Communications in Information and Systems
Volume 23, Number 2, 213–239, 2023

TSDFFilter: content-aware communication planning for remote 3D reconstruction

Xu-Qiang Hu, Yu-Ping Wang, Zi-Xin Zou, and Dinesh Manocha

We present a novel solution, TSDFFilter, for remote 3D reconstruction to relieve the high bandwidth requirement problem. Our approach is designed for scenarios where agents are used to collect data using an RGB-D camera and then transmit the information over the regular network to a high-performance server, where a global, dense, and volumetric model of a real-world scene is reconstructed. Our approach uses a content-aware communication planning framework in which agents can prune the gathered RGB-D information according to the transmission policy generated by the server. To generate the transmission policy, we introduce a confidence value to estimate how much each RGB-D pixel contributes to the reconstruction quality, and present an algorithm to find the confidence value. As a result, agents can transmit less RGB-D information without blindly compromising the reconstruction quality as the key-frame method and down-sampling method do. We implement our TSDFFilter framework to achieve real-time agent-assisted 3D reconstruction. Extensive evaluations show that comparing with the key-frame and down-sampling methods, our TSDFFilter framework can reduce the bandwidth requirement by up to 36% with similar reconstruction Chamfer distance, and reduce the reconstruction Chamfer distance by up to 78% with similar bandwidth requirement.

Keywords and phrases: Communication planning, remote 3D reconstruction, TSDF, transmission policy.

1. Introduction

Reconstructing dense, volumetric models of real-world 3D scenes is an important research topic in Visual Media [3, 41, 43, 36]. With the wide usage of consumer RGB-D cameras, gathering visual information for 3D reconstruction has become affordable, but data collection is still time-consuming. Thus, offline 3D reconstruction, where reconstruction is done after data collection, is time-consuming. With the widely usage of RGB-D cameras on mobile devices, and the development of robotics and drones, agents are being used

to automatically collect RGB-D data at remote locations [11, 6]. Such a 3D scene reconstruction framework is also called multi-agent collaborative dense scene reconstruction, or remote 3D reconstruction for short.

Under the remote 3D reconstruction framework, each agent is responsible for capturing part of the scene. For the reason of costs and energy, agents are not equipped with powerful GPUs which are usually needed for real-time reconstruction. Therefore, agents transmit the gathered RGB-D information to a high-performance server on which the global 3D model is reconstructed. However, this framework raises the requirements for high network bandwidth. In [11], the authors stated that the network bandwidth required to transmit the RGB-D images captured by a Kinect (640×480) is roughly 250Mbps, or 25Mbps if the same sequence is compressed, but their WiFi router can only provide a bandwidth of about 8Mbps. The situation was even worse when there were multiple RGB-D cameras, since they competed for the bandwidth of the WiFi router and the server's network card. Due to this issue, in practice, offline 3D reconstruction is usually preferred than remote 3D reconstruction in order to obtain more precise 3D model. Therefore, reducing the bandwidth requirements while retaining precision is essential for remote 3D reconstruction to scale to more agents and higher-resolution cameras.

Similar bandwidth problems also arise in remote simultaneous localization and mapping (SLAM) systems [9, 24, 33, 34]. These systems also gather and transmit high-resolution visual information to build a map and achieve localization. However, the main purpose of SLAM algorithms is localization, and building a sparse map of the scene is sufficient for accurate localization [8]. Therefore, not all collected points are valuable to SLAM algorithms. Transmitting only the feature points is sufficient and can relieve the bandwidth problem for remote SLAM systems [29]. However, this kind of solution cannot be used for remote 3D reconstruction frameworks.

In 3D reconstruction algorithms, all collected points are potentially valuable. They improve the reconstruction quality by either providing new information or reducing errors [27]. To relieve the bandwidth problem, remote 3D reconstruction systems currently have only two options: selecting key-frames or down-sampling the gathered images [11, 6]. Selecting key-frames results in the loss of some information provided only by the dropped frames and down-sampling the gathered images causes the details to be ignored. Reducing bandwidth requirements without compromising reconstruction quality is still a challenging problem for remote 3D reconstruction frameworks.

Main results In this paper, we present a communication planning algorithm for remote 3D reconstruction, TSDFFilter. Instead of totally dropping some of the RGB-D frames, our idea is to drop some RGB-D pixels. We introduce a ***confidence*** value for each RGB-D pixel and theoretically show that it can represent how much the RGB-D pixel contributes to the reconstruction quality. Based on the confidence value, the server can generate a transmission policy to the agents. Further, based on the transmission policy, the agents then transmit only the pixels that contribute more to the reconstruction quality, and thus the bandwidth requirement is reduced. We test our TSDFFilter framework to achieve real-time 3D scene reconstruction. Experimental results show that comparing with the key-frame and down-sampling methods, our TSDFFilter framework can reduce the bandwidth requirement by up to 36% with similar reconstruction Chamfer distance, and reduce the reconstruction Chamfer distance by up to 78% with similar bandwidth requirement. The main contributions of this paper include:

(1) We present a communication planning framework for remote 3D reconstruction, TSDFFilter, which can reduce the bandwidth requirement while retaining more useful details.

(2) We present the confidence value for each RGB-D pixel to estimate how much it contributes to the reconstruction quality, and an efficient algorithm to generate the confidence value.

(3) We apply our TSDFFilter framework to practical TSDF-based reconstruction system, InfiniTAM [17, 18], and implement it based on ROS [32], which is the de-facto robotic middleware.

(4) Our TSDFFilter framework is extensively evaluated with data from three datasets (the Scannet RGB-D dataset [4], the TUM RGB-D dataset [38] and the Cow & Lady RGB-D dataset [28] and show significant results. When the resulting reconstruction Chamfer distance is similar, our TSDFFilter framework can reduce the bandwidth requirement by up to 36% comparing with key-frame and down-sampling methods. When the bandwidth is similar, our TSDFFilter framework can reduce the reconstruction Chamfer distance by up to 78% comparing with key-frame and down-sampling methods.

2. Related work

2.1. 3D reconstruction

3D reconstruction is a common research interest in Computer Vision, Robotics, and Multimedia. Since the emergence of commodity RGB-D sensors (e.g., Microsoft's Kinect) and modern GPU programming frameworks

(e.g., NVidia's CUDA), real-time dense 3D reconstruction has become feasible on commodity hardware. KinectFusion [26, 16] was one of the first real-time volumetric reconstruction frameworks. To handle large-scale scenes, the state-of-the-art solutions [27, 17] organize voxels with a hash map. To achieve high reconstruction quality, the state-of-the-art frameworks [17, 18] employ the truncated signed distance function (TSDF) [2] as the data structure to store integrated depth images. Since the task of updating the TSDF value of each voxel is suitable for data-parallel algorithms [46], these frameworks rely on GPU to achieve real-time performance [39]. Our work is designed for frameworks employing voxel hashing and TSDF. Prior arts [1] consider pixel confidence maps in RGB-D reconstruction. Our work uses pixel confidence maps to generate transmission policy.

2.2. Remote SLAM and 3D reconstruction

Widely used mobile phone and commodity UAVs have provided more convenient ways to capture sensor data. Research has shown that these sensor data can be used for SLAM and 3D reconstruction. However, when these devices cannot provide the high performance required for real-time algorithms, the sensor data must be stored and processed off-line [49]. To process the sensor data in an online manner, remote SLAM [9, 24, 34, 44] and remote 3D reconstruction [11, 6] frameworks have been proposed. Such frameworks take advantage of the computing power provided by high-performance central server(s), and captured sensor data are transmitted to the server(s).

