3D OBJECT REGISTRATION IN THE WILD

Rishab Shah

May 7, 2018

Boston University

Department of Electrical and Computer Engineering

BOSTON UNIVERSITY

3D OBJECT REGISTRATION IN THE WILD

Rishab Shah

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

May 7, 2018

Contents

1 Introduction

RGB-D cameras became part of our daily life in applications such as game interaction and human-computer interface, just to cite a few. Because of their easy programming interface and response precision, such cameras have also been increasingly used for 3D reconstruction and movement analysis. Simultaneous localization and mapping (SLAM) is a method to simultaneously estimate positions of newly perceived landmarks and the position of the mobile robot itself while mapping. Localization is the process of establishing the spatial relationships between the robot and stationary objects, mapping is the process of establishing the spatial relationships among stationary objects. Localization and mapping are difficult because of their unobservable states in the real world. Perception sensors such as cameras, radar and laser range finders, and motion sensors such as odometry and inertial measurement units are noisy.

Over the past two decades, the Simultaneous Localization and Mapping (SLAM) problem has received considerable attention in the mobile robotics and artificial intelligence literature [1] [2].

One of the big problems is a lack of robustness. This concerns the difficulty of recognizing already mapped areas, which is also known as loop-closure detection. Those problems can be addressed by detecting when the robot is navigating through a previously visited place from local measurements. In order to close loops in a map, the system must recognize when it has returned to a previously mapped region of the world. Essentially, at this point two regions in the map are found to be the same region in the world even though their position is incompatible given the uncertainty estimate in the map – the classic loop closure problem. The system must then be able to calculate the transformation needed to align these two regions to ‘close the loop’. The naïve approach adopted in early SLAM work simply performs a nearest neighbor statistical gate on the likelihood of the current measurements given map and pose estimates. If the pose estimate is in gross error, while in reality the vehicle is in an already mapped area, the

likelihood of measurements being explained by the pose and the map estimate is very small. The resulting effect of this is that loop closure is not detected. Previously visited areas are re-mapped, but in the wrong global location, error accumulates without bound.

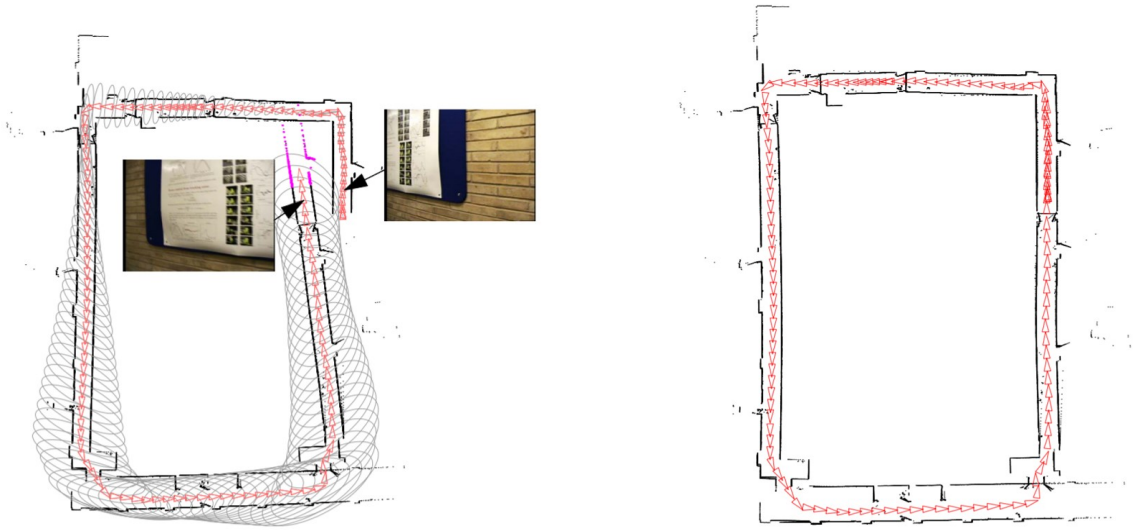


Fig. 1. (a) Shows a snapshot of the SLAM algorithm in [3] just before loop closing takes place. The vehicle poses stored in the state vector are shown as red triangles. Global uncertainty (gray ellipses) increases as the length of the excursion from the start location increases. A poor scan match at the bottom right introduces a small angular error which leads to a gross error in pose estimate when in reality the vehicle has returned to near its starting locations (top right). The inset images are the two camera views used in the loop-closing process. The left hand image is the query image and the right hand one the retrieved, matching image. (b) Shows the final map after applying the loop closing constraint.

Fig. 1, shown above, is an illustration of a poor case of loop closing from [3] and resolving the issue by relying on temporal relationships between local scenes.

In this paper, we consider robots equipped with 3D scanner (Kinect), and define the problem of loop closure detection as determining whether or not the point clouds obtained from the scanner are from the same location. We propose point cloud registration using point clouds recorded along the mapping path, based on object-level descriptors using convolutional neural networks for image classification and detection tasks and register them by implementing the Iterative Closest Point (ICP) algorithm. A point cloud is a data structure used to represent a collection of multi-dimensional points and is commonly used to represent three-dimensional data. In a 3D point cloud, the points usually represent the X, Y, and Z geometric coordinates of an underlying sampled surface. Point clouds can be acquired from hardware sensors such as stereo cameras, 3D scanners, or time-of-flight cameras, or generated from a computer program synthetically.

2 Literature review

Loop closures consist of recognizing a place that has already been visited in an excursion of arbitrary length. Problems arise when two different place from the environment are recognized as the same, this is known as perceptual aliasing. Detection methods for loop closure detection can be divided into three categories: map-to-map, image-to-image and image-to-map. Categories differ mainly from where the association data are taken from. However, the ideal would be to build a system that combines the advantages of all three categories. [4] propose a system where they use the notion of visual saliency to focus the selection of suitable image-feature descriptors for storage in a database. This time information is used to discover loop closing events, and this is achieved independently of estimated pose and the map. All uncertainties collapse after a loop closure whether or not the loop closure performed was correct or not.

