Proxy Clouds for Live RGB-D Stream Processing and Consolidation

Adrien Kaiser[1](✉), Jose Alonso Ybanez Zepeda[2], and Tamy Boubekeur[1]

[1] LTCI, Telecom ParisTech, Paris-Saclay University, Paris, France
{adrien.kaiser,tamy.boubekeur}@telecom-paristech.fr
[2] Ayotle, Le Kremlin Bicetre, France
alonso@hayo.io

Abstract. We propose a new multiplanar superstructure for unified real-time processing of RGB-D data. Modern RGB-D sensors are widely used for indoor 3D capture, with applications ranging from modeling to robotics, through augmented reality. Nevertheless, their use is limited by their low resolution, with frames often corrupted with noise, missing data and temporal inconsistencies. Our approach, named *Proxy Clouds*, consists in generating and updating through time a single set of compact local statistics parameterized over detected planar proxies, which are fed from raw RGB-D data. *Proxy Clouds* provide several processing primitives, which improve the quality of the RGB-D stream on-the-fly or lighten further operations. Experimental results confirm that our light weight analysis framework copes well with embedded execution as well as moderate memory and computational capabilities compared to state-of-the-art methods. Processing of RGB-D data with *Proxy Clouds* includes noise and temporal flickering removal, hole filling and resampling. As a substitute of the observed scene, our *proxy cloud* can additionally be applied to compression and scene reconstruction. We present experiments performed with our framework in indoor scenes of different natures within a recent open RGB-D dataset.

Keywords: RGB-D stream · 3D geometric primitives
Data reinforcement · Depth improvement · Online processing
Scene reconstruction

1 Introduction

Context. The real time RGB-D stream output of modern commodity consumer depth cameras can feed a growing set of end applications, from human computer interaction and augmented reality to industrial design. Although such devices are constantly improving, the limited quality of their stream still restraints their impact spectrum. This mostly originates in the low resolution of the frames and

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01231-1_16) contains supplementary material, which is available to authorized users.

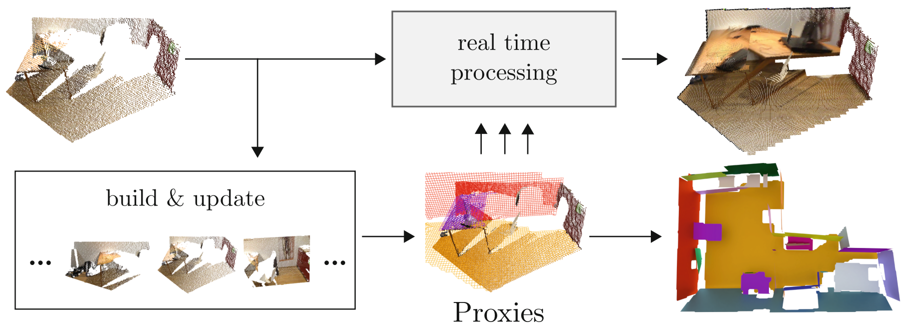

Fig. 1. Proxy clouds overview. From a stream of RGB-D frames (left), proxies are built on-the-fly and updated over time (bottom) and used to apply different real-time processing primitives to the incoming RGB-D frames (top). The system outputs an enhanced data stream and a planar model of the observed scene (right). The "build & update" procedure is detailed in Fig. 4.

the inherent noise, incompleteness and temporal inconsistency stemming from single view capture. With *Proxy Clouds*, we aim at improving real time RGB-D streams by analyzing them. A sparse set of detected 3D planes are parameterized to record statistics extracted from the stream and form a structure that we call *proxy*. This superstructure substitutes the RGB-D data and approximates the geometry of the scene. Using these time-evolving statistics, our plane-based framework improves the RGB-D stream on the fly by reinforcing features, removing noise and outliers or filling missing parts, under the memory-limited and real time embedded constraints of mobile capture in indoor environments (Fig. 1). We design such a lightweight planar superstructure to be stable through time and space, which gives priors to apply several signal-inspired processing primitives to the RGB-D frames. They include filtering to remove noise and temporal flickering, hole filling or resampling (Sect. 4). This allows structuring the data and simplifying or lightening subsequent operations, e.g. tracking and mapping, measurements, data transmission, rendering or physical simulations. While our primary goal is the enhancement of the RGB-D data stream, our framework can additionally be applied to compression and scene reconstruction (Sect. 6), as the generated structure is a representation of the observed scene.

Overview. In practice, our system takes a raw RGB-D stream as input to build and update a set of planar proxies on-the-fly. It outputs an enhanced stream together with a model of planar areas in the observed scene (see Fig. 2). On the contrary to previous approaches, which mostly rely on a full volumetric reconstruction to consolidate data, our approach is light weight, with a moderate memory footprint and a transparent interfacing to any higher-level RGB-D pipeline. To summarize, our contributions are:

- a stable and lightweight multiplanar superstructure for RGB-D data,
- construction and updating methods which are spatially and temporally consistent,
- a collection of RGB-D enhancement methods based on our structure which run on-the-fly.

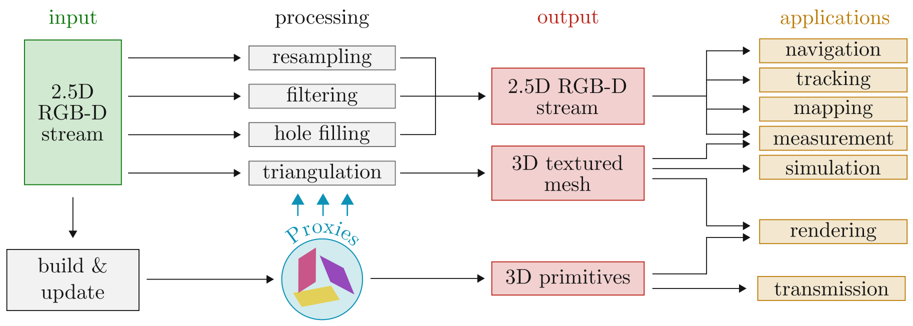

Fig. 2. Proxy clouds workflow. From a stream of 2.5D RGB-D frames, proxies are built on-the-fly and updated through time (Sect. 3). They are used as priors to process incoming frames through filtering, resampling or hole filling which allows better tracking, mapping, automated navigation or measurement. A selection of proxies based on the current RGB-D frame can be used for lightened transmission of the data. Proxies can be used as priors for triangulation and fast depth data meshing, with application to rendering or simulation. Eventually, the consolidation of the depth stream within the proxies leads to a reconstruction of planar areas in the observed scene. Processing modules are described in Sect. 4 and the consolidation of data is presented in Sect. 6.