For remote SLAM frameworks, a major issue is what data representation should be transmitted. Opdenbosch et al. [29] proposed a solution where fast feature extraction is performed at the agent, and the server performs the following SLAM by collecting the features. These features are sufficient to generate a sparse 3D map, but far from sufficient to reconstruct a dense volumetric model. Therefore, this kind of solution works well for remote SLAM, but is not suitable for remote 3D reconstruction.

For remote 3D reconstruction, the only options are selecting key-frames and down-sampling every frame [11, 6]. These two solutions can reduce the bandwidth requirement significantly, but they also damage the reconstruction quality. The key-frame methods treat each image as a whole and drop some full images without distinguishing which pixels might be valuable. The down-sampling methods consider all pixels to be of equal value, which is not always true.

Our idea is to distinguish which pixels are more valuable to the global model. This is difficult, if not impossible, without knowing the status of the

global map. Our solution is inspired by Giamou et al. [10]. This work aims to detect inter-robot loop closures for remote SLAM under a constrained network bandwidth. In this work, agents exchange some meta-data before actually exchanging visual information. Based on these meta-data, agents can design a policy regarding which data should be exchanged. Our solution to the remote 3D reconstruction also allows the server to give some hints to the agent, so the agent can distinguish which pixels are more valuable.

2.3. Video streaming

Dynamic Adaptive Streaming over HTTP (DASH) [37] is the de-facto standard for video streaming. By using the DASH technique, each video is partitioned into segments and each segment is encoded in multiple bitrates. Which bitrate to use for the next segment is determined by Adaptive Bitrate (ABR) algorithm. Generally, video streaming services aim to improve the Quality of Experience (QoE) by increasing the average video bitrate, reducing rebuffering events, and/or increasing video bitrate smoothness. These factors cannot be satisfied at the same time, and existing solutions have made significant improvements in balancing these factors [22, 31, 35, 47, 14, 7, 45, 48].

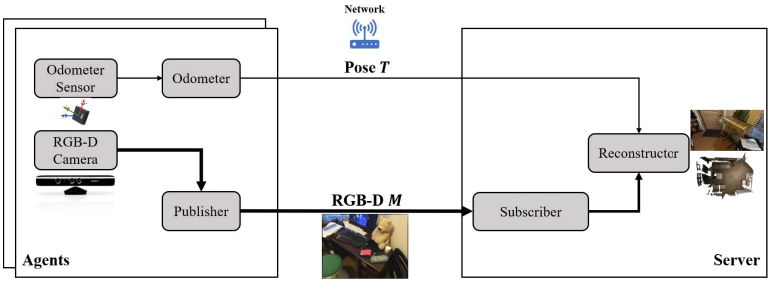
Volumetric video streaming, such as point cloud-based volumetric video [12, 20] or depth image (i.e., RGBD)-based volumetric video [21, 40], is more in line with application for 3D reconstruction, where remote agents transmit captured color and depth information to the server. Some researches extend DASH technique towards volumetric video streaming [13, 42].

However, the DASH technique is not well suited for 3D reconstruction services for two main reasons. On the one hand, down-sampling method used in DASH when bad network situation would lead to worse 3D reconstruction quality. On the other hand, some of content in sequence are redundant which will not be contributed to improve 3D reconstruction, e.g., for those area of already being well reconstructed. Our TSDFFilter algorithm only transmits the content which is useful for 3D reconstruction.

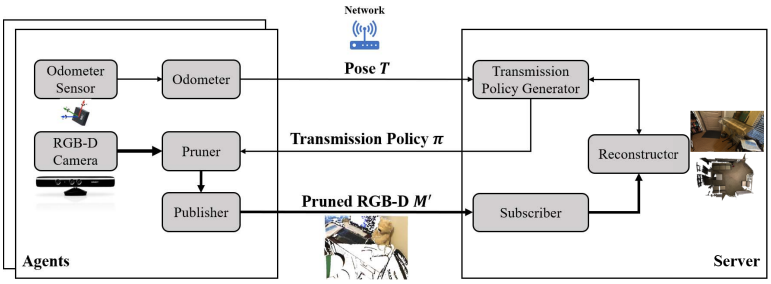
There are also some works [19] employ video compression techniques to transmit RGB-D streams and achieve competitive results. In all experiments of our framework and the comparing frameworks, we employ the same image compression algorithms for the convenience of implementation based on ROS [32]. It is feasible to change the compression algorithms to video compression algorithms, but we consider it orthogonal to our improvements.

3. Our approach

Figure 1(b) illustrates our TSDFFilter framework. In our framework, the agent transmits the pose T of the equipped camera to the server, the server generates a transmission policy $\pi(T, G)$ based on the pose T and the global status G of the server, and the agent prunes its RGB-D data from M to $M'(M, \pi)$ based on the transmission policy. In contrast, in the common framework (shown in Figure 1(a)), the agent does not know any information from the server and can only blindly transmit its poses and RGB-D data to the server. Down-sampling and selecting key-frames can only reduce the bandwidth requirement without knowing the needs of the server.



(a) Illustration of the common remote 3D reconstruction framework, where the agent blindly transmits its poses and RGB-D data to the server without knowing which data the server would prefer.



(b) Illustration of our TSDFFilter framework, where the server generates a transmission policy based on the pose and the agent can prune its RGB-D data based on the transmission policy.

Figure 1: An overview of the frameworks for remote 3D reconstruction. In our experiments, the overall bandwidth requirements are compared.

Here, we assume that each agent knows its own pose. This can be achieved separately by either low-cost SLAM algorithms [25], low-bandwidth

remote SLAM frameworks [5], or other sensors (e.g., IMU) equipped on the agent. Therefore, we can focus on how to transmit the information needed by the integration routines of 3D reconstruction frameworks.

3.1. Generating transmission policy

The key challenge is how to express and generate a transmission policy that can conform to the following principles:

- The pruned RGB-D data and the original RGB-D data should contribute similarly to the reconstruction quality.
- The transmission policy should be generated efficiently.
- The transmission policy should help reduce the overall bandwidth.

The first principle requires us to mine the properties of the reconstruction algorithm. As we have stated, to achieve high-quality 3D reconstruction, TSDF has become a de-facto expression of high-quality 3D models. TSDF is a volumetric representation of a scene for integrating depth images. When integrating a new depth image, new voxels that have never been captured before are allocated. Then, the TSDF value $TSDF(x)$ and the weight $W(x)$ of each associated voxel x are updated with Equation (1). t is the truncation parameter, $w_i(x)$ is the weight for each round of update, which is usually set as 1, $d_i(x)$ is the depth captured by depth camera, and $z_i(x)$ is the depth of the voxel.

$$
\begin{aligned}
TSDF_i(x) &= \frac{TSDF_{i-1}(x) \cdot W_{i-1}(x) + tsdf_i(x) \cdot w_i(x)}{W_{i-1}(x) + w_i(x)} \\
tsdf_i(x) &= max\left(-1, min\left(1, \frac{d_i(x) - z_i(x)}{t}\right)\right) \\
W_i(x) &= W_{i-1}(x) + w_i(x)
\end{aligned}
\tag{1}
$$

