A First Look into Privacy Leakage in 3D Mixed Reality Data

Jaybie A. de Guzman[1,2(✉)], Kanchana Thilakarathna[2,3], and Aruna Seneviratne[1,2]

[1] University of New South Wales, Sydney, NSW 2052, Australia
j.deguzman@student.unsw.edu.au, a.seneviratne@unsw.edu.au
[2] Data61 | CSIRO, Sydney, NSW 2015, Australia
[3] University of Sydney, Sydney, NSW 2006, Australia
kanchana.thilakarathna@sydney.edu.au

Abstract. We have seen a rise in *mixed* (MR) and *augmented reality* (AR) applications and devices in recent years. Subsequently, we have become familiar with the sensing power of these applications and devices, and we are only starting to realize the nascent risks that these technology puts over our privacy and security. Current privacy protection measures are primarily aimed towards known and well-utilised data types (i.e. location, on-line activity, biometric, and so on) while a few works have focused on looking into the security and privacy risks of and providing protection on MR data, particularly on 3D MR data. In this work, we primarily reveal the privacy leakage from released 3D MR data and how the leakage persist even after implementing spatial generalizations and abstractions. Firstly, we formalize the *spatial privacy* problem in 3D mixed reality data as well as the adversary model. Then, we demonstrate through an inference model how adversaries can identify 3D spaces and, potentially, infer more spatial information. Moreover, we also demonstrate how *compact* 3D MR Data can be in terms of memory usage which allows adversaries to create lightweight 3D inference models of user spaces.

Keywords: Mixed and augmented reality · 3D data · Point cloud data · Security and privacy

1 Introduction

Pokémon Go's release in 2016 arguably marked the beginning of augmented reality (AR) and mixed reality (MR) to be part of the mainstream mobile market. Soon after, Apple launched the ARKit in 2017 and, halfway through 2018, Google followed with the ARCore.[1] Microsoft, on the other hand, focused on the head-mounted displays (or HMDs) with the HoloLens and other OEM headsets running their Windows Mixed Reality platform.[2] These developments

[1] See https://developer.apple.com/documentation/arkit for Apple's ARKit See https://developers.google.com/ar/ for Google's ARCore.
[2] https://developer.microsoft.com/en-us/windows/mixed-reality.

K. Sako et al. (Eds.): ESORICS 2019, LNCS 11735, pp. 149–169, 2019.
https://doi.org/10.1007/978-3-030-29959-0_8

undoubtedly signifies the very near future with AR and MR being ubiquitous. (Henceforth, following Milgram's definition [18], we will be collectively calling both augmented and mixed reality as mixed reality or MR.)

Most mobile MR development platforms (i.e. ARCore, and ARKit) utilise a form of *visual odometry* combined with motion or inertial information to map the device's position relative to the real-world, while dedicated HMDs (i.e. HoloLens), leverage multiple cameras with depth sensors to understand the environment and create a virtual 3D map. Once a good mapping has been created, the virtual space (or a coordinate system) is shared with applications to allow synthetic or augmented content to interact with the physical world such as *anchoring* a virtual object on your desk.

However, this environment understanding capability required by MR poses unforeseen privacy risks for users. Once these captured 3D maps have been revealed to untrusted parties, potentially sensitive spatial information about the users' space are disclosed. Adversaries can vary from a background service that delivers unsolicited ads based on the objects detected from the user's surroundings to burglars who are able to map the user's house, and, perhaps, the locations and dimensions of specific objects in their house based on the released 3D data. Furthermore, turning off GPS tracking for location privacy may no longer be sufficient once the user starts using MR applications that can expose their locations through the 3D and visual data that are exposed.[3]

The recent EU-GDPR ruling aims to address these issues from a policy approach. It aims to empower the users and protect their data privacy. This highlights the importance of designing and developing *privacy-enhancing technologies* (PETs). Currently, there are numerous PETs designed for structured data such as k-anonymity [23], and *differential privacy* [4], as well as techniques for data aggregation during information collection [9]. However, current techniques protecting media are mostly for conventional data types, and are primarily focusing on *facial de-identification* for *identity privacy* [7,19,27] as well as protection against visual capture recording mechanisms [1,28]. (See [8] for a survey of MR-related security and privacy protection approaches.)

In this work, we focus on the *nascent risks* from captured and collected 3D data used for MR processing. To demonstrate the privacy leakage, we utilize actual 3D point cloud data, captured by a Microsoft HoloLens, to construct an adversarial inferrer that can identify spaces from the revealed 3D data. The inference performance is evaluated over both raw data and different 3D data generalizations. And we show how such generalizations are ineffective even with a simple *matching-based* inference attack. To the best of our knowledge, this

[3] For example, Google has unveiled their *Visual Positioning Service* (or VPS) using 3D data to locate users in space – an offshoot of Project Tango – during their 2018 I/O keynote event.

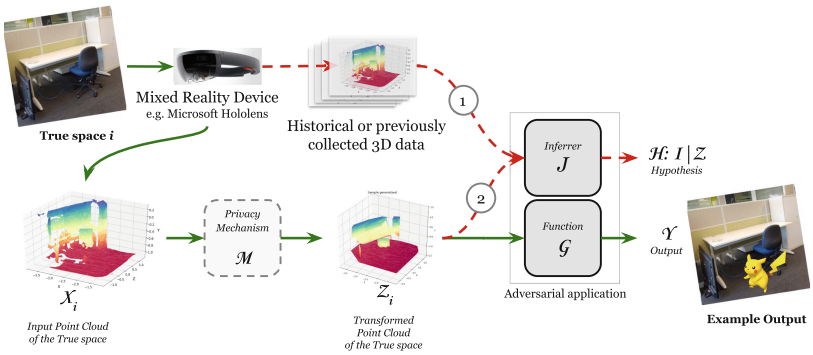


Fig. 1. Information flow (following the green solid arrows) for a desired MR functionality G with an intermediate privacy-preserving mechanism M; while an MR adversarial process (represented by the red broken arrows) may be done off line: (1) adversarial inference *modeling* or *learning* from, say, historical 3D data, and (2) adversarial inference or *matching* over released 3D data (Color figure online)

is the first work that aims to expose these risks. Consequently, we make the following specific contributions in this work:

1. We formalize the *3D spatial privacy problem* and define the privacy and utility metrics specific to 3D MR data.
2. We present a *3D adversarial inference model* to reveal the spatial privacy leakage and their effectiveness.
3. Using 3D point cloud data collected from Microsoft HoloLens, which is also the same 3D data representation format for Google's ARCore and Apple's ARKit, we demonstrate that 3D spatial inference attacks are possible on these MR platforms.
4. Lastly, results show the *insufficient protection* provided by spatial generalizations even by only using simple descriptor-matching for adversarial inference.

The rest of the paper is organized follows. Section 2 elaborates on the 3D MR data, i.e. point cloud data, and presents the theoretical framework of our *3D privacy problem*. In Sect. 4, we describe the evaluation methodology used to determine the privacy leakage in 3D data with and without spatial generalizations. The results are presented in Sect. 5 and the related work in Sect. 6. We conclude the paper in Sect. 7.