2 Previous Work

Camera Motion Estimation. Endres et al. [1] describe an egomotion estimation method that uses point features detected in the color component of the RGB-D frame. After detecting and matching SIFT, SURF or ORB features in subsequent color images, their 3D position in both frames is computed using the depth component. Using these matching 3D points, a robust RANSAC-based [2] estimation of the motion matrix allows discarding false positive matches. Sets of three matching points are randomly picked and the matrix transforming a set in the first frame into the second set is computed using a Least-Squares method [3]. Inliers of the transformation are estimated using their 3D position and orientation and the one giving the most inliers is kept. It is important to note that any existing method or device that localizes an RGB-D camera in its environment can be used instead. Some of them are presented in Sect. 2.

Plane Detection in RGB-D Stream. Methods that build high level models of captured 3D data are mostly based on *RANSAC* [2], the *Hough transform* [4] or *Region Growing* algorithms. In our embedded, real time, memory-limited context, we take inspiration from the RANSAC-based method of Schnabel et al. [5] for its time and memory efficiency, by repeating plane detection through time to acquire a consistent model and cope with the stochastic nature of RANSAC.

Their *Efficient RANSAC* implementation gives stochastic improvements to the critical steps of the algorithm in terms of complexity. For a regular RANSAC-based plane detection, minimal sets of three points would be randomly picked a fixed and large number of times. Then, the shape parameters are estimated from this minimal set and inliers of the estimated plane are computed. The shape with the highest score is kept, its inliers are removed from the point cloud and the algorithm is ran again on the remaining data. Schnabel et al. replace the fixed number of loops with a stochastic condition, based on the number of detected shapes and number of randomly picked minimal sets, to stop looking for planes in the dataset. Also, instead of searching the full point cloud for inliers of a given shape, they estimate this count in a random subset of the dataset and extrapolate it to the full point cloud. Other modifications allow improving the quality of detected shapes with a localized sampling and specific post-processing.

Again, our framework is not attached to a particular plane detection method, and other algorithms such as *point clustering* [6] or *agglomerative hierarchical clustering* [7], could be used. For a complete overview of plane detection methods in captured 3D data, we refer the reader to our survey [8].

Depth Processing. Depth maps can be denoised using spatial filters [9] e.g., Gaussian, median, bilateral [10–12], adaptive or anisotropic [13,14] filters, often refined through time, with the resulting enhanced stream potentially used for a full 3D reconstruction [15]. Other methods include non-local means [16], bilateral filters with time [12], Kalman filters [17], over-segmentation [18] and region-growing [19]. Wu et al. [20] present a shape-from-shading method using the color component to improve the geometry, which allows adding details to the low quality input depth. They show applications of their method to improve volumetric reconstruction on multiple small scale and close range scenes. Depth maps can be upsampled using cross bilateral filters such as *joint bilateral upsampling* [21] or *weighted mode filtering* [22]. Such methods are particularly useful to recover sharp depth regions boundaries and enforce depth-based segmentation.

Hole Filling. Depth sensing range limits and high noise levels often create holes in RGB-D data. Given the material of observed objects and the type of technology used, e.g. *time of flight*, *light coding* or *stereo vision*, some surfaces are harder to detect. The orientation of the surface with regards to the sensor and the perturbations due to light sources can also lower the quality of certain areas. In order to fill these holes in the depth component, one can use the same spatial filters as those used for denoising [16], or morphological filters [13,14].

Inpainting methods [23], over-segmentation [24] or multiscale [25] processing are also used to fill holes for e.g., *depth image-based rendering* (DIBR) under close viewing conditions.

Plane-Based Depth Processing. A set of 3D planes offers a faithful yet lightweight approximation for many indoor environments. Surprisingly, only a few methods have used planar proxies as priors to process 2.5D data, with in particular Schnabel et al. [26] who detect limits of planes to fill in holes in static 3D point clouds. *Fast Sampling Plane Filtering* [27] detects and merges planar patches in static indoor scenes. The detected planes allow filtering the planar surfaces of the input point cloud, however the primitives seem quite sensitive to the depth sensor noise and lack spatial consistency.

Dense SLAM. Online dense *Simultaneous Localization And Mapping (SLAM)* methods accumulate points within a map of the environment, while continuously localizing the sensor in this map. Recent dense SLAM systems include *RGB-D SLAM* [1], *RTAB-Map* [28], *KDP SLAM* [29] or *ORB-SLAM2* [30]. Point-based fusion [31] is also used to accumulate points without the need of a full volumetric representation. Several methods have been developed to include planar primitives in the SLAM system, either to smooth and improve the reconstruction [32,33] or improve the localization of the sensor [34–36]. Finally, a recent offline method [37] makes use of planes to estimate the geometry of a room in order to remove furnitures and model the lighting of the environment. This allows the user to re-light and re-furnish the room as desired.

Volumetric Depth Fusion. Online scene reconstruction methods using volumetric fusion were pioneered by *KinectFusion* [15], then made more efficient with *VoxelHashing* [38], and more accurate with *BundleFusion* [39]. However, the need for a voxel grid representing the space leads to high requirements of memory. Recent algorithms make use of planes to smooth and complete the data within the volume, such as methods by Zhang et al. [40] or Dzitsiuk et al. [41]. Offline improvement methods have been developed based on the volumetric representation of the scene, such as *3DLite* [42] that builds a planar model of the observed scene and optimizes it to achieve a high quality texturing of the surfaces.

3 Proxy Clouds

Model. Basically, *Proxy Clouds* model RGB-D data which is often seen and consistent through frames and space, hence revealing the dominant structural elements in the scene. To do so, they take the form of a multiplanar superstructure, where each proxy is equipped with a local frame, bounds and, within the bounds, a regular 2D grid of rich statistics, mapped on the plane and gathered from the RGB-D data.

