Dense point-cloud usage for localization

Anna Zderadickova, Michal Polic

2021/2022

1 Project overview

The goal of the project is finding more accurate camera poses in texture-less environments, e.g., construction side, see 1.

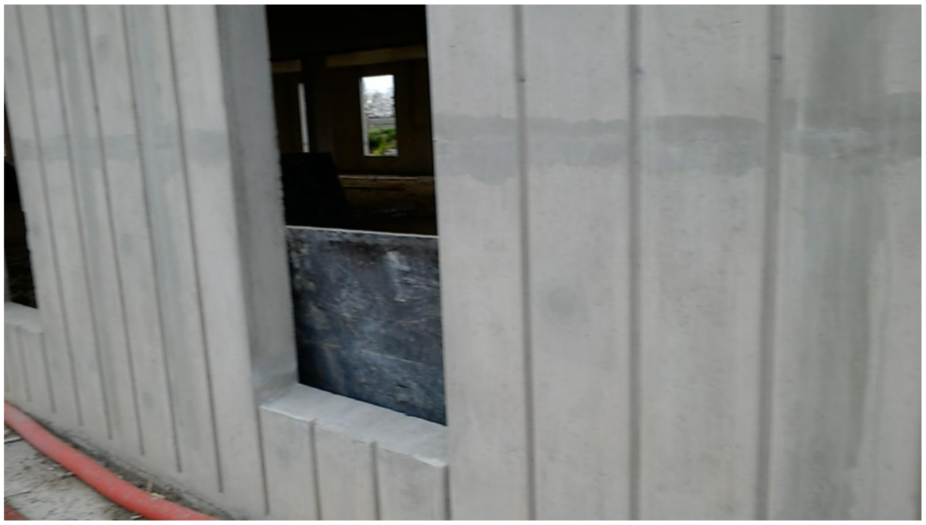


Figure 1: An example of Eiffage construction

We assume HoloLens device as the main sensor used for the localization and mapping of the environment. The HoloLens device consist of 4 gray-scale tracking, 1 RGB and one depth camera. The depth map is available even if the scene does not contain any significant textures which can be used for accurate localization. Therefore, we decided to directly align dense point-clouds, i.e., the map and HoloLens depth-maps in world coordinates, as the final stage of InLoc.

After the consultation with T. Staller we found out that similar strategy was published at least in [1]. We proposed to extend the current approach by using FCGF [2] to describe dense point-cloud (recommended as the best descriptor from ETH group focusing on dense point-cloud alignment) and TEASER++ [3]

or MAGSAC [4] to robustly estimate the SE(3) transformation between point-clouds. Recently, it was published on CVPR2021 the PREDATOR [5] for dense point-cloud alignment. We have also decided to include Robust ICP [6] in our approach.

In summary, we have running the algorithms:

- FCGF [2] - Fully Convolutional Geometric Features (FCGF) is used to calculate point cloud feature descriptors.
- TEASER++ [3] - TEASER++ stands for "truncated least squares estimation and semi definite relaxation". TEASER++ is used to align point-clouds. It decouples scale, rotation and translation using invariant measurements and uses truncated least squares to solve them. The scale is estimated using adaptive voting, rotation is estimated by semi definite relaxation and translation is again estimated by adaptive voting.
- Overlap PREDATOR [5] - Overlap PREDATOR is a model for pair-wise point-cloud registration specialized in point-clouds with low overlap ($\leq 30\%$ overlap). Interest points of the point clouds are sampled and their local neighborhood is described as feature descriptors, these feature descriptors are then matched to establish correspondences, the rotation and translation is then estimated by robustly minimizing the distance between correspondences. As PREDATOR focuses on low overlap point clouds it focuses on sampling interest points mostly in overlap areas.
- Robust ICP [6] - Robust ICP is a robust implementation of the Iterative Closest Point algorithm with fast convergence.
- *Few alignment methods from CVPR2020 were also tested in the past but we don't have them running now.*

The overview of the development process is in the Figure 2.

At first we wanted to have a Ground Truth measurements of HoloLens pose in the space to be able test individual methods and their accuracy. Here, we believe, that may be available datasets for this purpose but we are not aware of them.

The Figure 15 shows the accuracy of the localization by comparing the final dense point-cloud alignment with Matterport scan. We have accuracy of 2-3cm while aligning HoloLens camera poses to Vicon markers and 6-7 cm for point-cloud alignment. In the publication [7], authors have measured the tracking error smaller than 2 cm/sec and 2 deg/sec while depth sensor has also about 2cm error on the most distant measured points.

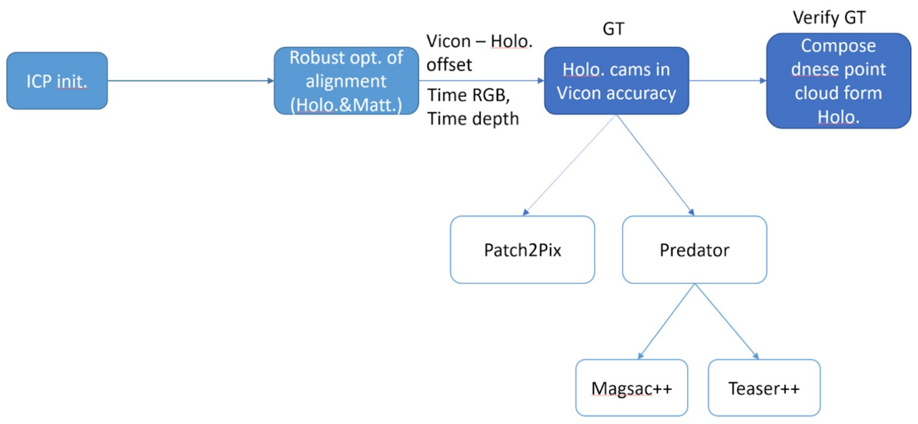

ICP init.
Robust opt. of alignment (Holo.&Matt.)
Vicon – Holo. offset
Time RGB, Time depth
GT
Holo. cams in Vicon accuracy
Verify GT
Compose dnese point cloud form Holo.
Patch2Pix
Predator
Magsac++
Teaser++

Figure 2: Proposed development scheme

The structure of the paper is following:

- Matterport and Vicon alignment - *the approach to transform Matterport to Vicon coordinate system*
- HoloLens and Vicon alignment - *the approach to obtain ground truth*
- Depthmaps alignment - *evaluation of previously mentioned point-cloud alignment methods*

2 Matterport and Vicon alignment

We have 3 coordinate systems that we are working with. First, we have the Vicon coordinate system in which we meassured the Vicon markers. The Matterport point-cloud of the room is in Matterport coordinate system and HoloLens cameras are in HoloLens world coordinate system. In order to align HoloLens depth point-clouds with Matterport point-cloud we need to transform both HoloLens and Matterport coordinate system to Vicon coordinate system.

The Matterport coordinate system was aligned by manualy measuring 8 markers on the floor of the room and matching them with same markers measured in Vicon using the procrustes method. We leave out gradually 1, 2, 3 and 4 points, align the rest and the alignment errors can be seen in Figure 3

As can be noticed in the histogram in figure 3 there are two clusters of errors. To further inspect this, we have plotted errors using 4 points to align over all points marking which point was used for alignment in fig. 4. After further investigation we noticed that there is large relational error between 5th and 6th points as can be seen in fig. 5

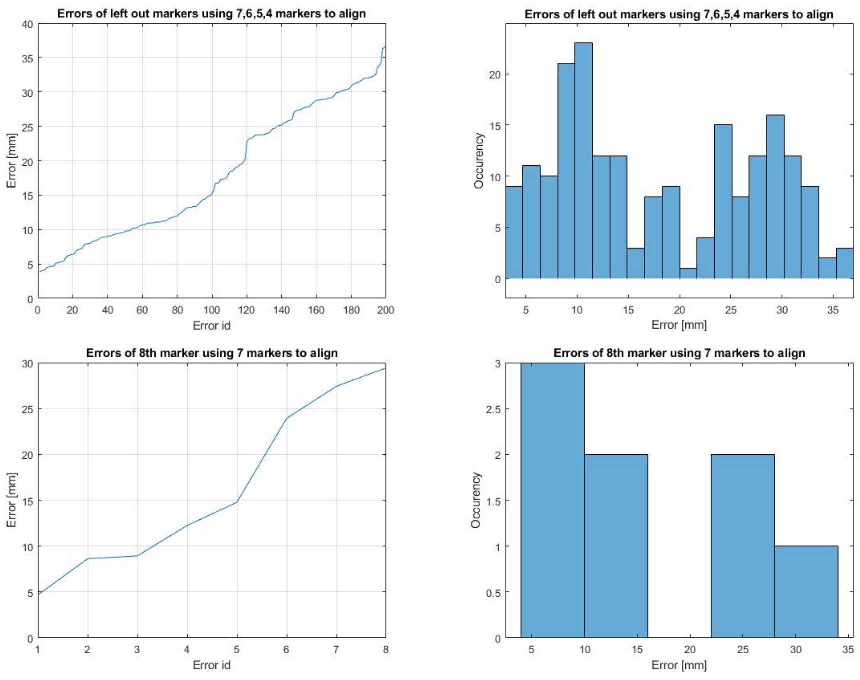


Figure 3: Gradually leaving out 1, 2, 3, 4 points out of Matterport to Vicon alignment and then calculating their error are in the first row. In the second row results of the Leave One Out error (LOO) can be seen.

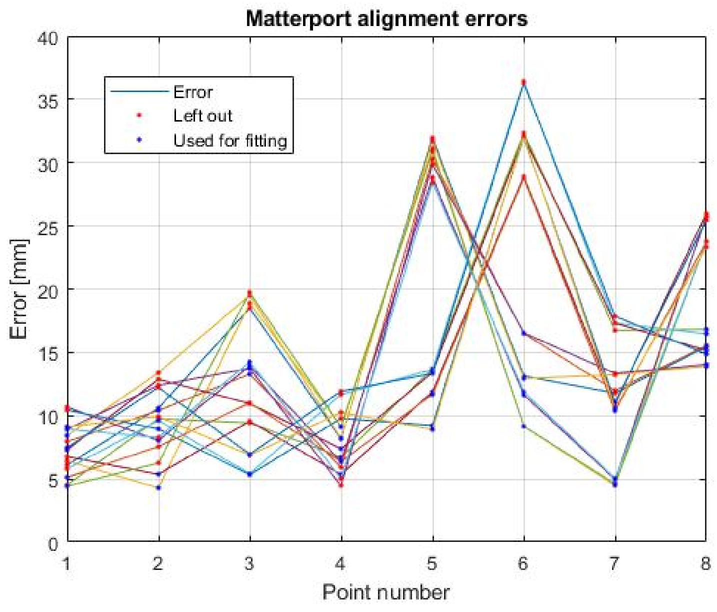


Figure 4: Alignment errors using 4 points to align.