2 3D Privacy Problem

2.1 Why 3D?

With images and video, what the machine sees is practically what the user sees and a great deal of privacy work have been done on these data forms. Contrariwise, in MR, the experience is exported as visual data (e.g. objects augmented

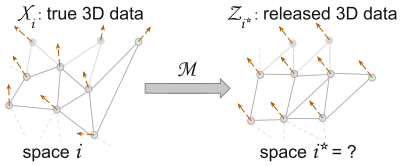


Fig. 2. A privacy preserving mechanism M transforms the raw point clouds X to a *potentially* privacy-preserving version Z to *hide* location identity ($i^* = ?$).

on the user's view) while its 3D nature, especially of the underlying data, is not exposed to the users: what the machine sees is different (arguably, even more) than what the user sees. That is, the digital representation of the physical world, the 3D point cloud data, is not exposed to the user. This inherent perceptual difference creates a disconnect and, perhaps, affects (or the lack thereof) how users perceive the sensitivity of 3D information. Furthermore, current MR platforms (i.e. Windows MR, ARCore and ARKit) directly operates on these 3D spatial maps or point clouds and, so far, no privacy preservation is applied before providing these data to third party applications.

3D Point Cloud Data. The 3D points comprising the 3D point cloud can be described by their $\{x, y, z\}$-position in space with an accompanying normal vector $\{n_x, n_y, n_z\}$. Figure 2 shows the point clouds as a mesh of 3D points with associated orientations represented by normal vectors. These are the minimum information necessary to capture the *geometric* properties of 3D spaces. Where normal vectors are not readily available, it is estimated from the points themselves. Sometimes, point clouds may also be accompanied by *photometric* information such as RGB or light intensity extracted from associated images or videos. For this work, we will only be focusing on the use of geometric information and leverage them for 3D description for emulating adversarial inference.

2.2 Defining the 3D Privacy Problem

We define the elements shown in Fig. 1: the space represented by a point cloud X identified by a label i; the privacy preserving mechanism M that transforms X to a privacy-preserved point cloud Z, i.e. $M : X \mapsto Z$ as shown in Fig. 2; an intended functionality G that produces an intended output Y, and from which we derive the utility function U; and an adversarial inferrer J that produces a hypothesis H to reveal the identity of a given unknown space. The adversarial processes may be done off line and not necessarily during MR function runtime. (See Appendix A for detailed definitions on X, M, Z, and G.)

Defining the Function Utility. For a given functionality G, an effective mechanism M aims to make the resulting outputs y_i from the raw point cloud x_i and its privacy-preserving version $z_{(i)}$ similar, i.e. $y_{x_i} \simeq y_{z_{(i)}}$, or their difference is small, $D_{Z;X} = |y_{x_i} - y_{z_{(i)}}| \to 0$. Or in terms of a utility function U which we intend to maximize (i.e. as close to 1 as possible if we assume that $D_{Z;X} \leq 1$),

$$U(Z;X) = 1 - D_{Z;X},\ \textit{where}\ Z = M(X). \tag{1}$$

The most common functionality in MR is the *anchoring* of virtual 3D objects on to real-world surfaces (e.g. the floor, walls, or tables) which requires near-truth 3D point cloud representations to provide consistent anchored augmentations.

Defining the Adversarial Inferrer. An inferrer J produces a hypothesis $h : i^* = i$ about the true location i of a given set of point clouds, x_{i^*} or $z_{(i^*)}$, for any query space i^* (i.e. $J : x_{i^*}\ or\ z_{(i^*)}$ for any $i^* \rightarrow h : i^* = i$) where the following inequality holds

$$P(h : i^* = i | x_{i^*}\ or\ z_{(i^*)}) > P(h : i^* = i^o,\ for\ any\ i^o \neq i | x_{i^*}\ or\ z_{(i^*)}). \tag{2}$$

The Privacy-Utility Problem. Consequently, we can now pose the following *privacy* function Π in terms of the error rate of the inferrer,

$$\Pi(Z;X) = \underset{iterations}{\text{mean}} \frac{|h : i_z \neq i_x|}{|\forall i|}, \tag{3}$$

which is simply the *mean misclassification rate* of an inferrer J about the query space i_z whose true identity is i_x. A few works in the literature uses the same error-based metric for privacy [22,26]. A desired M produces Z that maximizes both the privacy Π and the utility function U.

Privacy and Utility Metrics. Now, we define the specific privacy and utility metrics for this work. For privacy, we use the same notion of a high error rate as high privacy; thus, the same metric defined by Eq. 3 holds. For utility, we use the same similarity definition defined by Eq. 1 but define the specific components of the similarity function as,

$$U(Z;X) = \text{mean}(\alpha \cdot (1 - ||x - z||) + \beta \cdot (\boldsymbol{n_x} \cdot \boldsymbol{n_z})) \tag{4}$$

where the first component is the 3D point similarity of the true/raw point x from the transformed point z, the second component are their normal vector similarity, and α and β are contribution weights where $\alpha, \beta \in [0, 1]$ and $\alpha + \beta = 1$. We set $\alpha, \beta = 0.5$. We also insert a subjective acceptability metric $\gamma \in [0, 1]$ like so,

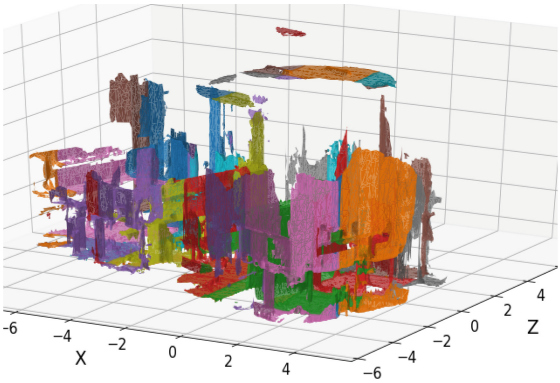
$$U(Z;X) = \text{mean}\left[\alpha \cdot \left(1 - \frac{\lceil ||x - z|| \rceil_\gamma}{\gamma}\right) + \beta \cdot \left(\lfloor \boldsymbol{n_x} \cdot \boldsymbol{n_z} \rfloor_{1-\gamma} - \frac{1-\gamma}{\gamma}\right)\right]. \tag{5}$$

γ allows us to specify the level of error or deviation of the released (i.e. generalized) spaces from the true space – any deviation beyond the set γ results to a zero utility. The range of $U(X, Z) \in [0, 1]$.

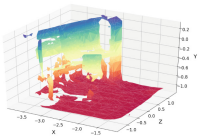
2.3 Adversary Model

Adversaries may desire to, at the very least, infer the location of the users using released 3D data. They may further infer user poses, movement in space, or,

even, detect changes in user environment. Furthermore, in contrast to video and image capture, 3D data, when generalized, can provide a much more lightweight and near-truth representation of user spaces which we will see later (Sect. 5.5). For our evaluation, we will focus on the minimum attack where the adversary infers the spatial location of the user given historical 3D raw data of user spaces. We also assume that the adversary is not aware of the generalizations that an MR platform can perform over 3D data before it is released.