Each cell of the grid includes an occupancy probability as well as a statistical model of the depth values. We choose to represent this local distribution using *smoothed local histograms* [43] made of Gaussian kernels. The contribution of an inlier p of distance $d(p)$ to the proxy is given in Eq. 1.

$$h_p(s) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(s-d(p))^2}{2\sigma^2}} \qquad h'_p(s) = -\frac{s-d(p)}{\sigma^2} h_p(s) \tag{1}$$

This compressed model stores the repartition of plane inliers distances to the proxy and makes possible estimating the diversity of the values within each cell by counting the number of modes in the distribution. If it has a single mode, then all values are similar and the surface of the proxy within the cell is most likely flat. If the distribution has two or more modes, then the values belong to different groups and the cell likely overlaps a salient area of the surface.

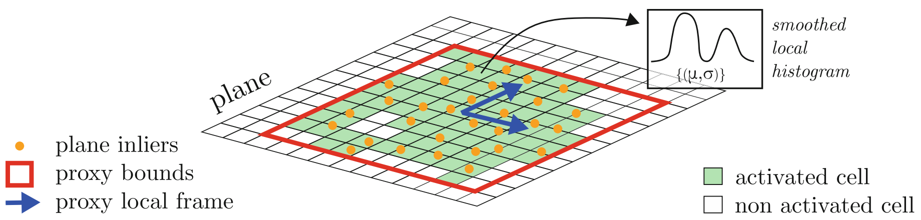

Fig. 3. Planar proxy model. Built upon a plane in 3D space, our proxy model is made of a local frame, bounds and a grid of cells which contain statistics. These statistics are the occupancy probability as well as a collection of mean μ and variance σ values for depth, representing a *smoothed local histogram* and gathered from the RGB-D data. Activated cells are the ones containing inliers from many frames.

Cells have a fixed size of 5 cm × 5 cm, which corresponds to about four times the area of a depth pixel at a typical distance of 8 meters[1]. Hence, this size ensures a minimum sampling of proxy cells by depth points even at far capture distances.

Cells are activated when their visitation percentage over the recent frames (the last 100 frames in our experiments) is greater than a threshold (25% in practice). Once activated, a cell stays so until the end of the processing. We consider a cell as visited as soon as it admits at least one inlier data point i.e., a data point located within a threshold distance to the cell under a projection in the direction from the sensor origin to the point. This activation threshold allows modeling the actual geometry of the observed scene, while discarding outliers observations due to the low quality of the sensor. Figure 3 gives visual insight of a planar proxy.

Building Proxies. We build planar proxies on-the-fly and update them through time using solely incoming raw RGB-D frames from the live stream. More precisely, for each new RGB-D image $X_t = \{I_t, D_t\}$ (color and depth), we run the procedure described in Algorithm 1 and Fig. 4.

The initial depth filtering (step 1) is based on a bilateral convolution [10] of the depth map using a Gaussian kernel associated with a range check to discard points further than a depth threshold from the current point, which could create artificial depth values if taken into account. In our experiments, we choose to set this threshold to 20 cm, which allows filtering together parts of the same object, while ignoring the influence of unrelated objects. Due to the embedded processing constraint, we estimate the normal field (step 2) through the simple computation of the depth gradient at each pixel, using the sensor topology as domain. The estimation of the camera motion from the previous frame (step 3) is inspired from the method introduced by Endres et al. [1], using point features from I_t. As previously stated, any egomotion estimation algorithm can be used at this step, as all we need is the values of the six degrees

[1] The area of a pixel at given depth Z is given by $a(Z) = tan(\frac{fov_H}{res_H})tan(\frac{fov_V}{res_V})Z^2$. With $fov = (60°, 45°)$ and $res = (320, 240)$, we have $a(8\,\text{m}) \approx 0.00068539\,\text{m}^2 \approx (2.6\,\text{cm})^2$.

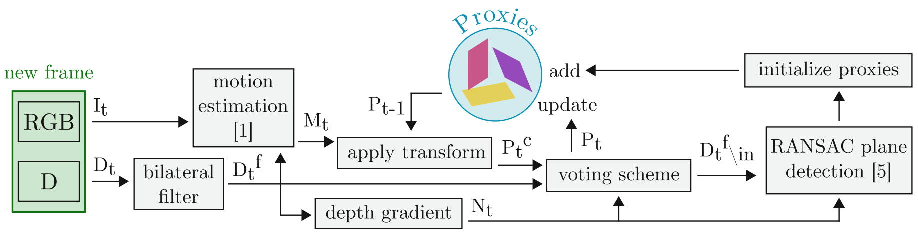

Fig. 4. Building proxies. This procedure is ran for each new RGB-D image X_t at frame t, made of a color image I_t and a depth map D_t. D_t^f: low-pass filtered version of D_t; M_t: camera motion matrix; N_t: normal vectors associated with D_t; P_{t-1}: proxies detected at frame $t-1$; P_t^c: candidate proxies; P_t: proxies at frame t; $D_t^f{}_{|in}$: low-pass filtered depth points without inliers from P_t;

Algorithm 1. Building Proxies

```
Notations: $X_t$: current RGB-D frame; $I_t$: current RGB frame; $D_t$: current depth
frame; $P_t$: current proxies;

$P_t \leftarrow \varnothing$
function BuildProxies($X_t = \{I_t, D_t\}$)
    1. filter $D_t$ with a bilateral color/depth convolution;
    2. estimate the normal field $N_t$ from $D_t$;
    3. estimate the camera motion $M_t$ from $X_{t-1}$ [1];
    4. search for previous proxies in $X_t$:
        4.1 register previous frame proxies to $X_t$ using $M_t$;
        4.2 cast votes from samples of $X_t$ to previous proxies;
        4.3 given the vote count for each previous proxy:
            keep it, update it with $X_t$ and add it to $P_t$;
            discard it and place it in probation state;
            purge it if it has been discarded for too long.
    5. detect new proxy planes in $X_t \setminus inliers(P_t)$:
        5.1 RANSAC-based plane detection [5];
        5.2 post-detection plane fusion;
        5.3 compute the local frame;
        5.4 initialize the new proxy with $X_t$.
        5.5 register new proxy to global space using $M_t$.
    return $P_t$
end function
```

of freedom modeling the camera motion. Examples of such algorithms are given in Sect. 2. In order to keep or discard previously detected proxies (step 4.2), we define a voting scheme where samples of X_t which are inliers of a given previous proxy cast their vote to this proxy and are marked. Then, the per-proxy vote count indicates whether the proxy is preserved or discarded (step 4.3). Preserved proxies are updated with X_t, hence see their parameters refined and occupancy statistics updated with new inliers. Discarded proxies are placed in *probation*

