# Point Cloud Noise and Outlier Removal for Image-Based 3D Reconstruction – Supplementary Material –

Katja Wolff[1,2] Changil Kim[2] Henning Zimmer[1] Christopher Schroers[1]
Mario Botsch[3] Olga Sorkine-Hornung[2] Alexander Sorkine-Hornung[1]

[1]Disney Research [2]Department of Computer Science, ETH Zurich [3]Bielefeld University

katja.wolff,sorkine,kimc@inf.ethz.ch
henning.zimmer,christopher.schroers,alex@disneyresearch.com botsch@techfak.uni-bielefeld.de

In this supplementary material, we first present the parameters used to create the meshes in Figure 4 of our paper with the screened Poisson surface reconstruction (PSR) method.

We then provide pseudocode for our noise and outlier removal algorithm. Finally, we present each dataset from Figure 4 of the paper in a more extensive manner.

For each reconstruction method (MVE, LFD, ACTS and PS) we show the unfiltered reconstructed point cloud, the triangle mesh constructed from this point cloud using PSR, the filtered point cloud using our outlier and noise removal algorithm, and the resulting mesh constructed by PSR. For PMVS, we show the reconstructed point cloud and the corresponding mesh only as a comparison. As PMVS does not produce depth maps, we cannot apply our algorithm here. **Please also refer to the supplementary video for a presentation of the results from Figure 4.**

## PSR parameters for Figure 4 in our paper

| Datasets | MVE | | | | LFD | | | | ACTS | | | | PS | | | | PMVS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSR | | **Ours+PSR** | | PSR | | **Ours+PSR** | | PSR | | **Ours+PSR** | | PSR | | **Ours+PSR** | | PSR | |
| | *pw* | *spn* | *pw* | *spn* | *pw* | *spn* | *pw* | *spn* | *pw* | *spn* | *pw* | *spn* | *pw* | *spn* | *pw* | *spn* | *pw* | *spn* |
| DECORATION | 0 | 5 | 1 | 5 | 0 | 20 | 0 | 20 | 0 | 20 | 0.1 | 5 | 0 | 20 | 0 | 20 | 0.25 | 1 |
| DRAGON | 0 | 5 | 1 | 3 | 0.1 | 20 | 1 | 5 | 0 | 5 | 4 | 3 | 0 | 20 | 0 | 20 | 0.5 | 1 |
| SCARECROW | 0 | 5 | 1 | 3 | 0.1 | 20 | 0 | 5 | 0.1 | 10 | 1 | 20 | 0 | 10 | 1 | 20 | 1 | 1 |
| STATUE | 0.1 | 1 | 4 | 10 | 0.1 | 10 | 1 | 20 | 0.1 | 20 | 0.1 | 3 | 0 | 5 | 1 | 20 | 0.25 | 1 |
| TORCH | 0 | 3 | 1 | 1 | 0 | 5 | 0 | 1 | 0 | 5 | 0 | 5 | 0 | 20 | 1 | 20 | 0.25 | 1 |

Table 1: Parameters used for generating meshes using PSR from the point clouds without (*columns titled PSR*) and with (*columns titled Ours+PSR*) our denoising method, in Figure 4 of our paper. Here, *pw* denotes the chosen point weight and *spn* denotes the samples per node.

## References

[1] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proc. ACM SIGGRAPH*, pages 71–78, 1992. 7