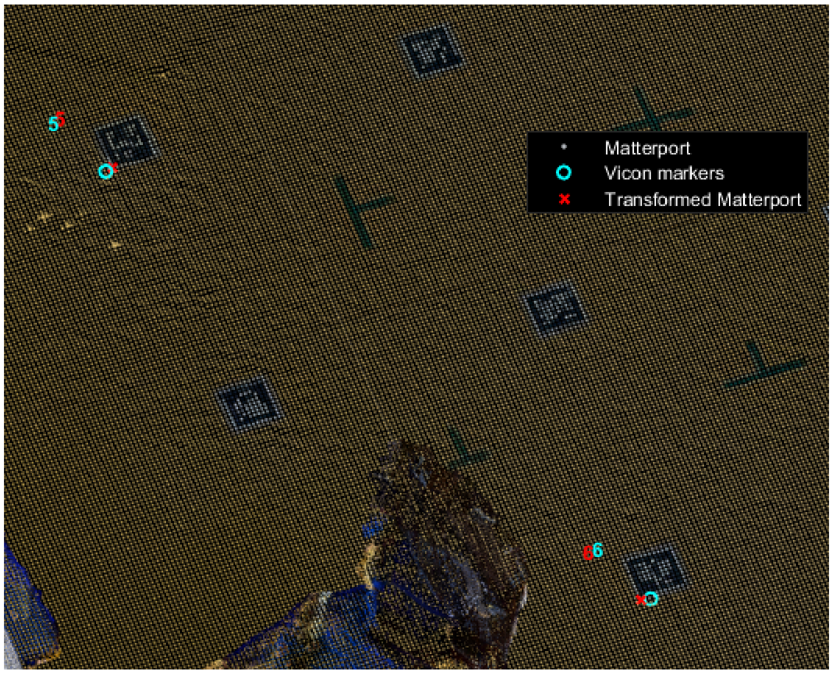


Figure 5: Error of 5th and 6th points.

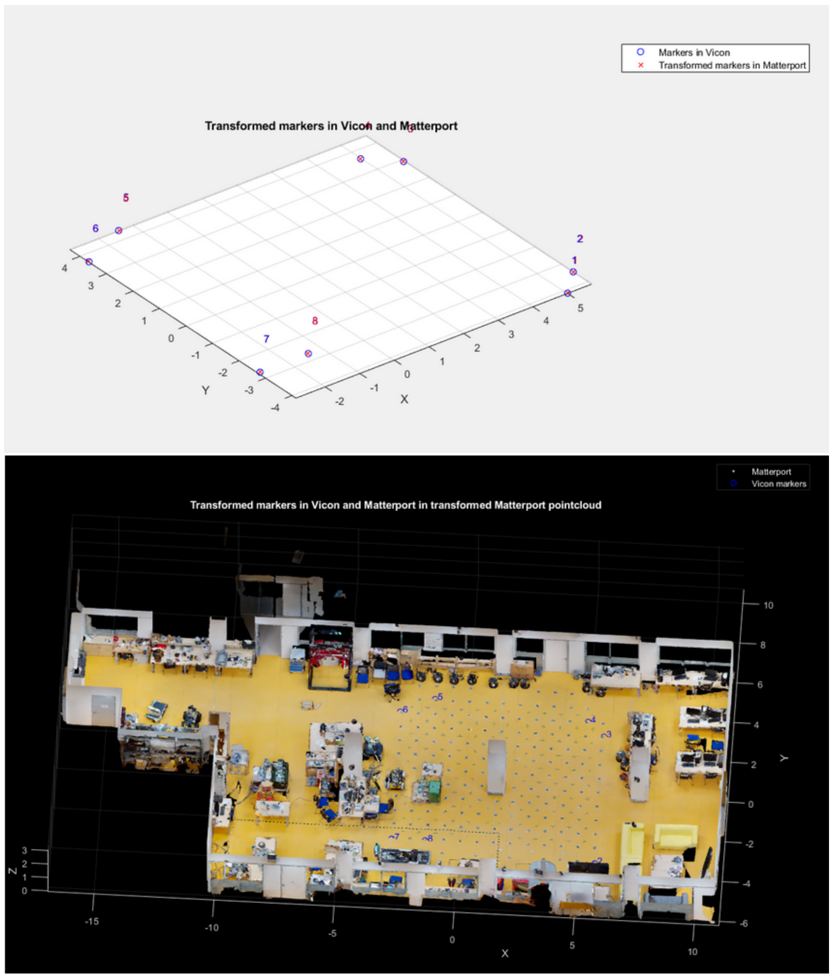


Figure 6: Matterport markers converted to Vicon coordinate system and the Matterport point-cloud converted to Vicon coordinate system.

3 HoloLens and Vicon alignment

3.1 Pipeline overview

We will quickly summarize the whole pipeline and then we will look at each step closely. First, we use the Robust ICP [6] (later in this work referenced to as R-ICP) to align HoloLens camera poses with Vicon markers. Then filter out outliers created by errors in Vicon and then we optimize our objective function to find transformation parameters, that align HoloLens to Vicon even closer. We run three option of the optimization and for each we run R-ICP [6] on the resulting point-clouds. The whole pipeline is described in figure 7.

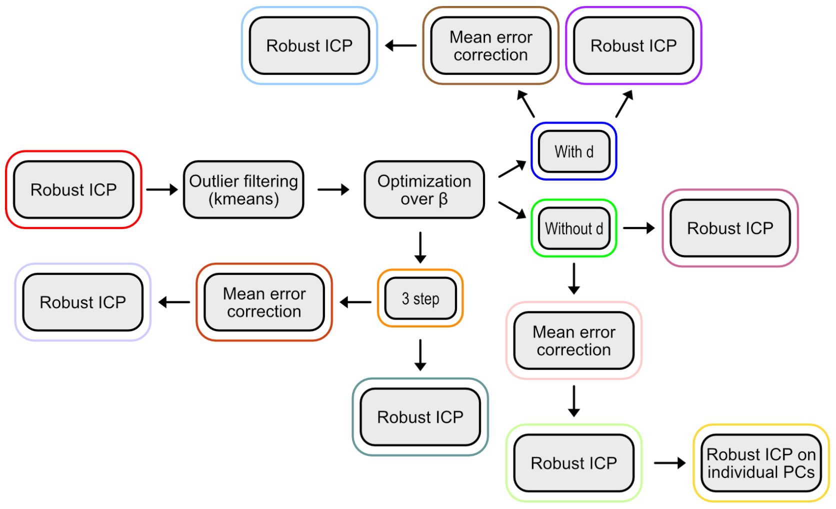


Figure 7: Scheme of the alignment pipeline. The color frames are representing the color of the results in following figures.

3.2 Objective function

We have the HoloLens camera poses and Vicon marker poses unaligned. Camera poses from all HoloLens sensors are taken into consideration. Then HoloLens camera poses are randomly rotated to help R-ICP converge to the right result and align them using R-ICP to Vicon markers. (We use 100 random rotations and then take the rotation with the lowest error (NN distances)).

We found out, that the framerate of HoloLens sensors is irregular (especially the depth sensor, as can be seen in Figure 9). This leads to the problem that we cannot assume regular timestamps/framerate in our calculations and we have to use timestamp steps calculated by subtracting the first timestamp of the HoloLens PV camera from all HoloLens timestamps. This lead us to the objective function, see Equation 3.

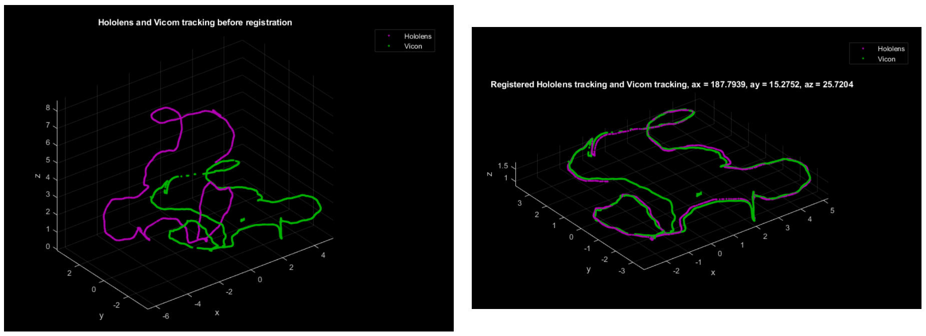


Figure 8: Hololens (purple) and Vicon (green) camera/marker poses before any alignment and after the random rotation and R-ICP. As can be noticed the Vicon data is much denser (100 FPS) on the contrary from HoloLens sensors we are getting approximately 15 FPS with the exception of depth camera, that is 1-3 FPS. The Vicon data has a cluster of error position that need to be filtered out for calculations.

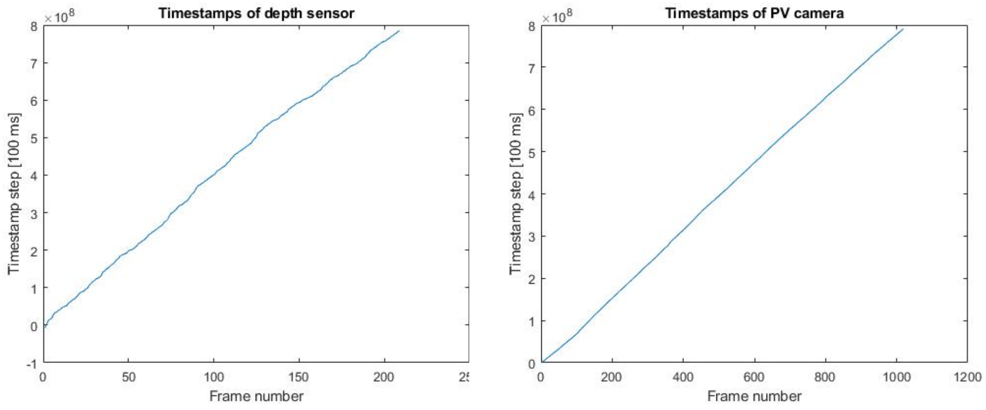


Figure 9: Timestamps of the depth camera and the PV camera.