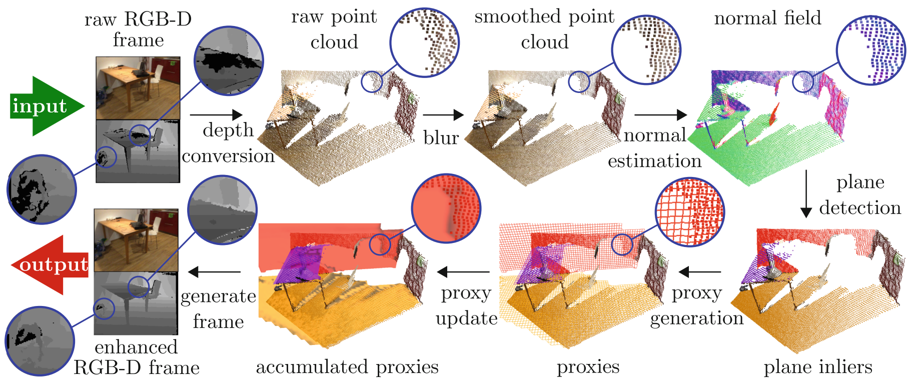

Fig. 5. Proxy building procedure. The input RGB-D frame is converted into a raw point cloud on which a low-pass filter is applied, followed by normal estimation (top). RANSAC-based plane detection [5] is applied and used as a basis for the construction or the update of our proxies, following the structure shown in Fig. 3 (bottom right). Accumulated proxy cells (bottom) are visualized with colors weighted by their occupancy probabilities: darker cells have a low probability and represent low confidence areas, whereas brighter cells represent areas with high confidence. Proxies are then used to generate enhanced RGB-D frames. The whole process runs on-the-fly.

state for near-future recheck with new incoming frames, and purged if discarded for too long. However, in order to avoid losing information on non-observed parts of the scene, we do not purge proxies that have been seen long enough, which stay in probation instead. When new proxies have been detected (step 5.1), similar ones are merged together in order to avoid modeling different parts of planar surfaces with multiple proxy instances (step 5.2). The proxy is then generated (step 5.3) with a bounding rectangle and a local frame computed to be aligned with the scene orientation (more details in the supplemental material). Using the global scene axes to compute the local frame leads to a fixed resolution and spatial consistency for the grid of all proxies and allows efficient recovery and fusion (step 5.2). Finally, occupancy statistics are initialized using X_t (step 5.4). In order to take into account the point of view when modeling the scene, inlier depth points are projected upon the detected shape following the direction between the camera and the point. The coordinates of the corresponding cell of the proxy grid are then recovered to update its statistics. As a last step, proxies are transformed from local depth frame into global 3D space in order to be tracked in the next frames (step 5.5).

Figure 5 shows the different steps of the construction of the proxy cloud on one specific example.

4 RGB-D Stream Processing

Our *Proxy Cloud* allows online recovery of the underlying structure of piecewise planar scenes (e.g., indoor) and is used as a prior to run on-the-fly processing on the incoming RGB-D frames. The different processing modules applied to the frames are presented below.

Filtering. While projecting the sensor's data points onto their associated proxy would allow removing the acquisition noise and quantization errors due to the sensor, this would lead to the flattening of all plane inliers. In order to minimize the loss of details on the planar surfaces while keeping a lightweight data structure, we instead use the planar proxies as a simple collaborative filter model. To that end, we designed a custom filter to leverage the *smoothed local histograms* stored in each cell of the proxies. As explained in Sect. 3, the number of detected modes allows distinguishing flat areas of the proxy surface from salient ones. For flat cells whose distribution has a single mode, we project the depth points on the plane along the direction between the camera and the point. We offset the points of the average distance to the plane only if it is above the noise threshold at the corresponding distance to the camera (see details on the noise threshold in the supplemental material). This allows smoothing surface areas that are exactly on the plane while keeping flat areas offset from the plane as they are in the scene. For flat cells whose distribution has two or more modes, we do not perform any projection in order to keep the saliency of the surface.

Equation 2 details the *smoothed local histograms*-based filtering of inlier $\mathbf{p}$ to $\mathbf{p}_f$, belonging to cell c with m_c modes and an average distance to the proxy of d_c, and a noise threshold of α. The proxy is based on a plane of normal $\mathbf{N}$ and distance to origin l.

$$\mathbf{p}_f = \begin{cases} \frac{l}{\mathbf{p}.\mathbf{N}}\mathbf{p} & \text{if } m_c = 1 \text{ and } d_c \leq \alpha \\ \frac{l}{\mathbf{p}.\mathbf{N}}\mathbf{p} + d_c\mathbf{N} & \text{if } m_c = 1 \text{ and } d_c > \alpha \\ \mathbf{p} & \text{if } m_c \neq 1 \end{cases} \tag{2}$$

The proxy can also be used as a high level range space for cross bilateral filtering [21], where inliers of different proxies will not be processed together.

Based on time-evolving data points, the proxies consolidate the stable geometry of the scene by accumulating observations in multiple frames. Averaging those observations over time removes temporal flickering, after a few frames only.

Hole Filling. Missing data in depth is often due to specular and transparent surfaces such as glass or screens. With our *Proxy Cloud*, observed data is reinforced over multiple frames from the support of stable proxies, augmenting the current frame with missing samples from previous ones. In practice, the depth data that is often seen in incoming frames creates activated cells with sufficient occupancy probability to survive within the model even when samples for these cells are missing.