For each round of updates, the captured depth $d_i(x)$ is an approximate value of the depth from the view point to the real-world surface, $s_i(x)$. We can assume that $d_i(x)$ is a random variable that follows a normal distribution with mean $s_i(x)$, and each $d_i(x)$ is independent (see Figure 2). The variance σ_i^2 reflects the error introduced by the RGB-D camera. Thus, we can deduce in Equation (2) that the $TSDF_i(x)$ also follows a normal distribution after $W_i(x)$ rounds of update. In this normal distribution, the expectation reflects the real-world $TSDF(x)$. Note that the variance becomes smaller as the update round $W_i(x)$ increases. Specifically, when all σ_j is the same, the

variance becomes $\frac{\sigma^2}{t^2 \cdot W_i(x)}$ which is inversely proportional to $W_i(x)$.

$$
\begin{aligned}
d_i(x) &\sim N\left(s_i(x), \sigma_i^2\right) \\
tsdf_i(x) &\sim N\left(\frac{s_i(x) - z_i(x)}{t}, \frac{\sigma_i^2}{t^2}\right) \\
TSDF_i(x) &\sim N\left(\frac{\Sigma(s_j(x) - z_j(x))}{t \cdot W_i(x)}, \frac{\Sigma\sigma_j^2}{t^2 \cdot W_i^2(x)}\right)
\end{aligned} \tag{2}
$$

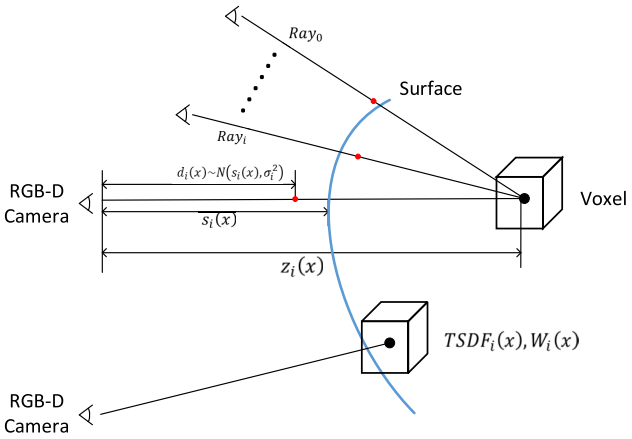


Figure 2: The depth captured by the RGB-D camera can be modeled as a normal distribution. For each voxel x, the $TSDF_i(x)$ also follows a normal distribution after $W_i(x)$ round of updates.

For the second and the third principles, our purpose is to express the transmission policy, which decides whether each RGB-D pixel needs to be transmitted. If we can find a voxel x on the ray corresponding to the RGB-D pixel with its $TSDF(x) = 0$, we can estimate the real-world $TSDF(x)$ with the normal distribution of voxel x. If we can estimate the real-world $TSDF(x)$ with enough confidence, we do not need further updates. Since smaller variance indicates more confidence, and the variance is inversely proportional to $W_i(x)$, we can set a W_{MAX} to indicate the confidence threshold. Thus, the error caused by the generated transmission policy follows a normal distribution in Equation (3), and we can expect the reconstruction Chamfer distance of the result dense model is about $\frac{\sigma^2}{t^2 \cdot W_{\text{MAX}}}$. Since σ^2 is a constant that reflects the error introduced by the RGB-D camera, and t is a constant set by the 3D reconstruction algorithm, the reconstruction

Chamfer distance of the resulting dense model is expected to be inversely proportional to W_{MAX}.

$$Error(x) \sim N\left(0, \frac{\sigma^2}{t^2 \cdot W_{\mathrm{MAX}}}\right) \tag{3}$$

Algorithm 1: Generate a Transmission Policy(on the server)

Data: modle, the current 3D model; W_{MAX}, the confidence threshold.
Input: pose, the current pose of robot.
Output: res, a binary image.

```
procedure genTransPolicy(pose):
    for each pixel p of res do
        while loop ray-casting from p, pose do
            x ← current voxel
            if (x is unexplored) then
                p ← 1
            else if (TSDF(x) > 0) then
                continue ray-casting.
            else
                interpolate a position with TSDF = 0.
                compute the average W at the interpolate position.
                if W >= W_MAX then
                    p ← 0 // this pixel need not to be transmitted
                else
                    p ← 1
                end
            end
        end
    end
    return res
end
```

Overall, the algorithm for generating the transmission policy is shown in Algorithm 1. The transmission policy is expressed with a binary image with the same size of the RGB-D images. For each RGB-D pixel, we try to find a voxel x on the corresponding ray that satisfies $TSDF(x) = 0$ and $W(x) \geq W_{\mathrm{MAX}}$. If such a voxel is found, the transmission policy decides not to transmit the RGB-D pixel or, otherwise, the transmission policy decides to still transmit the RGB-D pixel. This routine is highly parallelizable and can be accomplished quickly on the server with modern GPU, which meets the

second principle. The binary image that expresses the transmission policy can be further compressed to reduce the required bandwidth. We will verify the third principle in Section 4.

3.2. Alternative solutions

We can also theoretically analyze the error introduced by the key-frame method and the down-sample method.

For the key-frame method, let key-frame ratio $K \leq 100\%$ represent the ratio of preserved frames, e.g., $K = 25\%$ indicates that one out of every 4 frames is preserved. On average, this configuration is equivalent to the case where each frame has a probability of K to be dropped. Thus, for a voxel x that could be updated for $W(x)$ rounds, there are only $K \cdot W(x)$ rounds after key-frame selection. We showed in Equation (2), the error for a voxel x whose $TSDF(x) = 0$ is inversely proportional to $W(x)$. Key-frame method reduces every $W(x)$ by K times, and therefore increase the error for every voxel by K^{-1} times. If $W(x)$ is large for all voxels originally, the key-frame method introduces fewer errors; however, if $W(x)$ is small, the key-frame method could introduce large errors. In extreme cases, for some voxel x whose $W(x) = 1$ originally, it could be reduced to $W(x) = 0$, which means the voxel x is completely lost. In this case, the error depends on the distance to the next nearest voxel, which is not predictable.

For the down-sampling method, let down-sampling ratio $R \leq 100\%$ represent the ratio of preserved pixels on each width and height, e.g., $R = 25\%$ indicates that a 1024×768 image will be down-sampled into a 256×192 image. On average, this configuration is equivalent to the case where each ray of each frame has a probability of R^2 to be dropped. Similar to the analysis of the key-frame method, we can expect that the down-sampling method to increase the error for every voxel by R^{-2} times. In practice, however, the down-sampled RGB-D images are up-sampled by interpolation back to the original resolution before being used to achieve 3D reconstruction. If some part of the scene is a plane, linear interpolation can accurately restore the depth of the plane. However, on other parts of the scene, interpolation introduces a large error in the depth of the observed depth $d_i(x)$.

