Shifting profilometry depth estimation technique [4]. Note that both systems use structured-light and thus come with the same limitations when it comes to this manner of estimating depth. We won't cover these limitations in detail here, as it is beyond the scope of the project.

The Kinect is well known for being an affordable and widely available 3D scanning device. It has the benefit of streaming data at 30fps, both in color and depth images, which can be reconstructed as point clouds with color information. However, the resolution in both cameras is

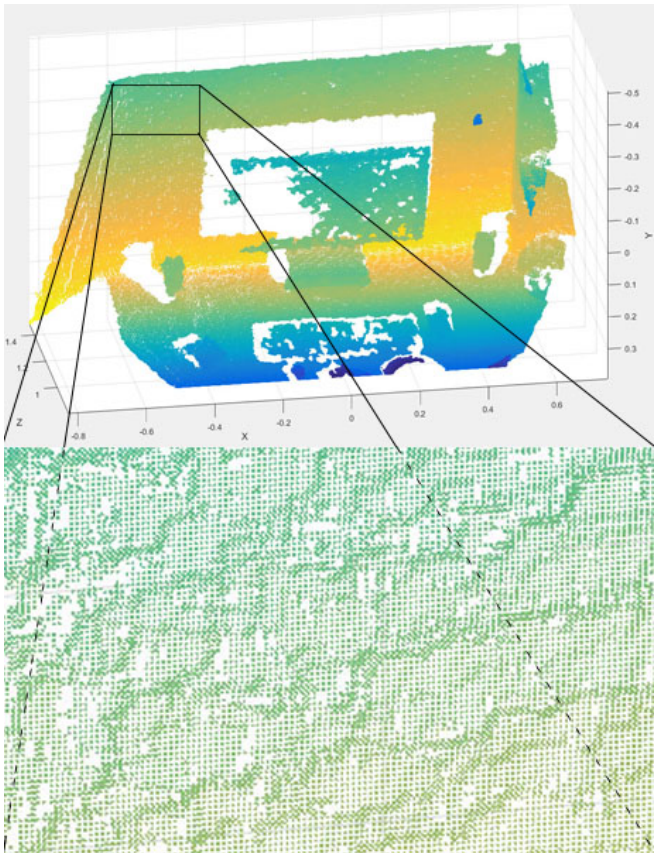

Figure 1: A Kinect sample frame, with close-up showing noise.

quite limited: 320x240 pixels for depth, 640x480 pixels for color. A sample of a Kinect frame and a close-up is shown in *Figure 1*, where we can observe the poor resolution and noise of the images. The noise, which is non-constant between two consecutive frames, makes these data streams highly suitable for static fusion (averaging multiple captures with the same viewpoint to reduce noise).

The Kinect also comes with the ability to create point clouds with color, thanks to its color camera. The color information, however, does not provide additional input for localization, and was not used in this project.

The structured-light system, on the other hand, does not propose an interactive framerate, as it takes several minutes to capture one depth image of a scene. Changing the viewpoint of the system also requires a new calibration, which is a vital step for correct depth estimation, unless the system is moved in an absolutely rigid frame, or if the scene is moved instead of the system. The second option is quite practical and feasible for small objects.

However, the structured-light system does have a much better resolution. It is, in this case, limited by either the camera's or the projector's resolution. The system used has a camera resolution of 4752x3168 pixels, while the projector has a resolution of 1920x1080 pixels, making it the system's bottleneck for precision.

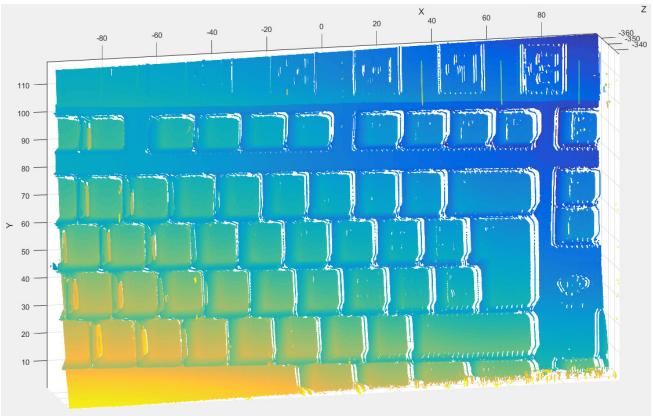

Figure 2: A sample output frame from our structured-light system.