This hole filling, stemming naturally from the proxy structure, is completed by two additional steps. First, the extent of the proxies is extrapolated to the intersection of adjacent proxies - this is particularly useful to complete unseen areas under furniture for example. Second, we perform a morphological closing [44] on the grid of cells, with a square structural element having a fixed side of seven cells. This corresponds to closing holes of maximum 35 cm by 35 cm, which allows filling missing data due to small specular surfaces, e.g. computer screens or glass-door cabinets, while keeping larger openings such as windows or doors.

Resampling. RGB-D streams can be *super-sampled* on the fly, by enriching their low definition geometric component using the higher resolution color component structured in the proxies to guide the process. This results in high definition RGB-D data with controllable point density on the surface of the shapes.

Lossy Compression. Compression of the input data is achieved by using directly the proxy cloud as a compressed, lightweight geometric substitute to the huge amount of depth data carried in the stream, avoiding storing uncertain and highly noisy depth regions, while still being able to upsample back to high resolution depth using a bilateral upsampling. In particular, this is convenient to broadcast captures of indoor scenes where planar regions are frequent.

5 Experiments

Proxy Clouds are implemented through hardware and software components. The hardware setup is made of a computer with Intel Core i7 at 3.5 GHz and 10 GB memory. No GPU is used. The software setup has a client-server architecture, where the server runs in a single thread within an embedded environment with low computational power and limited memory to trigger the sensor and process the data. The client's graphical user interface allows controlling the processing parameters and getting a real time feedback of the stream. A limited range of intuitive parameters allow the user to control the trade-off between quality of the output and performance of the processing.

We run all of our experiments on the *3DLite* [42] dataset[2], containing 10 scenes acquired with a Structure sensor[3] under the form of RGB-D image sequences. This choice was motivated by the availability of ground truth poses along with the visual data, as well as result meshes and performance metrics provided from processing with both *BundleFusion* [39] and *3DLite* [42], with which we compare our method in Sect. 6.

Geometric statistics on the generated proxies are available as supplemental material for all processed scenes (Table 1). We also provide in Fig. 1 of the supplemental material, a plot of the increment of the average depth over time, to show the fast convergence of the proxy statistics after about 30 accumulated samples. The accuracy of the proxy representation can be quantitatively assessed through the PSNR values in Table 1.

Performance. The current time required to build and update planar proxies using our (single-threaded) implementation is around 130 ms for an input depth image of 320×240 pixels. A detailed graph presenting the time needed for all steps is available as supplemental material (Fig. 2).

[2] 3DLite dataset: http://graphics.stanford.edu/projects/3dlite/#data.
[3] Structure sensor: http://structure.io.

Live RGB-D Stream Processing. Figure 6 shows examples of data improvement using the processing modules of our framework. Experiments show that the *Proxy Cloud* is particularly efficient to remove noise over walls and floors while keeping salient parts, and helps reducing holes due to unseen areas, specular areas, such as lights or glass, and low confidence areas, such as distant points. Resampling the point cloud allows recovering structure if the sensor did not give enough data samples, e.g. on lateral planar surfaces. More examples of RGB-D stream improvement on other scenes are available as supplemental material.

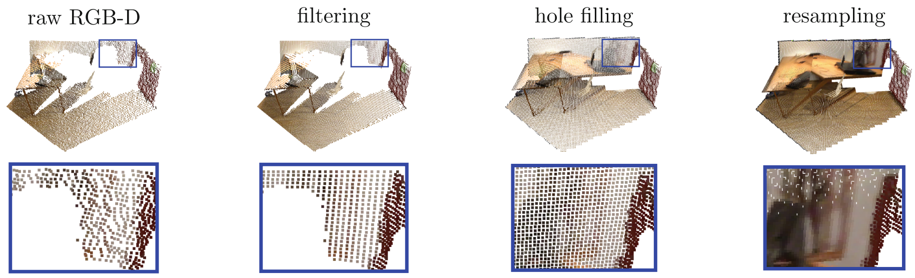

Fig. 6. Data improvement. Raw RGB-D and *Proxy Clouds*-improved data showing results of real time filtering, hole filling and resampling. The texture applied to the completed data is extracted from the raw color frame, which does not make sense for all parts of unseen geometry, as we can see on the hole filling and resampling examples. (Color figure online)

Compression. Substituting the *Proxy Cloud* to the RGB-D stream provides a simple yet effective lossy compression scheme for transmission, with the practical side effect of removing many outliers. Our efficient data structure leads to good compression ratios while keeping high Peak Signal-to-Noise Ratio (PSNR) and being fast for compression and decompression (see Table 1 for evaluation metrics of the compression using proxies). The proxies are stored as simple grids of statistics with a local frame and bounding rectangle. As such, the compressed structure itself, i.e. the proxies, can benefit from image-based compression schemes such as JPEG [45] for offline export and storage, for which we report compression ratios and PSNR values in Table 1. In addition to the bandwidth saving, the compressed *Proxy Cloud* representation enables smooth super-sampling of the geometric data, where the output point cloud density over proxy surfaces *can be increased as desired.* The planar parameterization of each proxy offers a suitable domain for point upsampling operators, while a similar approach performed directly on the RGB-D stream is blind to the scene structure.

6 RGB-D Stream Consolidation

While being lightweight and fast to compute, the *Proxy Cloud* represents a superstructure modeling the dominant planar elements of an indoor scene. In

Table 1. Compression metrics for all processed scenes. Compression ratios are based on the raw size of a 320×240 depth map and the size of the proxies without the outliers. The JPEG export ratio is between the sizes of the raw and exported proxy clouds. The export and load procedure for all proxies takes an average of about 40 ms. The Peak Signal-to-Noise Ratio (PSNR) is computed using the average Root Mean Square Error (RMSE) between raw depth points and *proxy*-filtered ones. The time values correspond to the compression and decompression. The former is the building of proxies averaged over all frames, while the latter is the generation of a depth map by applying visibility tests to the proxies. We compare our compression performance to a state-of-the-art method based on H.264 [46] with a quality profile of 50. The corresponding compression and decompression times are given for the whole frame set.