3.3. Pruning RGB-D data

When the agent receives the transmission policy from the server, it is used to prune the RGB-D data. Since we have represented the transmission policy with a binary image, we can simply prune each pixel of the RGB-D data

Algorithm 2: Tailoring the RGB-D data(on the agent)

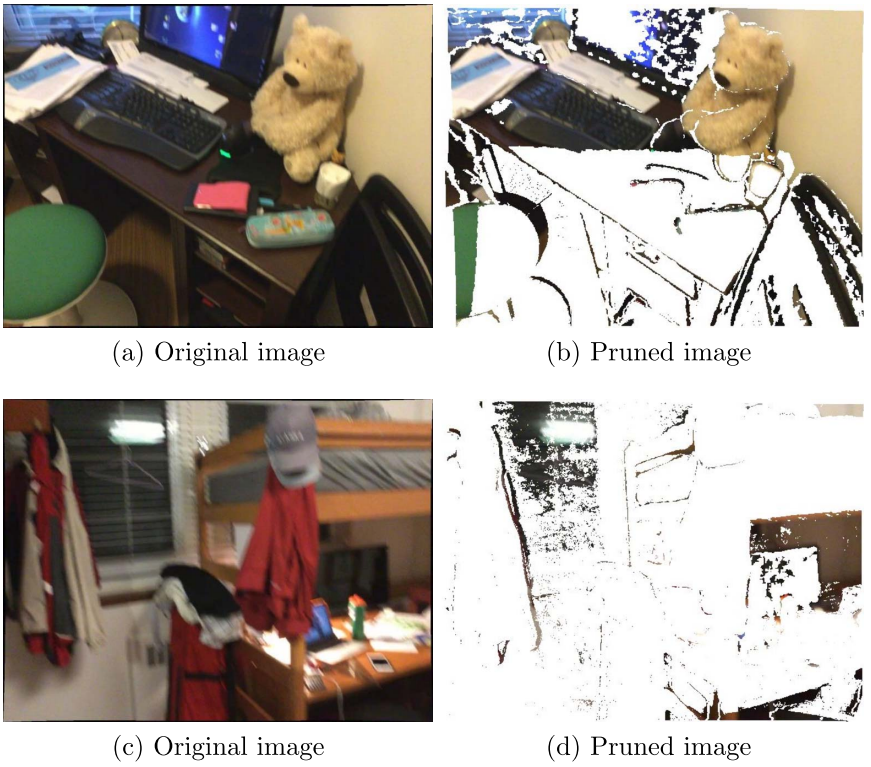
```
Input: rgb, the RGB image; depth, the depth image; trans, the
       transmission policy(a binary image).
procedure tailorRGBD(rgb, depth, trans):
    for each pixel depth[i][j] do
        if trans[i][j] is 0 then
            depth[i][j] ← 0
            rgb[i][j] ← (0, 0, 0)
        end
    end
end
```

when the corresponding pixel on the binary image is 0. The pseudo code of pruning the RGB-D data is shown in Algorithm 2.

Figure 3 shows two examples of how the pruning works. Figure 3(a) shows an RGB image in the Scannet RGB-D dataset, and Figure 3(b) shows the image after our pruning routine. At run time, the camera is moving to the upper-right. At the rays corresponding to the upper-right side of the pose, there is not enough information, and the transmission policy of those pixels is 1. Therefore, we cannot drop those pixels. In contrast, the bottom-left side of the pose has already been updated enough times, and most of the pixels are pruned. Similarly, for Figure 3(c) and (d), the camera is moving to the right, and most pixels on the left are pruned.

Before the server can achieve the reconstruction task with the pruned RGB-D data, there is an issue about how to transmit pruned RGB-D data efficiently. In our pruned RGB-D images, many pixels are pruned and replaced with depth 0 and the color black (i.e., (R, G, B) = (0, 0, 0)). We find that ordinary compression methods for RGB images and depth images work well, since connected regions of the same value can be greatly compressed. In order to quantify the effectiveness when image compression methods working on our pruned RGB-D images, we conduct a study. We generate a series of images by pruning pixels from the same image, compress them with the same PNG compression level, and measure their sizes. The result is shown in Figure 4. As we can see, as more pixels are pruned, the size of the compressed image decreases in a linear fashion. The decrease in size will directly result in a reduction in bandwidth, which is exactly what our framework needs.

Finally, after the server receives the pruned RGB-D data, it can achieve the reconstruction task. Since the depths of the pruned pixels are 0, they will not contribute to the reconstruction.



(a) Original image (b) Pruned image

(c) Original image (d) Pruned image

Figure 3: Examples of how the pruning works.(a) and (c) are RGB images before pruning. (b) and (d) are the pruned RGB images (white pixels are pruned), and the required bandwidth can be reduced about 50% and 90% respectively. For (a) and (b), the camera is moving up. For (c) and (d), the camera is moving to the right.

4. Evaluation

To show practical results, our experiments are carried out completely online. We use three datasets, Scannet RGB-D dataset [4], TUM RGB-D SLAM dataset [38] and Cow & Lady real-world RGB-D dataset [28]. We select 6 sequences of different lengths from the three datasets, because we consider them representative to different scenarios. The detail and characteristic of these 6 sequences are shown in Table 1. Each data sequence is replayed on the agent side and transmitted to the server side for 3D reconstruction. On the server side, we run the InfiniTAM framework [17, 18], a state-of-the-art 3D reconstruction framework. All software modules (i.e., 3D reconstruction

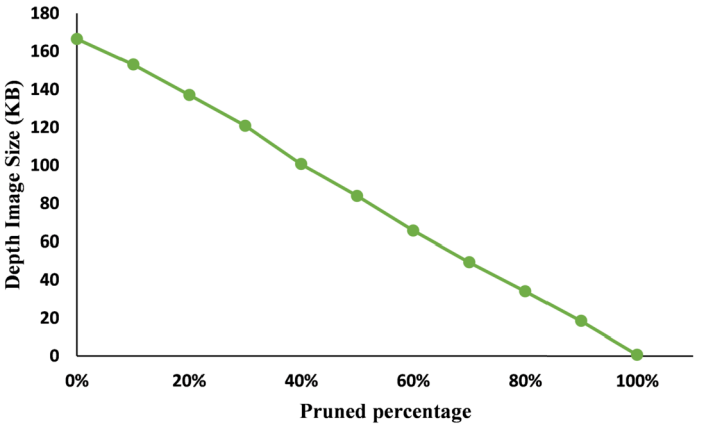


Figure 4: The effectiveness when image compression methods working on our pruned RGB-D images. As more pixels are pruned, the size of the compressed image decreases in a linear fashion.

and data sequence replaying) are connected with ROS middleware. For convenience, we use image compression algorithms employed by ROS in data transmission. The ROS version is Melodic on Ubuntu 18.04.

4.1. Comparison with mobile reconstruction methods

In the remote 3D reconstruction scenario, agents transmit the collected data to the server to utilize the powerful computing resources of the server to complete the reconstruction task and generate a reconstruction model. However, we can also use the extremely limited computing resources on agents to complete the reconstruction task. We refer this kind of solutions as mobile reconstruction methods. Voxfield [30] is the state-of-the-art TSDF-based mobile reconstruction framework, which can complete reconstruction tasks without the support of high-performance GPU.

