Fast 3D point-cloud segmentation for interactive surfaces

Everett M. Mthunzi
mondliwethu.everett.mthunzi@uni-weimar.de
Bauhaus-Universität Weimar
Weimar, Germany

Christopher Getschmann
cget@cs.aau.dk
Aalborg University
Aalborg, Denmark

Florian Echtler
floech@cs.aau.dk
Aalborg University
Aalborg, Denmark

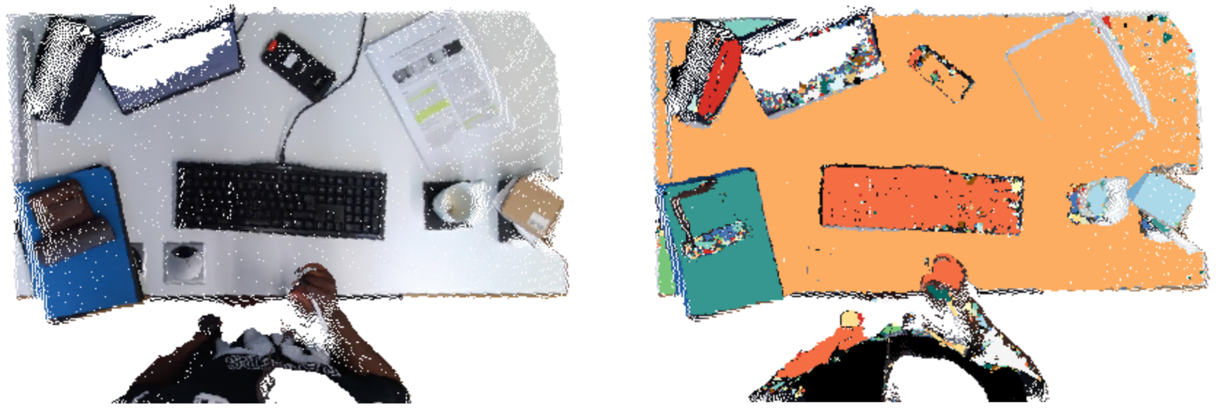


Figure 1: Segmenting candidate interaction regions on tabletop environments.

ABSTRACT

Easily accessible depth sensors have enabled using point-cloud data to augment tabletop surfaces in everyday environments. However, point-cloud operations are computationally expensive and challenging to perform in real-time, particularly when targeting embedded systems without a dedicated GPU. In this paper, we propose mitigating the high computational costs by segmenting candidate interaction regions near real-time. We contribute an open-source solution for variable depth cameras using CPU-based architectures. For validation, we employ Microsoft's Azure Kinect and report achieved performance. Our initial findings show that our approach takes under 35 *ms* to segment candidate interaction regions on a tabletop surface and reduces the data volume by up to 70%. We conclude by contrasting the performance of our solution against a model-fitting approach implemented by the SurfaceStreams toolkit. Our approach outperforms the RANSAC-based strategy within the context of our test scenario, segmenting a tabletop's interaction region up to 94% faster. Our results show promise for point-cloud-based approaches, even when targeting embedded solutions with limited resources.

CCS CONCEPTS

• **Human-centered computing** → *Interactive systems and tools.*

KEYWORDS

depth cameras, fast 3D point-cloud segmentation, interactive tabletop surfaces

ACM Reference Format:
Everett M. Mthunzi, Christopher Getschmann, and Florian Echtler. 2021. Fast 3D point-cloud segmentation for interactive surfaces. In *Interactive Surfaces and Spaces (ISS '21 Companion), November 14–17, 2021, Lodz, Poland.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3447932.3491141

1 INTRODUCTION

Commodity depth cameras have promoted using projector-camera systems to augment un-instrumented surfaces such as tabletops [5], floors [18], and walls [8]. One prominent strategy is combining image processing and depth perception-based techniques [9, 15, 17, 19]. While 2D techniques enable detecting interactions [2–4, 10, 16], solely relying on 2D methods limits the solution space of interactive surface applications. One such limitation is the inability to determine interactions on coplanar surfaces, e.g., object-to-object interactions. A promising alternative to address this shortcoming is employing 3D point-cloud representations of scenes and augmenting 2D techniques with 3D information. However, processing 3D point clouds is non-trivial and computationally expensive, particularly with limited hardware resources. Wilson and Benko suggested constraining computations to an *interaction volume*, i.e., a 3D region 10 *cm* above a target tabletop surface [17]. Similarly, Kaltenbrunner and Echtler suggested defining a *fish tank*; a rectangular 3D volume extending above the table surface. [11].