Scene	Proxy clouds				JPEG export		H.264 [46] (qp = 50)		
	Frame ratio	Scene ratio	PSNR	Time (ms)	Ratio	PSNR	Scene ratio	PSNR	Time (s)
apt	5.19	835.7	43.0 dB	147/32	7.65	36.7 dB	11.1	25.0 dB	267/232
offices	3.75	945.5	42.0 dB	154/30	11.5	36.0 dB	10.5	22.1 dB	828/926
office0	8.72	2324.5	42.4 dB	122/19	6.75	35.7 dB	14.4	30.3 dB	692/902
office1	6.61	1944.8	40.8 dB	116/27	6.30	34.5 dB	10.5	26.9 dB	693/692
office3	8.02	1415.8	41.7 dB	122/32	7.80	35.9 dB	12.3	29.0 dB	424/444
scene0220_02	5.26	707.3	39.0 dB	127/41	7.25	33.1 dB	11.1	18.3 dB	165/175
scene0271_01	5.45	942.0	37.5 dB	119/39	6.04	33.4 dB	10.5	18.8 dB	165/162
scene0294_02	3.99	1113.4	41.9 dB	133/39	6.69	36.1 dB	9.4	17.2 dB	238/200
scene0451_05	3.59	674.7	37.1 dB	128/41	5.61	32.3 dB	7.9	15.5 dB	190/138
scene0567_01	6.09	806.5	40.6 dB	137/28	8.13	35.5 dB	15.6	20.0 dB	170/151

addition to being used to filter the input point cloud and generate an enhanced RGB-D stream as output, proxies themselves are a way to consolidate the input RGB-D frames. Hence, meshing the proxy cells leads to a lightened organized structure and aggregating all proxies in the global space allows reconstructing a higher quality surface model of the observed scene, generated on-the-fly. In this section, we compare the performance and quality of scene reconstruction using *Proxy Clouds* to state-of-the-art methods *BundleFusion* [39] and *3DLite* [42].

Qualitative Results. Figure 7 presents the reconstructed planar models based on the corresponding *Proxy Cloud*. More reconstructed scenes are available as supplemental material. As we can see, most large planar surfaces such as walls and floors are modeled with a single proxy instance.

Quantitative Results. Tables 2 and 3 present performance and quality metrics for the 10 scenes of the dataset. The reconstruction using proxies can be quantitatively assessed and compared to *3DLite* through the values of RMSE with *BundleFusion*. These metrics show that the lightweight and simple structure of the *Proxy Cloud* leads to better performance both in timing and memory consumption, while keeping a quality comparable to that of state-of-the art methods.

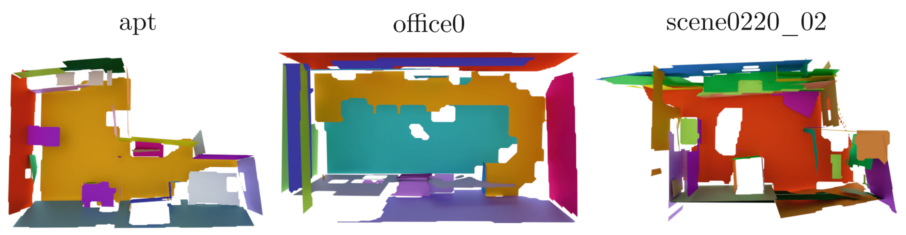

Fig. 7. Scene reconstruction. Examples of reconstructed scenes of the *3DLite* [42] dataset using *Proxy Clouds*. The meshes are made of quads when four activated cells are adjacent, and triangles otherwise.

With its low runtime and memory needs, *Proxy Clouds* offer a lighter alternative to most recent reconstruction methods characterized by *volumetric* or *deep learning* approaches, which have high requirements in computation costs and memory consumption. The generic format and implementation of the proxies avoid the need for tedious platform-specific tuning and make them well suited for embedded operation and modern mobile applications. In addition to the fact that our proxies are built and updated on the fly, the processing runs in a single thread and requires far less memory than modern embedded devices offer.

Table 2. Quantitative comparison of scene reconstruction processing. The metrics are compared between *BundleFusion* (BF), *3DLite* (3DL) and our *Proxy Clouds* (PC). The processing time to process one frame is averaged over all frames in the scene. The memory consumption is the maximum used memory during processing.

Scene	#Frames	Processing time			Memory consumption	
		BF	3DL	PC	BF	PC
apt	2865	–	5.5 h	**147 ms**	–	**135 MB**
offices	8518	–	10.8 h	**154 ms**	–	**251 MB**
office0	6159	26.4 ms	3.6 h	**122 ms**	21.4 GB	**136 MB**
office1	5730	27.7 ms	3.1 h	**116 ms**	21.1 GB	**138 MB**
office3	3820	27.7 ms	4.5 h	**122 ms**	16.9 GB	**218 MB**
scene0220_02	2026	–	2.8 h	**127 ms**	–	**119 MB**
scene0271_01	1904	–	3.0 h	**119 ms**	–	**105 MB**
scene0294_02	2369	–	5.0 h	**133 ms**	–	**110 MB**
scene0451_05	1719	–	5.1 h	**128 ms**	–	**126 MB**
scene0567_01	2066	–	3.1 h	**137 ms**	–	**206 MB**

Table 3. Quantitative comparison of scene reconstruction data. We compare our *Proxy Clouds* (PC) to *BundleFusion* (BF) and *3DLite* (3DL). The model size for BF and 3DL includes texture information, while the PC model only contains geometry. The Root Mean Square Error (RMSE) is computed using the *Metro* tool [47] between the *BundleFusion* mesh, taken as reference, and the *3DLite* and *Proxy Clouds* meshes.