There is a plethora of attempts to visually detect closed loops based on regular (intensity) camera images [5], [6], [7]. The general consensus about the approach chosen for visual loop closure detection seems to contain first finding the previous images that are most similar to the current image (based on a content-based image retrieval scheme) and afterwards discarding those that cannot be loops due to heuristics or geometry consistency checks. The most commonly used image retrieval scheme is Bag of Visual Words. Some algorithms contain both interest point detection and local descriptor extraction, e.g. Normal Aligned Radial Features (NARF) [8] or SIFT [9]. NARF interest points can be found near the borders of objects but not directly on these borders, which is usually good for depth images or point clouds. Local descriptors are designed to describe an image patch (usually located around an interest point) in a compact but distinctive way. Similar to the location of interest points, a point's descriptor is usually desired to be invariant to changes in scale and perspective. An example of loop closure detection for 2D point clouds is the work by Bosse et al [10]. They use consecutive point clouds to build submaps, which are then compressed using orientation and projection histograms as a compact description of submap characteristics. For the similar problem of object

recognition using 3D points, regional shape descriptors have been used [11, 12]. These techniques operate by determining point-to-point correspondences within the scan given the current alignment offset estimate, refining that estimate by solving an optimization problem, then repeating until convergence or a maximum number of iterations. Iterative methods often converge to a good solution quickly, but the requirement of a good initial guess suggests that they are not applicable in all situations. Additionally, the fact that scans do not match perfectly (due to incomplete overlap, occlusion, and noise) suggests that iterative techniques should be made robust to outliers.

3 Our method

The implementation of our method used here is described as follows: objects in the scene are detected and recognized using the Mobilenet-Single Shot multibox Detector (SSD) convolutional neural network using the color image frame to obtain the bounding boxes around the required object, the corresponding point cloud is cropped according to the detection in the previous step, the point cloud is filtered to remove outliers to make the registration process more susceptible to convergence, two different point clouds from different angles are registered using the ICP algorithm.

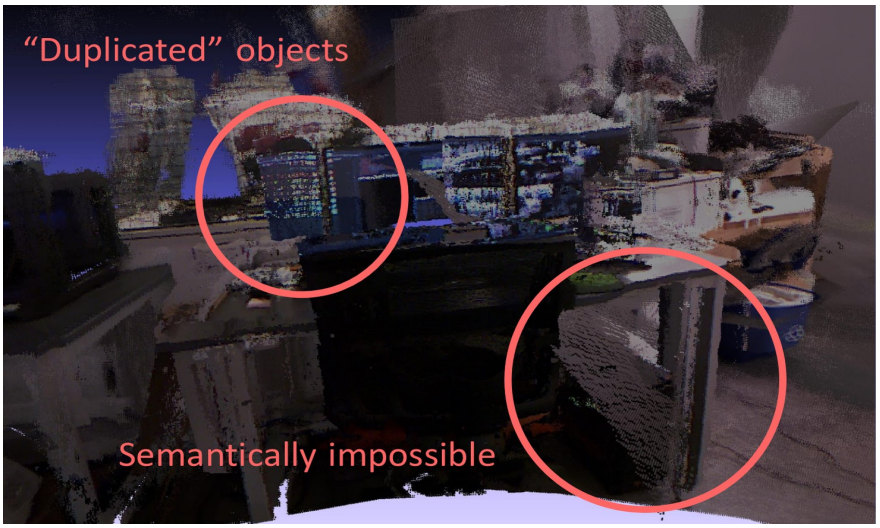
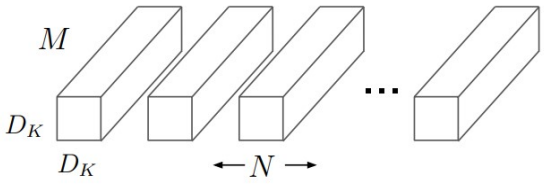


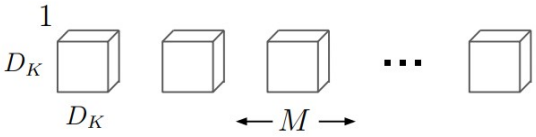
Fig. 2. This image shows how we might get the presence of “duplicate” objects while performing point cloud registration without using any semantic information.

3.1 Object detection and recognition

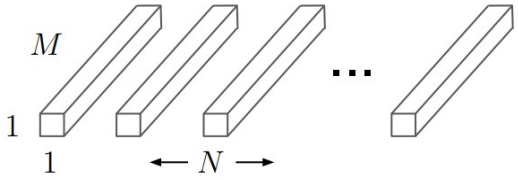
Deep learning has fueled tremendous progress in the field of computer vision in recent years, with neural networks repeatedly pushing the frontier of visual recognition technology. While many of those technologies such as object, landmark, logo and text recognition are provided for internet-connected devices through the Google Cloud Vision API, the ever-increasing computational power of mobile devices can enable the delivery of these technologies into the hands of users, anytime, anywhere, regardless of internet connection. However, visual recognition for on device and embedded applications poses many challenges - models must run quickly with high accuracy in a resource-constrained environment making use of limited computation, power and space. Since our goal is to use a minimal amount of computation on board a robot, we use MobileNets [13], which are small, low-latency, low-power models parameterized to meet the resource constraints of a robot's computation capacity. The MobileNet model is based on depthwise separable convolutions which is a form of factorized convolutions which factorize a standard convolution into a depthwise convolution and a 1x1 convolution called a pointwise convolution. For MobileNets the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. Fig. 3 shows how a standard convolution is factorized into a depthwise convolution and a 1x1 pointwise convolution.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Fig. 3. The standard convolutional filters in (a) are replaced by two layers in (b) and pointwise convolution in (c) to build a depthwise separable filter. D_K is the spatial of the kernel assumed to be square, M is the number of input channels, and N is the number of output channels.

We use a detection model pre-trained on the COCO dataset (a dataset containing over 250,000 images for object detection and recognition). The dataset contains objects in various categories e.g., cars, chairs, keyboards, humans, books, etc. In addition to this, we use a detection model called the Single Shot multibox Detector (SSD) to predict different objects. SSD is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network. This makes SSD easy to train and straightforward to integrate into systems that require a detection component. Experimental results [14] on the PASCAL VOC, COCO, and ILSVRC datasets have shown that SSD has competitive accuracy to methods that utilize an additional object proposal step and is much faster. For 300x300 input form the COCO dataset, SSD achieves results in an average of 30ms.