(a) Complete captured raw point cloud: different regions are differently colored



(b) Sample Region

(c) Photo of sample region

Fig. 3. Render of the gathered point cloud (1 unit is roughly 1 m in the real-world)

Using the definitions in Sect. 2.2, we can formalize the adversary models as previously shown in Fig. 1. We assume that the adversary has *prior knowledge* about the spaces which they can use as reference for building their inference model J. Prior knowledge can be made available through (1) *historical* or publicly available 3D spatial data of the user spaces, (2) *previously provided* data by the user themselves or other users, or (3) from a *colluding application* or service that has access to raw or higher resolution 3D data.

Adversarial Inference. Our adversarial inference is a two-step process as labelled in Fig. 1: (1) the creation of a reference description model or dictionary using the 3D descriptor algorithms (Sect. 3.2) over the previously known spaces as reference, (2) and the inference of unknown spaces by *matching* their 3D descriptors to that of the reference descriptors from step 1. The construction of the inference model is detailed in the next section.

3 3D Description and Inference

3.1 3D MR Data

We gathered real 3D point cloud data using the Microsoft HoloLens in an office environment to demonstrate the leakage from actual human-scale spaces in which

an MR device is usually used.[4] The render of the gathered 3D space is shown in Fig. 3a. We sliced our gathered point cloud into roughly 2.5×2.5 squares about the xz-plane (i.e. the floor plane) to create a synthetic set of multiple spaces.[5] The resulting number of spaces after slicing is 38. Also, we treat the spaces to be non-contiguous – specifically, spaces that are truly adjacent do not inform adversarial inference.

3.2 Describing the 3D Space

The 3D point clouds can then be used by the adversary to train an inference model. Features that describe and discriminate among 3D spaces are usually used for inference modelling. There are considerable features in 3D point clouds for it to be directly used as a 3D descriptor, albeit a crude one, and it won't be translation- and rotation-invariant by itself. Hence, invariant descriptors are necessary for adversarial inference models to be resilient.

To provide invariance, we utilize existing 3D description algorithms.[6] The curvature-reliant self-similarity (SS) descriptors [10] are very sensitive to point cloud variations, due to the curvature estimation. To counter this, we explored the use of non-curvature reliant spin image (SI) descriptors [13,14]. SI descriptors only use the normal vector unlike the SS approach which uses *local curvature maxima* for key point selection. Thus, a vanilla SI computes the descriptor for every point in the point cloud which produces a dense descriptor space. For our SI implementation, we extract key points and descriptors from the subsampled space by factor of 3 (Fig. 5 shows that significant errors only appear at resolutions < 3) to create a lighter weight descriptor set. Also, the spinning effect reduces the impact of variations within that spin which makes SI descriptors more robust compared to SS descriptors. Furthermore, as we will describe in Sect. 4.1, plane generalization removes curvatures which makes its use as a geometric description information impractical. Validation of the inference performance of these descriptors are detailed in Sect. 3.3.

3.3 Inferring the 3D Space

For the adversarial inference model, we built two types of *inferrers*: (1) a *baseline 3D Bayesian* inference model using directly the 3D point cloud data, and (2) a *matching-based* inference model using the rotation-invariant descriptors.

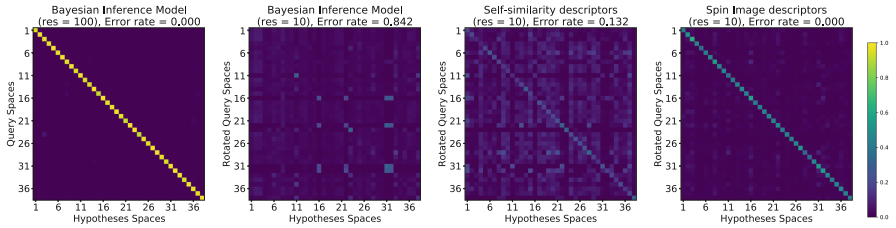
[4] There are numerous 3D point cloud datasets such as those listed in http://cvgl.stanford.edu/resources.html but most of these available 3D data sets are models of objects or of city-scale models.

[5] Note: the resulting surface are of the slice varies due to the walls, and objects within a slice. It can also be less than 2.5×2.5 due to gaps on the space.

[6] For a concise discussion and bench marking of different 3D description algorithms, we direct the reader to [3].

Inference Using the Rotation-Invariant Descriptors. It is challenging to create a straightforward 3D inference model as we would have in a 3D Bayesian model.[7] As a work around, we utilize the standard matching-based approach that is used over high-dimensional descriptors. This approach is rather *deterministic* as opposed to the *probabilistic* Bayesian inference model.

This deterministic approach used for the rotation-invariant descriptors utilizes a *matching-based voting mechanism* with a reference set of descriptors to determine a match; then, *nearest neighbor distance ratio* (or NNDR) is used to qualify a match. Thus, instead of the probabilistic maximization described in Eq. 2, we utilize this NNDR-based approach for deterministic inference. See Appendix B for more details on this descriptor matching process.



(a) Bayesian (res = 100) (b) Bayesian, rotated queries (c) Self Similarity (d) Spin Images

Fig. 4. Inference performance heatmaps of the different 3D description approaches

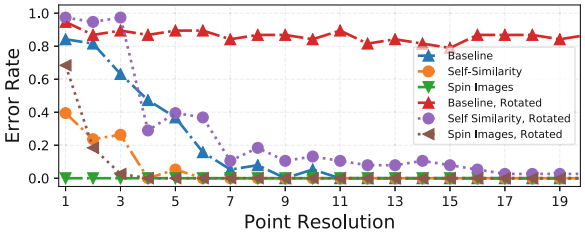


Fig. 5. Performance of the different 3D description/inference for different resolutions

Validating the Inference Models. We conducted a preliminary validation to check the effectiveness of the chosen description and inference approaches. To validate our inference models, we feed them the same data as queries.

[7] For example, our spin image description implementation have 200 (i.e. 10×20) dimensions; it'll require 10^{200} bins for every key point to be described if we are to approximate that each dimension will have 10 bins.

Using the Bayesian Inference Model. When complete versions of the set of points x_i for each space i is given as a query data, the baseline Bayesian inference model performs very well as shown by the solid yellow diagonal in the heatmap/confusion matrix in Fig. 4a. Figure 5 shows the results of varying the resolution from $1 \leq res < 20$. For un-rotated query spaces, the Bayesian inference model only starts to have errors at resolutions ≤ 10, while its error rate for rotated query spaces is ≥ 0.8 for all resolutions. As we have indicated earlier in Sect. 3.2, the baseline inference model is not rotation-invariant and it is clearly observed here. For example, Fig. 4b shows a heat-map for a lower resolution of res $= 10$ with rotated query spaces; we can not see a distinguishable diagonal to signify good inference performance.