The frames are acquired by projecting and capturing up to 36 monochrome or binary patterns on the scene. The captured images are then processed, with the help of the calibration information set up beforehand. An example of our system's output is shown in *Figure 2*.

The only scene captured with the structured-light system is a computer keyboard, due to time constraints. Three viewpoints were captured by moving the keyboard horizontally by a few centimeters each time, another viewpoint was captured by adding a rotation around the vertical axis, with the first (non-translated) viewpoint as reference, and a last viewpoint was setup by adding slightly more rotation to the object. This dataset was chosen for presenting good test cases and a relatively clean depth estimation, ~~without too big discontinuities~~.

3. PROPOSED SOLUTION

We propose a solution that relies on two main algorithms: SIFT and ICP. Let us detail how each is used with our dataset in order to solve the problem at hand.

Firstly, recall that we captured an additional RGB picture for each viewpoint for which we have depth information. When fusing two different depth maps correspnding to different viewpoint, we run both images in a SIFT mosaicing algorithm. This algorithm was meant to propose a 2D stitching of the two input images as one bigger image, where features are automatically detected and aligned. See *Figure 3* for a mosaicing example, which is the basic mosaicing sample proposed by VLFeat[1].

The mosaicing internally works by computing keypoints and their descriptors for both input images, then matching up the two sets to find keypoints and descriptors

1 Open-source library which implements SIFT, among others.

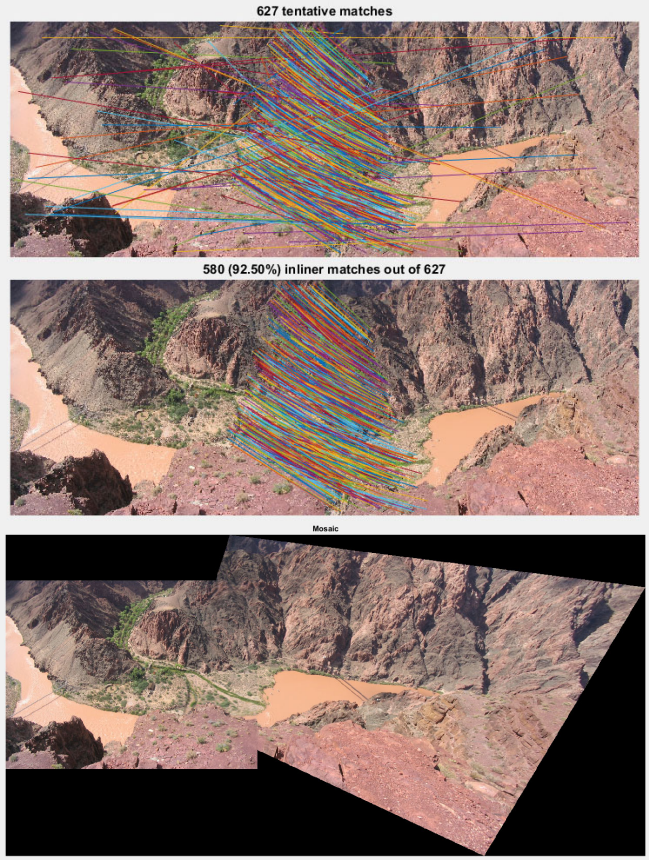

Figure 3: SIFT mosaicing example. Top: detected features and matches. Middle: outliers removal. Bottom: Mosaicing result.

that offer the most similarity. A random sample consensus (RANSAC) is then run on the matches set to remove outliers, as can be seen on *Figure 3*, and a homography is computed on the remaining inliers of the set. The homography is then possibly refined by minimizing its residuals, and the mosaic image is composed by averaging the two images (the first one is left untouched, while the second one has the homography applied to it).