In order to show the difference between the remote reconstruction framework and the mobile reconstruction framework, we first conduct an experiment comparing the results of Voxfield and InfiniTAM. We run the experiment multiple times under varying voxel size settings. The processing time per frame is recorded to compare the execution efficiency, and the error is calculated to compare the quality of the resulting models generated by two frameworks. The error is calculated as the reconstruction Chamfer distance [23] between the generating reconstruction models and the ground-

Table 1: Sequences used in experiments

Dataset	Sequence	Characteristic	Description
Scannet	scene0709_00	The scene is a medium kitchen.	Captured with the camera moving around a kitchen.
	scene0710_00	The scene is a room and the camera aim at nearly the same direction.	Captured with the camera translating and aiming at nearly the same direction.
	scene0717_00	The scene is large and most voxels has a small W.	Captured with the camera moving in a big college dormitory.
TUM	fr1/xyz	The scene contain people and there is information overlap between the frames of this sequence.	Recorded laboratory desks and a student sitting in a chair.
	fr1/desk	The scene doesn't contain any people and there is large overlap between the frames of this sequence.	Recorded laboratory desks without people.
Cow & Lady	cow and lady	The scene is large and there are not many effective pixels measured by the depth camera.	Recorded an indoor scene with a cow, mannequin and a few other typical office accessories.

truth model provided by datasets. The Chamfer distance is calculated between the reconstructed mesh N and the ground-truth G is:

$$d_{\mathrm{CD}}(N,G) = \frac{1}{2N}\sum_{n\in N}\min_{g\in G}\|n-g\|_2^2 + \frac{1}{2G}\sum_{g\in G}\min_{n\in N}\|g-n\|_2^2 \tag{4}$$

Table 2 shows the results. We can see that with the support of high-performance GPUs, the InfiniTAM method only takes about one percent of the processing time of Voxfield method to complete the reconstruction task with the same voxel size, and generate a 3D model with smaller error at the same time. Therefore, in order to obtain higher quality models and more efficient execution efficiency, we need the agent to transmit data to the server for 3D reconstruction. When the voxel size is small, it will be difficult to complete the reconstruction task only by relying on the agents' limited computing resources.

Table 2: Comparison between the state-of-the-art mobile 3D reconstruction framework (Voxfield with only CPU) and remote 3D reconstruction framework (InfiniTAM with GPU) under varying voxel sizes. The "V / I Ratio" column means the value of Voxfield divided by the corresponding value of InfiniTAM. With the same voxel size, the processing time of Voxfield is more than 100 times that of InfiniTAM, and the reconstruction Chamfer distance is also larger

Voxel Size	Process Time (s)			Reconstruction Chamfer Distance (m)		
	Voxfield	InfiniTAM	V / I Ratio	Voxfield	InfiniTAM	V / I Ratio
0.080	0.0887	0.000597	148.7	0.0631	0.0526	120%
0.040	0.0959	0.000572	167.5	0.0341	0.0295	115%
0.020	0.1035	0.000609	169.9	0.0223	0.0188	119%
0.010	0.1379	0.000618	223.0	0.0234	0.0222	106%
0.005	0.4244	0.000636	667.5	0.0323	0.0256	126%

4.2. Numeric comparison

The main purpose of our TSDFFilter framework is to reduce the bandwidth requirement while retaining more useful details. To measure detail retention, we use the *rostopic bw* tool on each ROS topic. The overall bandwidth requirement is the sum of transmitting the RGB-D data from the agent to the server and transmitting the transmission policy from the server to the agent. In order to measure detail retention capabilities, we measure the Chamfer distance between the resulting reconstruction models and the ground-truth model. The ground-truth model is achieved by running the same reconstruction algorithm in an off-line manner, because we would like to show the effect of our communication planning algorithm by showing the difference of its results with that of the off-line results.

There is a parameter in our TSDFFilter framework that can affect the result, i.e., the threshold W_{MAX}. We run the experiment multiple times with different values of W_{MAX} to show the influence of this value on the result. For comparison, we run the experiment with the key-frame method and the down-sampling method.

Since our TSDFFilter framework prunes the transmitted information based on its own features, it is expected that these results would differ when using different sequences. To make a fair and extensive comparison, we run the experiment with 6 sequences from 3 different datasets, shown in Table 1.

All of the results are organized into Figure 5. The figure shows the reconstruction errors that can be obtained by the above three methods under different bandwidth conditions. With the increase of W_{MAX}, our method can obtain smaller model error under the same bandwidth condition. Each

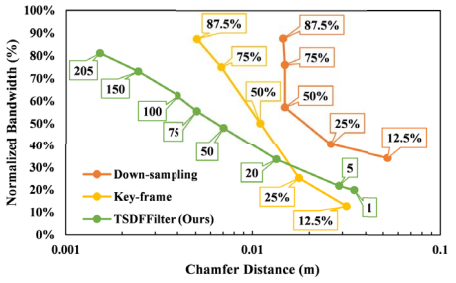
figure includes the results of handling a different sequence in three methods, including down-sampling, key-frame and our TSDFFilter. For the key-frame method, each point has different a key-frame ratio K, resulting in different bandwidths and reconstruction Chamfer distance results; for the down-sampling method, each point has different down-sampling ratio R; for our TSDFFilter framework, each point has different threshold W_{MAX}. Each sub-figure includes the normalized bandwidth requirement (the horizontal axis) and reconstruction Chamfer distance (the vertical axis). The bandwidth requirement is normalized by dividing by the bandwidth required to generate the ground-truth.

In general, with a lower key-frame ratio K, a lower down-sampling ratio R, or a lower threshold W_{MAX}, the bandwidth requirement becomes lower while the reconstruction Chamfer distance become higher. The bandwidth requirement when using the key-frame method or the down-sampling method is less affected by the content of different sequences. On the other hand, the bandwidth requirement when using our TSDFFilter framework differs when handling different sequences. The result of reconstruction Chamfer distance values verifies the analysis in Section 3 that the reconstruction Chamfer distance using our TSDFFilter is inversely proportional to W_{MAX}. In Figure 5, we show that W_{MAX} values varies from 1 to 205. Specially, we show the extreme result when the threshold $W_{\mathrm{MAX}} = 1$, which indicates a pixel will be pruned as long as it has been observed once. Another extreme result is when the threshold $W_{\mathrm{MAX}} = 255$. This threshold is so high that our TSDFFilter framework can prune little pixels. In this case, the resulting normalized bandwidth requirement is about 80%, which also indicates that the bandwidth required for transmitting the transmission policy is low. However, the resulting reconstruction Chamfer distance is not zero because of the uncontrollable randomness. For $W_{\mathrm{MAX}} = 50$ and 75, the results are comparable with the key-frame method and the down-sampling method. With similar bandwidth requirements, our TSDFFilter framework can achieve about 70% lower reconstruction Chamfer distance.

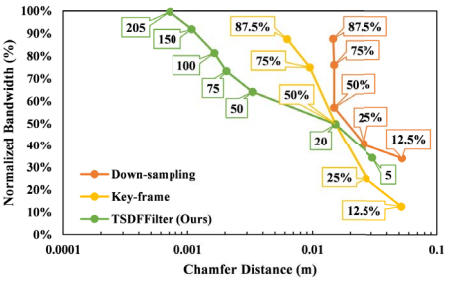
We should note that, for the *Cow & Lady* sequence, the bandwidth for our TSDFFilter framework is high even when $W_{\mathrm{MAX}} = 1$. This is probably because of the characteristics of the *Cow & Lady* sequence. In this sequence, the camera moves along without turning back, and thus each frame is valuable.

4.3. Visualized reconstruction results

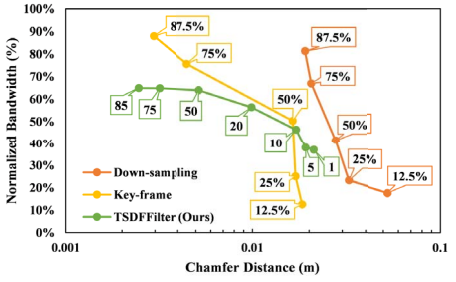
In addition to the numeric results, we also visualize an example of the results to show the advantage of our TSDFFilter framework.