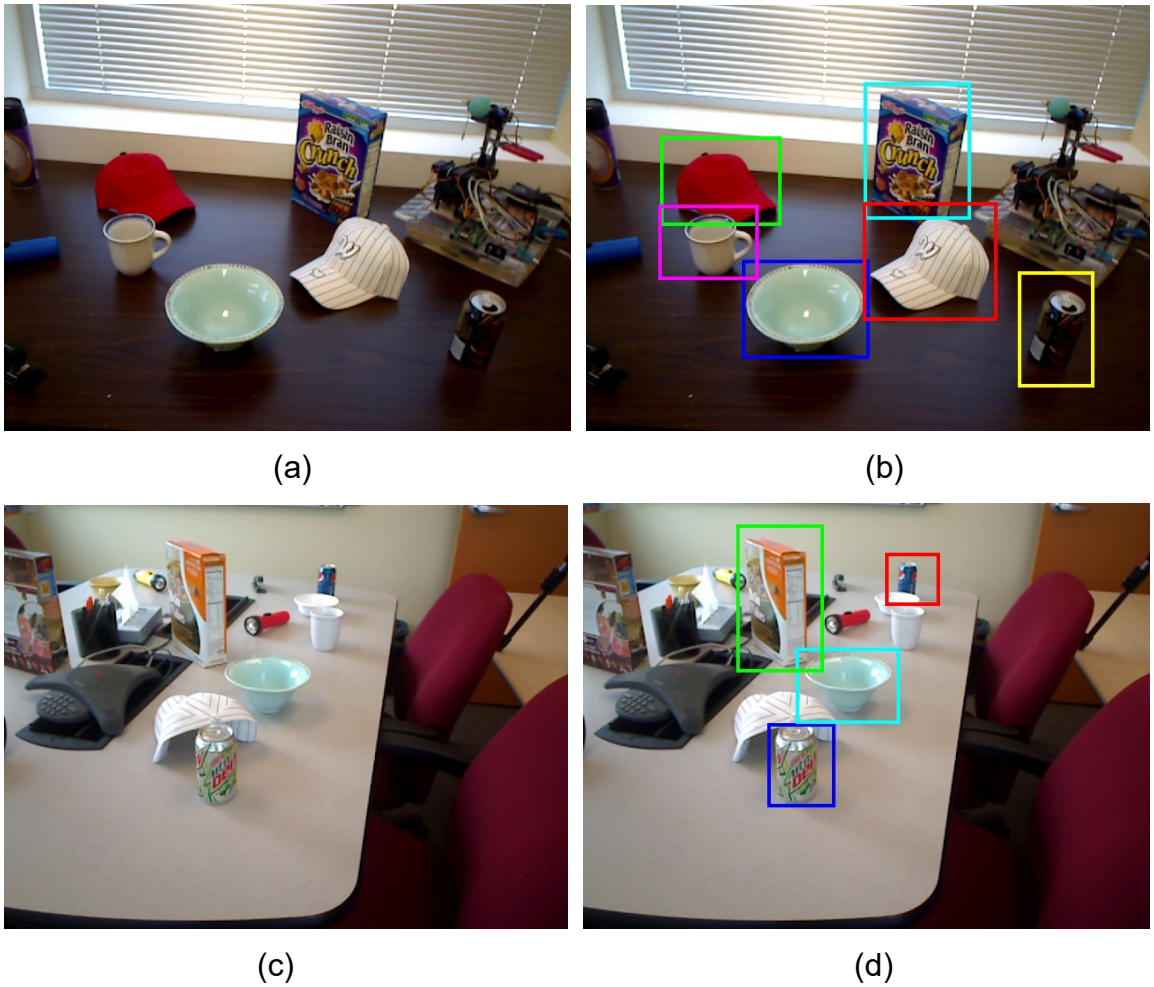


Fig. 4. (a) and (c) show two scenes, a desk and a meeting table, strewn with different objects. (c) and (d) show the result of the object recognition in the form of bounding boxes around different recognized objects such as the cap, cereal box, soda can, etc.

3.2 Point cloud segmentation

We segment the point clouds corresponding to the color image frames by making use of the regions of interest that we obtained in the form of bounding boxes and cropping the point clouds. We observe from the Fig. 5 that there are some outliers present in the segmented point cloud that prevents the point cloud registration from converging.