However, since we do not want image stitching but depth map alignment and fusion, we discard the results obtained after the matching features are found. We use those matches by finding their 3D position in the point clouds, and from these matching 3D points, we estimate a rigid transform with the help of the singular value decomposition (SVD) of the covariance matrix between the two (matching) sets of 3D points. Note that this is actually similar to running a step of the ICP algorithm [3] on our data, the most notable difference being that the initial 3D points sets are made up of supposedly only matching points, while the ICP correspondences are established by looking, for all points in the first point cloud, at their nearest neighbours in the second point cloud.

The rigid transform computed with the SIFT correspondences is then applied to the second point cloud, which replaces the original second point cloud in the following computations.

The ICP algorithm is then run on downsampled versions of the two point clouds. The downsampling not only helps with performance, as the high resolutions of the camera ~~and projector~~ creates point clouds of the order of millions of points, but also with the precision of the alignment. Indeed, more points imply more noise, which will make the algorithm run more iterations for a potentially worse result. The point clouds are downsampled with a box grid filter, instead of random sampling, in order to avoid a point distribution that could be similar to the distribution of the noise.

As mentionned before, the ICP algorithm iterates several transfom estimations from the SVD between the two point clouds, and stops when a maximum number of iterations threshold or a minimum error threshold is reached. The output is the estimated transform, which is then applied to the original (non-downsampled) second point cloud.

The next step simply consists of merging the now aligned point clouds. This can either be done trivially, by simply creating a bigger point cloud created from the set of ~~the~~ 3D points from both point clouds after alignment. However, this increases the noise in the resulting point cloud, as both point clouds' noise will be added together. A better way to merge them is to, again, use a box grid filter on the resulting point cloud. A compromise appears in this case, since the box grid filter defines a voxel view of our scene, and we need to balance the precision of our depth estimation against the desired noise reduction. A bigger grid step size indeed means a smoother averaging of the points, but it also means a smaller depth resolution.

4. RESULTS

The first simple result that we show is how our method can remove noise in depth maps. This can be easily shown by capturing multiple frames of a scene that includes flat surfaces with a static viewpoint. By fitting a plane to the flat areas, we can quantify the error by computing the root mean squared error (RMSE) between the point cloud and the plane. *Figure 4* shows a plane (in red) fitted to the back wall of the scene for a single frame ~~of a scene~~ and for 4 ~~aligned~~ fused frames from the same scene. The fit is slightly smoother for the fused point cloud, as the wall has a more complete representation.

Since the Kinect depth maps are very noisy, flat surfaces are actually made up of several layers of points.