Using the Rotation-Invariant Descriptors. With un-rotated query spaces, the SS descriptors' maximum error rate is only 0.4 as shown in Fig. 5, while the SI descriptors stays 0 even at the smallest resolution of 1. With rotated query spaces, errors increased for both but significant errors (i.e. ≥ 0.1) only appear at res ≤ 3 for the SI descriptors, while errors for the SS descriptors already appear even at higher resolutions of res ≤ 14.

The excellent performance of the spin image descriptors can be better visualized with the heatmaps shown in Fig. 4 with res $= 10$. As can be observed, the spin images discriminates well as demonstrated by the clearer diagonal in Fig. 4d as compared to Fig. 4c. Thus, in the succeeding experiments described in the next section (with results in Sect. 5), we will only be using spin image descriptors.

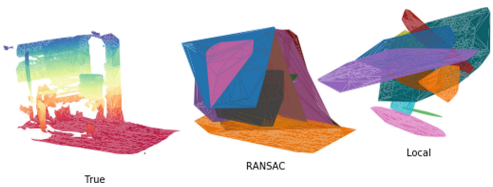


Fig. 6. Surface generalization, i.e. plane fitting, example: (left) sample raw space, (center) RANSAC generalization, and (right) locally-originated generalization.

4 Evaluation Setup

For evaluating the performance of an adversary as described in Sect. 2.3, we check its inference performance over released modified point clouds. We use the descriptor set extracted from the 3D raw point cloud data as the reference set available to the adversary (labelled 1 in Fig. 1). We, then, implement various information reduction techniques to investigate how well can the adversary infer the identity, i.e. spatial location, of the released and modified point cloud.

4.1 3D Information Reduction Strategies

To limit the amount of information released with the point clouds, (1) plane generalizations and (2) partial releasing can be utilised to provide MR applications the least information necessary to deliver the desired functionality.

Plane Fitting Generalization. For the generalizations, as we do not intend to determine an efficient 3D generalization algorithm for our data, we have employed two simple techniques: the popular Random Sample Consensus (or RANSAC) plane fitting method [6], and a simple locally-originated plane generalization (we use label LOCAL henceforth). Figure 2 earlier shows what structurally occurs during plane-fitting generalization which can potentially preserve spatial privacy. Please see Appendix C for the generalization pseudo-code (Algorithms 1 and 2).

RANSAC. For our implementation, we directly utilize the accompanying normal vector of each point to estimate the planes in the plane fitting process instead of computing or estimating them from the neighbouring points. Algorithm 1 (in Appendix C) shows the pseudo-code of our RANSAC implementation, while an example RANSAC spatial generalization is shown in Fig. 6-center.

LOCAL. On the other hand, LOCAL generalization is an oversimplification of RANSAC as can be seen in Algorithm 2. We removed the point and plane test (i.e. Lines 12 and 14 in Algorithm 1) which ensures that a point is a valid member of the candidate plane and that the candidate plane is the best, i.e. largest, among all candidate planes. This results in more inaccurate generalizations as we go further away from the initial test point from which the candidate plane originated. Figure 6-right shows a sample LOCAL generalization.

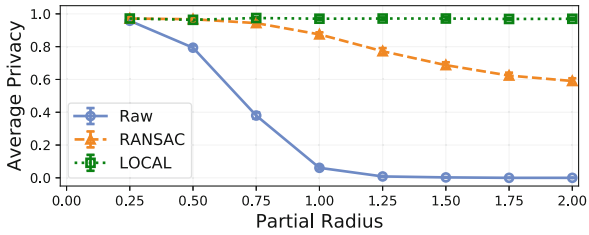


Fig. 7. Average privacy (i.e. *mean error rate* ± *margin of error with 95% confidence*) over one-time released partial spaces with varying radii and generalizations

Partial Spaces. In partial spaces, we only release *segments* of the space with varying radius. This demonstrates the case when an MR application is provided with limited 3D spatial information *only once*, such as a specific surface, a plane or an anchor point. We apply this technique to both raw and generalized point clouds. For every partial space level (i.e. radius), we get 10 sample random

iterations per space as a user can initiate an MR application from any point within a space; to demonstrate rotation-invariance, we further vary the spaces by doing 5 random rotations which results to a total of 50 iterations per space. We, then, get the mean error rate (with confidence intervals) over these iterations.

4.2 Successive Release of Partial Spaces

We use the information reduction techniques described in Sect. 4.1 as strategies for privacy protection. First, we evaluated adversarial performance over *one-time* released partial spaces as described in Sect. 4.1. Then, we introduced more information by successively releasing partial spaces.

To demonstrate the case when users are moving around and their physical space is gradually revealed, we included an experimental setup that successively releases partial spaces. Following the described abstraction strategies in Sect. 4.1, we have the following different 3D data setups for successively releasing of partial spaces: (1) from collected raw points, (2) from RANSAC generalized planes, and (3) from LOCAL generalized planes. Similar to the one-time partial release case, we do 10 sample iterations, and 5 random rotations for each case in the successive release setup. (For the extended LOCAL shown in Fig. 8d, we do 10 sample iterations but only did one random rotation for demonstration purposes.)

5 Results and Discussion

In the succeeding discussions, we would like to emphasize the trends and relative values rather than absolute empirical values themselves. We also presented takeaways whose discussions on trends and relationships can be generalized.

5.1 Inference of Partial Spaces

Figure 7 shows the performance of our adversarial inference over partial spaces with raw points and of the two generalized cases. For the raw-points case, at radius $r = 0.25$, the average privacy Π_{Raw} is very high, but immediately drops below $\Pi_{Raw} < 0.8$ at $r \geq 0.5$. With RANSAC generalization applied, it can be seen that the inference success is reduced, or essentially prevented, with radii $r \leq 1.0$, but average privacy Π_{RANSAC} starts to decrease for $r > 1.0$; thus, RANSAC generalizations are not effective protection strategies. This should not come as a surprise, since the RANSAC algorithm will try to fit planes as close to the true/raw space.

On the other hand, locally-originated plane generalizations can prevent inference for this one-time partial release case. Regardless of the size of the revealed space, the average privacy stays at $\Pi_{LOCAL} > 0.9$ as shown in Fig. 7. In fact, contrary to RANSAC generalizations, locally-originated plane generalizations will maintain a high Π_{LOCAL} with larger revealed spaces because the LOCAL algorithm will only produce a generalized plane from a *singular* local reference point which may not even be from a true plane or have a normal vector consistent with its neighbours. This results in plane generalizations that are more likely to be very different from the surfaces of the true spaces.

5.2 Successive Release of Partial Spaces

Following the partial spaces performance, it is tempting to say that we can maintain privacy by only releasing partial spaces of $r \leq 0.25$ even with raw captured data, but that is only for the single one-time release case. In this section, as described in Sect. 4.2, we will now show the privacy or inference performance when we *successively* release partial spaces.