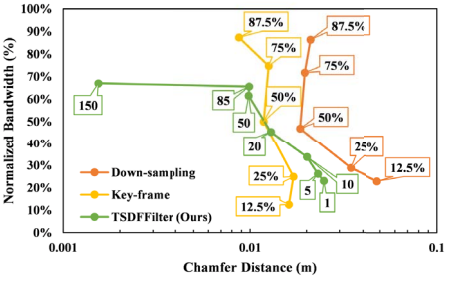
(a) TUM-fr1/xyz



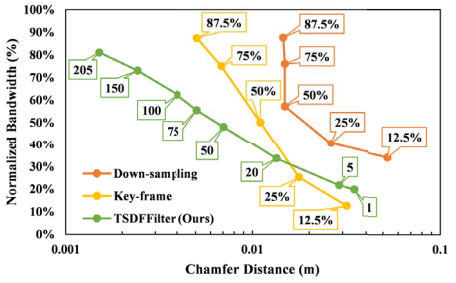
(b) TUM-fr1/desk



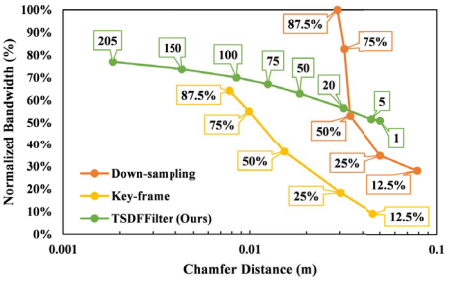
(c) Scannet-scene0709_00



(d) Scannet-scene0710_00



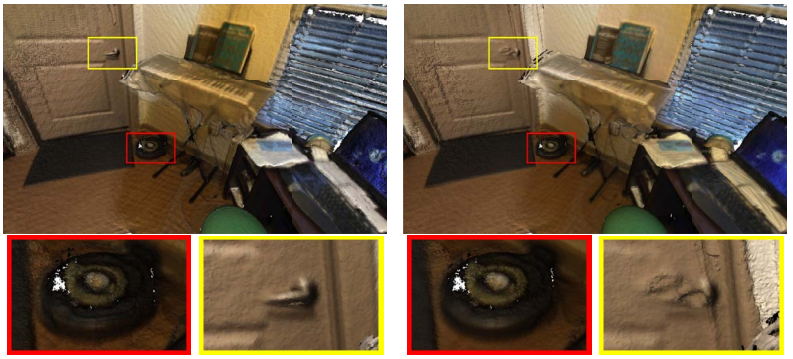
(e) Scannet-scene0714_00



(f) Cow & Lady

Figure 5: The numeric comparison results. Each figure includes the results of handling a different sequence with three method: down-sampling, key-frame and our TSDFFilter. The figure shows the bandwidth requirements that need with different the reconstruction errors using these three methods. With the increase of W_{MAX}, our method can work in less bandwidth requirement conditions with the same reconstruction errors.

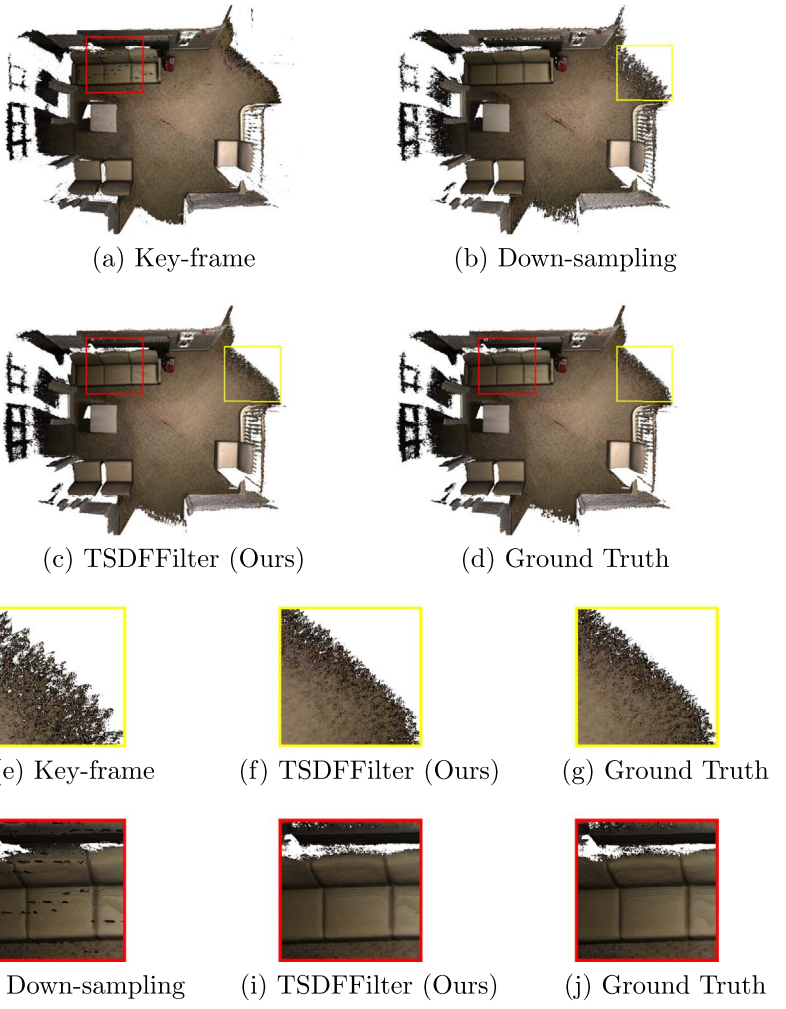
(a) Key-frame (b) Down-sampling



(c) TSDFFilter (Ours) (d) Ground Truth

Figure 6: Visualized reconstruction results of the Scannet scene0710_00 sequence. Each subfigure is generated by one of the four methods. In each subfigure, the first row illustrates higher views of the reconstructed models and the second row illustrates the detail within the yellow box and red box, which shows the detail on the door handle and sweeping robot. In the yellow boxes, we can clearly see the door handle, which indicates that our TSDFFilter framework successfully retains more details. The down-sampling method loses more detail because it has more depth values which are estimated by interpolation; the key-frame method, on the other hand, tends to smooth the result.

Figure 6 shows the visualized results of the Scannet scene0710_00 sequence. Figure 6(a) illustrates the reconstruction result with the key-frame method ($K = 0.75$); Figure 6(b) illustrates the reconstruction result with the down-sampling method ($R = 0.75$); Figure 6(c) illustrates the recon-



(a) Key-frame (b) Down-sampling

(c) TSDFFilter (Ours) (d) Ground Truth

(e) Key-frame (f) TSDFFilter (Ours) (g) Ground Truth

(h) Down-sampling (i) TSDFFilter (Ours) (j) Ground Truth

Figure 7: Visualized reconstruction results of the Scannet scene0714_00 sequence. Each subfigure is generated by one of the four methods. The first and second rows illustrate higher views of the reconstructed models. The third and fourth rows illustrate the detail within the yellow boxes and red boxes. The key-frame method loses a lot of points, and the down-sampling method produces lots of outliers.

struction result with our TSDFFilter framework ($W_{\mathrm{MAX}} = 50$); Figure 6(d) illustrates the reconstruction result with off-line manner (i.e., the ground truth). The bandwidth requirements for these sets of results are similar.

The first row of each subfigure illustrates higher views of the reconstructed models. The second row of each subfigure illustrates the detail within the yellow and red boxes, which shows the detail on the door handle and sweeping robot. From the parts within the yellow boxes, we can clearly see the door handle, which indicates that our TSDFFilter framework successfully retains more details. Interestingly, our TSDFFilter framework successfully retains more details around the door knob (shown in the yellow boxes) than the ground truth. This is because our TSDFFilter framework stops updating voxels whose confidence value is high enough, and can probably avoid these voxels being damaged by future inaccurate scans.