Scene	Geometry size (vertices/faces)			Model size			RMSE with BF	
	BF	3DL	PC	BF	3DL	PC	3DL	PC
apt	1.7M/3.3M	62K/93K	**58K/55K**	70 MB	9.8 MB	**4.5 MB**	**3.65 m**	5.26 m
offices	1.4M/2.8M	**116K**/174K	153K/**144K**	58 MB	19 MB	**12.7 MB**	**3.81 m**	4.72m
office0	5.7M/11.3M	**42K**/63K	45K/**42K**	238 MB	6.3 MB	**3.6 MB**	**0.10 m**	0.29m
office1	6.0M/11.8M	**46K**/69K	50K/**46K**	251 MB	7.2 MB	**4.5 MB**	**0.19 m**	0.45m
office3	6.4M/12.6M	**42K**/64K	46K/**43K**	266 MB	6.2 MB	**3.9 MB**	0.30 m	**0.24 m**
scene0220_02	0.3M/0.6M	55K/83K	**49K/46K**	12 MB	9.1B	**3.7 MB**	**2.84 m**	3.75m
scene0271_01	0.2M/0.4M	39K/59K	**34K/33K**	9 MB	5.8 MB	**2.7 MB**	2.32 m	**2.31 m**
scene0294_02	0.3M/0.5M	39K/59K	**36K/34K**	10 MB	6.1 MB	**3 MB**	2.75 m	**2.74 m**
scene0451_05	0.3M/0.6M	60K/90K	**43K/39K**	12 MB	9.2 MB	**3.9 MB**	5.01 m	**3.84 m**
scene0567_01	0.3M/0.4M	**29K**/43K	44K/**41K**	9 MB	4.5 MB	**3.4 MB**	2.21 m	**1.67 m**

7 Conclusion and Future Work

We introduced *Proxy Clouds*, a unified plane-based framework for real-time processing of RGB-D streams. It takes the form of stable proxies modeling the dominant geometric scene structure through a set of rich statistics. Our method provides a compact, lightweight and consistent spatio-temporal support for the processing primitives designed to enhance data or lighten subsequent operations. It runs at interactive rates on mobile platforms and allows fast enhancement and transmission of the captured data. Our structure can be meshed and used as a model of the observed scene, generated on-the-fly. Its implementation makes possible real-time feedback and its control relies on a limited range of parameters. Compared to *BundleFusion* and *3DLite*, *Proxy Clouds* provide a good balance between processing time, memory consumption and approximation quality.

In the future, we plan to develop a parallel implementation using multi-core CPU and mobile GPUs to achieve a higher processing rate on embedded platforms. To that end, the primitives we use in our algorithm are naturally parallel scalable. While our current proxy model stores statistics on a uniform (yet sparse) grid, it could be improved using a sparse adaptive structure [48]. We also plan to extend the geometry of proxies to other simple shapes, such as boxes, spheres and cylinders, while still maintaining a unified representation for all of them, interfacing them seamlessly to the processing primitives. Last, we plan to use our stable proxies to estimate the position and orientation of the camera and track it within the scene, in a similar spirit to Raposo et al. [49].

Acknowledgements.. This work is partially supported by the French National Research Agency under grant ANR 16-LCV2-0009-01 ALLEGORI and by BPI France, under grant PAPAYA. We also wish to thank the authors of *3DLite* [42], *BundleFusion* [39] and *ScanNet* [50] for providing the dataset we use.

References

1. Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W.: 3-D mapping with an RGB-D camera. IEEE Trans. Robot. **30**(1), 177–187 (2014)
2. 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)
3. Umeyama, S.: Least-squares estimation of transformation parameters between two point patterns. IEEE Trans. Pattern Anal. Mach. Intell. **13**(4), 376–380 (1991)
4. Hulik, R., Spanel, M., Smrz, P., Materna, Z.: Continuous plane detection in point-cloud data based on 3D Hough transform. J. Vis. Commun. Image Representation **25**(1), 86–97 (2014)
5. Schnabel, R., Wahl, R., Klein, R.: Efficient RANSAC for point-cloud shape detection. Comput. Graph. Forum **26**(2), 214–226 (2007)
6. Holz, D., Holzer, S., Rusu, R.B., Behnke, S.: Real-time plane segmentation using RGB-D cameras. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS (LNAI), vol. 7416, pp. 306–317. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32060-6_26
7. Feng, C., Taguchi, Y., Kamat, V.R.: Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 6218–6225. IEEE (2014)
8. Kaiser, A., Ybanez Zepeda, J.A., Boubekeur, T.: A survey of simple geometric primitives detection methods for captured 3D data. In: Computer Graphics Forum (2018, to appear)
9. Li, L.: Filtering for 3D time-of-flight sensors. Technical report SLOA230, Texas Instruments, January 2016
10. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Sixth International Conference on Computer Vision, pp. 839–846. IEEE (1998)
11. Shao, L., Han, J., Kohli, P., Zhang, Z. (eds.): Computer Vision and Machine Learning with RGB-D Sensors. Advances in Computer Vision and Pattern Recognition. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-319-08651-4
12. Essmaeel, K., Gallo, L., Damiani, E., De Pietro, G., Dipandà, A.: Temporal denoising of kinect depth data. In: Eighth International Conference on Signal Image Technology and Internet Based Systems (SITIS), pp. 47–52. IEEE (2012)
13. Liu, S., Chen, C., Kehtarnava, N.: A computationally efficient denoising and hole-filling method for depth image enhancement. In: Kehtarnavaz, N., Carlsohn, M.F. (eds.) SPIE Conference on Real-Time Image and Video Processing, SPIE, April 2016
14. Le, A.V., Jung, S.W., Won, C.S.: Directional joint bilateral filter for depth images. Sensors **14**(7), 11362–11378 (2014)
15. Newcombe, R.A., et al.: Kinectfusion: real-time dense surface mapping and tracking. In: IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 127–136. IEEE, October 2011