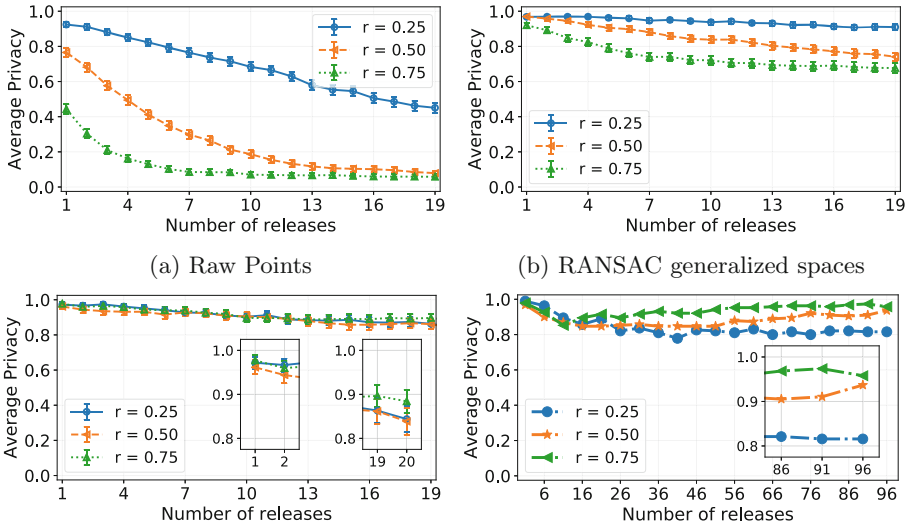
Raw-Points Spaces. Figure 8a shows the inference performance of successively released partial raw-points spaces. This is consistent with the results presented in Fig. 7. After a good number of releases, the space is slowly revealed; thus, the dropping average privacy. For $r = 0.25$, the Π_{Raw} drops below 0.8 after 4 or more releases, while for the larger radii, $r \geq 0.5$, the average privacy quickly drops and even starts at $\Pi_{Raw} < 0.8$ at the first release.

RANSAC Generalized Planes. For the successively released, RANSAC generalized partial spaces, as shown in Fig. 8b, after 4 releases, $\Pi_{RANSAC} \leq 0.8$ for radius $r = 0.75$. Similar to the performance shown in Fig. 7, at higher radii, Π_{RANSAC} for successive release eventually falls below ≤ 0.6 after a good number of releases. Specifically, for $r \geq 0.5$, $\Pi_{RANSAC} \leq 0.6$ after about 14 releases.

Compared to the successively released partial spaces from raw points, the RANSAC generalization already contributes some errors to the released spaces. This reflects on the rather slow drop of Π_{RANSAC}. Nonetheless, if RANSAC spaces are continuously released, regardless of its size, the space will be revealed. However, keeping RANSAC spaces to a small size, i.e. $r \leq 0.5$, and limiting release, e.g. no more than 10 releases, RANSAC can be a potential inference protection aside from being a generalization technique.

Local Generalized Planes. Similar to the results in Fig. 7, the inference performance from successively released and locally generalized partial spaces, as shown in Fig. 8c, presents error rates above 0.8 within 20 releases. To check inference performance for more releases using LOCAL, we extend the number of releases to 96 and checked the inference performance every multiple of 5 successive releases as shown in Fig. 8d. Now, the average Π_{LOCAL} do drop to ≤ 0.8 for $r = 0.25$ ($r = 0.75$ approaches 0.8 at release 10) but eventually increases with more releases. Due to the high inaccuracy provided by localized generalizations, especially at larger partial spaces, more releases do not contribute to improved inference and only *misleads* adversarial inference. Partially released planes with nearby originating points with different normals will produce planes within the same vicinity but of different orientations. This confuses the inferrer. Thus, if spatial privacy is a priority, localized generalizations can be used.

Takeaway. *Privacy can be arranged as $\Pi_{Raw} < \Pi_{RANSAC} < \Pi_{LOCAL}$, based on the form of released data; for continuously released large spaces ($r > 0.5$), RANSAC cannot provide adequate privacy, but for small enough spaces ($r \leq 0.5$), it can be a potential form of inference protection coupled with limited or controlled releasing.*



(a) Raw Points

(b) RANSAC generalized spaces

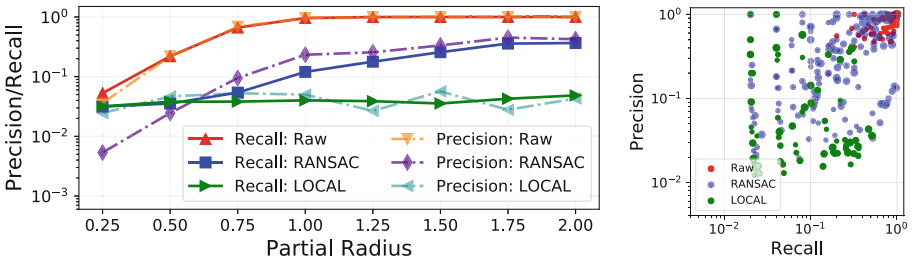
(c) LOCAL generalized spaces

(d) LOCAL generalized extended

Fig. 8. Average privacy (*mean error rate* ± *margin of error with 95% confidence*) over *successively* released partial spaces. For Fig. 8a–c, we perform up to 20 releases per iteration. For Fig. 8c, we extend the LOCAL case to see long-term inference.

5.3 Inference Trends with Spatial Properties

Precision and Recall. We also checked the precision and recall as an inference performance metric. These values were checked for every space as well as the impacts of spatial properties on inference and/or privacy. Figure 9a shows the average precision and recall of our adversarial inferrer as we vary the radius of partial spaces. As expected, for raw-points and RANSAC-generalized spaces, precision and recall increases as the radius increases. On the other hand, precision and recall of LOCAL stays low, < 0.1, and only ever so slightly increases –



(a) Precision and recall vs radius

(b) Precision vs recall

Fig. 9. Precision and recall over partial spaces

from 0.032 to 0.048 for recall, and from 0.024 to 0.043 for precision – but not consistently (as we can see with the dips in precision at $r = 1.25$ & 1.75).

Figure 9b shows the scatter plot of the precision and recall values for all spaces and iterations (averaged in Fig. 9a) with the radius (relatively) depicted by the size of the circle. We can see that the values for the raw-points spaces crowd on the upper right quadrant, i.e. high precision and recall area, while that of RANSAC generalized spaces is slightly more scattered but also crowds on the upper right quadrant. For the locally-generalized spaces, most of the green circles reside on the lower half which means that recall is spread from low to mid-high but precision values are mostly very low.

Despite the bad performance of our adversarial inferrer, looking more closely in to the spaces reveals some consistency. We looked into the top 10 spaces for raw points, RANSAC generalized, and LOCAL generalized in terms of *number of false positives*, *precision*, *recall*, and *least errors/privacy*. (In the interest of space, we no longer show the list of top 10 spaces.) The list reveals that the spaces with high recall and least errors are almost the same; thus, high recall and least errors have a high correlation (i.e. $\rho_{recall,least-errors} \approx 0.964$).

