Dense range images from sparse point clouds using multi-scale processing

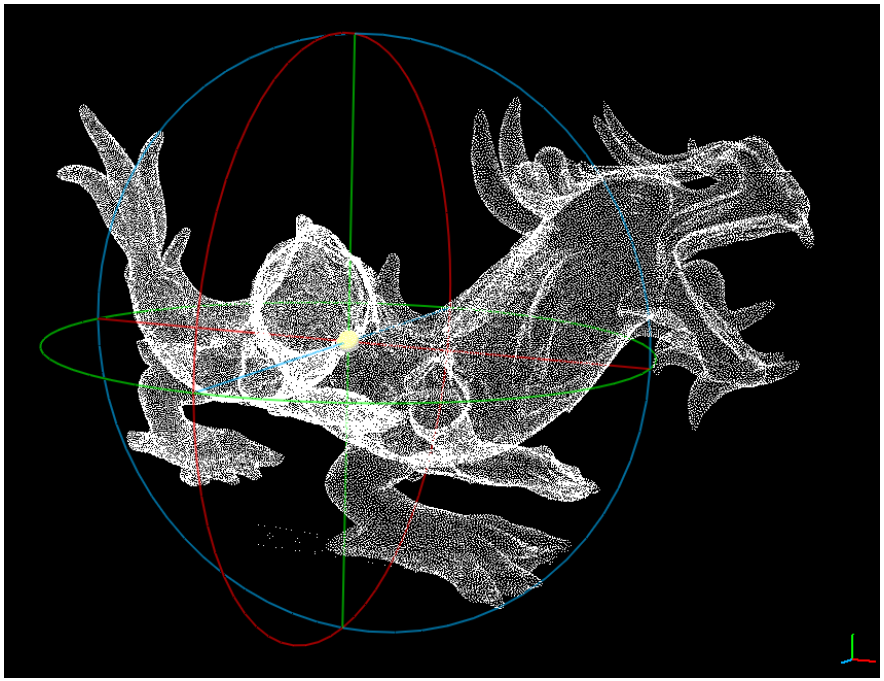
Luat Do, Lingni Ma, Peter H.N. de With 2012

Tim Lenertz / Renzhi Deng

Université Libre de Bruxelles

10 december 2014

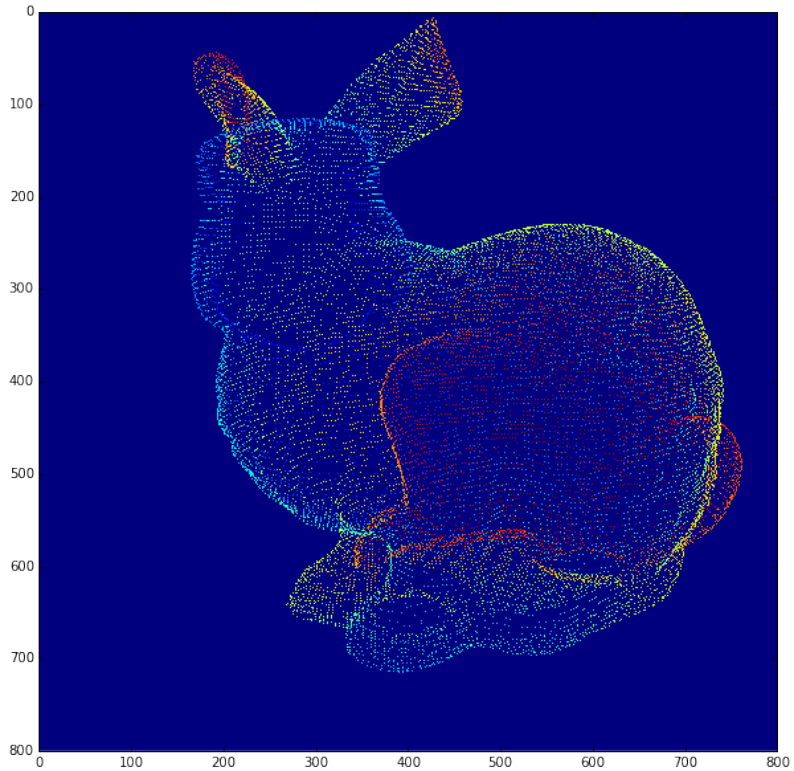
Point cloud projection



Point cloud
Set of X,Y,Z points

Range image
Depth values on X,Y grid
Output of 3D scanner
Only information seen from one view point