Figure 7 shows the visualized results of the Scannet scene0714_00 sequence. Each subfigure illustrates the reconstruction result with one of the four methods. The first and second rows illustrate the full views of the reconstructed models. The third and fourth rows illustrate the detail within the red and yellow boxes, which shows the detail on the sofa and wall. From the parts within the yellow boxes, we can clearly see our TSDFFilter framework can obtain more points compapred to the key-frame method. From the parts within the red boxes, we can clearly see the sofa, which indicates that our TSDFFilter framework successfully retains more details and the down-sampling method introduce outliners. The down-sampling method loses more detail because it has more depth values that are estimated by interpolation; the key-frame method, on the other hand, tends to smooth the result; and our TSDFFilter framework retains much of the detail.

5. Conclusion and future work

We have presented a communication planning framework for remote 3D reconstruction called TSDFFilter. In our TSDFFilter framework, agents do not blindly transmit their data but are instead able to prune their data according to the transmission policy generated by the server. To generate the transmission policy, we present the confidence value for each RGB-D pixel to estimate how much it contributes to the reconstruction quality and an efficient algorithm to generate the confidence value. Experimental results show that our TSDFFilter framework can reduce the bandwidth requirement and overcome the disadvantages of down-sampling and key-frame methods.

As far as we know, this is the first remote 3D reconstruction framework that applies feedback from the server to guide the agents how to transmit data. Besides, our TSDFFilter framework focuses on TSDF-based reconstruction method, and cannot be directly applied on semantic-aware approaches, such as [50, 15]. But it would be an interesting future work.

References

[1] Cao, Y., Kobbelt, L. and Hu, S. (2018). Real-time high-accuracy three-dimensional reconstruction with consumer RGB-D cameras. *ACM Trans. Graph.* **37** 171.

[2] Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4–9, 1996* (J. Fujii, ed.) 303–312. ACM.

[3] Dai, A., Nießner, M., Zollhöfer, M., Izadi, S. and Theobalt, C. (2017a). BundleFusion: real-time globally consistent 3D reconstruction using on-the-fly surface reintegration. *ACM Trans. Graph.* **36** 24:1–24:18.

[4] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T. and Nießner, M. (2017b). ScanNet: richly-annotated 3D reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE.*

[5] Ding, X., Wang, Y., Tang, L., Yin, H. and Xiong, R. (2019). Communication constrained cloud-based long-term visual localization in real time. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019, Macau, SAR, China, November 3–8, 2019* 2159–2166. IEEE.

[6] Dong, S., Xu, K., Zhou, Q., Tagliasacchi, A., Xin, S., Nießner, M. and Chen, B. (2019). Multi-robot collaborative dense scene reconstruction. *ACM Trans. Graph.* **38** 84:1–84:16.

[7] Duanmu, Z., Ma, K. and Wang, Z. (2017). Quality-of-experience of adaptive video streaming: exploring the space of adaptations. In *Proceedings of the 25th ACM International Conference on Multimedia* 1752–1760.

[8] Engel, J., Koltun, V. and Cremers, D. (2018). Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **40** 611–625.

[9] Forster, C., Lynen, S., Kneip, L. and Scaramuzza, D. (2013). Collaborative monocular SLAM with multiple Micro Aerial Vehicles. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3–7, 2013* 3962–3970. IEEE.

[10] Giamou, M., Khosoussi, K. and How, J. P. (2018). Talk resource-efficiently to me: optimal communication planning for distributed loop

closure detection. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21–25, 2018* 1–9. IEEE.

[11] Golodetz, S., Cavallari, T., Lord, N. A., Prisacariu, V. A., Murray, D. W. and Torr, P. H. S. (2018). Collaborative large-scale dense 3D reconstruction with online inter-agent pose optimisation. *IEEE Trans. Vis. Comput. Graph.* **24** 2895–2905.

[12] Han, B., Liu, Y. and Qian, F. (2020). ViVo: visibility-aware mobile volumetric video streaming. In *MobiCom '20: The 26th Annual International Conference on Mobile Computing and Networking, London, United Kingdom, September 21–25, 2020* 11:1–11:13. ACM.

[13] Hosseini, M. and Timmerer, C. (2018). Dynamic adaptive point cloud streaming. In *Proceedings of the 23rd Packet Video Workshop* 25–30.

[14] Huang, T., Zhang, R.-X., Zhou, C. and Sun, L. (2018). Qarc: video quality aware rate control for real-time video streaming based on deep reinforcement learning. In *Proceedings of the 26th ACM International Conference on Multimedia* 1208–1216.

[15] Huang, S.-S., Chen, H., Huang, J., Fu, H. and Hu, S.-M. (2021). Real-time globally consistent 3D reconstruction with semantic priors. *IEEE Transactions on Visualization & Computer Graphics* **01** 1–1.

[16] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R. A., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A. J. and Fitzgibbon, A. W. (2011). KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, October 16–19, 2011* (J. S. Pierce, M. Agrawala and S. R. Klemmer, eds.) 559–568. ACM.

[17] Kähler, O., Prisacariu, V. A., Ren, C. Y., Sun, X., Torr, P. H. S. and Murray, D. W. (2015). Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Trans. Vis. Comput. Graph.* **21** 1241–1250.

[18] Kähler, O., Prisacariu, V. A., Valentin, J. P. C. and Murray, D. W. (2016). Hierarchical voxel block hashing for efficient integration of depth images. *IEEE Robotics and Automation Letters* **1** 192–197.

[19] Lawrence, J., Goldman, D. B., Achar, S., Blascovich, G. M., Desloge, J. G., Fortes, T., Gomez, E. M., Häberling, S., Hoppe, H.,

Huibers, A., Knaus, C., Kuschak, B., Martin-Brualla, R., Nover, H., Russell, A. I., Seitz, S. M. and Tong, K. (2021). Project starline: a high-fidelity telepresence system. *ACM Trans. Graph.* **40** 242:1–242:16.

[20] Li, A. Q., Cheung, W., Kawiak, R., Robbins, D., Chen, M., Quesada, P., Tran, T., Juang, J. and Jiang, C. (2019). An investigation of Volumetric VOD streaming compression technique.

[21] Lu, J., Benko, H. and Wilson, A. D. (2017). Hybrid HFR depth: fusing commodity depth and color cameras to achieve high frame rate, low latency depth camera interactions. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06–11, 2017* (G. Mark, S. R. Fussell, C. Lampe, Schraefel, J. P. Hourcade, C. Appert and D. Wigdor, eds.) 5966–5975. ACM.

[22] Mao, H., Netravali, R. and Alizadeh, M. (2017). Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21–25, 2017* 197–210. ACM.

[23] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S. and Geiger, A. (2019). Occupancy networks: learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* 4460–4470.

[24] Mohanarajah, G., Usenko, V., Singh, M., D'Andrea, R. and Waibel, M. (2015). Cloud-based collaborative 3D mapping in real-time with low-cost robots. *IEEE Trans Autom. Sci. Eng.* **12** 423–431.

[25] Mur-Artal, R. and Tardós, J. D. (2017). ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Trans. Robotics* **33** 1255–1262.