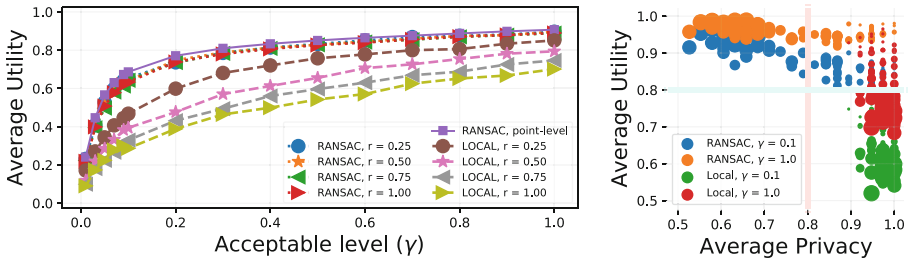
Furthermore, for the raw and RANSAC cases, the average number of planes of the top 10 spaces with high false positives are small, i.e. 4.21 and 4.38, respectively, while those of the top 10 spaces in terms of precision have higher averages at 14.44 and 13.77, respectively. Thus, raw or RANSAC spaces with more planes have lower uncertainty in being inferred or identified, and, perhaps, if privacy is desired, we may only release a lower number of planes, i.e. < 5. However, for the LOCAL generalized case, there is no observable trend among the inference performance and that of the number of planes.

Takeaway. *Raw and RANSAC spaces with higher number of observable or generalized planes are more likely to be inferred with higher precision; thus, releasing spatial generalizations with lower number of planes (i.e. < 5) can confuse adversarial inference.*

5.4 Computing Utility of Generalizations

Plane-fitting generalizations contribute variations to the released point clouds from true spaces. Figure 10a shows the computed average utility based on Eq. 5 for the different generalizations with varying partial radius and acceptability metric γ. A γ value of 1 means that we accept variations for up to 1 unit-combined-difference (see Eq. 5) of the true point from the released point and the true normal from the released normal.

For reference, we include the point-level (synonymous to $r = 0$) utility computation from RANSAC points which produces the highest utility trend, while other RANSAC generalizations of partial spaces with $r > 0$ comes close second. The average utility provided by RANSAC generalizations are consistent regardless of the size of the released generalized spaces. It does decrease as we decrease the acceptability value γ, but it does not go too low, i.e $U_{RANSAC} \geq 0.5$ for $\gamma \geq 0.1$, such that the generalizations are rendered unacceptable. This is due to how RANSAC generalizations tries to approach the true spaces.



(a) (b)

Fig. 10. (a) Utility of the generalizations (Note: Utility of true points and planes are always 1.); (b) Scatter plot of utility and error rate of different partial spaces (radius is relatively indicated by marker size)

On the other hand, LOCAL generalizations have lower utility trends and go much lower as the radius increases. This is due to the increased inaccuracies in the localized generalizations as it disregards point locations and normals other than the randomly chosen origin point. As a result, the utility trend further decreases as we increase the radius, and this is true for any γ. In fact, at $\gamma = 0.1$, $U_{LOCAL} \leq 0.5$ at $r = 0.25$. As expected, if we are to set the acceptable utility at ≥ 0.8, only localized generalizations of radius $r \leq 0.5$ can provide such utility and $r = 0.5$ barely makes the cutoff at $\gamma = 1.0$. Any γ lower than that, only generalizations with $r \leq 0.25$ can provide an average utility ≥ 0.8.

In reality, these U_{LOCAL} values are unacceptable. If we are to set an acceptability level of $\gamma \leq 0.2$, there is only at most 0.6 chance of getting a locally generalized point that is close to the true point including its orientation. Thus, for the rest of the points from a locally generalized point cloud, augmentations are translated by at most 0.2 m (in any direction) and/or rotated by at most $cos^{-1}(0.2)$ or 78.5°.

The difference in utility and error rate as we vary the radius of partial spaces is better visualized by the scatter plot in Fig. 10b. U_{RANSAC} stays ≥ 0.8 and

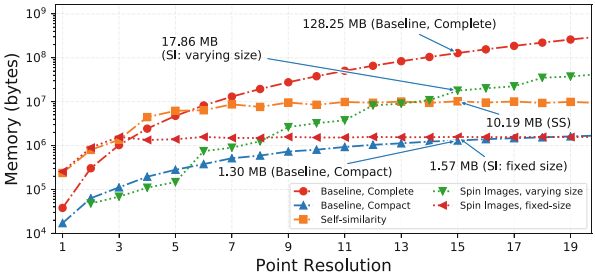


Fig. 11. Used memory by inference models and descriptors extracted from different point cloud resolutions.

privacy drops as we increase the size, while U_{LOCAL} is only ≥ 0.8 for smaller partial size and the privacy is consistently ≥ 0.8. The relatively higher utility of smaller LOCAL releases is further corroborated by the average privacy values of the successive release case shown in Fig. 8c and d which shows smaller spaces having lower privacy compared to larger spaces with more releases.

For LOCAL, points nearby the reference point will most likely have similar normal vector directions, but as we go further away from the reference point on the same locally generalized plane the variation increases, and thus the utility drops. Conversely, RANSAC contributed variations are fairly consistent and low regardless of a point's distance from a reference point with which the generalized plane was produced, since it tries to do a good representation of the true space.

Takeaway. *Overall,* ***LOCAL*** *generalizations provides high average privacy but can only provide adequate utility for smaller spaces; for example, utility of* $U > 0.5$ *for* $\gamma \leq 0.2$ *can only be achieved with spaces of small radius* $r \leq 0.25$*.*

5.5 Memory Compactness of Descriptors and Inference Models

Another interesting aspect is how a very good inferrer can be constructed at a low resolution $res \leq 10$ with discriminative performance similar to that of higher resolutions (see Fig. 5). As shown in Fig. 11, the memory size exponentially increases as we increase the resolution. A baseline Bayesian inference model with a low resolution of 15 requires a memory size of about 128 MB. This memory usage is undesirably huge due to the almost complete representation of the point probabilities in 3D space. However, we can take advantage of the sparsity of the data points to make it compact. The memory usage by the compact representation is also shown in Fig. 11. At res = 15, the compact memory usage is now just 1.30 MB from the original 128 MB – almost 2 orders of magnitude smaller.

For the rotation-invariant descriptors, at res = 15, a corresponding set of SS descriptors takes about 10.19 MB, but a corresponding set of SI descriptors – which, anyway, performs better than SS descriptors – with a fixed descriptor size is as compact as the baseline inference model (that is not rotation-invariant) at only 1.58 MB. In fact, we used res = 3 (as previously stated in Sect. 3.2) for the descriptors used in the inference evaluation discussed in the previous subsections.

Thus, any MR application (trusted or not) with access to 3D data produced by the user's MR device can efficiently create a lightweight inference model of the user's space. (For reference, the original point-cloud data is about 13 MB; thus, our inferrer is a much more compact representation of the point-cloud data at res = 15.)