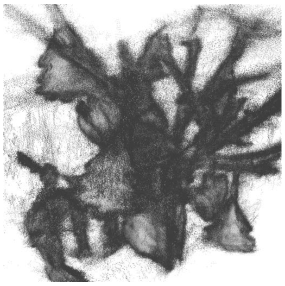
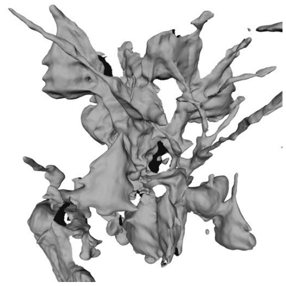
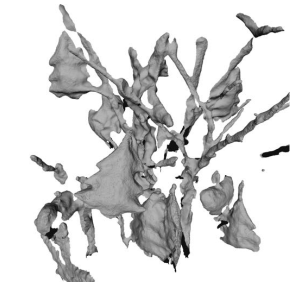
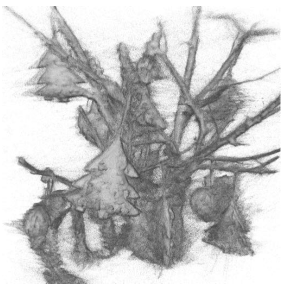
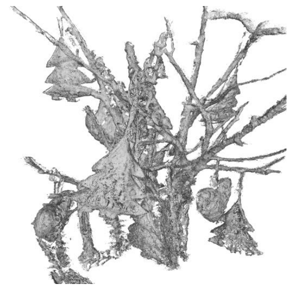
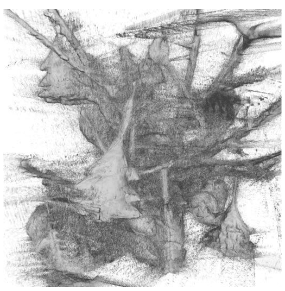
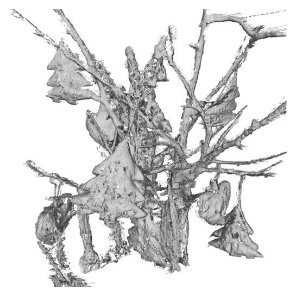
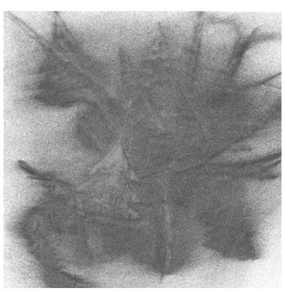
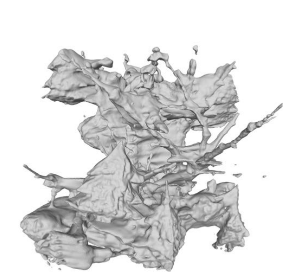
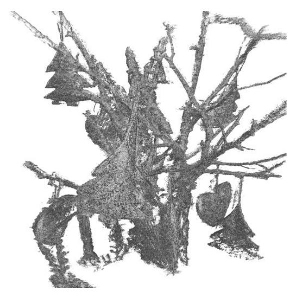
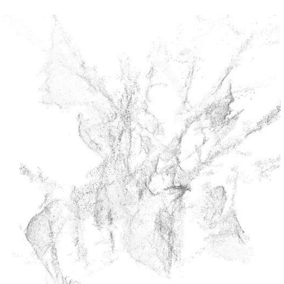
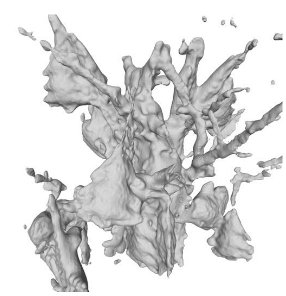
| | point cloud | PSR | filtered point cloud | PSR on filtered point cloud |
|---|---|---|---|---|
| MVE |  |  |  |  |
| LFD |  |  |  |  |
| ACTS |  |  |  |  |
| PS |  |  |  |  |
| PMVS |  |  | | |

Table 2: Results for the DECORATION dataset.