16. Bapat, A., Ravi, A., Raman, S.: An iterative, non-local approach for restoring depth maps in RGB-D images. In: Twenty First National Conference on Communications (NCC), pp. 1–6. IEEE (2015)
17. Camplani, M., Salgado, L.: Adaptive spatio-temporal filter for low-cost camera depth maps. In: IEEE International Conference on Emerging Signal Processing Applications (ESPA), pp. 33–36. IEEE (2012)
18. Schmeing, M., Jiang, X.: Color segmentation based depth image filtering. In: Jiang, X., Bellon, O.R.P., Goldgof, D., Oishi, T. (eds.) WDIA 2012. LNCS, vol. 7854, pp. 68–77. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40303-3_8
19. Chen, L., Lin, H., Li, S.: Depth image enhancement for kinect using region growing and bilateral filter. In: 21st International Conference on Pattern Recognition (ICPR), pp. 3070–3073. IEEE (2012)
20. Wu, C., Zollhöfer, M., Nießner, M., Stamminger, M., Izadi, S., Theobalt, C.: Real-time shading-based refinement for consumer depth cameras. ACM Trans. Graph. (TOG) **33**(6), 200 (2014)
21. Kopf, J., Cohen, M.F., Lischinski, D., Uyttendaele, M.: Joint bilateral upsampling. ACM Trans. Graph. (ToG) **26**(3), 96 (2007)
22. Min, D., Lu, J., Do, M.N.: Depth video enhancement based on weighted mode filtering. IEEE Trans. Image Process. **21**(3), 1176–1190 (2012)
23. Liu, R., et al.: Hole-filling based on disparity map and inpainting for depth-image-based rendering. Int. J. Hybrid Inf. Technol. **9**(5), 145–164 (2016)
24. Buyssens, P., Daisy, M., Tschumperlé, D., Lézoray, O.: Superpixel-based depth map inpainting for RGB-D view synthesis. In: IEEE International Conference on Image Processing (ICIP), pp. 4332–4336. IEEE (2015)
25. Solh, M., AlRegib, G.: Hierarchical hole-filling for depth-based view synthesis in FTV and 3D video. IEEE J. Sel. Topics Sig. Process. **6**(5), 495–504 (2012)
26. Schnabel, R., Degener, P., Klein, R.: Completion and reconstruction with primitive shapes. Comput. Graph. Forum **28**(2), 503–512 (2009)
27. Biswas, J., Veloso, M.: Planar polygon extraction and merging from depth images. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3859–3864. IEEE (2012)
28. Labbé, M., Michaud, F.: Online global loop closure detection for large-scale multi-session graph-based slam. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2661–2666. IEEE, September 2014
29. Hsiao, M., Westman, E., Zhang, G., Kaess, M.: Keyframe-based dense planar slam. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 5110–5117. IEEE, May 2017
30. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. IEEE Trans. Robot. **33**(5), 1255–1262 (2017)
31. Keller, M., Lefloch, D., Lambers, M., Izadi, S., Weyrich, T., Kolb, A.: Real-time 3D reconstruction in dynamic scenes using point-based fusion. In: International Conference on 3D Vision (3DV), pp. 1–8. IEEE, June 2013
32. Salas-Moreno, R.F., Glocken, B., Kelly, P.H., Davison, A.J.: Dense planar slam. In: IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 157–164. IEEE, September 2014
33. Elghor, H.E., Roussel, D., Ababsa, F., Bouyakhf, E.H.: Planes detection for robust localization and mapping in RGB-D slam systems. In: International Conference on 3D Vision (3DV), pp. 452–459. IEEE, October 2015

34. Dou, M., Guan, L., Frahm, J.-M., Fuchs, H.: Exploring high-level plane primitives for indoor 3D reconstruction with a hand-held RGB-D camera. In: Park, J.-I., Kim, J. (eds.) ACCV 2012. LNCS, vol. 7729, pp. 94–108. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37484-5_9
35. Kaess, M.: Simultaneous localization and mapping with infinite planes. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 4605–4611. IEEE, May 2015
36. Gao, X., Zhang, T.: Robust RGB-D simultaneous localization and mapping using planar point features. Robot. Auton. Syst. **72**, 1–14 (2015)
37. Zhang, E., Cohen, M.F., Curless, B.: Emptying, refurnishing, and relighting indoor spaces. ACM Trans. Graph. (TOG) **35**(6), 174 (2016)
38. Nießner, M., Zollhöfer, M., Izadi, S., Stamminger, M.: Real-time 3D reconstruction at scale using voxel hashing. ACM Trans. Graph. (ToG) **32**(6), 169 (2013)
39. Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., Theobalt, C.: Bundlefusion: real-time globally consistent 3D reconstruction using on-the-fly surface reintegration. ACM Trans. Graph. (TOG) **36**(3), 24 (2017)
40. Zhang, Y., Xu, W., Tong, Y., Zhou, K.: Online structure analysis for real-time indoor scene reconstruction. ACM Trans. Graph. (TOG) **34**(5), 159 (2015)
41. Dzitsiuk, M., Sturm, J., Maier, R., Ma, L., Cremers, D.: De-noising, stabilizing and completing 3D reconstructions on-the-go using plane priors. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3976–3983. IEEE, May 2017
42. Huang, J., Dai, A., Guibas, L., Niessner, M.: 3Dlite: towards commodity 3D scanning for content creation. ACM Trans. Graph. (TOG) **36**(6), 203 (2017)
43. Kass, M., Solomon, J.: Smoothed local histogram filters. ACM Trans. Graph. (TOG) **29**(4), 100 (2010)
44. Serra, J.: Image Analysis and Mathematical Morphology. Academic Press, Inc., Cambridge (1983)
45. Wallace, G.K.: The JPEG still picture compression standard. IEEE Trans. Consum. Electron. **38**(1), xviii–xxxiv (1992)
46. Nenci, F., Spinello, L., Stachniss, C.: Effective compression of range data streams for remote robot operations using H. 264. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3794–3799. IEEE, September 2014
47. Cignoni, P., Rocchini, C., Scopigno, R.: Metro: measuring error on simplified surfaces. Comput. Graph. Forum **17**(2), 167–174 (1998)
48. Lefebvre, S., Hoppe, H.: Compressed random-access trees for spatially coherent data. In: Kautz, J., Pattanaik, S. (eds.) Proceedings of the 18th Eurographics Conference on Rendering Techniques, pp. 339–349. Eurographics Association (2007)
49. Raposo, C., Lourenco, M., Goncalves Almeida Antunes, M., Barreto, J.P.: Plane-based odometry using an RGB-D camera. In: British Machine Vision Conference (BMVC). Elsevier, September 2013
50. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: richly-annotated 3D reconstructions of indoor scenes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, July 2017