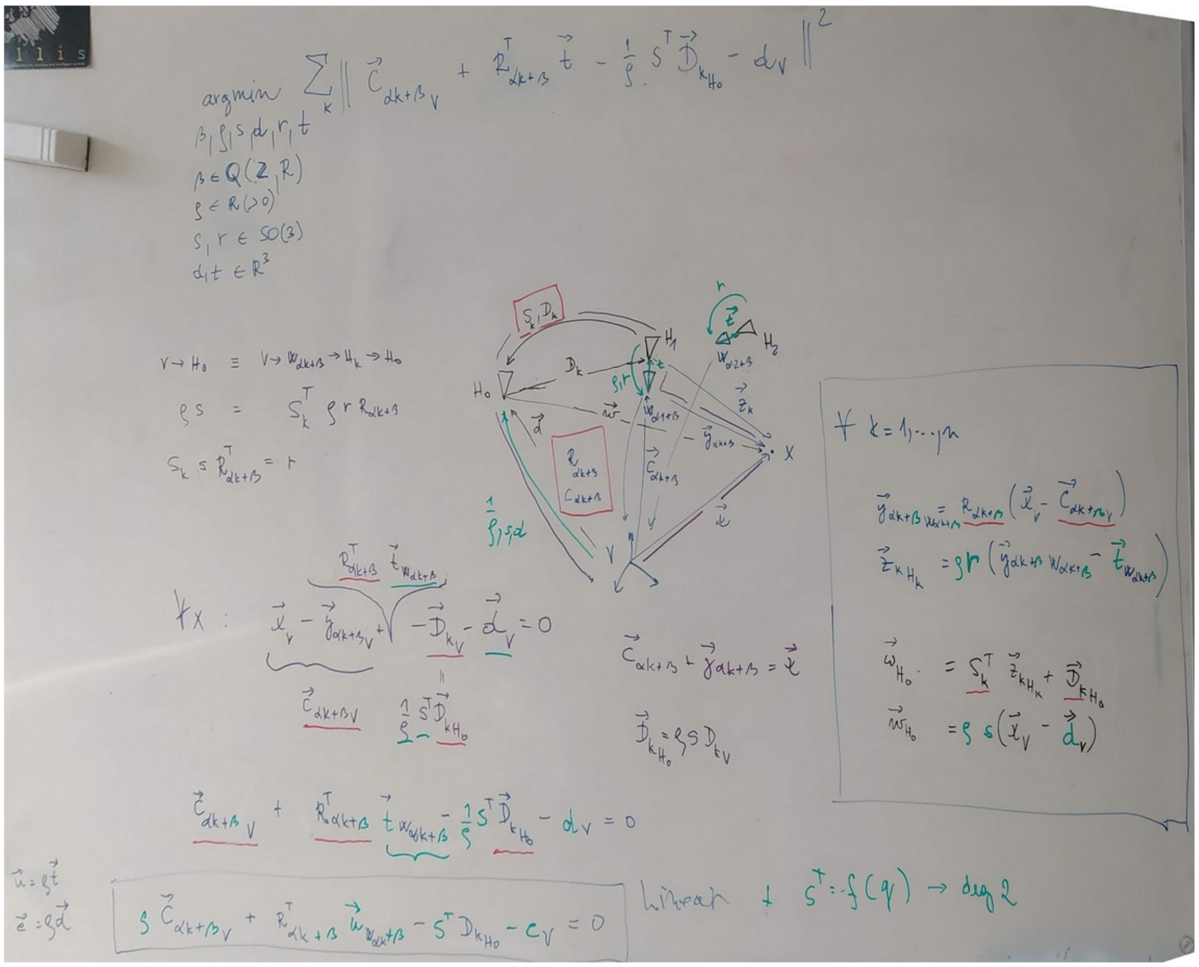

argmin β, ρ, s, d, r, t Σ_k ‖ C_{αk+β V} + R^T_{αk+β} t − 1/ρ s^T D_{k H0} − d_V ‖²
β ∈ Q(Z, R)
ρ ∈ R(>0)
s, r ∈ SO(3)
d, t ∈ R³
V → H0 ≡ V → W_{αk+β} → H_k → H0
ρ s = S_k^T ρ r R_{αk+β}
S_k s R^T_{αk+β} = r
H0
H1
H2
D_k
S_k, D_k
R_{αk+β} C_{αk+β}
1/ρ, ρ s d
V
X
∀ k = 1, ..., n
y_{αk+β W_{αk+β}} = R_{αk+β} (x_V − C_{αk+β V})
z_{k H_k} = ρ r (y_{αk+β W_{αk+β}} − t_{W_{αk+β}})
ω_{H0} = S_k^T z_{k H_k} + D_{k H0}
ω_{H0} = ρ s (x_V − d_V)
∀x : x_V − y_{αk+β V} + R^T_{αk+β} t_{W_{αk+β}} − D_{k V} − d_V = 0
C_{αk+β V}
1/ρ S^T D_{k H0}
C_{αk+β} + y_{αk+β} = x
D_{k H0} = ρ s D_{k V}
C_{αk+β V} + R^T_{αk+β} t_{W_{αk+β}} − 1/ρ S^T D_{k H0} − d_V = 0
u = ρ t
e = ρ d
ρ C_{αk+β V} + R^T_{αk+β} ω_{W_{αk+β}} − S^T D_{k H0} − e_V = 0
Linear + S^T = f(q) → deg 2

Figure 10: Graphical interpretation of equation 3. In practise, because of a irregular HoloLens framerate, we substitude $\alpha\ \tau$ where τ is $\bar{t}_k - \bar{t}_1$.

$$V_{V2H} = C_{\tau+\beta_V} + R^T_{\tau+\beta} t_i \tag{1}$$

$$H_V = \frac{1}{\rho} S^T D_{k_H 0} - d_V \tag{2}$$

$$V_{V2H} - H_V = 0 \tag{3}$$

In this formula we have:

- V_{V2H} represents Vicon markers shifted info HoloLens sensors
- H_V stands for HoloLens camera poses transformed to Vicon
- τ - here τ is $\bar{t}_k - \bar{t}_1$. This means that we use directly the time of HoloLens capture instead of multiples of framerate.
- β - time shift as that HoloLens and Vicon didn't start recording at the same time
- $C_{\tau+\beta_V}$ - Vicon marker origin position in Vicon coordinate system
- $R_{\tau+\beta}$ - Vicon marker rotation in Vicon coordinate system
- t_i - translation between Vicon marker and camera center of HoloLens in Vicon coordinate system, it differs depending on each sensor
- ρ is a scale of the Vicon vectors onto HoloLens vectors
- S stands for camera rotation in HoloLens coordinate system
- $D_{k_H 0}$ represents camera center in HoloLens coordinate system
- d_V is the origin of HoloLens coordinate system in Vicon coordinate system.

We know: C, R, D and these parameters are unknown and optimized: t, ρ, S, d, β.

In the graph 12 we can see the function values both initial and optimized. The value is equal to

$$\sum(\|f\|^2) + 10(\|t\| + \|d_V\|) \tag{4}$$

where

$$f = C_{\tau+\beta_V} + R^T_{\tau+\beta} t_i - \frac{1}{\rho} S^T D_{k_H 0} - d_V \tag{5}$$

while filtering out the Vicon errors. Vicon errors are filtered by splitting all f values into two clusters (using kmeans) and picking the cluster with lower errors to be taken into account. We then optimize the function value over all possible βs (here we try β from 0 to 100) to determine the best time shift.

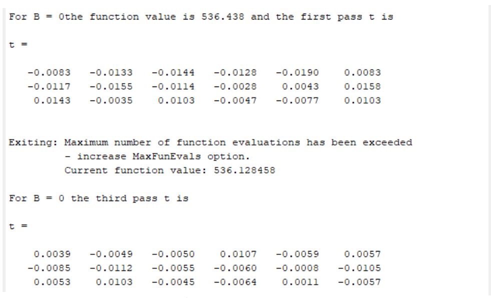
To determine wherever it is beneficial to use d_V in the objective function, we run the optimization for the objective function without d_V and thus the function value changes to:

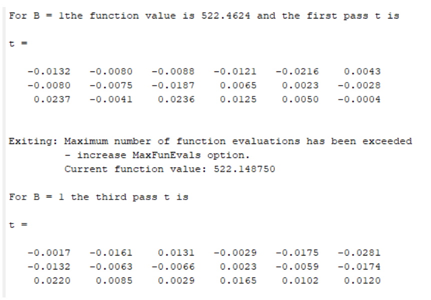
$$\sum(\|f\prime\|^2) + 10\|t\|. \tag{6}$$

where

$$f\prime = C_{\tau+\beta_V} + R^T_{\tau+\beta} t_i - \frac{1}{\rho} S^T D_{k_H 0} \tag{7}$$

We also try to remove HoloLens error in shift and rotation by using a three-step optimization. In first step we optimize t, ρ, S, then with these optimized parameters we optimize the HoloLens camera poses ($D_{k_H 0}$) and lastly we once again optimize t, ρ, S. We should see that the values of optimized t are equal to those in first pass if we found the right β and the drift of HoloLens cameras distribution is Gaussian. The three-step optimization is unfortunately running slowly in MATLAB and it took a day to get values for the first two betas. We are considering rewriting the optimization in CERES. However from the first two betas we have noticed that t from third pass is not equal to the first pass t (as can be seen in fig. 11). We then ran the three step optimization only for the best found beta. The optimized t from the third pass differs from the first pass as well.


```
For B = 0the function value is 536.438 and the first pass t is

t =

   -0.0083   -0.0133   -0.0144   -0.0128   -0.0190    0.0083
   -0.0117   -0.0155   -0.0114   -0.0028    0.0043    0.0158
    0.0143   -0.0035    0.0103   -0.0047   -0.0077    0.0103


Exiting: Maximum number of function evaluations has been exceeded
         - increase MaxFunEvals option.
         Current function value: 536.128458

For B = 0 the third pass t is

t =

    0.0039   -0.0049   -0.0050    0.0107   -0.0059    0.0057
   -0.0085   -0.0112   -0.0055   -0.0060   -0.0008   -0.0105
    0.0053    0.0103   -0.0045   -0.0064    0.0011   -0.0057
```


```
For B = 1the function value is 522.4624 and the first pass t is

t =

   -0.0132   -0.0080   -0.0088   -0.0121   -0.0216    0.0043
   -0.0080   -0.0075   -0.0187    0.0065    0.0023   -0.0028
    0.0237   -0.0041    0.0236    0.0125    0.0050   -0.0004


Exiting: Maximum number of function evaluations has been exceeded
         - increase MaxFunEvals option.
         Current function value: 522.148750

For B = 1 the third pass t is

t =

   -0.0017   -0.0161    0.0131   -0.0029   -0.0175   -0.0281
   -0.0132   -0.0063   -0.0066    0.0023   -0.0059   -0.0174
    0.0220    0.0085    0.0029    0.0165    0.0102    0.0120
```

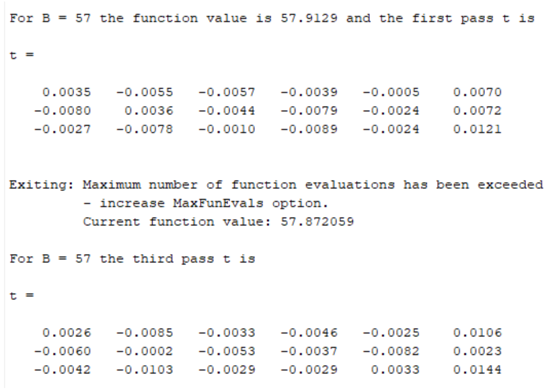

```
For B = 57 the function value is 57.9129 and the first pass t is

t =

    0.0035   -0.0055   -0.0057   -0.0039   -0.0005    0.0070
   -0.0080    0.0036   -0.0044   -0.0079   -0.0024    0.0072
   -0.0027   -0.0078   -0.0010   -0.0089   -0.0024    0.0121


Exiting: Maximum number of function evaluations has been exceeded
         - increase MaxFunEvals option.
         Current function value: 57.872059

For B = 57 the third pass t is

t =

    0.0026   -0.0085   -0.0033   -0.0046   -0.0025    0.0106
   -0.0060   -0.0002   -0.0053   -0.0037   -0.0082    0.0023
   -0.0042   -0.0103   -0.0029   -0.0029    0.0033    0.0144
```

Figure 11: Output t for the first two *beta* and for the best *beta*.