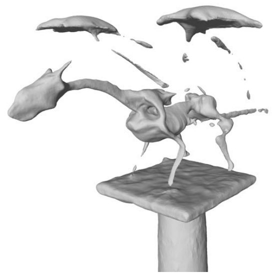
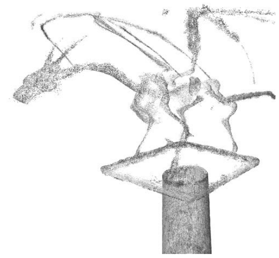
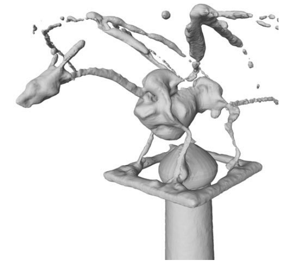
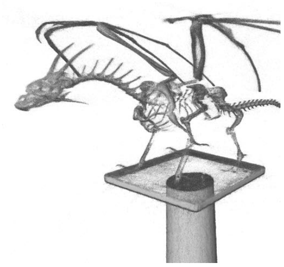
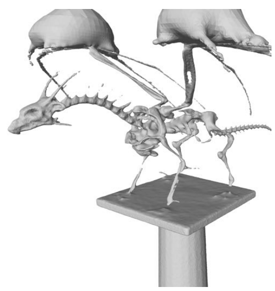
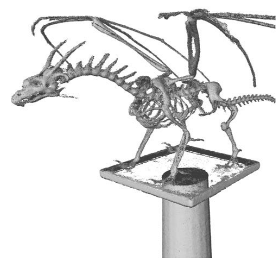
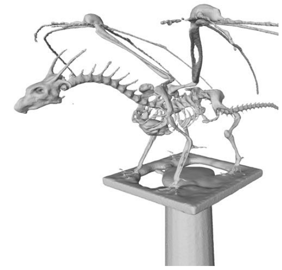
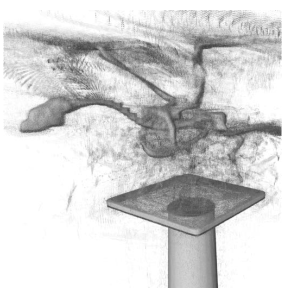
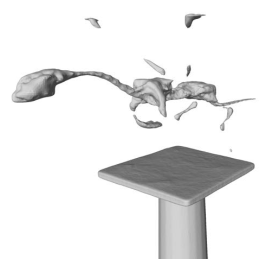
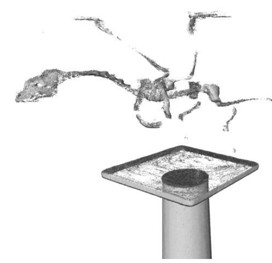
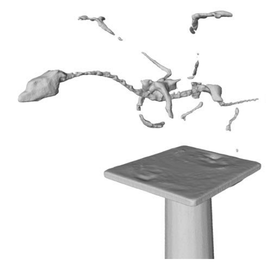
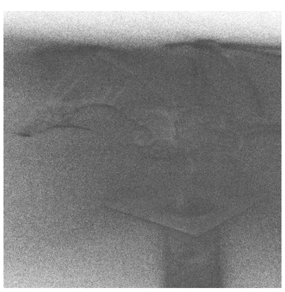
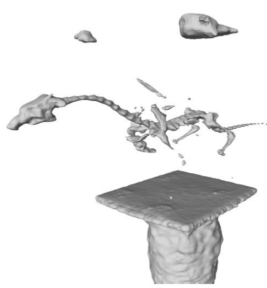
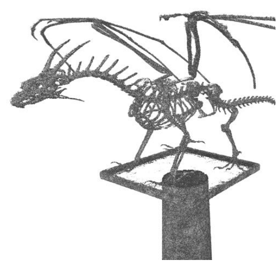
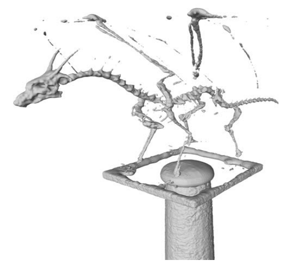
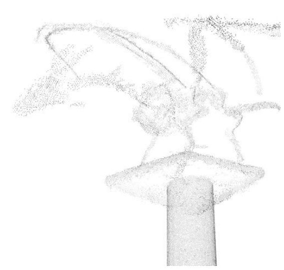
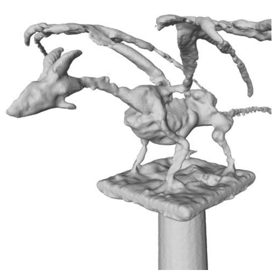
| | point cloud | PSR | filtered point cloud | PSR on filtered point cloud |
|---|---|---|---|---|
| MVE |  |  |  |  |
| LFD |  |  |  |  |
| ACTS |  |  |  |  |
| PS |  |  |  |  |
| PMVS |  |  | | |

Table 3: Results for the DRAGON dataset.