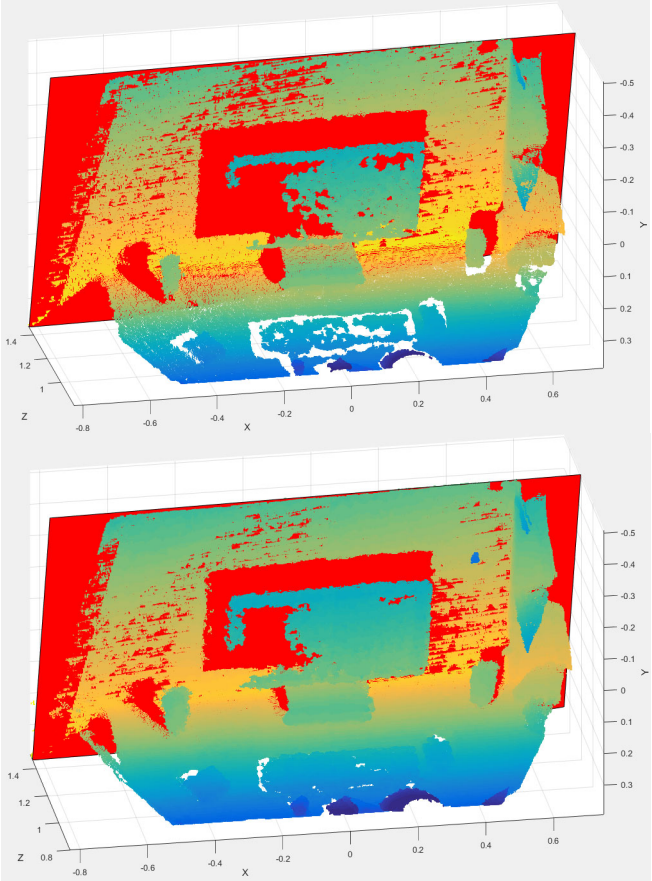

Figure 4: Plane fitting for asserting noise reduction.

This makes quantifying the fitting error quite hard to do precisely, as there are multiple solutions (planes) that align to the flat surfaces well enough to qualify. RMSE values are, however, dropping a bit for the fused point cloud on *Figure 4*, with a value of 4.65e-5, against the value of 4.99e-5 for the first frame.

As mentionned in Section 3, the noise reduction depends on the grid step chosen in the actual fusion step. *Table 1* shows the RMSE values dropping as expected when the voxel size increases. The heavy downsampling applied to the fused depth map, however, makes it sparse, as seen in *Figure 5*.

grid step	0.001	0.05	0.01
RMSE	4.97e-5	4.91e-5	4.65e-5

Table 1. RMSE values corresponding to different voxel sizes

The most interesting result, however, is that our method can increase the completeness of the scene representation. For example, occlusion might hide objects or small areas behind protruding elements of a scene, which means the data is lost in a given viewpoint. Capturing the scene from another viewpoint, with a sufficiently high

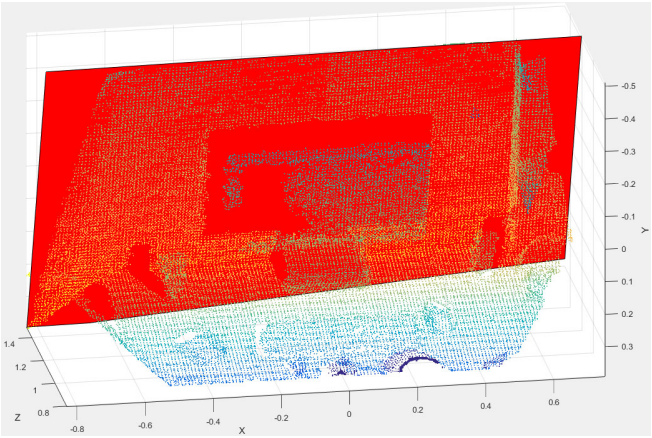

Figure 5: Result of downsampling for minimized RMSE.

overlap rate between the two viewpoints, will align the common parts of the depth maps and effectively add up the new information that was lost because of occlusion. This can be seen in *Figure 6*, where the sides of the keys were missing in the first viewpoint, and present ~~and smoother~~ in the fused point cloud.

Another way of increasing the completeness of the representation is to stitch together multiple close-up captures of a large scene, for example a room sized 3D map of which we might desire a precise 3D representation. Compared to a single capture of the whole scene from a distance, multiple captures of smaller parts can be aligned and fused together one by one and provide higher depth resolution and precision. This method evidently takes much more time, and introduces another compromise, this time between capture time and desired level of precision.

5. DISCUSSION

There are several failure cases in our method, due to the nature of the underlying algorithms. For example, ICP, SIFT or the combination of the two can fail on some examples, while one algorithm alone can produce good results. Let us discuss these failure cases in this chapter.