Given a 3D point-cloud representation of a tabletop's environment, segmenting the interaction volume can also be framed as a necessary preprocessing step that mitigates high computational costs in subsequent steps.

Existing model fitting techniques for segmenting the interaction volume favor primitive shapes, such as planes, cylinders, and spheres. However, the shape of the target surface is unknown ahead of time. Therefore, there is no guarantee that a model-fitting approach would generalize well for fitting arbitrary volumes. Despite a wealth of well-understood algorithms, prevailing segmentation techniques do not target interactive surface implementations. To our knowledge, no study has explored an open solution for segmenting candidate interaction regions as a preprocessing solution, which we consider essential to allow for near real-time applications using limited hardware resources (e.g. embedded or small-form-factor solutions without dedicated GPU).

In this paper, we propose mitigating high computational costs on CPU architectures by segmenting candidate interaction regions near real-time. We contribute an open solution for preprocessing 3D point clouds using variable depth cameras. For validation, we employ Microsoft's Azure Kinect and report achieved performance. Additionally, we provide an open-source project to promote exploring the proposed approach and enable community-based revision to enable a broader range of depth cameras.

2 OUR APPROACH

We propose a pipeline with four filters: an outlier removal filter, a coarse segmentation filter, a final segmentation filter, and a clustering filter. Assumptions that need to be considered for these pipeline operations to be valid are as follows: (a) the depth sensor is mounted directly above a target tabletop surface; (b) the sensing direction is approximately perpendicular to the target tabletop; (c) the line-of-sight to the target tabletop is unoccluded; and (d) the sensing range, i.e., the distance between the depth sensor and the target tabletop surface, is within the operational limits specified by the vendor.

2.1 Outlier removal

We denote an unorganized 3D point cloud using $\boldsymbol{p} = \{p_1, p_2, \ldots, p_m\}$, $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$ and the corresponding centroid using $\overline{p} = (\overline{x}, \overline{y}, \overline{z}) \in \mathbb{R}^3$. We denote the set of Euclidean distance measures from the centroid $\overline{p}$ to all points $\{p_1, p_2, \ldots, p_m\}$ using $\boldsymbol{d} = \{d_1, d_2, \ldots, d_m\}$, $d_i = d_i(p_i, \overline{p}) \in \mathbb{R}^1$. Linear discriminant analysis [20] and statistical analysis are used for removing outliers. Given a raw 3D point cloud, within-point variance is evaluated, and *"distance-outliers"* [12] are identified and filtered out using the interquartile range test for normality of distribution: $\{d_i : d_i < \overline{d} + 1.5\sigma\}$ with mean $\overline{d}$ and the standard deviation σ.

2.2 Coarse segmentation

Given a set of denoised points $\boldsymbol{p'}$, $\overline{p'}$ is used to denote the centroid. The set of points $\boldsymbol{p'}$ are translated into distance vectors $\boldsymbol{r}$ such that $\boldsymbol{r} = \{r_1, r_2, \ldots, r_m\}$, $r_i \mapsto p'_i - \overline{p'} \in \mathbb{R}^3$. Points that exist on a tabletop's surface are denoted using $\boldsymbol{t}$ such that $\boldsymbol{t} = \{t_1, t_2, \ldots, t_m\}$, $t_i = (x_i, y_i, z_i) \in \mathbb{R}^3$. The set of points $\boldsymbol{q} = \{q_1, q_2, \ldots, q_m\}$, $q_i = (x_i, y_i, z_i) \in \mathbb{R}^3$ is used to denote a coarse segment such that $\boldsymbol{t} \subseteq \boldsymbol{q}$ and $\boldsymbol{q} \subseteq \boldsymbol{p'}$. Given assumptions 1 and 2, we exploit the depth sensor's view perspective: knowledge that the centroid $\overline{p'}$ exists in the subset of points $\boldsymbol{q}$. Coarse segmentation, therefore, considers determining the face normal of the tabletop and growing a region of interest that corresponds to the interaction volume. In order to determine the face normal of the tabletop, we frame the task as a least squares problem and employ the Eckart-Young-Mirsky matrix approximation theorem [7], computing the singular value decomposition over $\boldsymbol{p'}$ as $A = U\Sigma V^T$ where $U = (u_1, \ldots, u_m), V = (v_1, \ldots, v_3), \Sigma = diag(\sigma_1, \ldots, \sigma_3)$, m denoting the number 3D points. Given the standard form of the equation of a plane and the normal vector of each point ${p'}_i$ (i.e., $\vec{N_i} = \langle A, B, C \rangle$), the left singular vectors (U) corresponds to the x, y, z column vectors. The minimized error in determining norm (i.e., the Frobenius norm n) of the tabletop's surface is given by σ_3, which is associated with v_3. Finally, the coarse segment is determined using constraint $\boldsymbol{E} = argmin \sum_{i=1}^{m} n \cdot r_i$.