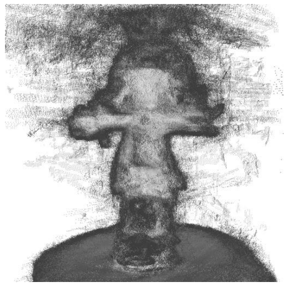
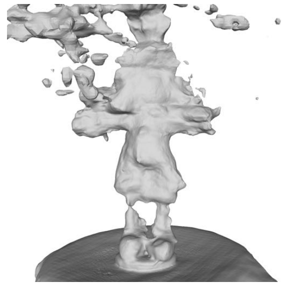
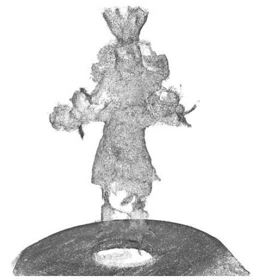
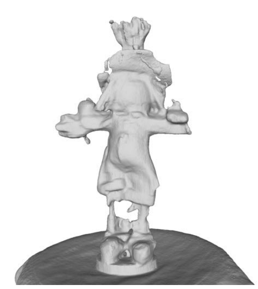
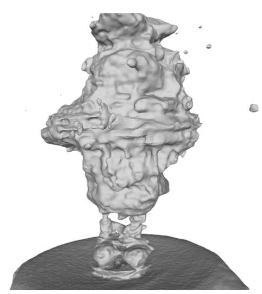
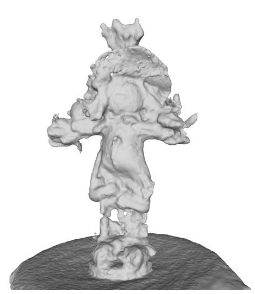
| | point cloud | PSR | filtered point cloud | PSR on filtered point cloud |
|---|---|---|---|---|
| MVE |  |  |  |  |
| LFD |  |  |  |  |
| ACTS |  |  |  |  |
| PS |  |  |  |  |
| PMVS |  |  | | |

Table 4: Results for the SCARECROW dataset.

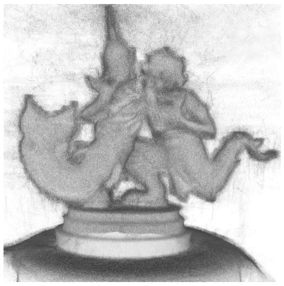
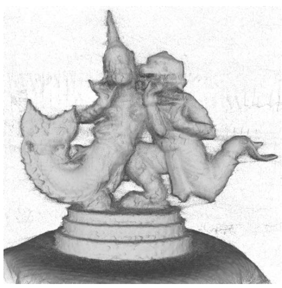
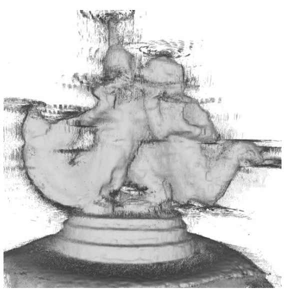
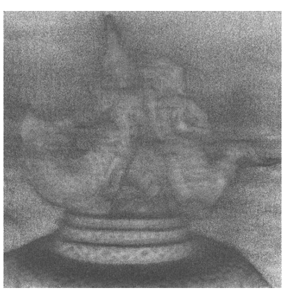
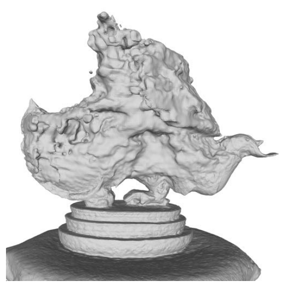
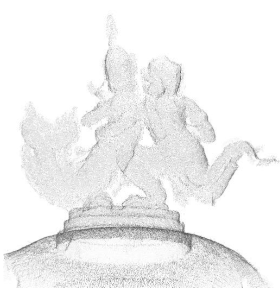
| | point cloud | PSR | filtered point cloud | PSR on filtered point cloud |
|---|---|---|---|---|
| MVE |  |  |  |  |
| LFD |  |  |  |  |
| ACTS |  |  |  |  |
| PS |  |  |  |  |
| PMVS |  |  | | |

Table 5: Results for the STATUE dataset.