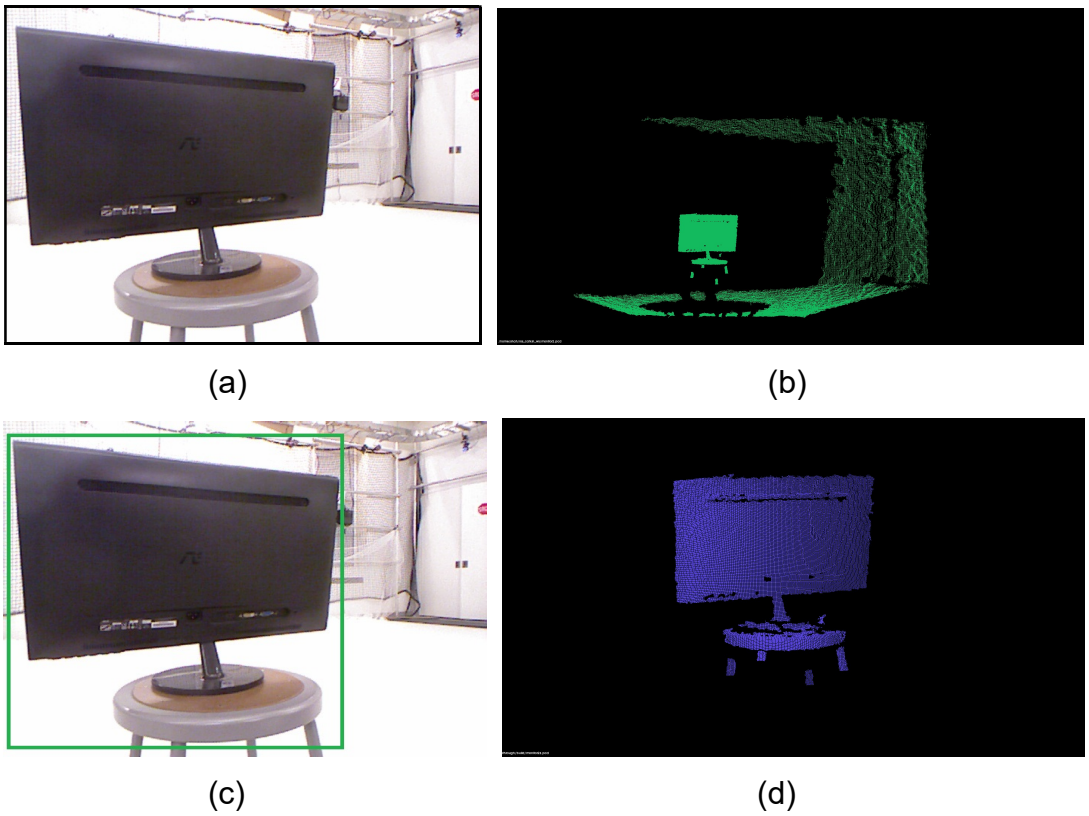


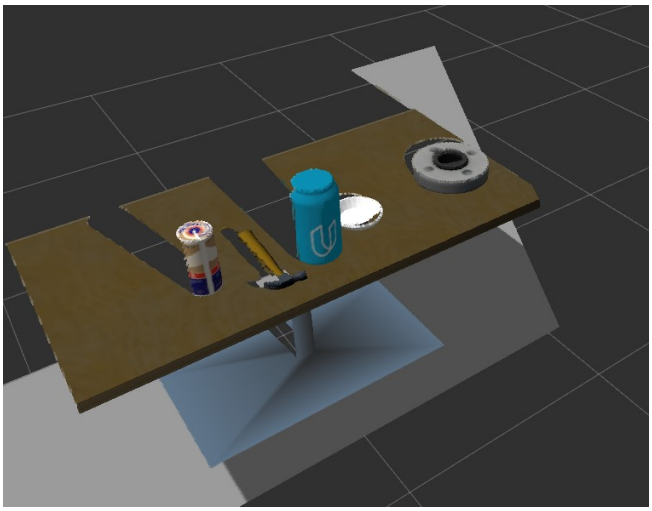
Fig. 5. (a) represents the color image frame of a monitor. (b) represents the corresponding point cloud. The points in the background need to be cropped and we use the object recognition to do so. (c) is the result of the object recognition using the MobileNets-SSD model. (d) is the result of cropping the point cloud according to the bounding box.

To reduce the point cloud and remove the outliers, we use two methods: a passthrough filter and RANSAC. A passthrough filter cuts off values that are either inside or outside a given user range. In our case, we assume all images are recorded from a close range since we are in an indoor environment and objects are suspended above the ground as they might be on a table surface or similar, we introduce a threshold to our point cloud of z = 1.75m and y = 0.75m. This means, we crop anything beyond these values in our point cloud's coordinate system. In addition, we use a geometric filtration algorithm in the form of Random Sampling and Consensus (RANSAC). RANSAC extracts the outliers to remove points corresponding to the table surface to get a better approximation prior to ICP. Random sample and consensus (RANSAC) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates.

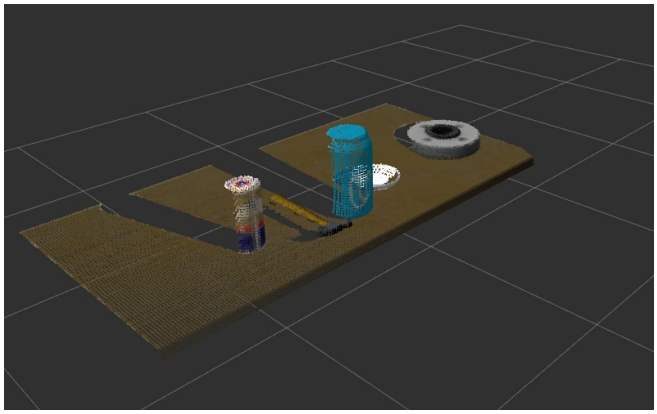
In the first step, a sample subset containing minimal data items is randomly selected from the input dataset. A fitting model and the corresponding model parameters are computed using only the elements of this sample subset. The cardinality of the sample subset is the smallest sufficient to determine the model parameters.

In the second step, the algorithm checks which elements of the entire dataset are consistent with the model instantiated by the estimated model parameters obtained from the first step. A data element will be considered as an outlier if it does not fit the fitting model instantiated by the set of estimated model parameters within some error threshold that defines the maximum deviation attributable to the effect of noise.

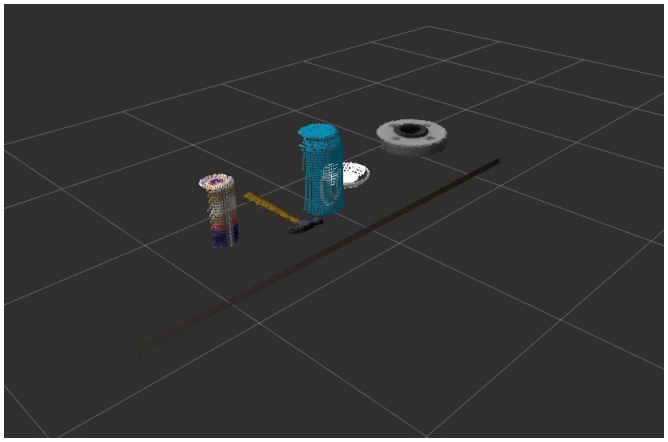
The set of inliers obtained for the fitting model is called consensus set. The RANSAC algorithm will iteratively repeat the above two steps until the obtained consensus set in certain iteration has enough inliers.



(a)



(b)



(c)

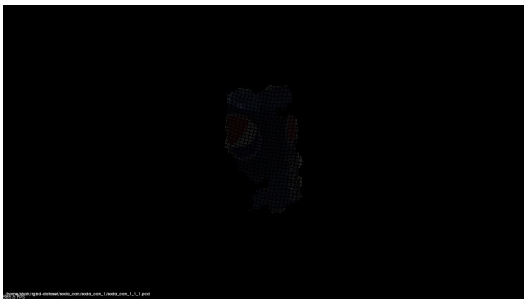
Fig. 6. This figure is an illustration of the passthrough filter and the RANSAC algorithm. (a) shows the point cloud of a scene in which there are 4 objects on a table. (b) is the result of using the passthrough filter and removing point beyond a threshold. (c) is the result after using the RANSAC algorithm to remove point on the table surface which are deemed as outliers.