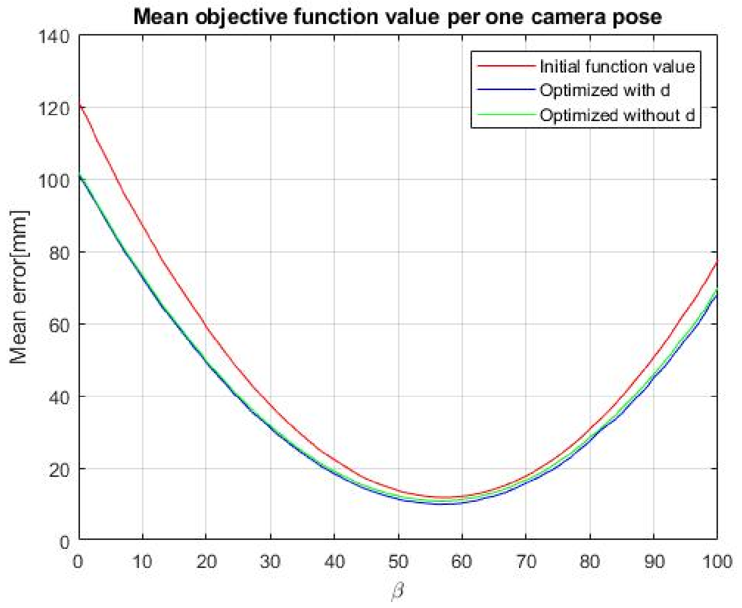


Figure 12: Graph of the function values dependent on β.

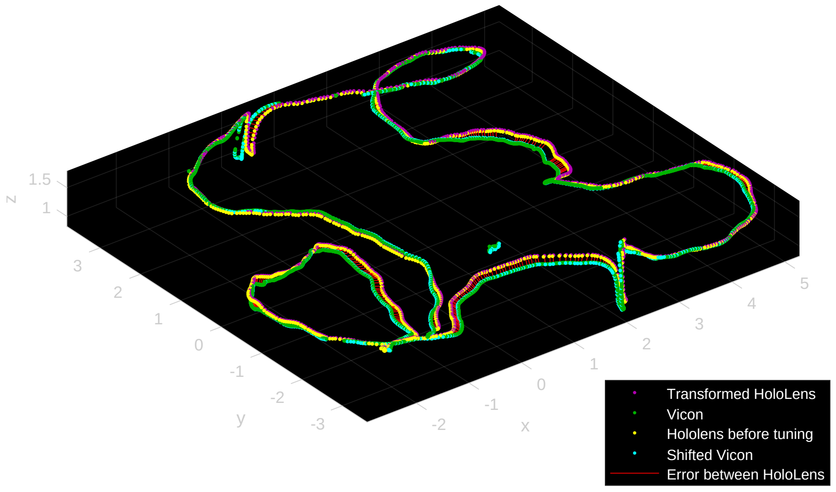
The best β is determined to be 57. In Figure 13 we can see examples of aligned cameras with HoloLens after tuning, using the best β and the parameters optimized for this β. Errors of the alignment can be seen in figure 14

3.3 Mean error correction

Taking into consideration that HoloLens can have a tracking error we tested the possibility of improving our results by using a mean error correction. We calculated distances between aligned HoloLens camera poses and the used the mean of these distances in the distance calculations (by adding it to transformed HoloLens).

3.4 Point cloud alignment

We then take HoloLens depth point-clouds and use the same transformation calculated from HoloLens camera poses and Vicon markers to align it to a Matterport point-cloud that has been transformed to Vicon. For each point-cloud (X) we get two matrices A_{D2O} and A_{C2D} to transform the point cloud to HoloLens world coordinate system. We then get the depth point-cloud transformed to Vicon by:



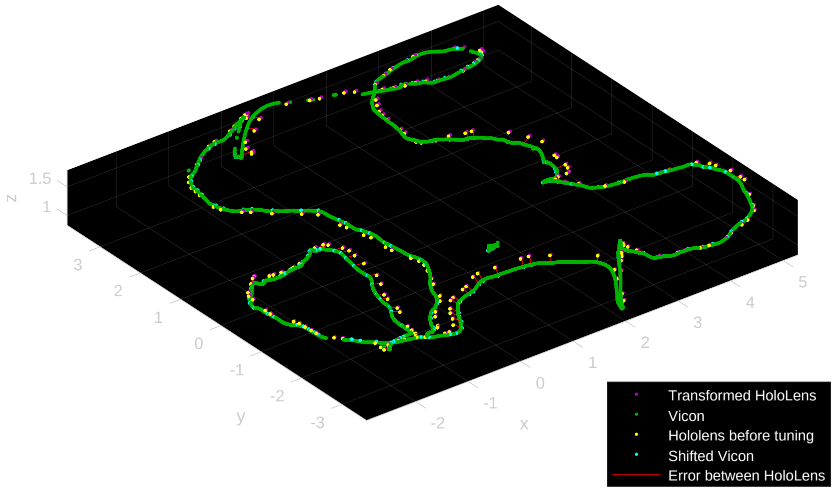


Figure 13: Example of HoloLens PV camera and depth (long throw depth) sensor aligned with Vicon. Aligned HoloLens cameras are purple, unaligned HoloLens are yellow, Vicon is green, and turquoise are Vicon markers shifted by t and matched by time to HoloLens. Red lines represents the errors between aligned HoloLens and Vicon.

$$\frac{1}{\rho} P S^T R_{ii}^T A_{D2O}\ A_{C2D}\ X + T + R_V t + correction \tag{8}$$

Where ρ is a scale of the Vicon vectors onto HoloLens vectors, P is transformation matrix obtained from Robust ICP [6], S is camera rotation in HoloLens coordinate system, R_{ii} is the random rotation and rotation acquired by ICP, T

is $-d_V$ for optimization using d_V or a zero vector otherwise.
(d_V is the origin of HoloLens coordinate system in Vicon coordinate system), R_V is Vicon marker rotation in Vicon coordinate system. t represents the translation between Vicon marker and camera center of HoloLens in Vicon coordinate system it differs depending on each sensor and correction represents the shift between aligned HoloLens and Vicon (the errors in Figure 13).

For each result we then run Robust ICP [6] on aligned point-clouds and apply the calculated transform onto the transformed HoloLens camera poses. All distances between HoloLens camera poses and Vicon markers can be seen in fig. 14.

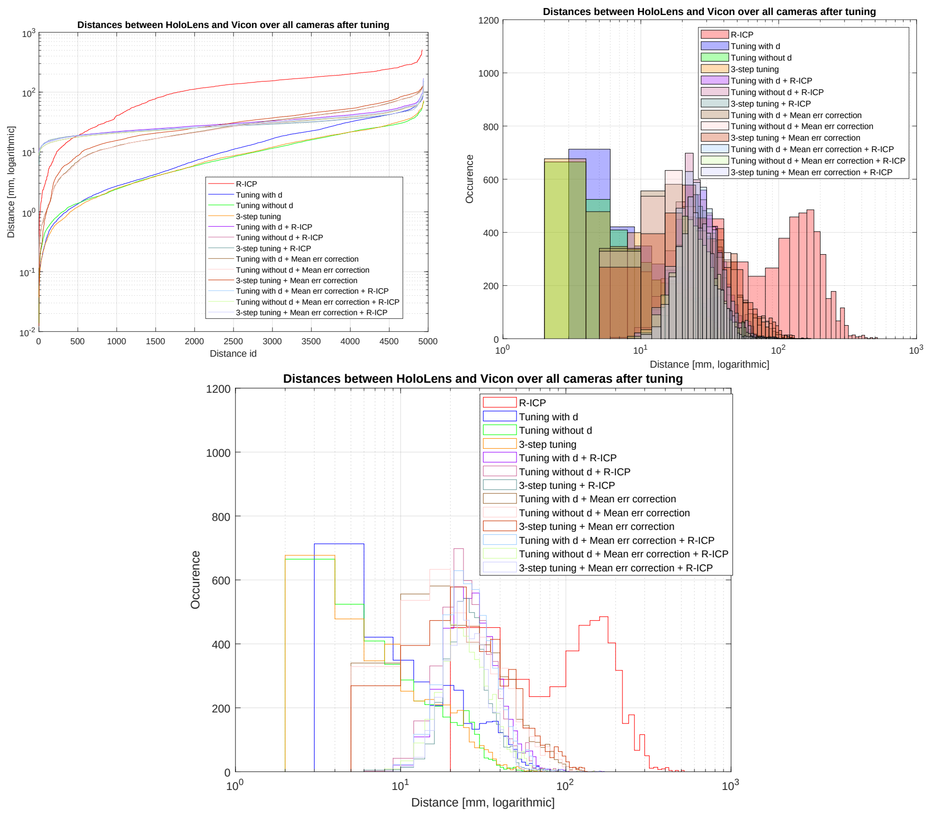
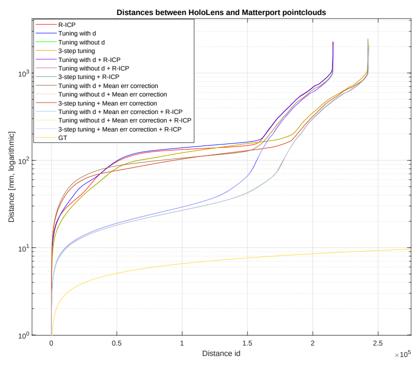


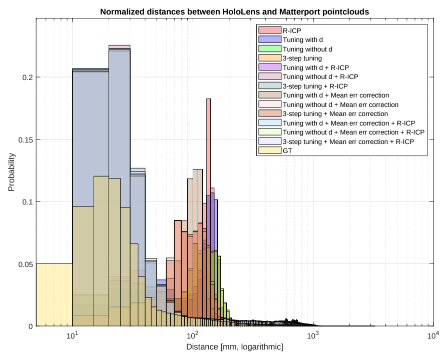
Figure 14: Distances between all aligned HoloLens camera centres and Vicon marker centres.

For point-cloud alignment we merge all depth maps obtained from HoloLens, then we subsample both the HoloLens point cloud and the Matterport point cloud (we consider every 10th point) and crop Matterport point cloud according to aligned HoloLens point cloud. This leads to different number of points in point-clouds for each optimization branch so to give the most objective results, we normalize the histograms (using the probability function, where the value of the bin equals to occurrence/sum(occurrences)). We then run Robust ICP [6] on aligned point-clouds to align them even closer. To measure distances between both point-clouds we use NN search, all distances can be seen in fig. 15

As can be seen from figures presented the best result is using tuning without

d_V, mean error correction and R-ICP so we have determined these poses to be ground truth. As mentioned previously HoloLens poses can have a tracking error up to 2 cm/sec and 2 deg/sec. To fix this error we ran R-ICP on each HoloLens depth point-cloud separately. Our obtained ground truth point-clouds can be seen in figures as yellow lines.