2.3 Final segmentation

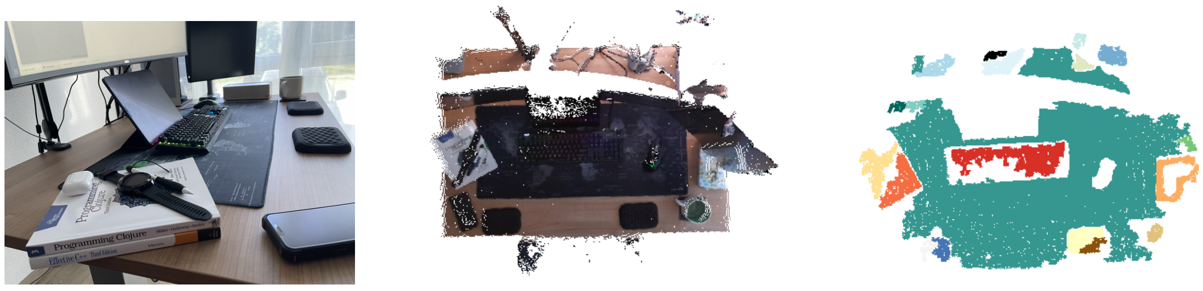
Coarse segmentation yields the desired segment height above the tabletop surface. However, the segment is overfitted laterally. Final segmentation considers confining the area of the coarse segment to that of the tabletop's area to correctly define the desired interaction volume. Our approach to detecting the tabletop's boundary is two-fold: First, we employ Silhouette edge detection. Then we assess the discontinuity in depth measures. The normal-vector components utilized for Silhouette edge detection (i.e., A, B, C), are those computed during coarse segmentation. We exploit the knowledge that face normals of the tabletop's edge are perpendicular to the sensor's viewing direction. Therefore, face normals that correspond to the tabletop boundary are characterized by a vanishing depth component (i.e., a vanishing z normal value). Given Assumptions 1 and 2, points in proximity to the centroid are a known subset of $\boldsymbol{t}$. We re-frame the analysis of discontinuity in depth measures as a rate-of-change problem. As such, we heap and sort the set of points $\boldsymbol{q}$ in descending order of depth measures. We then compute the maximum second derivative, which coincides with the tabletop's edge.

2.4 Clustering candidate interaction regions

Given the set of points corresponding to interaction volume, the segment is clustered into candidate interaction regions using DBSCAN [6]. We optimize the original algorithm [6], using nanoflann [1], achieving a worst-case computational complexity of $O(n^2)$. For a single camera, a more straightforward approach purely based on pixel neighborhood would suffice. However, we target a more general approach to account for cases with two or more depth cameras for larger surfaces regardless of camera-to-camera calibration. DBSCAN requires the approximation of two hyperparameters: k and ε. We set $k = 4$ as suggested by Ester et al.[6], and estimate the ε neighborhood size in an iterative pre-processing step, which only is required once as part of the initial setup for the depth sensor. However, if the sensor is changed or displaced significantly, ε hyperparameter recalculation is required for optimal clustering.

3 INITIAL PERFORMANCE REPORT

A tabletop with walls and protruding structures in its vicinity was employed for validation. The Kinect was mounted directly above the tabletop surface, 1.42 *m* above the tabletop, with a view direction approximately perpendicular to the target surface. The Kinect itself was configured with a color resolution of 720p and a 2x2



(a) Illustrative scene. (b) Segmented interaction volume. (c) Candidate interaction regions.