(a)

(b)

(c)

(d)



(e)

(f)

Fig. 7. (a) and (d) are the point clouds of the scenes in Fig. 4. They are represented using their corresponding RGB values for better visualization. (b), (c), (e) and (f) are the segmented point clouds of a cap, a cereal box, a soda can and a chair respectively.

3.3 Point cloud registration

The problem of consistently aligning various 3D point cloud data views into a complete model is known as registration. Its goal is to find the relative positions and orientations of the separately acquired views in a global coordinate framework, such that the intersecting areas between them overlap perfectly. For every set of point cloud datasets acquired from different views, we therefore need a system that is able to align them together into a single point cloud model, so that subsequent processing steps such as segmentation and object reconstruction can be applied. We sometimes refer to the

problem of registering a pair of point cloud datasets together as pairwise registration, and its output is usually a rigid transformation matrix (4x4) representing the rotation and translation that would have to be applied on one of the datasets (let's call it source) in order for it to be perfectly aligned with the other dataset (let's call it target, or model).

The steps performed in a pairwise registration step are shown in the diagram below.

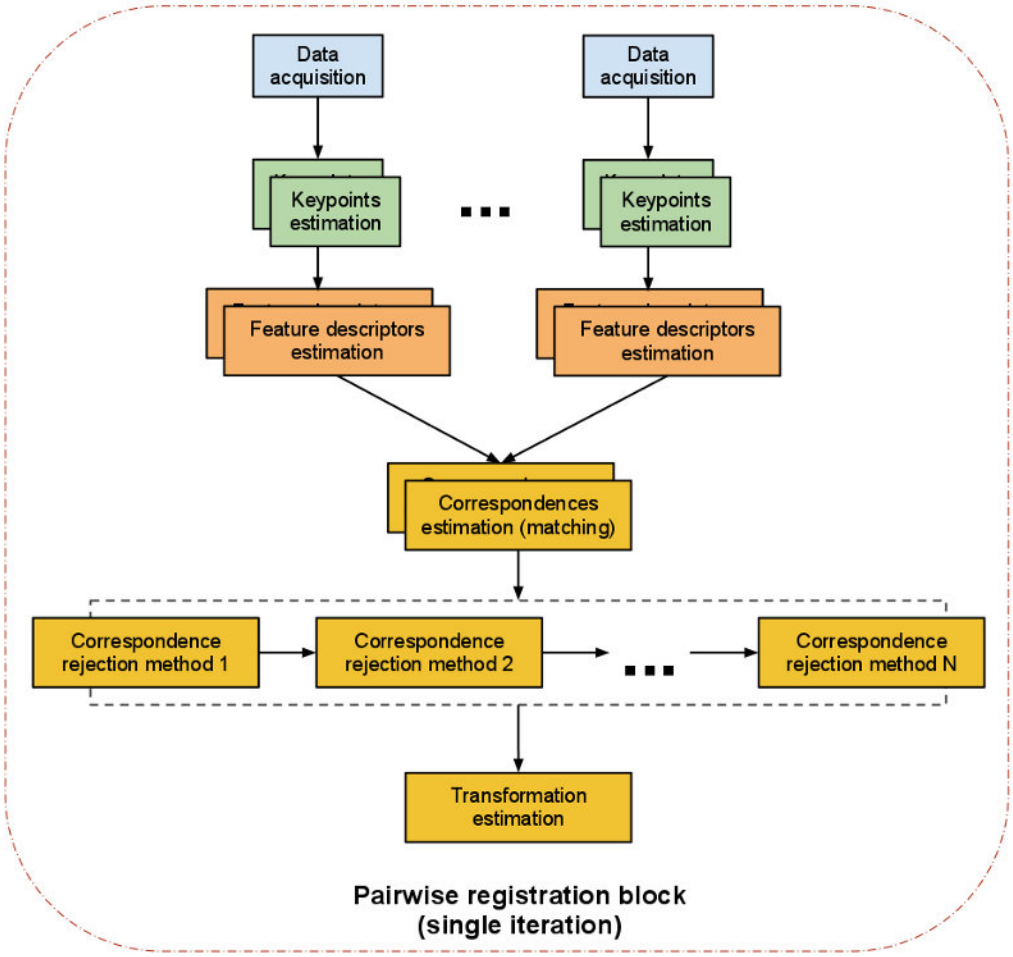


Fig. 8. A block diagram of registration of point clouds using ICP. Please note that we are representing a single iteration of the algorithm. The programmer can decide to loop over any or all of the steps.

In the Iterative Closest Point or, in some sources, the Iterative Corresponding Point, one point cloud (vertex cloud), the reference, or target, is kept fixed, while the other one, the source, is transformed to best match the reference. The algorithm iteratively revises the transformation (combination of translation and rotation) needed to minimize an error

metric, usually a distance from the source to the reference point cloud, such as the sum of squared differences between the coordinates of the matched pairs.

- For each point (from the whole set of vertices usually referred to as dense or a selection of pairs of vertices from each model) in the source point cloud, Match the closest point in the reference point cloud (or a selected set).
- Estimate the combination of rotation and translation using a root mean square point to point distance metric minimization technique which will best align each source point to its match found in the previous step. This step may also involve weighting points and rejecting outliers prior to alignment.
- Transform the source points using the obtained transformation.
- Iterate (re-associate the points, and so on).

Let's have a look at the steps of the pipeline. Keypoints are interest points that have a "special property" in the scene, like the corner of a book or you can take every point as a keypoint as well. For correspondence estimation, we use kd-tree nearest neighbor search (FLANN) as a form of point matching. Fast Library for Approximate Nearest Neighbors (FLANN) is a library for performing fast approximate nearest neighbor searches in high dimensional spaces. Naturally, not all estimated correspondences are correct. Since wrong correspondences can negatively affect the estimation of the final transformation, they need to be rejected. This could be done using RANSAC or by trimming down the amount and using only a certain percent of the found correspondences.

A special case is one to many correspondences where one point in the model corresponds to a number of points in the source. These could be filtered by using only the one with the smallest distance or by checking for other matchings nearby.

