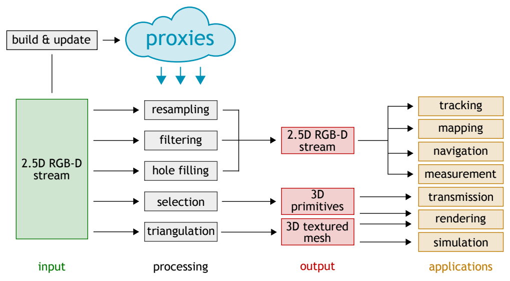
Proxy Clouds for RGB-D Stream Processing: An Insight

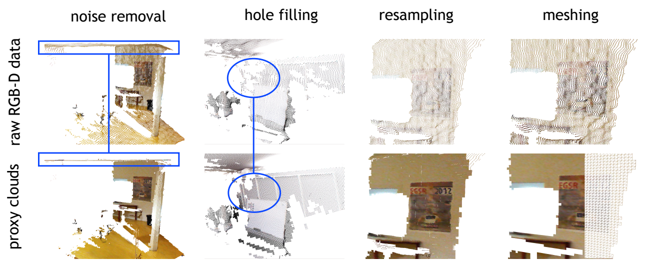
Adrien Kaiser
Telecom ParisTech, Paris-Saclay University & Ayotle

Jose Alonso Ybanez Zepeda
Ayotle

Tamy Boubekeur
Telecom ParisTech, Paris-Saclay University



(a) Proxy Clouds Workflow



(b) Data Improvement

Figure 1: *(a) Workflow.* From a stream of RGB-D frames, proxies are built and updated through time. They are used as priors to process the frames and improve subsequent operations. A selection of proxies based on the current RGB-D frame can be used for lightening data transmission or as triangulation prior for fast depth data meshing, with application to rendering or simulation. *b) Data Improvement.* Raw RGB-D data (top) and *Proxy Clouds*-improved data after 100 frames (bottom) showing results of real time noise removal, hole filling, point cloud resampling and meshing. Blue surrounded areas highlight regions where improvement using *Proxy Clouds* is significant compared to the low quality input RGB-D frames.

ABSTRACT

Modern RGB-D sensors are widely used for indoor 3D capture, with applications ranging from modeling to robotics, through gaming. Nevertheless, their use is limited by their low resolution, with frames often corrupted with noise, missing data and temporal inconsistencies. In order to cope with all these issues, we present *Proxy Clouds*, a multiplanar superstructure for unified real-time processing of RGB-D data. By generating and updating through time a single set of rich statistics parameterized over planar proxies from raw RGB-D data, several processing primitives can be applied to improve the quality of the RGB-D stream on-the-fly or lighten further operations. We illustrate the use of *Proxy Clouds* on several applications, including noise and temporal flickering removal, hole filling, resampling, color processing and compression. We present experiments performed with our framework in indoor scenes of different natures captured with a consumer depth sensor.

CCS CONCEPTS

•**Computing methodologies →Shape representations;** *Scene understanding; Shape analysis;* Reconstruction; Modeling;

KEYWORDS

RGB-D stream, Plane detection, Depth improvement, Scene analysis

ACM Reference format:
Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. 2017. Proxy Clouds for RGB-D Stream Processing: An Insight. In *Proceedings of SIGGRAPH '17 Talks, Los Angeles, CA, USA, July 30 - August 03, 2017,* 2 pages.
DOI: http://dx.doi.org/10.1145/3084363.3085031

1 INTRODUCTION

Objectives. The real time RGB-D stream output of modern commodity consumer depth cameras can feed a growing set of end applications, from AR to design. Although such technologies made recently great progress, the limited quality of their RGB-D stream still limits their impact spectrum. This mostly originates in the low resolution of the frames and the inherent noise, incompleteness and temporal inconsistency stemming from single view capture. With *Proxy Clouds*, we aim at improving such streams by extracting statistics which are recorded on a sparse set of detected 3D planes. Using these time-evolving statistics, we can improve the RGB-D stream on the fly by reinforcing features, removing noise and outliers or filling missing parts, under memory-limited, real time embedded constraints. We design such a lightweight planar superstructure to be stable through time which gives priors to apply several signal-inspired processing primitives to the RGB-D frames, to structure the data and simplify or lighten subsequent operations. In practice, our system takes a raw RGB-D stream as input and outputs an enhanced stream together with the optional set of proxies associated with the current RGB-D data (see Fig. 1).