Figure 2: Segmenting candidate interaction regions using 3DINTACT.

binned depth resolution. A desktop computer with 16G RAM and an AMD Ryzen 7 3700X 4GHz CPU was also employed. On average, segmenting the interaction volume took under 16 *ms*, with an average data reduction of 76.67%. Clustering candidate interaction regions within the segmented interaction volume took under ~19.5 *ms*. Accordingly, graphed performance measures indicate that the runtime for both processes increases steadily with an increase in point cloud size (Fig. 3—a). As expected, contrasting runtimes of clustering entire unsegmented point clouds representations against clustering segmented interaction volumes (Fig. 3—b) indicates a 44.28% reduction in runtime penalty in favor of the segmented point cloud. It is also necessary to note that resulting spatial density clusters from the unsegmented point cloud would still require additional computational strategies to determine the clusters that correspond to the actual interaction volume.

4 APPLICATION AND BENEFITS

To streamline exploring our approach, we also put forward 3DINTACT: an open-source project for segmenting interaction regions on tabletop surfaces near real-time [13]. The toolkit abstracts the proposed pipeline operations into small modular libraries that developers can modify, adapt, and extend flexibly. The open-source project elaborates using ready-made solutions for applications, including finding vacant surface space for interactive projection, real-time rendering, and object detection. To show how our approach can be leveraged to benefit existing applications, we form a concrete case using the SurfaceStreams toolkit [4].

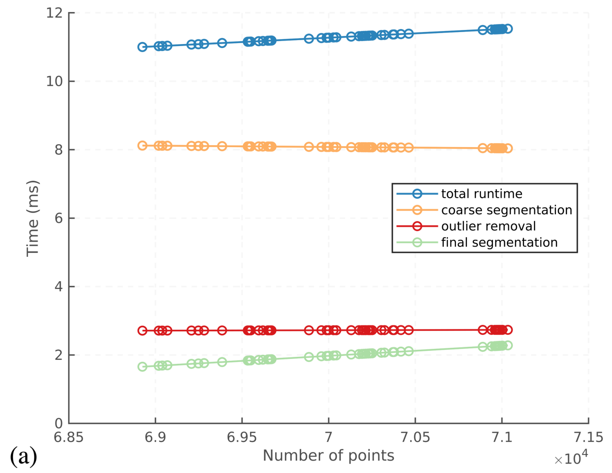
SurfaceStreams is a toolkit that enables multiple display-camera systems to record and share digital interactive media. A critical step that the toolkit considers is segmenting the interaction volume. For this task, it employs a RANSAC-based model-fitting strategy. The resulting computational cost and high runtime (~295 *ms*) obligate SurfaceStreams to run the algorithm only once at startup. This approach leaves the problem of incidental target surface displacement unaddressed. In place of the model-fitting strategy, the toolkit could employ 3DINTACT to continuously segment the interaction volume correctly. For the given tabletop environment (Fig. 4), benchmarking our approach against the RANSAC-based strategy yielded results that indicate our approach to be faster by up to 94.89%; a promising initial result. The significance of segmenting the interaction volume near real-time is addressing the previously mentioned problem while also mitigating subsequent processing costs currently necessary to determine the interaction area [4]. 3DINTACT also simplifies real-time rendering processed 3D point clouds which could be useful for applications such as telepresence and spatial augmented reality.

5 DISCUSSION

While the contribution we put forward targets a general solution for variable depth cameras, initial validation has been limited to using the Kinect over a unique tabletop environment. A well-rounded generalization of runtime and data reduction requires extending validation to variable depth sensors and tabletop environments. Although the computational complexity for each pipeline operation is outlined, the measured runtimes are subject to the capabilities of the specific computer in use. Future validation considers employing variable depth sensors, tabletop environments, and computers to realize a well-rounded generalization on performance. Given that our approach trades off robustness offered by exhaustive redundant operations [14] for minimal runtime penalty, future work must aim to optimize the robustness of the proposed approach without compromising generality or incurring additional runtime penalty. The work presented in this paper is a building block for interpreting how persons and objects interact within a tabletop's environment in real-time. Our next research step aims to employ the segmented interaction regions for spatial awareness and enabling interaction between mobile devices and tabletop surfaces using dynamic surface projection.