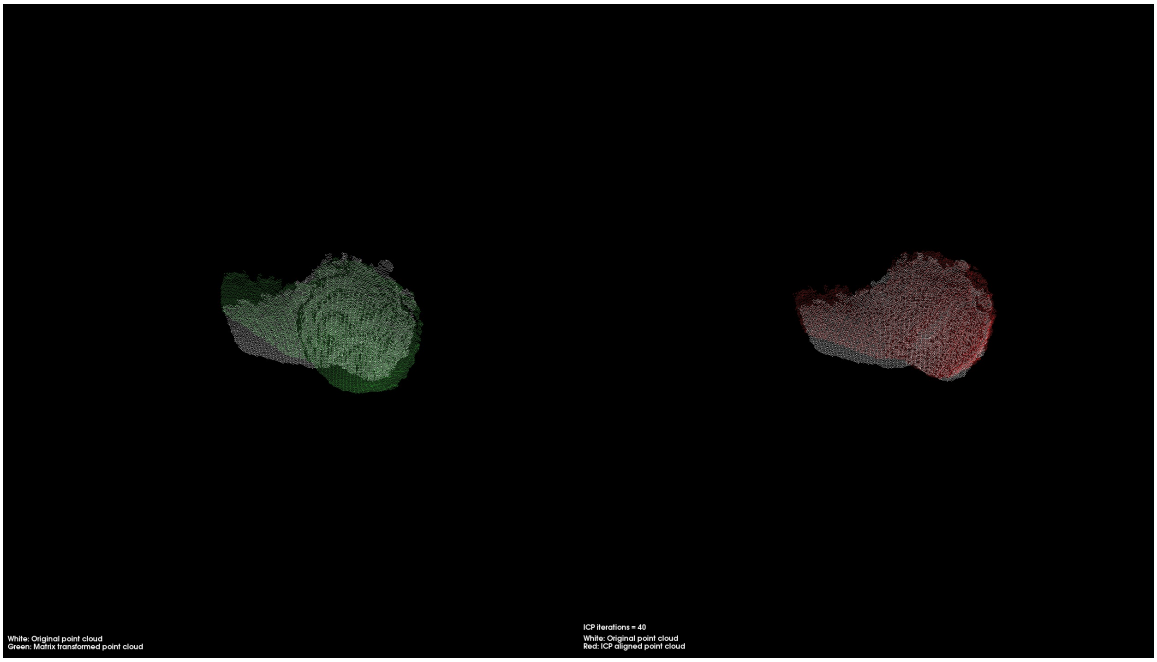
The last step is to calculate the transformation. This is done as follows:

- Evaluate the error metric (sum of squared differences between the coordinates of the matched pairs) based on correspondence.

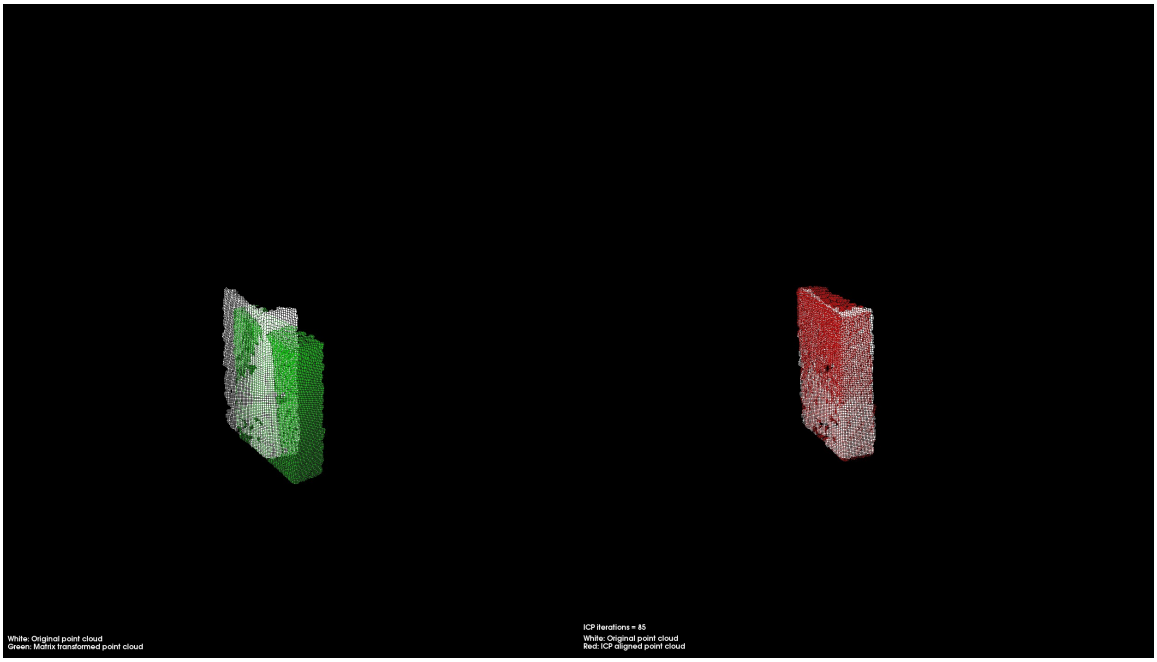
- Estimate a (rigid) transformation between camera poses and minimize error metric, in our case, we use Singular-Value Decomposition (SVD) for the motion estimate.
- Optimize the structure of the points.
- Use the rigid transformation to rotate/translate the source onto the target, and potentially run an internal ICP loop with all points.
- Iterate until some convergence criterion is met.