Previous Work. Methods that build high level models of captured 3D data are mostly based on *RANSAC*, the *Hough transform* or *Region Growing* algorithms. In our embedded, real time, memory-limited context, we take inspiration from the RANSAC-based method of

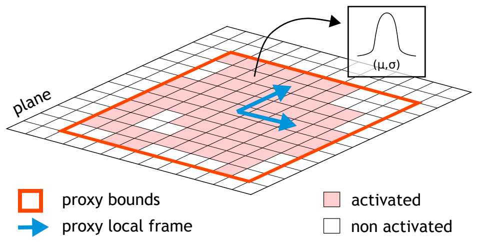


Figure 2: *Planar Proxy Model.* Built upon a plane in 3D space, the model is made of a local frame, spatial bounds and a grid of cells which contain statistics (mean μ and variance σ) gathered from the RGB-D data. Activated cells are the ones containing inliers from many frames.

Schnabel et al. [2007] 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. Regarding RGB-D processing, depth maps can be denoised using spatial filters e.g., gaussian, median, bilateral, adaptive or anisotropic filters, often refined through time, with the resulting enhanced stream potentially used for a full 3D reconstruction [Izadi et al. 2011]. They can be upsampled using cross bilateral filters and their holes can be filled with either spatial or morphological filters, together with inpainting or multiscale processing for e.g., *depth image-based rendering* (DIBR) under close viewing conditions. Only a few methods have used planar proxies as priors to process 2.5D data, with in particular Schnabel et al. [2009] who detect limits of planes to fill in holes in static 3D point clouds.

2 MODEL

Proxy clouds. capture the subset of RGB-D data which is often seen and consistent through frames, hence revealing the dominant structural elements in the scene. They take the form of a multiplanar superstructure, where each proxy is equipped with a local frame, spatial bounds and a regular 2D grid of statistics, mapped on the plane and gathered from the RGB-D data, including mean and variance for color and depth (see Fig. 2). We build 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 following algorithm:

```
1. filter $D_t$ with a bilateral color/depth convolution;
2. estimate the normal field $N_t$ from $D_t$ using the depth gradient;
3. estimate the camera motion $M_t$ from $X_{t-1}$ [Endres et al. 2014];
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. update the previous proxies with $X_t$ – or discard proxies
         given their vote count;
5. detect new proxy planes in $X_t \setminus inliers$ using RANSAC [Schnabel
   et al. 2007] and initialize local frames and statistics with $X_t$;
6. refine camera motion $M_t$ using the proxy cloud.
```

RGB-D Stream Processing. Our proxy cloud allows recovering 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: **noise removal** is performed by projecting the sensor's data points onto their associated proxy; **temporal flickering** is suppressed by accumulating observations in multiple frames, using the proxies to consolidate the stable geometry of the scene; **hole filling** is conducted by reinforcing data from the support of stable proxies over multiple frames, augmenting the current frame with missing samples from previous ones; **super sampling** is realized by enriching the low definition geometric component of the stream using the higher resolution color component structured in the proxies to guide the process; **color processing** exploits proxies as priors to define the kernel support onto which standard convolution-based image manipulations, such as *blurring* or *sharpening*, are executed; **compression** 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.

3 RESULTS AND DISCUSSION

Proxy Clouds are currently implemented through hardware and software components. The hardware setup is made of an Orbbec Astra RGB-D sensor and a mobile device with Intel Core i7, 8 cores at 2.0GHz and 4GB memory. The software setup has a client-server architecture, where the server runs on an embedded environment to trigger the sensor and process the data. The client's GUI allows controlling the processing parameters and getting a real time feedback. A limited range of intuitive parameters allow the user to control the quality-performance trade-off. The current timing to build and update planar proxies using our (single-threaded) implementation is around 200 ms for an input depth image of 320x240 pixels.
In contrast to previous RGB-D data processing pipelines, our *Proxy Clouds* model dominant geometric components with local information, allowing to faithfully represent high frequency borders or openings. Their refinement over time gives them more stability and consistency, even in noisy and occluded environments.
As future work, we plan to develop a parallel implementation using multi-core CPU and mobile GPUs. Also, our current grid model is regular and statistics are stored cell-wise, which could be improved by using a sparse adaptive structure. Last, we plan to extend the geometry of proxies to other simple shapes, such as boxes, spheres and cylinders, while still maintaining an unified representation for the statistics recorded on them.

REFERENCES

Felix Endres, Jurgen Hess, Jurgen Sturm, Daniel Cremers, and Wolfram Burgard. 2014. 3-D mapping with an RGB-D camera. *IEEE Transactions on Robotics* 30, 1 (2014), 177–187.
Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proc. ACM Symp. on User Interface Software and Technology.* 559–568.
Ruwen Schnabel, Patrick Degener, and Reinhard Klein. 2009. Completion and reconstruction with primitive shapes. *Computer Graphics Forum* 28, 2 (2009), 503–512.
Ruwen Schnabel, Roland Wahl, and Reinhard Klein. 2007. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum* 26, 2 (June 2007), 214–226.