6 CONCLUSION

Given an unorganized 3D point-cloud representation of a tabletop's environment, we have presented one possible approach to reducing high computational costs for downstream operations on CPU architectures. The preprocessing strategy we propose considers segmenting the interaction volume of a tabletop surface and clustering the segment into candidate interaction regions. Our initial experiments have indicated that the proposed approach requires under ~16 *ms* to segment the interaction volume and ~19.5 *ms* to cluster candidate interaction regions. These initial runtimes can serve to benchmark comparable approaches in the future. In addition to presenting an initial performance report, we have shown tacit benefits for an existing toolkit. Equally exciting is the prospect



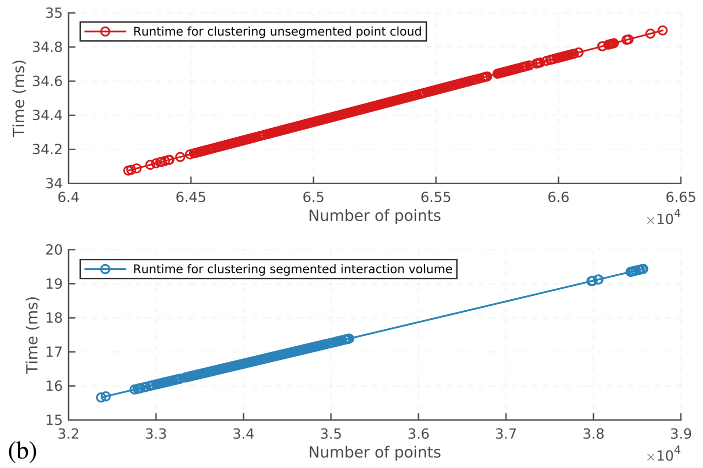


Figure 3: Initial performance report: Runtime performance over 1000 run instances. In (a), outlier removal took under 6 ms, coarse segmentation took under 8 ms, and final segmentation took under 4 ms. In (b), clustering interaction regions without segmenting the interaction volume first took between 34 ms and 35 ms. Clustering interaction regions after segmenting the interaction volume took between 15.7 ms and 19.5 ms.

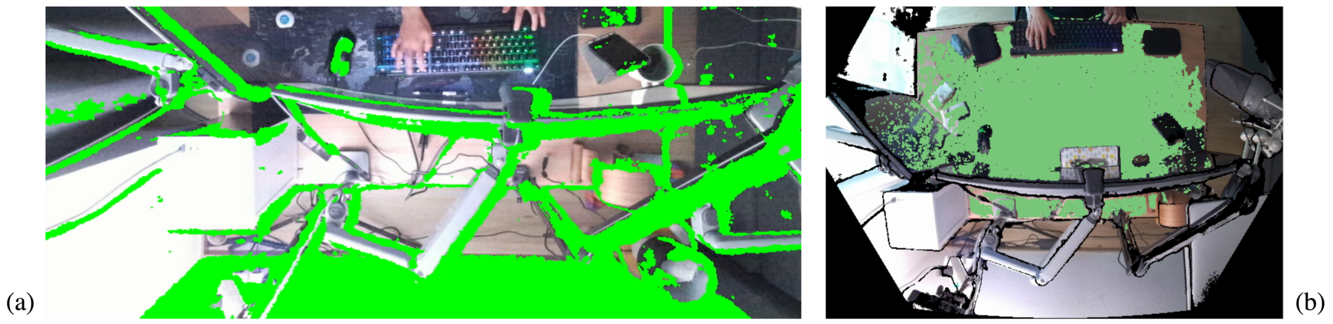


Figure 4: Addressing challenges using our approach. In addition to RANSAC's high runtime penalty, fitting the dominant plane can yield undesired results in cases where the dominant surface does not correspond to the target tabletop surface as illustrated in (a), where the detected plane is the wall next to the tabletop. In (b), our approach correctly segments the desired interactive region on the tabletop.

of leveraging 3D point-cloud representations for interactive surface applications, which we look to explore in future work.

ACKNOWLEDGMENTS

This paper was supported by the German Federal Ministry for Education and Research (BMBF) through grant number 16SV8288 within the VIGITIA project.

REFERENCES

[1] Pranjal Kumar Blanco, Jose Luis and Rai. 2014. nanoflann: a C++ header-only fork of {FLANN}, a library for Nearest Neighbor with KD-trees. https://github.com/jlblancoc/nanoflann

[2] Patrick Chiu, Chelhwon Kim, and Hideto Oda. 2018. Recognizing gestures on projected button widgets with an RGB-D camera using a CNN. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*. ACM, New York, NY, USA, 369–374. https://doi.org/10.1145/3279778.3279907

[3] Michael den Bergh and Luc Van Gool. 2011. Combining RGB and ToF cameras for real-time 3D hand gesture interaction. In *IEEE Workshop on Applications of Computer Vision (WACV)*. IEEE, Kona, HI, USA, 66–72. https://doi.org/10.1109/WACV.2011.5711485