Firstly, ICP is an iterative algorithm, and, as mentionned before, only stops after a certain number of iterations or a minimum error change between two iterations is reached. However, it is not a globally optimized algorithm, and might in some cases fall in a local minimum. A good example of failure when using ICP alone is shown in *Figure* 7, where the alignment is good on the Y and Z axis, but is off by a few keys in the X axis. The keyboard is a very regular object, with repeating patterns (the keys) with only occasional changes in their shape. More importantly, their height is supposed to be the same, up to a certain manufacturing error. When the centroids of the point clouds are less than half a (small) key away from each other, ICP

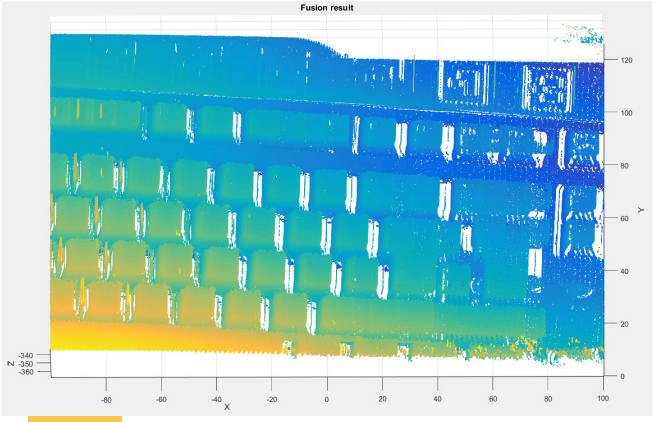

Figure 7: ICP failure case: translation of the keyboard.

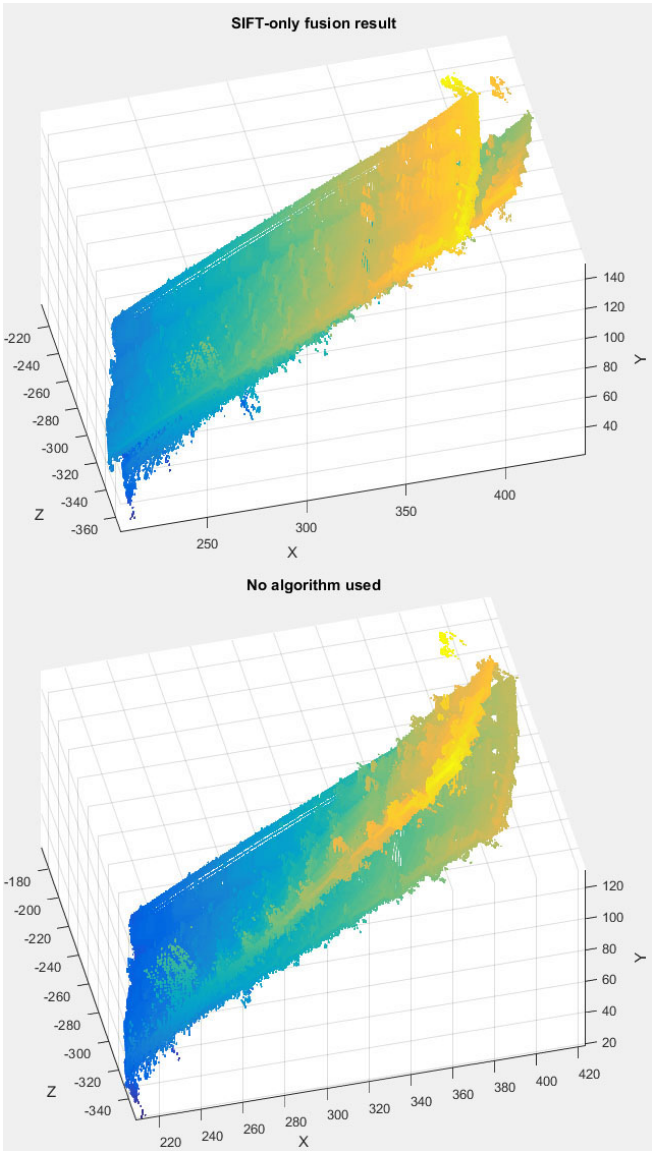

Figure 8: SIFT failure case: rotation of the keyboard.