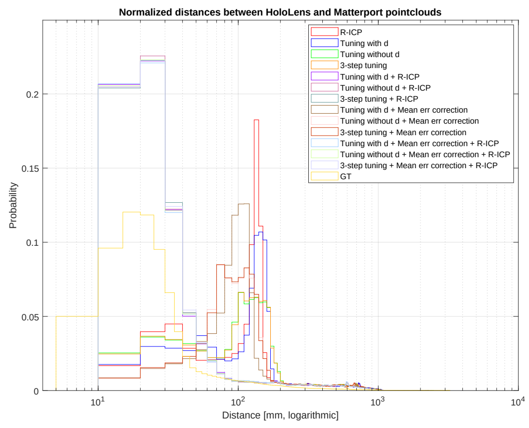


Figure 15: Distances between aligned HoloLens point-clouds and Matterport point cloud. (Note that the number of points in point-clouds differ as we are cutting point-clouds accordingly to alignment, thus leading us to normalizing the histograms (the value of the bin equals to occurrence/sum(occurrences)))

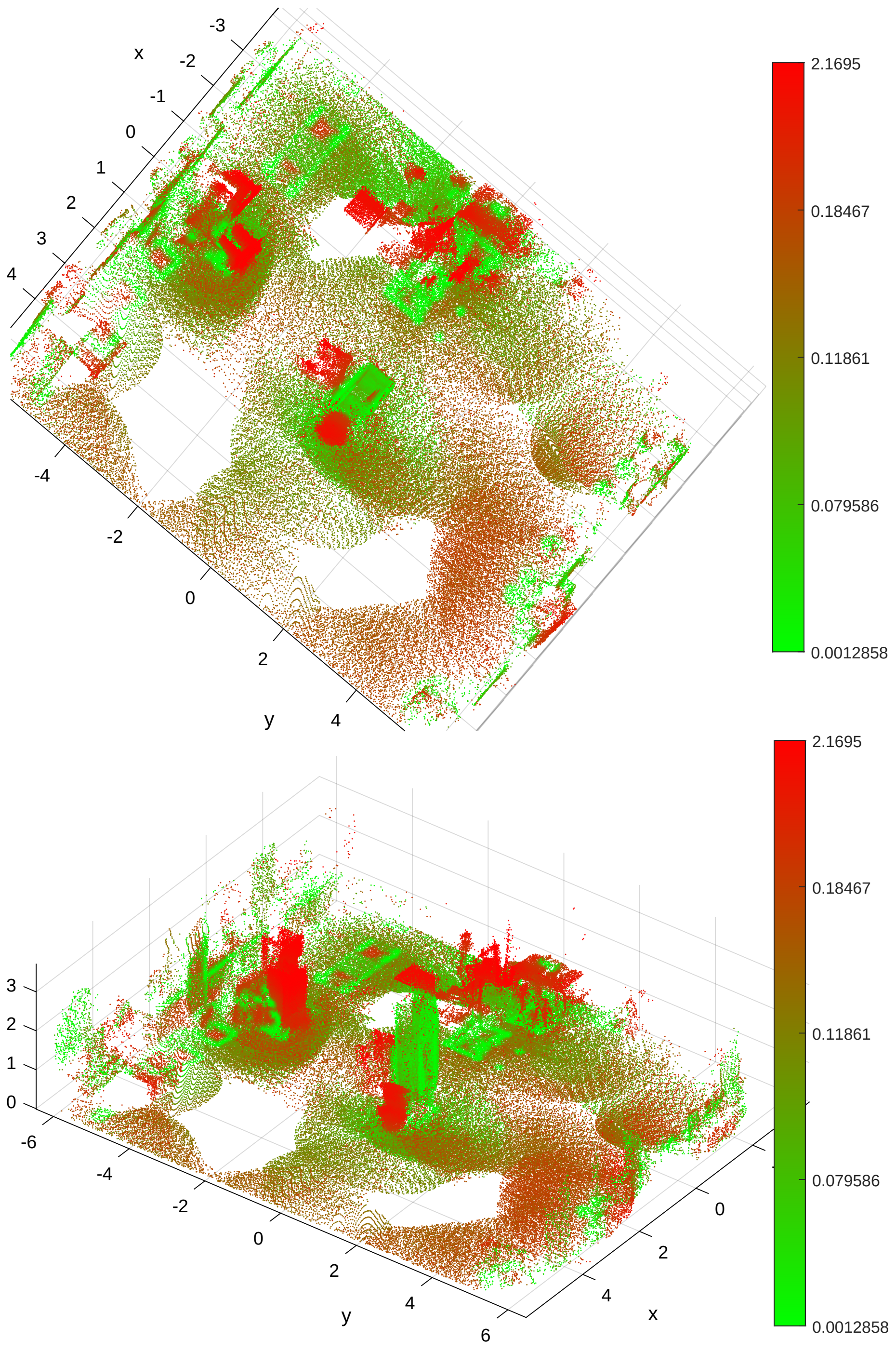


Figure 16: HoloLens point-clouds colored accordingly to distance of NN in Matterport point-cloud using 3step tuning, mean error correction and R-ICP to align.

4 Depthmaps alignment

We have tried several methods for point-cloud alignment: FCGF[2] combined with RANSAC, FCGF[2] combined with TEASER++[3], Overlap PREDATOR[8] and Robust ICP [6]. In this section we will discuss the process and compare alignment results for each of these methods.

To evaluate mentioned methods we have taken the ground truth point-clouds (we have 193 point-clouds) and noised it. (We applied translation from 0 to 100 centimeters in 10 centimeters increase and random rotation from 0 to 20 degrees in 4 degrees increase). We then used methods to align noised point-clouds back to cutouts of Matterport point-cloud and evaluated errors in translation and in rotation in comparison to ground truth camera poses. The translation error is calculated as a distance between ground truth camera pose and an aligned noised camera pose. The rotation error is calculated:

$$e = arccos(0.5 * (trace(R) - 1)) \tag{9}$$

where

$$R = R_{GT}^{T} * R_{noised} \tag{10}$$

R_{GT} is the rotation matrix of ground truth camera pose and R_{noised} is the rotation matrix of noised and aligned camera pose.

The FCGF[2] + RANSAC method required we first ran FCGF to obtain feature descriptors for both point-clouds. We are using pre-taught model (ResUNetBN2C) provided by the authors that was trained on indoor datasets. FCGF calculates the feature descriptors from voxels of the point-cloud so it requires a voxel size. In this experiment, the voxel size is set to 2.5 centimeters. After getting feature descriptors from FCGF, we calculated tentative matches between a pair of point-clouds by calculating the feature vectors distances and finding the closest ones. Finally we ran RANSAC implemented in MATLAB on obtained feature matches and got a transformation structure containing scale, translation and rotation. Results of this aligment method can be seen in figures 17 and 18. Noised point-clouds with rotations of 12 and 20 degrees result in smaller errors whereas 8 and 16 degrees noised point-clouds result in large errors.

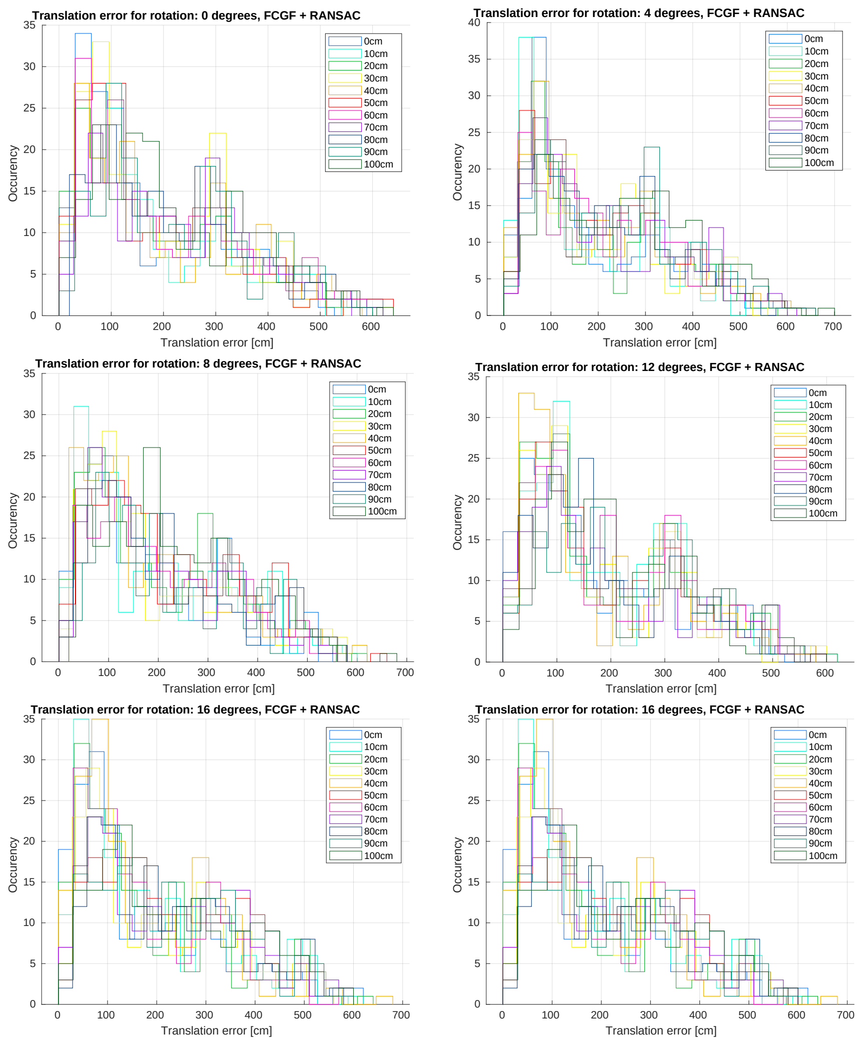


Figure 17: Translation error for FCGF + RANSAC alignment over all translation and rotation noised point-clouds.