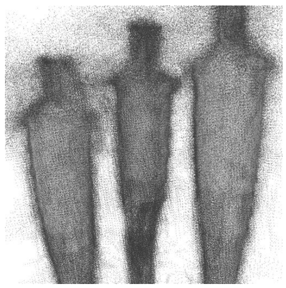
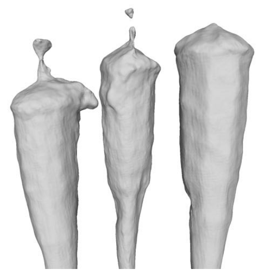
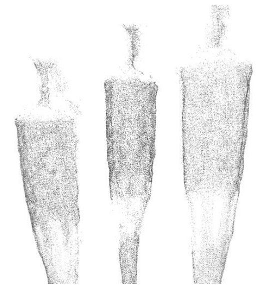
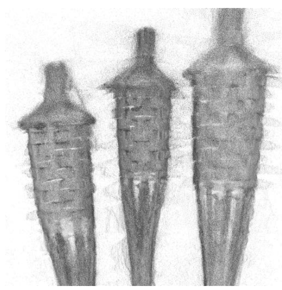
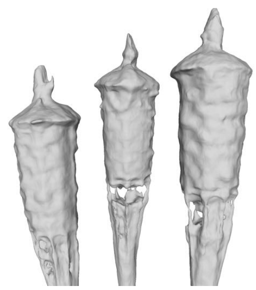
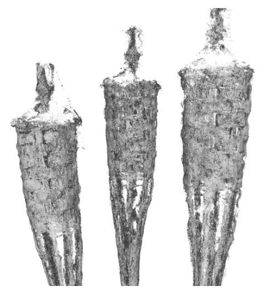
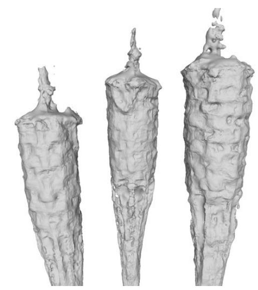
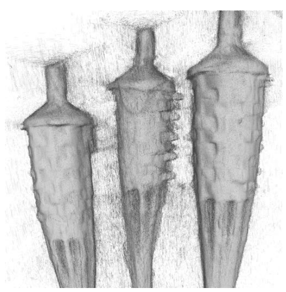
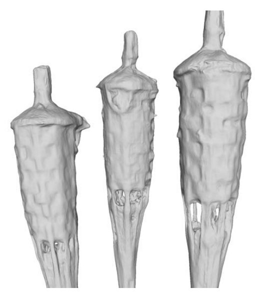
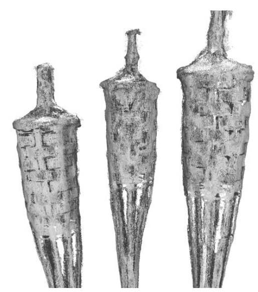
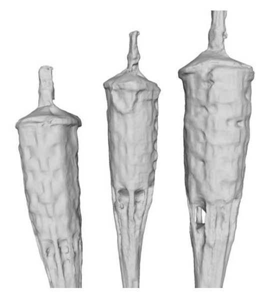
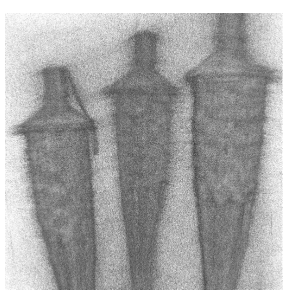
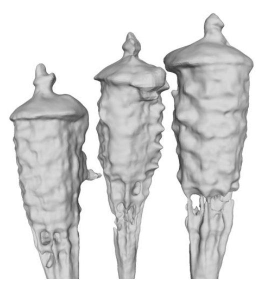
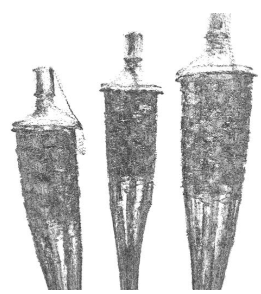
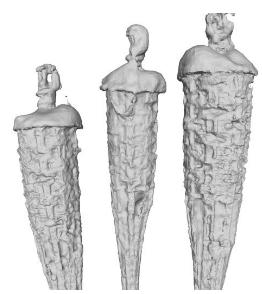
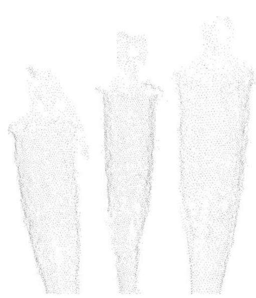
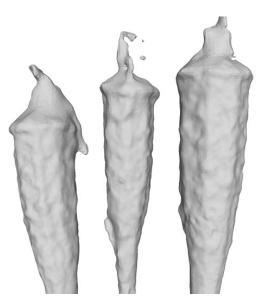
| | point cloud | PSR | filtered point cloud | PSR on filtered point cloud |
|---|---|---|---|---|
| MVE |  |  |  |  |
| LFD |  |  |  |  |
| ACTS |  |  |  |  |
| PS |  |  |  |  |
| PMVS |  |  | | |

Table 6: Results for the TORCH dataset.