[26] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S. and Fitzgibbon, A. W. (2011). KinectFusion: real-time dense surface mapping and tracking. In *10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011, Basel, Switzerland, October 26–29, 2011* 127–136.

[27] Nießner, M., Zollhöfer, M., Izadi, S. and Stamminger, M. (2013). Real-time 3D reconstruction at scale using voxel hashing. *ACM Trans. Graph.* **32** 169:1–169:11.

[28] Oleynikova, H., Taylor, Z., Fehr, M., Siegwart, R. and Nieto, J. (2017). Voxblox: incremental 3D Euclidean signed distance fields for on-board

MAV planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[29] Opdenbosch, D. V., Oelsch, M., Garcea, A., Aykut, T. and Steinbach, E. G. (2018). Selection and compression of local binary features for remote visual SLAM. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21–25, 2018* 7270–7277. IEEE.

[30] Pan, Y., Kompis, Y., Bartolomei, L., Mascaro, R., Stachniss, C. and Chli, M. (2022). Voxfield: non-projective signed distance fields for online planning and 3D reconstruction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022)*.

[31] Petrangeli, S., Famaey, J., Claeys, M., Latré, S. and Turck, F. D. (2016). QoE-driven rate adaptation heuristic for fair adaptive video streaming. *TOMM* **12** 28:1–28:24.

[32] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R. and Ng, A. (2009). ROS: an open-source robot operating system. In *IEEE International Conference on Robotics and Automation, ICRA, Workshop on Open Source Software, Kobe, Japan*.

[33] Saeedi, S., Trentini, M., Seto, M. and Li, H. (2016). Multiple-robot simultaneous localization and mapping: a review. *J. Field Robotics* **33** 3–46.

[34] Schmuck, P. (2017). Multi-UAV collaborative monocular SLAM. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29–June 3, 2017* 3863–3870. IEEE.

[35] Sengupta, S., Ganguly, N., Chakraborty, S. and De, P. (2018). HotDASH: hotspot aware adaptive video streaming using deep reinforcement learning. In *2018 IEEE 26th International Conference on Network Protocols, ICNP 2018, Cambridge, UK, September 25–27, 2018* 165–175. IEEE Computer Society.

[36] Shi, H., Wang, Z., Lv, J., Wang, Y., Zhang, P., Zhu, F. and Li, Q. (2021). Semi-supervised learning via improved teacher-student network for robust 3D reconstruction of stereo endoscopic image. In *MM '21: ACM Multimedia Conference, Virtual Event, China, October 20–24, 2021* (H. T. Shen, Y. Zhuang, J. R. Smith, Y. Yang, P. Cesar, F. Metze and B. Prabhakaran, eds.) 4661–4669. ACM. MR4466222

[37] Stockhammer, T. (2011). Dynamic adaptive streaming over HTTP –: standards and design principles. In *Proceedings of the Second Annual ACM SIGMM Conference on Multimedia Systems, MMSys 2011, Santa Clara, CA, USA, February 23–25, 2011* (A. C. Begen and K. Mayer-Patel, eds.) 133–144. ACM.

[38] Sturm, J., Engelhard, N., Endres, F., Burgard, W. and Cremers, D. (2012). A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*.

[39] Takikawa, T., Litalien, J., Yin, K., Kreis, K., Loop, C., Nowrouzezahrai, D., Jacobson, A., McGuire, M. and Fidler, S. (2021). Neural geometric level of detail: real-time rendering with implicit 3D shapes. *CoRR* **abs/2101.10994**.

[40] Tang, Z., Feng, X., Xie, Y., Phan, H., Guo, T., Yuan, B. and Wei, S. (2020). VVSec: securing volumetric video streaming via benign use of adversarial perturbation. In *MM '20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October 12–16, 2020* (C. W. Chen, R. Cucchiara, X. Hua, G. Qi, E. Ricci, Z. Zhang and R. Zimmermann, eds.) 3614–3623. ACM.

[41] van der Hooft, J., Wauters, T., Turck, F. D., Timmerer, C. and Hellwagner, H. (2019a). Towards 6DoF http adaptive streaming through point cloud compression. In *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21–25, 2019* (L. Amsaleg, B. Huet, M. A. Larson, G. Gravier, H. Hung, C. Ngo and W. T. Ooi, eds.) 2405–2413. ACM.

[42] van der Hooft, J., Wauters, T., De Turck, F., Timmerer, C. and Hellwagner, H. (2019b). Towards 6dof http adaptive streaming through point cloud compression. In *Proceedings of the 27th ACM International Conference on Multimedia* 2405–2413.

[43] Wang, Y., Lu, Y., Xie, Z. and Lu, G. (2021a). Deep unsupervised 3D SfM face reconstruction based on massive landmark bundle adjustment. In *MM '21: ACM Multimedia Conference, Virtual Event, China, October 20–24, 2021* (H. T. Shen, Y. Zhuang, J. R. Smith, Y. Yang, P. Cesar, F. Metze and B. Prabhakaran, eds.) 1350–1358. ACM.

[44] Wang, Y., Zou, Z., Wang, C., Dong, Y., Qiao, L. and Manocha, D. (2021b). ORBBuf: a robust buffering method for remote visual SLAM.

In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27–Oct. 1, 2021* 8706–8713. IEEE.

[45] Wang, Y., Wang, W., Liu, D., Jin, X., Jiang, J. and Chen, K. (2022). Enabling edge-cloud video analytics for robotics applications. *IEEE Transactions on Cloud Computing*.

[46] Werner, D., Al-Hamadi, A. and Werner, P. (2014). Truncated signed distance function: experiments on voxel size. In *Image Analysis and Recognition – 11th International Conference, ICIAR 2014, Vilamoura, Portugal, October 22–24, 2014, Proceedings, Part II* (A. J. C. Campilho and M. S. Kamel, eds.). *Lecture Notes in Computer Science* **8815** 357–364. Springer. MR3295522

[47] Yeo, H., Jung, Y., Kim, J., Shin, J. and Han, D. (2018). Neural adaptive content-aware internet video delivery. In *13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8–10, 2018* (A. C. Arpaci-Dusseau and G. Voelker, eds.) 645–661. USENIX Association.

[48] Zhang, B., Jin, X., Ratnasamy, S., Wawrzynek, J. and Lee, E. A. (2018). Awstream: adaptive wide-area streaming analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* 236–252.

[49] Zhang, H., Wang, G., Lei, Z. and Hwang, J. (2019). Eye in the sky: drone-based object tracking and 3D localization. In *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21–25, 2019* (L. Amsaleg, B. Huet, M. A. Larson, G. Gravier, H. Hung, C. Ngo and W. T. Ooi, eds.) 899–907. ACM.

[50] Zheng, T., Zhang, G., Han, L., Xu, L. and Fang, L. (2020). Building fusion: semantic-aware structural building-scale 3D reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xu-Qiang Hu
Department of Computer Science and Technology
Tsinghua University
Haidian District, Beijing
China
E-mail address: huxq@outlook.com

Yu-Ping Wang
School of Computer Science and Technology
The Beijing Institute of Technology
Haidian District, Beijing
China
E-mail address: wyp_cs@bit.edu.cn

Zi-Xin Zou
Department of Computer Science and Technology
Tsinghua University
Haidian District, Beijing
China
E-mail address: zouzx19@mails.tsinghua.edu.cn

Dinesh Manocha
Department of Computer Science
University of Maryland
College Park, MD 20742
USA
E-mail address: dmanocha@gmail.com

Received January 20, 2023