4 Experimental results

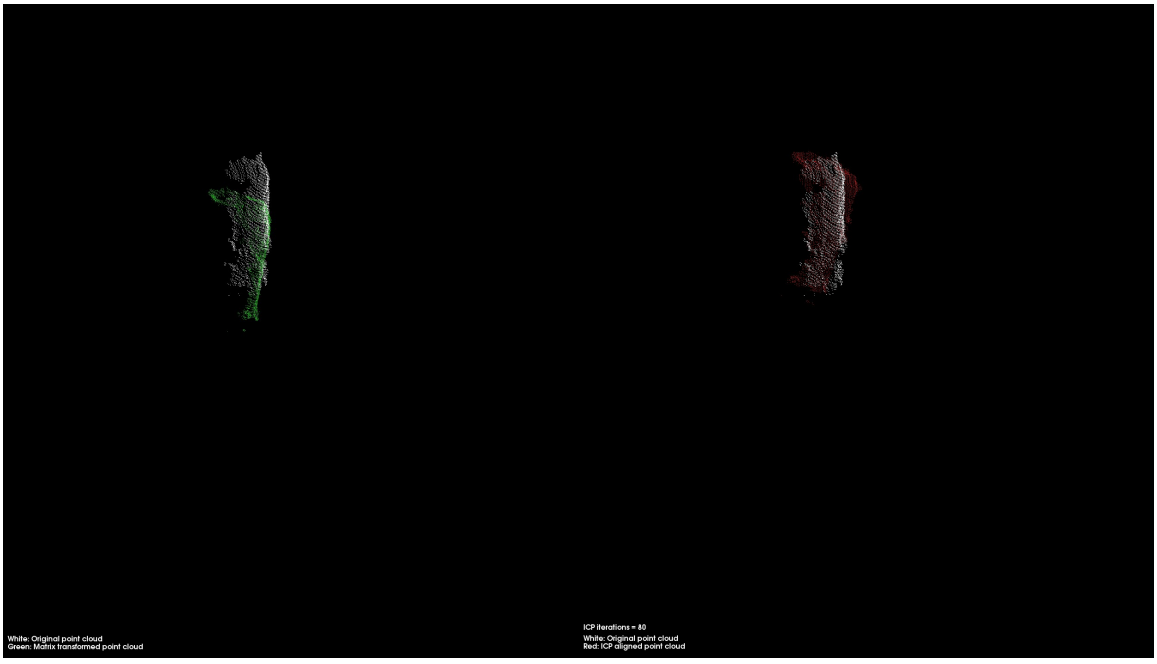
The results of the registration are illustrated as follows.



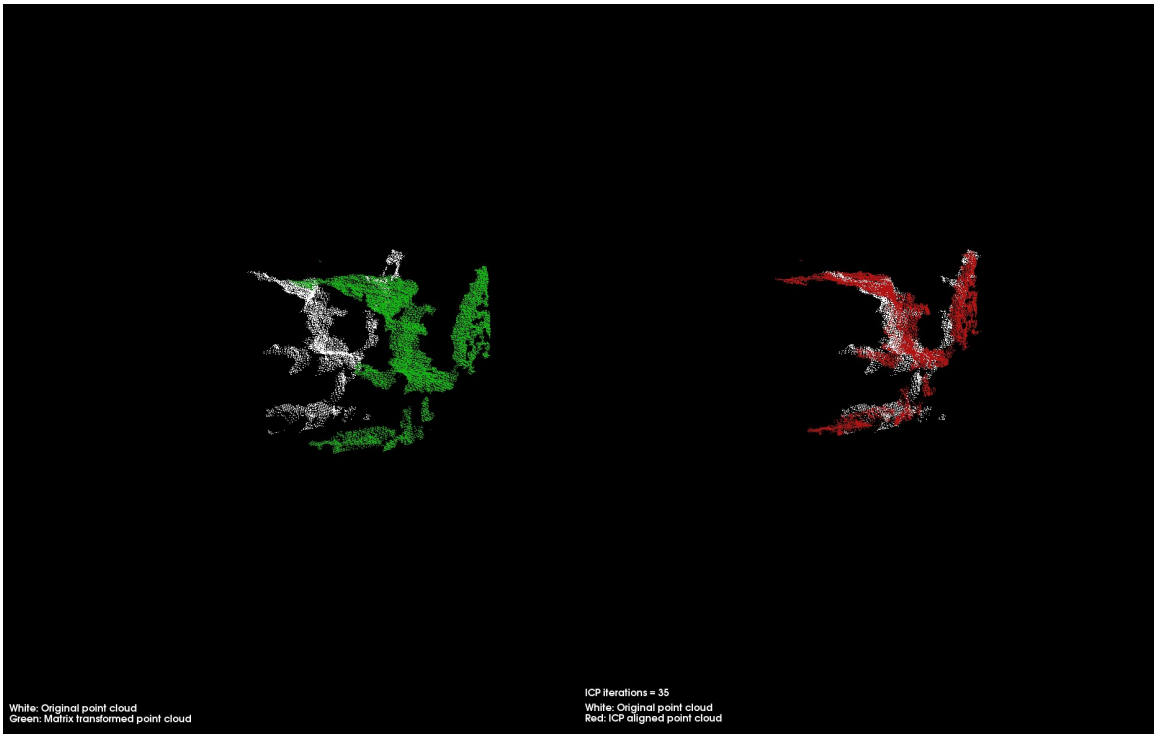
(a)


White: Original point cloud
Green: Matrix transformed point cloud
ICP iterations = 85
White: Original point cloud
Red: ICP aligned point cloud

(b)


White: Original point cloud
Green: Matrix transformed point cloud
ICP iterations = 80
White: Original point cloud
Red: ICP aligned point cloud

(c)



(d)

Fig. 9. The above figures represent the results of ICP for the cap (a), cereal box (b), soda can (c), and chair (d).

Table 1 Errors and iterations for convergence for various viewing angles

	No. of iterations	Alignment error (in m)	Change in viewing angle (X-axis [in degrees])	Change in viewing angle (Y-axis [in degrees])	Change in viewing angle (Z-axis [in degrees])
Cap	40	0.0000445	30	10	2
Cereal Box	85	0.00056	192	-3	0
Soda Can	80	0.2563	55	0	0
Chair	35	0.0000236	15	6	0
Monitor	26	0.000178	140	48	6
Bowl	68	0.0945	39	15	5
Flashlight	107	0.3586	23	2	-5
Book	24	0.000423	63	-12	0
Keyboard	78	0.00628	36	-5	-3

The results show very low alignment errors for different objects. Alignment error is the average Euclidean distance between closest points measured in meters. Note that we see higher errors for the soda can, the bowl and the flashlight. This is due to the fact that there are very few points in the cloud to adequately perform ICP. The viewing angles represented in the table are the different angles from which the second point cloud was recorded in reference to the first. These values were obtained by placing 3 colored markers on top of the Kinect and recording their ground-truth pose using the Opti-track cameras.