Takeaway. *A compact and efficient inferrer of 3D spaces can be created from raw point cloud data released by any MR-capable device (which, now, can be any device with a vision sensor and adequate processing power).*

6 Related Work

Most privacy work for MR were primarily focused on *visual* information or media (i.e. image and video) sanitization [12,20,21]. Aside from that are *abstraction* approaches to privacy protection. In the specific 3D use case, significant work have been done on protecting physiological information using abstractions [5,11] using the idea of *least privilege* [25]. The same approach has also been used for providing visual privacy when using 3D MR browsers [24]. However, these works did not specifically work on protecting 3D MR data against spatial inference.

Other recent works have focused on protecting MR outputs specifically in ensuring user safety [15,16]. Furthermore, as MR devices allow for new modes of *collaboration*, issues on *power imbalance* brought by the *directionality* of MR interfaces [2] are now being studied as well [17]. Again, these works do not focus on spatial inference using 3D MR data.

7 Conclusion

In this work, we demonstrated how we can infer and reveal spaces employing descriptor-based inference over 3D point cloud data collected using the Microsoft HoloLens. The same point cloud data representation is also used by Google's ARCore and Apple's ARKit. Therefore, it is possible to easily extend this work to these mobile MR platforms as well. Currently, these MR platforms do not apply privacy preservation on released 3D MR data to third party applications which can allow adversaries to easily perform spatial inference attacks similar to what we have demonstrated. In addition, we have demonstrated how leakage can persist even after implementing spatial generalizations: RANSAC generalizations can't provide adequate protection when continuous successive generalizations are released, while LOCAL generalizations provide promise in protecting spatial privacy but utility is currently undesirably low. If directly applied, LOCAL generalizations cause augmentations to be shifted, translated, and/or rotated by a great degree, i.e. a maximum combined error of 0.2 with maximum average utility of only 0.6.[8] Moreover, we show how compact in terms of memory usage these 3D inference models can be, which allows adversaries to keep models for every users' set of 3D spaces.

In our future work, we aim to develop a hybrid generalization technique as a potential privacy solution combining desirable properties from RANSAC and LOCAL to; perhaps, in conjunction with *controlled releasing*, where we do not release a new portion of the space if the requested 3D space overlaps with those released earlier. Moreover, limiting released generalizations to no more than 4 planes, and/or limiting the number of partial successive releases may also provide inference protection. Furthermore, we intend to extend the proposed *geometric information* based inference strategy to use additional *photometric* information such as (RGB) color profile as well as employing advanced techniques for adversarial inference.

[8] Combined error in terms of rotation ($\cos \Delta\theta$) and translation (Δx); see Eq. 5.

A 3D Spatial Definitions

Defining the Input Space. Let X_i be the raw representation of space i in the physical world. A point-cloud extractor F takes pose information vector $v \in R^3$ and releases a point cloud $x_{i,v}$ relative to that pose,
$F : X_i, v \rightarrow x_{i,v}$, for any 3D space with location i and a reference pose v.

Combining $x_{i,v}$ produces a complete point-cloud representation of space X_i, which we label as $\hat{X}_i = \bigcup_v x_{i,v}\ \forall v$. An extension of this is that for any pose $v \in R^3$, we get a partial point-cloud representation $X_{i,v}$ of the true space. And that there exists a set of poses $v_s \subset V$ such that $\hat{X}_{i,v_s} = \bigcup_{v \in v_s} x_{i,v}$ spans X_i or $\hat{X}_{i,v_s} = X_i$.

Defining the Abstraction. A privacy-preserving mechanism M transforms any released point cloud $x_{i,v}$ to a privacy-preserving version $z_{(i),v}$,
$M : x_{i,v} \rightarrow z_{(i),v}$, where we denote the privacy-preservation of i by (i) – that is, the true i of a released z is not divulged or kept secret. Figure 2 shows a simple visualization of the transformation that can occur. In this specific case, the normal vectors of the adjacent points are aligned to create a flat surface.

Similar to the raw point-clouds $x_{i,v}$, combining the privacy-preserving point-cloud representations $z_{(i),v}$ produces $\hat{Z}_{(i)} = \bigcup_v z_{(i),v}$ for all $v \in V$,
or $Z_{(i)} = \bigcup_v z_{(i),v}$.

Defining the Intended Functionality. An intended deterministic output y produced by an intended application or functionality G upon taking point clouds as the input, expressed as $G : x_i, or\ z_{(i)} \rightarrow y_{(i)}$.

B Defining the Feature Matching Process Using Rotation-Invariant Descriptors

A matching function Υ maps two sets of features f_a and f_b, of spaces a and b, like so: $\Upsilon : f_a \mapsto f_b$.

To determine good matches, we use the descriptor Euclidean distance as a measure of their similarity. To *accept* a match for a key point $x_{a,1}$ with feature $f_{a,1}$ of an unknown query space $a = i^*$, we get the *nearest neighbor distance ratio* (NNDR) of the features like so: $\frac{||f_{a,1} - f_{b,1}||}{||f_{a,1} - f_{b,2}||} < threshold$, where descriptor $f_{b,1}$ of $x_{b,1}$ (i.e. key point x_1 of known space $b = i$) is the nearest neighbor of descriptor $f_{a,1}$ of $x_{a,1}$ (i.e. key point x_1 of unknown query space $a = i^*$) and $f_{b,2}$ is the second nearest neighbor, and see if the NNDR falls below a set threshold (e.g. 0.75 for the self-similarity, or 0.9 for the spin-image descriptors). Then, we maximum-normalize the distance of the accepted matches to make the maximum distance be 1. The mean of the distances is multiplied with a Bayesian-inspired weight, $\frac{|\{f_{x_{i^*}} \mapsto f_{x_i}\}|}{|\{f_{x_{i^*}}\}|}$, where $|\{f_{x_{i^*}} \mapsto f_{x_i}\}|$ is the number of matched descriptors of an unknown query space $x_{a=i^*}$ from one of the known reference spaces $x_{b=i}, i \in \forall i$, and $|\{f_{x_{i^*}}\}|$ is the number of key points or descriptors extracted from the

query space x_{i^*}. This allows us to create a hypothesis, i.e. $h : i^* = i$, also via argument-maximization as follows,

$$\arg\max_i \left(1 - \underset{\{f_{x_{i^*}} \mapsto f_{x_i}\}}{\text{mean}} \{||f_{x_{i^*}} - f_{x_i}||\}\right) \cdot \frac{|\{f_{x_{i^*}} \mapsto f_{x_i}\}|}{|\{f_{x_{i^*}}\}|}, \tag{6}$$

where the first product term is the mean similarity (i.e. 1 - *mean difference*) while the second term is the Bayesian-inspired weight.