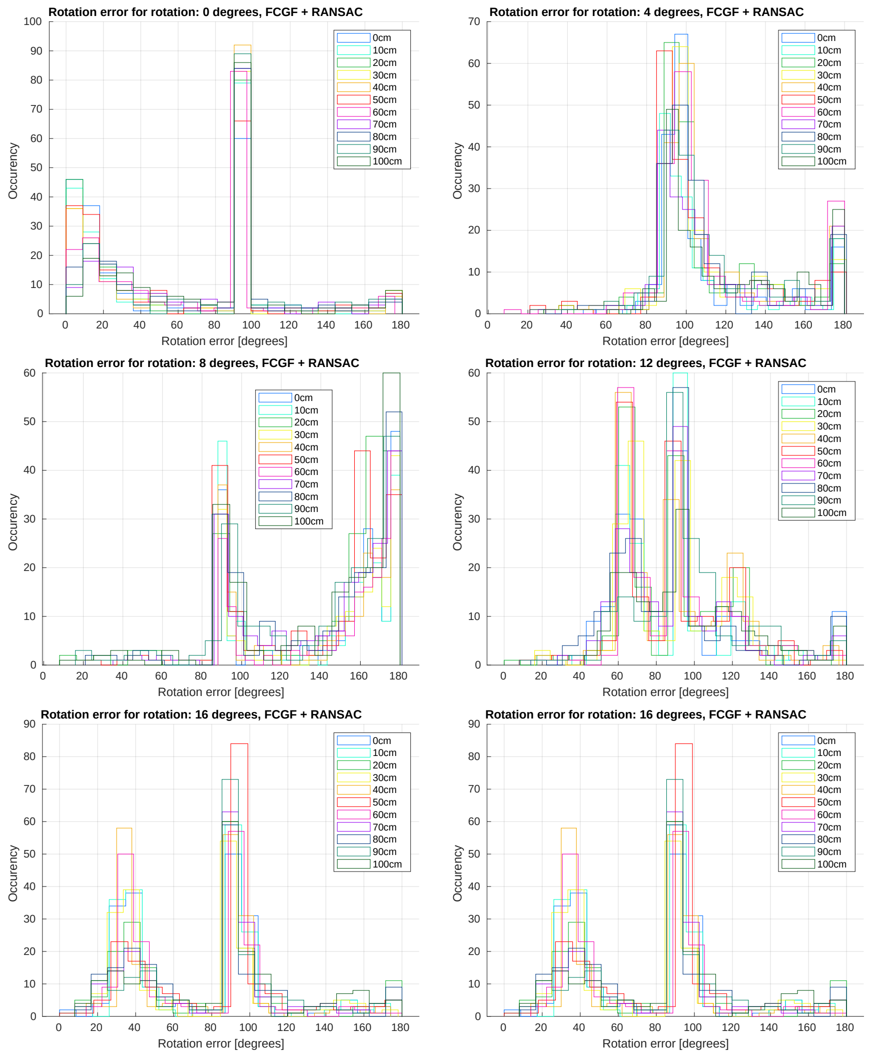


Figure 18: Rotation error for FCGF + RANSAC alignment over all translation and rotation noised point-clouds.

The code of TEASER++[8] is publicly available as C++ and Python libraries under the MIT license. We used the Python library for the evaluation. The FCGF[2] + TEASER++[8] method required similar approach as the previous method and again we had to match feature descriptors as TEASER++ assumes correspondences between two point-clouds to be the same row in the .ply file. Unfortunately even though TEASER++ ran on smaller point-clouds that we tested it on, when presented with point-clouds created from feature matches, it would run out of memory and get killed by the OS. According to the authors, this problem is ”due to the algorithm of generating the compatibility graph, the memory consumption is on the order of N^2.” [9] and since we have approximately between 20 to 30 thousand correspondences the computer runs out of memory. This could be solved by either sub-sampling point-cloud correspondences, sub-sampling point-clouds at the beginning or changing voxel size. To see TEASER++ alignment on two smaller HoloLens point-clouds view figure 19

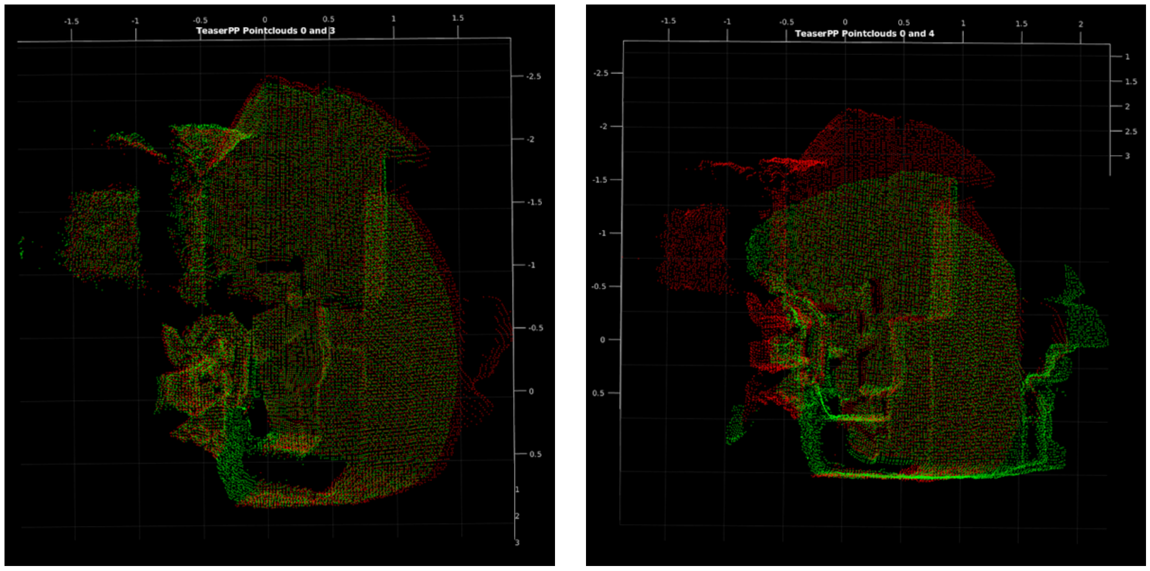


Figure 19: Example of point-clouds from HoloLens being aligned to the first HoloLens depth point-cloud, using FCGF[2] and TEASER++[8] for alignment.

Overlap Predator [8] is a model for pairwise point-cloud registration specialized in point-clouds with low overlap ($\leq 30\%$ overlap) by focusing sampling to overlap areas. The code is publicly accessible [5] under the MIT license. It is available in Python using Pytorch and CUDA, we are using pre-trained model for indoor scenes. As can be seen figure 8 even point-clouds with small overlaps are aligned correctly. From the Predator [8] alignment we get a 4x4 transformation matrix. While Overlap Predator seems to be working well on a pair of HoloLens depth point-clouds as can be seen in fig. 20. When we take tried to run Predator on a HoloLens point-cloud and a Matterport point-cloud cutout the resulting alignment was incorrect (fig. 21). The two point-clouds have very different density of points and shape of the Matterport cutout does not copy the shape of HoloLens point-cloud. To inspect why alignment fails we have decided to subsample both point-clouds (HoloLens and Matterport) to the density of 1 point per 1 centimeter and to cut the Matterport point-cloud accordingly to the HoloLens one to observe if a tighter fit of the Matterport point-cloud cutout would affect the alignment result. We have generated Matterport cutouts based on distances of points from NN in HoloLens point-cloud we then chose points 10, 9, 8, ..., 1cm distant from HoloLens point-cloud. Results of this experiment are in the fig. 22. After this experiment it was observed that Predator does not behave consistently and for the same input we get varying results (fig. 23).

TODO at last: Nahradit RANSAC v Predatoru necim jinym (Magsac, Teaser++)

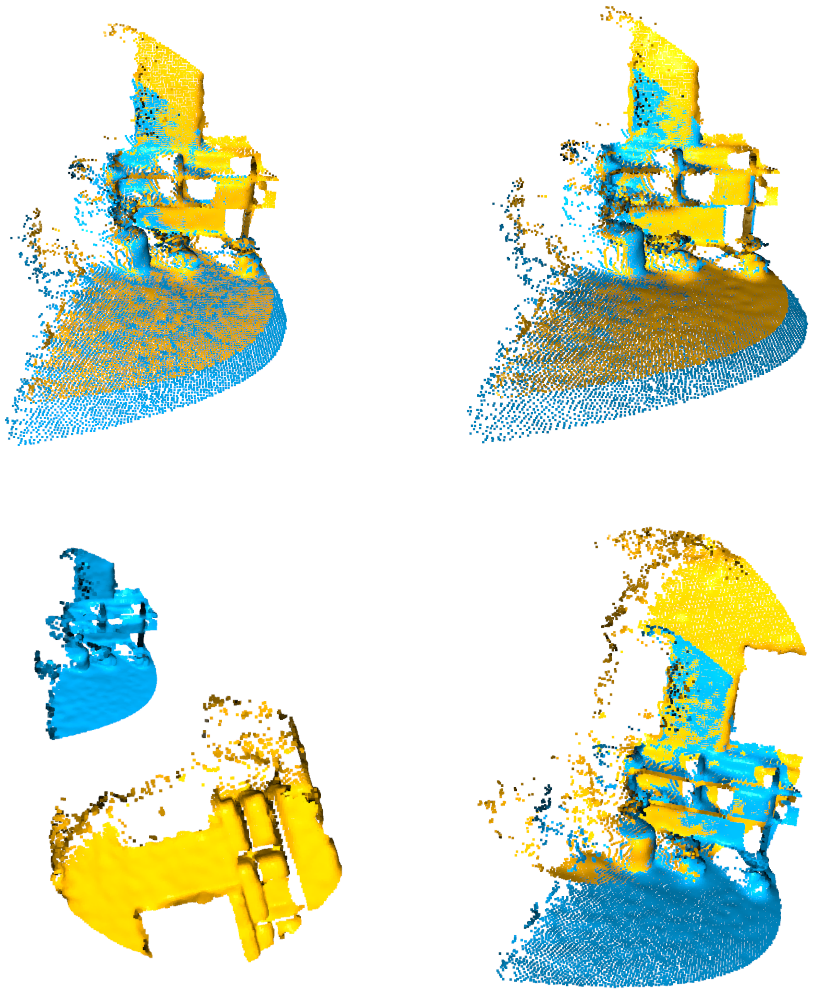


Figure 20: Example of a pair of HoloLens point-clouds aligned by Predator[8]. Left is input, right is Predator alignment, upper row are two point-clouds from the same recording session and the lower row are two point-clouds of the same object from different recording sessions.

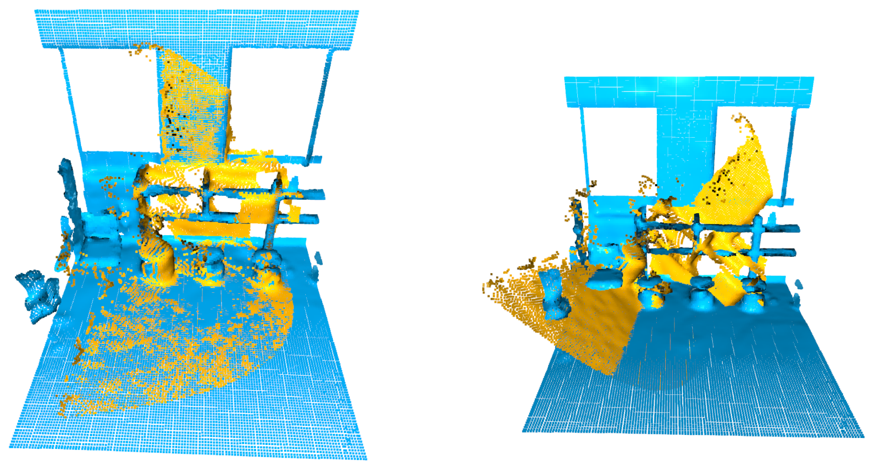


Figure 21: Result of HoloLens and Matterport point-clouds alignment by Predator[8], left is ground truth, right is Predator alignment.