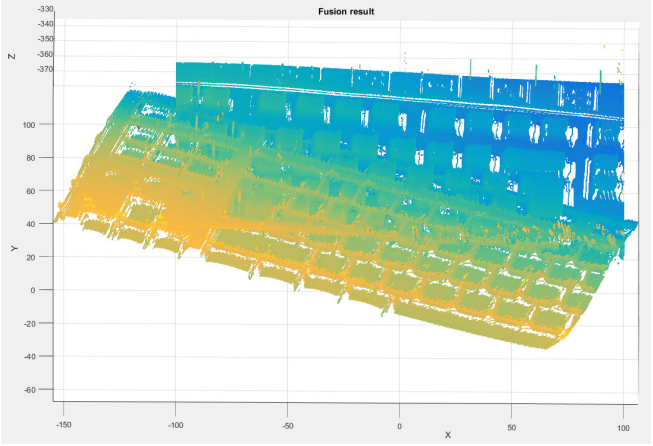

Figure 9: SIFT and ICP failure case: heavy rotation of the keyboard.

will estimate a good rotation transform, and the translation will be small enough. However, for more distant centroids, as seen in *Figure 7* where the keyboard was translated, ICP aligns small keys ~~together~~ and ignores bigger keys or empty spaces. This happens because the error actually increases for a few iterations when moving towards the global optimum and ICP rejects those solutions.

This could be improved by constraining the algorithm to a minimum acceptable error between the two registered point clouds. However, this functionnality is not available in Matlab's point cloud registration method, `pcregrigid`, and rewriting our own version of ICP is out of the scope of this project.

SIFT, on the other hand, works nicely with the translation cases that fail with ICP alone, but has more trouble when rotations are at play, as shown in *Figure 8*. As stated earlier, the computation to estimate the transform is the same as one iteration of ICP and as such cannot be optimal. Another possible weak point of our SIFT implementation is the use of the correspondences found: we select all matches, as the outliers were already removed by ~~the~~ RANSAC ~~elimination in SIFT~~. Their 3D position in the point clouds, however, might be noise ~~points instead of data points~~, and this may lead to a wrong estimation of the transform. One possible improvement would be to select only matches that map to real 3D points in the depth-maps, which is hard to provide automatically.

Unfortunately, both algorithms' weak points do not cancel each other out by simply combining them, and our SIFT and ICP pipeline also has failure cases, for example when the second viewpoint is heavily rotated compared to the first viewpoint. This can be seen in *Figure 9*. This particular rotation is simply too much for any of the two algorithms used to align properly.

The noise reduction example with Kinect data also fails to some level, as the point cloud becomes too sparse to

be interesting as the grid step is increased to lower the noise. This is, as mentionned in Section 4, due to the Kinect depth maps having heavy noise in flat areas, as the noise creates multiple layers of points. Downsampling with a box grid filter cannot denoise these layers to a flat region unless the grid step is very high, thus losing a lot of information in the rest of the point cloud.

One possible improvement could be applied to flat areas: first extracting them by fitting planes to the point cloud, which will possibly return multiple regions, and downsample those regions in the thickness direction, according to the corresponding plane's normal vector. This would not, however, remove noise in the non-flat areas of the point cloud.

6. REFERENCES

[1] R. Newcombe, et al., "KinectFusion, Real-Time Dense Surface Tracking and Mapping", *Mixed and augmented reality (ISMAR)*, 2011

[2] F. Blais, et al., "Accurate 3D acquisition of freely moving objects", *Proceedings of the 2nd International Symposium on 3D Data processing, Visualization and Transmission*, 2004

[3] P. Besl, N. McKay, "A Method for Registration of 3D Shapes", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol 14, 1992

[4] J. Salvi, et al., "A state of the art in structured light patterns for surface profilometry", *Pattern Recognition*, Vol 43, Issue 8, 2010