C Plane Generalization

Our RANSAC plane generalization, shown in Algorithm 1, mainly follows the described algorithm in [6] except for the normal estimation which we skip and instead use the estimated normal vectors directly provided by the spatial mesh produced by the HoloLens. On the other hand, the algorithm for the locally-originated plane generalization, shown in Algorithm 2, is a crude and simplified generalization which removes the point (Line 12) and plane (Line 14) discrimination process from RANSAC.

Algorithm 1. RANSAC algorithm [6]

F the number of planes to find = 30
T the point-plane distance threshold = 0.05
R the number of RANSAC trials = 100
Data: $X = \{x_1, x_2, ..., x_n\}$, a set of 3D points
Result: $P = \{p_{x_m} : \{x_{p_1}, x_{p_2}, ...\}\}$, a set of planes (a 3D point, and a normal) and their associated co-planar points

```
for f ← 1 to F do
    bestPlane = {0, 0}
    bestPoints = {}
    for r ← 1 to R do
        S = s_1 = a point at random from X
        thisPlane = {s_1, normal_{s_1}}
        thisPoints = {}
        for x_i ∈ X do
            if (distance(thisPlane, x_i) ≤ T) then
                thisPoints ← thisPoints + x_i
        if |thisPoints| > |bestPoints| then
            bestPlane ← thisPlane
            bestPoints ← thisPoints
    P ← P + {bestPlane, coPlanarTransformed(bestPoints)}
    X ← X − bestPoints
```

Algorithm 2. Locally-originated plane generalization

F the number of planes to find = 30
r the radius of the local region (e.g. 0.5)
Data: $X = \{x_1, x_2, ..., x_n\}$, a set of 3D points
Result: $P = \{p_{x_m} : \{x_{p_1}, x_{p_2}, ...\}\}$, a set of planes (a 3D point, and a normal) and their associated co-planar points
for $f \leftarrow 1$ **to** F **do**
    $S = s_1$ = a point at random from X
    $thisPlane = \{s_1, normal_{s_1}\}$
    $thisPoints = \{x_i \in X : |x_i - s_1| \leq r\}$
    $P \leftarrow P + \{thisPlane, coPlanarTransformed(thisPoints)\}$
    $X \leftarrow X - thisPoints$

References

1. Acquisti, A.: Privacy in the age of augmented reality (2011)
2. Benford, S., Greenhalgh, C., Reynard, G., Brown, C., Koleva, B.: Understanding and constructing shared spaces with mixed-reality boundaries. ACM Trans. Comput. Hum. Interact. (TOCHI) **5**(3), 185–223 (1998)
3. Bronstein, A.M., et al.: SHREC 2010: robust feature detection and description benchmark. In: Proceedings of EUROGRAPHICS Workshop on 3D Object Retrieval (3DOR) (2010)
4. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Found. Trends® Theor. Comput. Sci. **9**(3–4), 211–407 (2014)
5. Figueiredo, L.S., Livshits, B., Molnar, D., Veanes, M.: Prepose: privacy, security, and reliability for gesture-based programming. In: 2016 IEEE Symposium on Security and Privacy (SP), pp. 122–137. IEEE (2016)
6. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
7. Gross, R., Sweeney, L., de la Torre, F., Baker, S.: Semi-supervised learning of multi-factor models for face de-identification. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8, June 2008. https://doi.org/10.1109/CVPR.2008.4587369
8. de Guzman, J.A., Thilakarathna, K., Seneviratne, A.: Security and privacy approaches in mixed reality: a literature survey. arXiv preprint arXiv:1802.05797 (2018)
9. He, W., Liu, X., Nguyen, H.V., Nahrstedt, K., Abdelzaher, T.: PDA: privacy-preserving data aggregation for information collection. ACM Trans. Sens. Netw. (TOSN) **8**(1), 6 (2011)
10. Huang, J., You, S.: Point cloud matching based on 3d self-similarity. In: 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 41–48. IEEE (2012)
11. Jana, S., et al.: Enabling fine-grained permissions for augmented reality applications with recognizers. In: USENIX Security (2013)
12. Jana, S., Narayanan, A., Shmatikov, V.: A scanner darkly: protecting user privacy from perceptual applications. In: 2013 IEEE Symposium on Security and Privacy (SP), pp. 349–363. IEEE (2013)
13. Johnson, A.E., Hebert, M.: Using spin images for efficient object recognition in cluttered 3d scenes. IEEE Trans. Pattern Anal. Mach. Intell. **5**, 433–449 (1999)

14. Johnson, A.E., Hebert, M.: Surface matching for object recognition in complex three-dimensional scenes. Image Vis. Comput. **16**(9–10), 635–651 (1998)
15. Lebeck, K., Kohno, T., Roesner, F.: How to safely augment reality: challenges and directions. In: Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications, pp. 45–50. ACM (2016)
16. Lebeck, K., Ruth, K., Kohno, T., Roesner, F.: Securing augmented reality output. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 320–337. IEEE (2017)
17. Lebeck, K., Ruth, K., Kohno, T., Roesner, F.: Towards security and privacy for multi-user augmented reality: foundations with end users. In: Towards Security and Privacy for Multi-User Augmented Reality: Foundations with End Users, p. 0. IEEE (2018)
18. Milgram, P., Kishino, F.: A taxonomy of mixed reality visual displays. IEICE Trans. Inf. Syst. **77**(12), 1321–1329 (1994)
19. Newton, E.M., Sweeney, L., Malin, B.: Preserving privacy by de-identifying face images. IEEE Trans. Knowl. Data Eng. **17**(2), 232–243 (2005)
20. Raval, N., Srivastava, A., Razeen, A., Lebeck, K., Machanavajjhala, A., Cox, L.P.: What you mark is what apps see. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, pp. 249–261. ACM (2016)
21. Roesner, F., Molnar, D., Moshchuk, A., Kohno, T., Wang, H.J.: World-driven access control for continuous sensing. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 1169–1181. ACM (2014)
22. Shokri, R., Theodorakopoulos, G., Le Boudec, J.Y., Hubaux, J.P.: Quantifying location privacy. In: 2011 IEEE Symposium on Security and Privacy, pp. 247–262. IEEE (2011)
23. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **10**(05), 557–570 (2002)
24. Vilk, J., et al.: SurroundWeb: mitigating privacy concerns in a 3d web browser. In: 2015 IEEE Symposium on Security and Privacy (SP), pp. 431–446. IEEE (2015)
25. Vilk, J., et al.: Least privilege rendering in a 3d web browser. Technical report (2014)
26. Wagner, I., Eckhoff, D.: Technical privacy metrics: a systematic survey. ACM Comput. Surv. **51**(3), 57:1–57:38 (2018). https://doi.org/10.1145/3168389, http://doi.acm.org/10.1145/3168389
27. Wu, Y., Yang, F., Ling, H.: Privacy-protective-gan for face de-identification. arXiv preprint arXiv:1806.08906 (2018)
28. Zarepour, E., Hosseini, M., Kanhere, S.S., Sowmya, A.: A context-based privacy preserving framework for wearable visual lifeloggers. In: 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), pp. 1–4. IEEE (2016)