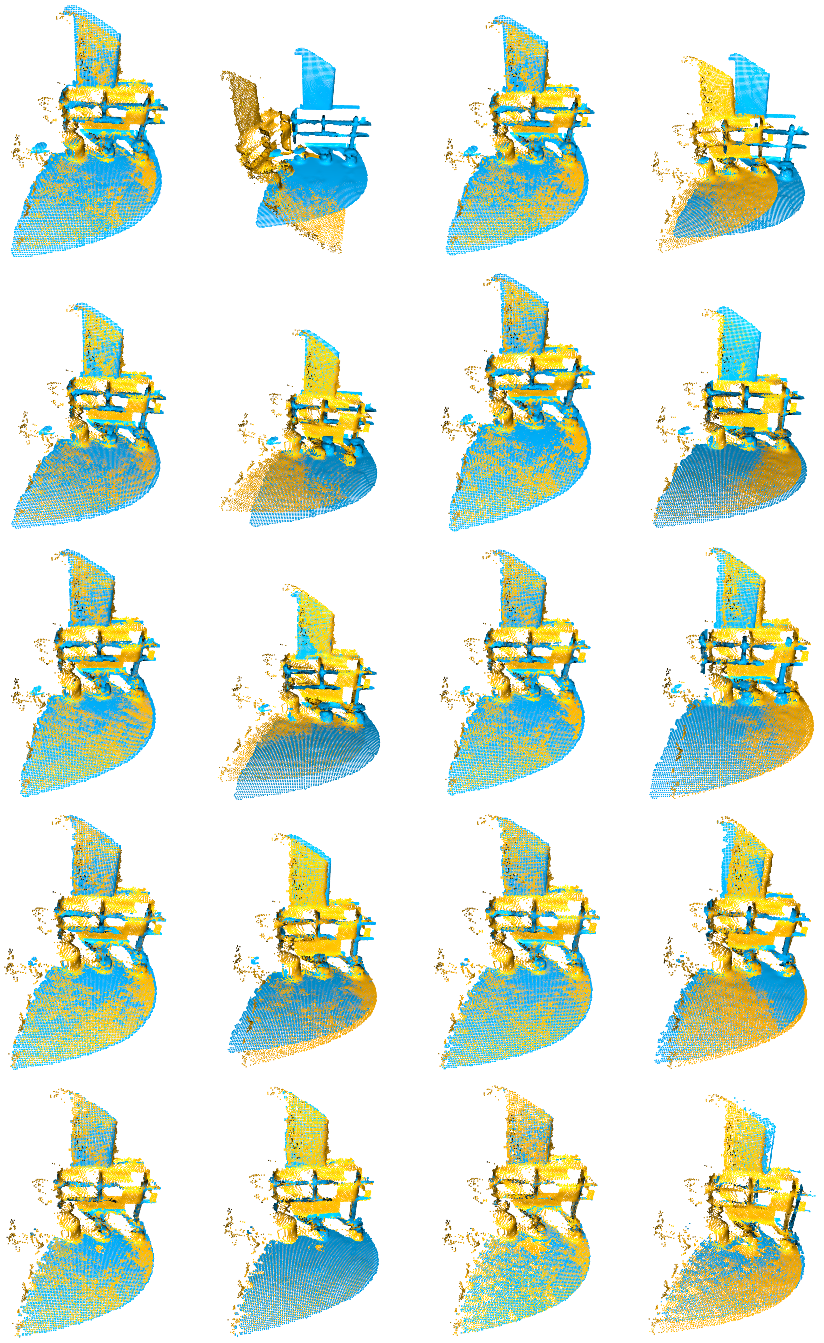


Figure 22: Results of HoloLens and Matterport point-clouds alignment by Predator[8], with changing fit of point-clouds from 10cm (upper left) to 1cm (lower right).

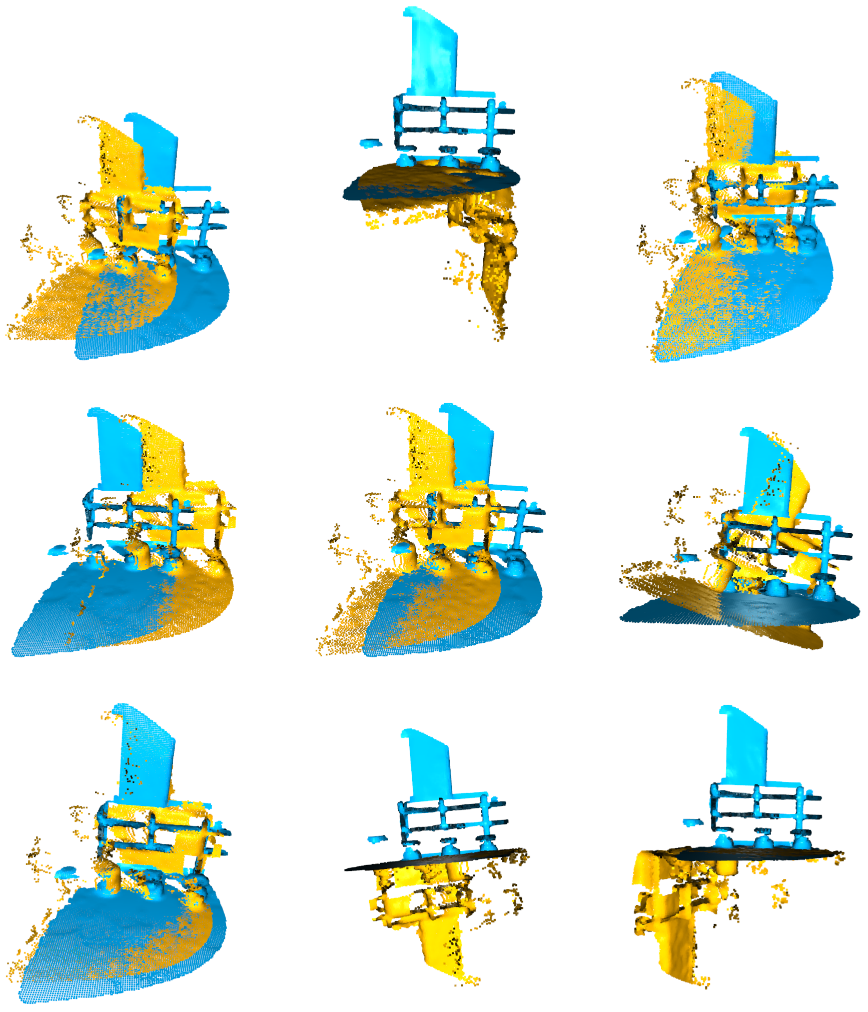


Figure 23: Results of HoloLens and Matterport point-clouds alignment by Predator[8] several times with same pair of point-clouds and same parameters, each time the alignment gave a different result.

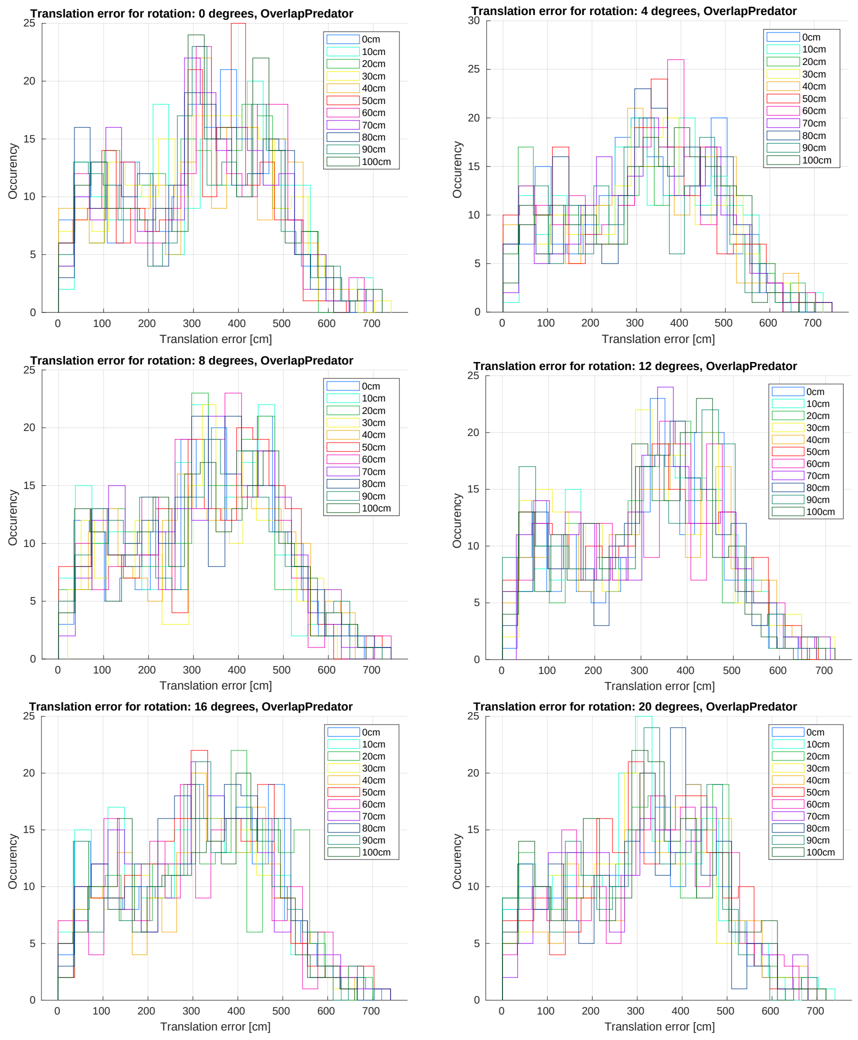


Figure 24: Translation error for Overlap Predator alignment over all translation and rotation noised point-clouds.