Table 2 Alignment errors for change in viewing angles in X-axis

Viewing angle	Cap	Cereal Box	Soda Can	Chair
-180	0.000963	0.0002	0.0323	0.000165
-170	0.0005884	0.00046	0.02205	0.00045
-160	0.000656	0.0006416	0.014	0.000845
-150	0.0006548	0.000895	0.0651	0.003
-140	0.0009614	0.005651	0.0561	0.00654
-130	0.001259	0.0069	0.048	0.00894
-120	0.003645	0.00982	0.0459741	0.00917
-110	0.0099687	0.0078	0.03641	0.00981
-100	0.0429	0.0754	0.051	0.00743
-90	0.066	0.088	0.025	0.0989
-80	0.0261	0.0625	0.0365	0.0649
-70	0.001256	0.049	0.0478	0.0189
-60	0.0008745	0.005489	0.0216	0.005654
-50	0.000696	0.002183	0.0312	0.0064
-40	0.000565	0.0009841	0.0451	0.00456
-30	0.00036	0.0005511	0.0519	0.00111
-20	0.000123	0.004774	0.0526	0.00053
-10	0.0000484	0.000086	0.03259	0.00015
0	0	0	0	0
10	0.00002638	0.000089	0.0561	0.0000916

20	0.00008974	0.00046	0.0215	0.0002184
30	0.00036	0.0005511	0.0384	0.00111
40	0.00053	0.000561	0.04698	0.0054
50	0.006169	0.00145	0.03129	0.007845
60	0.009542	0.002693	0.0231	0.008843
70	0.0236	0.026	0.0489	0.01453
80	0.0659	0.0489	0.0513	0.0259
90	0.0916	0.0841	0.0356	0.0684
100	0.0874	0.0387	0.039978	0.0617
110	0.00578	0.012	0.03364	0.0312
120	0.00123	0.00236	0.00489	0.00951
130	0.000962	0.00165	0.02854	0.00537
140	0.000949	0.000841	0.05145	0.00459
150	0.000621	0.00048	0.0415	0.00216
160	0.0004236	0.000519	0.045	0.000863
170	0.0002145	0.0003515	0.041	0.0005312
180	0.0001126	0.000151	0.03263	0.000127

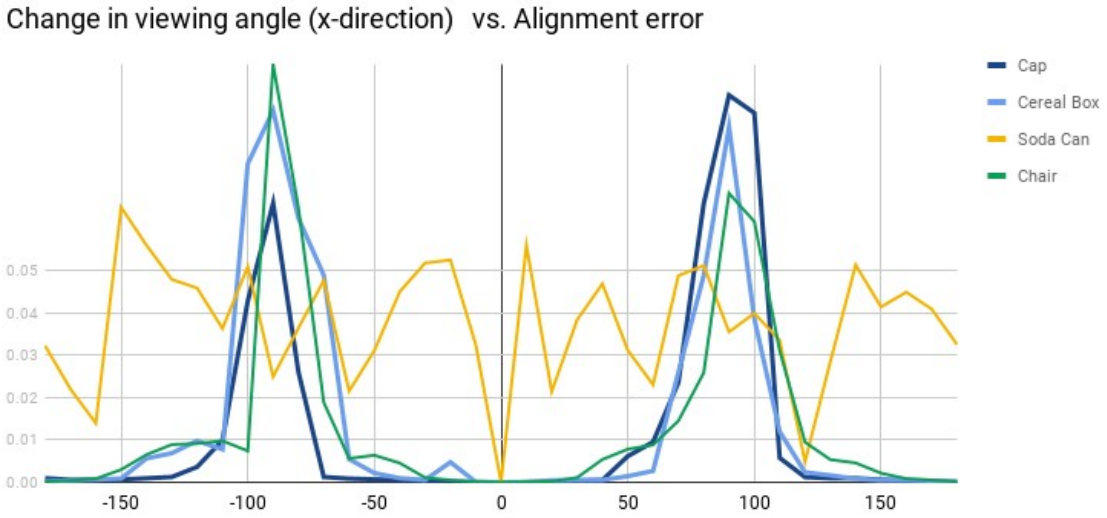


Fig. 10. A line graph displaying the above table.

The above table shows the variation in alignment error for four objects when the viewing angle is changed (in X-axis only). We observe from the table that the errors for the soda

can are very high and for the others, the errors peak at around ±70º to ±90º. This is due to the face that they are mostly symmetrical objects and around abovementioned angles, the view of the object completely changes causing errors while trying to register the point clouds. Almost all of the procedures to register the point clouds took around 2 to 4s. This is slower than required as ideally we would need to work in real time.

5 Conclusion and future work

Experiments showed good results with low alignment errors for cloud point registration of different objects. In the future, it would be better to use multiple point clouds for each viewing angle and perform another ICP registration initially and again try to register them for different viewing angles. This would increase the range of viewing angles for which the point clouds converge using ICP.

References

[1] Smith, R., Self, M. and Cheeseman, P., 1990. Estimating uncertain spatial relationships in robotics. In Autonomous robot vehicles (pp. 167-193). Springer, New York, NY.

[2] Thrun, S., 2002. Robotic mapping: A survey. Exploring artificial intelligence in the new millennium, 1, pp.1-35.

[3] Ho, K.L. and Newman, P., 2006. Loop closure detection in SLAM by combining visual and spatial appearance. Robotics and Autonomous Systems, 54(9), pp.740-749.

[4] P. M. Newman, On the solution to the simultaneous localization and map building problem", Ph.D. dissertation, Dept. Mechan. Eng., Australian Centre for Field Robotics, Univ. Sydney, Sydney, Australia, 1999.

[5] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and mapping in the Space of Appearance," The International Journal of Robotics Research, vol. 27, no. 6, pp. 647–665, 2008.

[6] K. Konolige and M. Agrawal, "FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping," IEEE Trans. Robotics, vol. 24, no. 5, pp. 1066–1077, 2008.

[7] H. Zhang, "BoRF: Loop-closure detection with scale invariant visual features," in Robotics and Automation (ICRA), 2011 IEEE International Conference on, 2011, pp. 3125–3130.

[8] B. Steder, R. Rusu, K. Konolige, and W. Burgard, "Point feature extraction on 3D range scans taking into account object boundaries," in Robotics and Automation (ICRA), 2011 IEEE International Conference on, 2011, pp. 2601–2608.

[9] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," Int. J. Comput. Vision, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[10] M. C. Bosse and R. Zlot, "Map matching and data association for largescale two-dimensional laser scan-based SLAM," International Journal of Robotics Research, vol. 27, no. 6, pp. 667–691, 2008.

[11] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 1, pp. 433 – 449, May 1999.

[12] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in Proceedings of the European Conference on Computer Vision (ECCV), Prague, Czech Republic, May 2004.

[13] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[14] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., 2016, October. Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.