**Algorithm 1:** Pseudocode for our noise and outlier removal algorithm.

*Please refer to Section 3 in our paper for more details.*

---

**input** : $N$ depth maps $D_i$, $N$ camera matrices $P_i$ (including the camera view vectors $\mathbf{v}_i$), $N$ RGB images $I_i$, parameters: $\sigma$, $t_d$, $t_p$, $t_v$

**output** : filtered 3D points with normals

**forall the** *depth maps* $D_i$ **do**

- $\{\mathbf{p}\} \leftarrow$ project all pixels of $D_i$ into 3D (using camera matrices $P_i$);
- **forall the** *3D points* $\mathbf{p}$ *of* $D_i$ **do**
  - compute normal (using PCA on a patch of size 7×7) `// refer to [1]`
  - compute weights $w_i(\mathbf{p})$ from normals `// see Equation 3`

filtered_points $\leftarrow \emptyset$

**forall the** *depth maps* $D_i$ **do**

- **forall the** *3D points* $\mathbf{p}$ *of* $D_i$ **do**
  - $d(\mathbf{p}), w(\mathbf{p}), v(\mathbf{p}), s, s2 \leftarrow 0$
  - **forall the** *depth maps* $D_j$ **do**
    - $(u,v,z) \leftarrow$ project $\mathbf{p}$ into $D_j$ (using camera matrices $P_i$) `// here u and v denote the coordinates in image space and z denotes the depth value`
    - get the triangle in which $(u, v)$ is contained `// see Figure 2: this means checking whether (u,v) is contained in image space and determining three integer coordinates for the triangle corners - an upper left triangle or a lower right triangle`
    - **if** $\mathbf{v}_i^\mathsf{T}\mathbf{v}_j > 0$ ***or*** *(u,v) lies in no triangle* ***or*** *the smallest angle of the triangle < 1 degree* **then**
      - continue
    - interpolate weight $w$ from triangle corner weights $w_i(\mathbf{p})$ `// see Figure 2`
    - interpolate depth value $z(\mathbf{p})$ from the depth values of triangle vertices `// see Figure 2`
    - $d \leftarrow z(\mathbf{p}) - z$
    - **if** $d < -\sigma$ **then** `// p could not have been observed from this view`
      - continue
    - **if** $d > \sigma$ **then** `// p is far from the observed range surface, therefore truncate d`
      - $d \leftarrow \sigma$
    - $d(\mathbf{p}) \leftarrow \frac{w(\mathbf{p})d(\mathbf{p}) + (wd)/\sigma}{w(\mathbf{p}) + w}$ `// see Equation 4`
    - $w(\mathbf{p}) \leftarrow w(\mathbf{p}) + w$
    - **if** $d \neq \sigma$ **then** `// update photoconsistency only for range surfaces close to p`
      - interpolate color value $\mathbf{c}$ from the color values of triangle vertices (from the RGB image $I_j$); `// see Figure 2`
      - $s \leftarrow s + c$
      - $s2 \leftarrow s2 + c^\mathsf{T} c$
      - $v(\mathbf{p}) = v(\mathbf{p}) + 1$
  - $p(\mathbf{p}) = \sqrt{(s2 - s^\mathsf{T} s / v(\mathbf{p}))/v(\mathbf{p})} \cdot \frac{2}{255\sqrt{3}}$ `// see Equation 7; scaled to [0,1]`
  - **if** $-t_d < d(\mathbf{p}) < 0$ ***and*** $p(\mathbf{p}) < t_p$ ***and*** $v(\mathbf{p}) > t_v$ **then**
    - filtered_points $\leftarrow$ filtered_points $\cup$ $\mathbf{p}$

**return** filtered_points