[4] Florian Echtler. 2018. Surfacestreams: A content-agnostic streaming toolkit for interactive surfaces. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 10–12. https://doi.org/10.1145/3266037.3266085

[5] Florian Echtler and Raphael Wimmer. 2014. The interactive dining table, or pass the weather widget, please. In *Proceedings of the 9th ACM International Conference on Interactive Tabletops and Surfaces*. ACM, New York, NY, USA, 419–422. https://doi.org/10.1145/2669485.2669525

[6] Martin Ester, Hans-Peter Kriegel, J"org Sander, Xiaowei Xu, and Others. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*. AAAI Press, Portland, Oregon, USA, 226–231.

[7] Gene H Golub, Alan Hoffman, and Gilbert W Stewart. 1987. A generalization of the Eckart-Young-Mirsky matrix approximation theorem. *Linear Algebra and its applications* 88 (1987), 317–327.

[8] Yizheng Gu, Chun Yu, Zhipeng Li, Weiqi Li, Shuchang Xu, Xiaoying Wei, and Yuanchun Shi. 2019. Accurate and low-latency sensing of touch contact on any surface with finger-worn IMU sensor. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. ACM, New Orleans, LA, USA, 1059–1070. https://doi.org/10.1145/3332165.3347947

[9] Jan Gugenheimer, Enrico Rukzio, Pascal Knierim, and Julian Seifert. 2014. UbiBeam: An interactive projector-camera system for domestic deployment. In *Proceedings of the 9th ACM International Conference on Interactive Tabletops and*

Surfaces. ACM, New York, NY, USA, 305–310. https://doi.org/10.1145/2669485.2669537

[10] Nikhita Joshi and Daniel Vogel. 2019. An evaluation of touch input at the edge of a table. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 2865–2874. https://doi.org/10.1145/3290605.3300476

[11] Martin Kaltenbrunner and Florian Echtler. 2018. The TUIO 2.0 protocol: An abstraction framework for tangible interactive surfaces. *Proceedings of the ACM on Human-Computer Interaction* 2, EICS (jun 2018), 35. https://doi.org/10.1145/3229090

[12] Edwin M Knorr, Raymond T Ng, and Vladimir Tucakov. 2000. Distance-Based Outliers: Algorithms and Applications. *The VLDB Journal* 8, 3–4 (feb 2000), 237–253. https://doi.org/10.1007/s007780050006

[13] Everett M Mthunzi. 2021. 3DINTACT: an open-source CXX_11 project for segmenting interaction regions on tabletop surfaces near real-time. https://github.com/edisonslightbulbs/3DINTACT

[14] A Nguyen and B Le. 2013. 3D point cloud segmentation: A survey. In *6th IEEE Conference on Robotics, Automation and Mechatronics*. IEEE, Manila, Philippines, 225–230. https://doi.org/10.1109/RAM.2013.6758588

[15] Yoon Jung Park, Hyocheol Ro, Jung-Hyun Byun, and Tack-Don Han. 2019. Adaptive Projection Augmented Reality with Object Recognition Based on Deep Learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion*. ACM, New York, NY, USA, 51–52. https://doi.org/10.1145/3308557.3308678

[16] Alexander J B Trevor, Suat Gedikli, Radu B Rusu, and Henrik I Christensen. 2013. Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration* 1 (2013), 6.

[17] Andrew D. Wilson and Hrvoje Benko. 2010. Combining multiple depth cameras and projectors for interactions on, above, and between surfaces. In *23rd ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 273–282. https://doi.org/10.1145/1866029.1866073

[18] Christian Winkler, Julian Seifert, David Dobbelstein, and Enrico Rukzio. 2014. Pervasive information through constant personal projection: The ambient mobile pervasive display (AMP-D). In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 4117–4126. https://doi.org/10.1145/2556288.2557365

[19] Qin Wu, Jiayuan Wang, Sirui Wang, Tong Su, and Chenmei Yu. 2019. MagicPAPER: Tabletop interactive projection device based on tangible interaction. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*. ACM, Los Angeles, CA, USA, 2. https://doi.org/10.1145/3306214.3338575

[20] Jieping Ye. 2007. Least Squares Linear Discriminant Analysis. In *Proceedings of the 24th International Conference on Machine Learning*. ACM, New York, NY, USA, 1087–1093. https://doi.org/10.1145/1273496.1273633