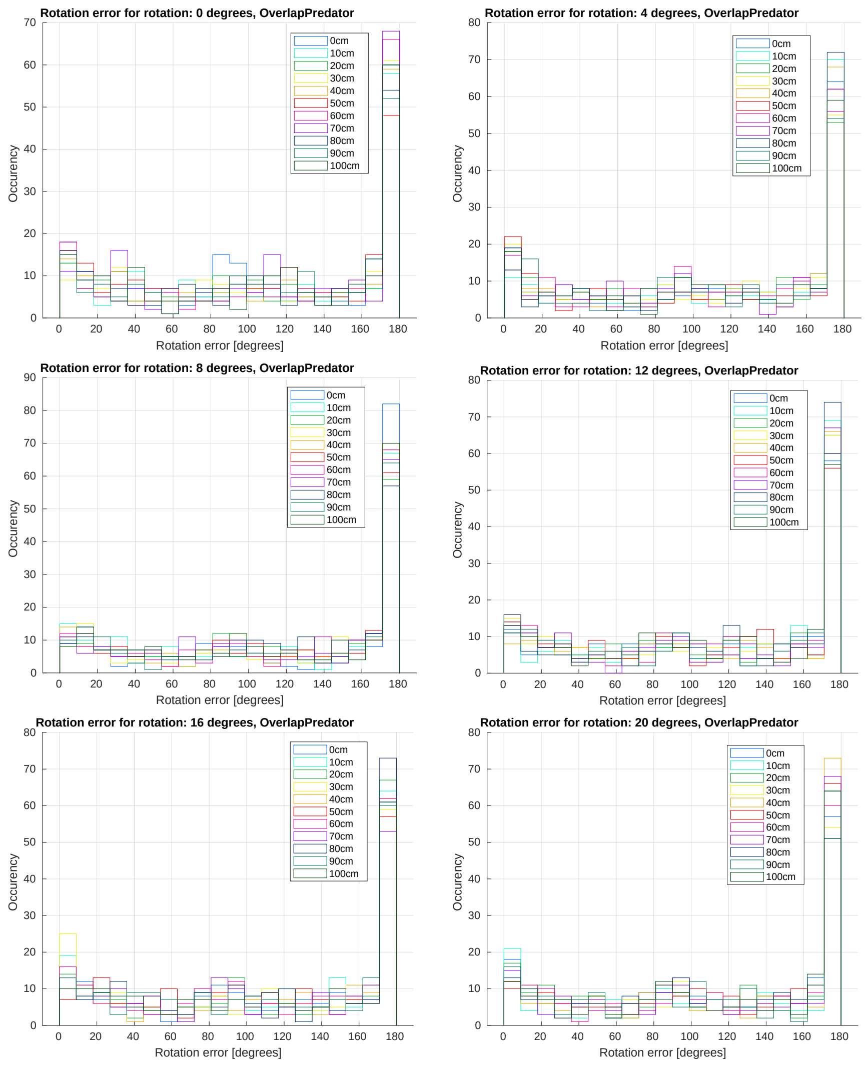


Figure 25: Rotation error for Overlap Predator alignment over all translation and rotation noised point-clouds.

Robust ICP [6], as the name indicates, is a robust version of the iterative closest point algorithm with fast convergence. It is publicly available under the MIT license as a C++ library. Output of the R-ICP is a transformed point-cloud and a 4x4 transformation matrix. As can be seen in figures 26 and 27, errors for just translated point-clouds are relatively small compared to errors while applying rotation.

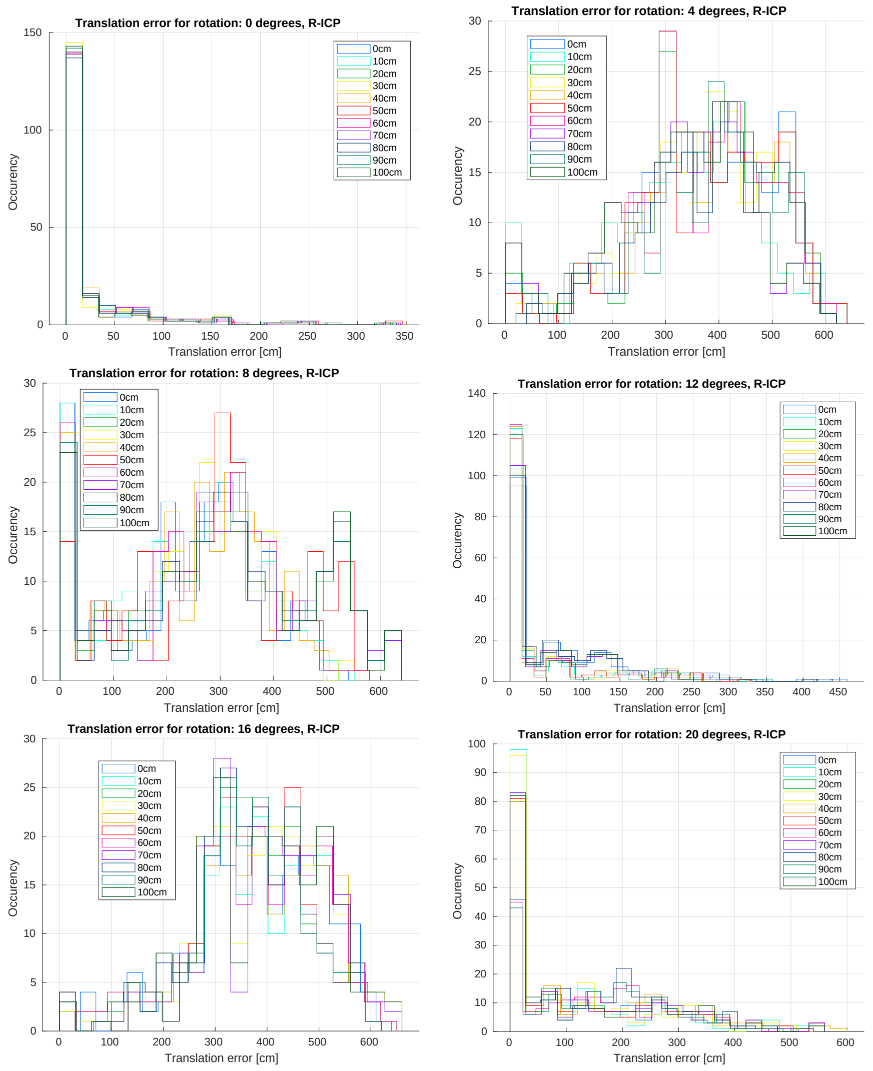


Figure 26: Translation error for R-ICP alignment over all translation and rotation noised point-clouds.

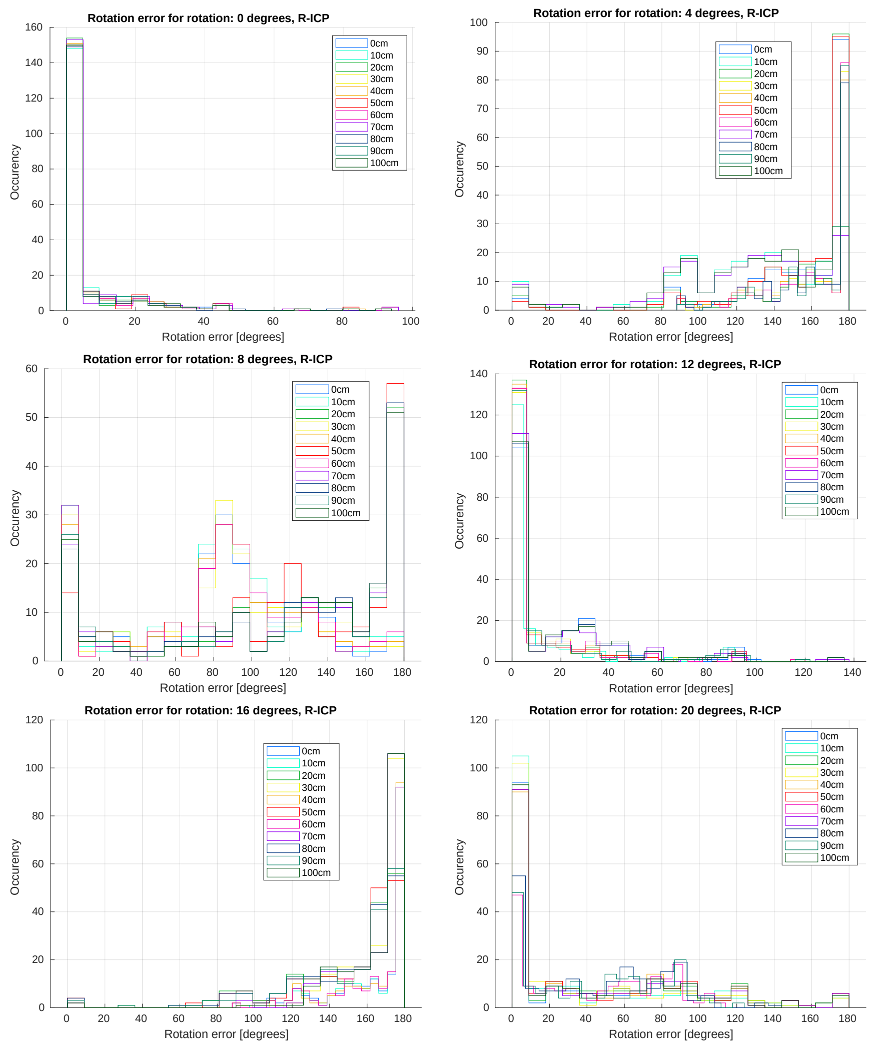


Figure 27: Rotation error for R-ICP alignment over all translation and rotation noised point-clouds.

5 Conclusion

In this work we presented our process of obtaining ground truth data from HoloLens depth point-clouds, Vicon data and a Matterport point-cloud of a room. We then evaluated several point-cloud alignment methods using the acquired ground truth. There is still some room for development like for example training point-cloud aligning methods on our datasets.

References

1. WANG, Jikai; WANG, Peng; DAI, Deyun; XU, Meng; CHEN, Zonghai. Regression forest based rgb-d visual relocalization using coarse-to-fine strategy. *IEEE Robotics and Automation Letters*. 2020, vol. 5, no. 3, pp. 4431–4438.
2. CHOY, Christopher; PARK, Jaesik; KOLTUN, Vladlen. Fully convolutional geometric features. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 8958–8966.
3. YANG, Heng; SHI, Jingnan; CARLONE, Luca. Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*. 2020, vol. 37, no. 2, pp. 314–333.
4. BARATH, Daniel; MATAS, Jiri; NOSKOVA, Jana. MAGSAC: marginalizing sample consensus. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10197–10205.
5. HUANG, Shengyu; GOJCIC, Zan; USVYATSOV, Mikhail; WIESER, Andreas; SCHINDLER, Konrad. Predator: Registration of 3D Point Clouds With Low Overlap. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 4267–4276. Available also from: `https://github.com/overlappredator/OverlapPredator5`.
6. ZHANG, Juyong; YAO, Yuxin; DENG, Bailin. Fast and Robust Iterative Closest Point. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2021, pp. 1–1.
7. HÜBNER, Patrick; CLINTWORTH, Kate; LIU, Qingyi; WEINMANN, Martin; WURSTHORN, Sven. Evaluation of HoloLens tracking and depth sensing for indoor mapping applications. *Sensors*. 2020, vol. 20, no. 4, p. 1021.
8. HUANG, Shengyu; GOJCIC, Zan; USVYATSOV, Mikhail; WIESER, Andreas; SCHINDLER, Konrad. Predator: Registration of 3D Point Clouds With Low Overlap. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 4267–4276. Available also from: `https://arxiv.org/abs/2011.13005`.
9. SHI, Jingnan. The cloudPoint size more than 5000, RobustRegistrationSolver will be collapse. In: 2021. Available also from: `https://github.com/MIT-SPARK/TEASER-plusplus/issues/118`.