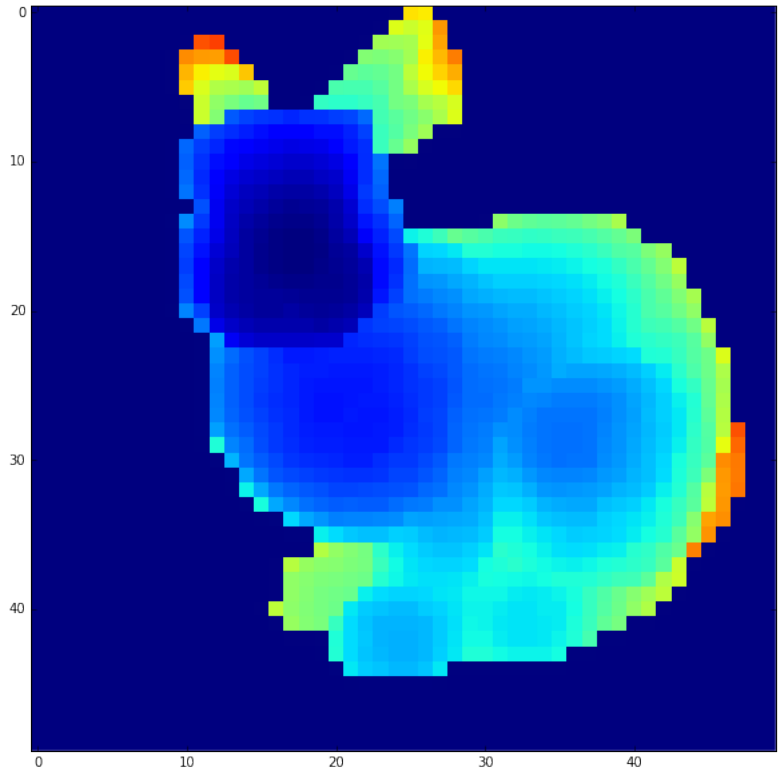
Sparse Range Image



800x800 pixel

Sparse

Pixels with no corresp. point



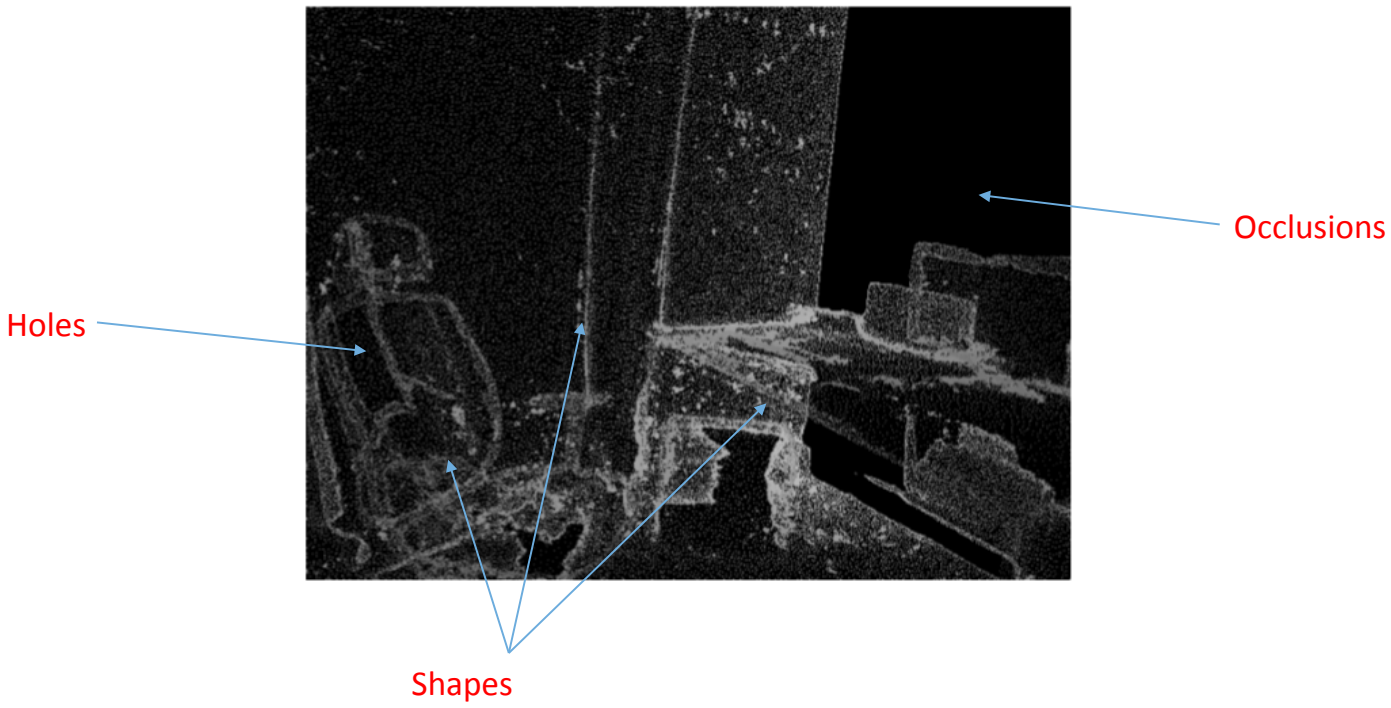
50x50 pixel

Dense, no holes

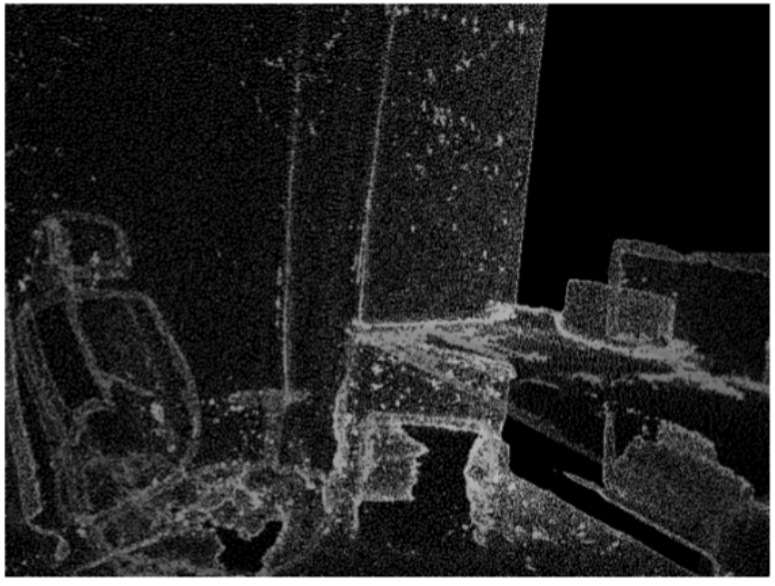
But lower quality

→ **Aim:** Combine information from both

Three major issues



Idea



Problem?

Algorithm: pseudo code

Algorithm 1 RANGE IMAGES FROM POINT CLOUDS

Require: Point Cloud Ψ, Image stack S, range image Lr
Ensure: Densed range image Hr

1: **while** holes **do**
2: project Ψ to Lr using Z-buffering
3: find small holes in Lr
4: **if** holes **then**
5: $S \leftarrow$ push Lr into image stack
6: downscale Lr with factor 2
7: **end if**
8: **end while**

9: **while** S not empty **do**
10: $Hr \leftarrow$ pop image from S
11: resize Lr to the image size of Hr
12: remove strong edges from Lr
13: $Hr \leftarrow \min(Lr, Hr)$
14: $Lr \leftarrow Hr$
15: **end while**

16: fill in remaining small holes in Hr

Kernel: $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

Occlusion map

Bilinear filtering

K-means clustering in a foreground and background cluster

fill the holes

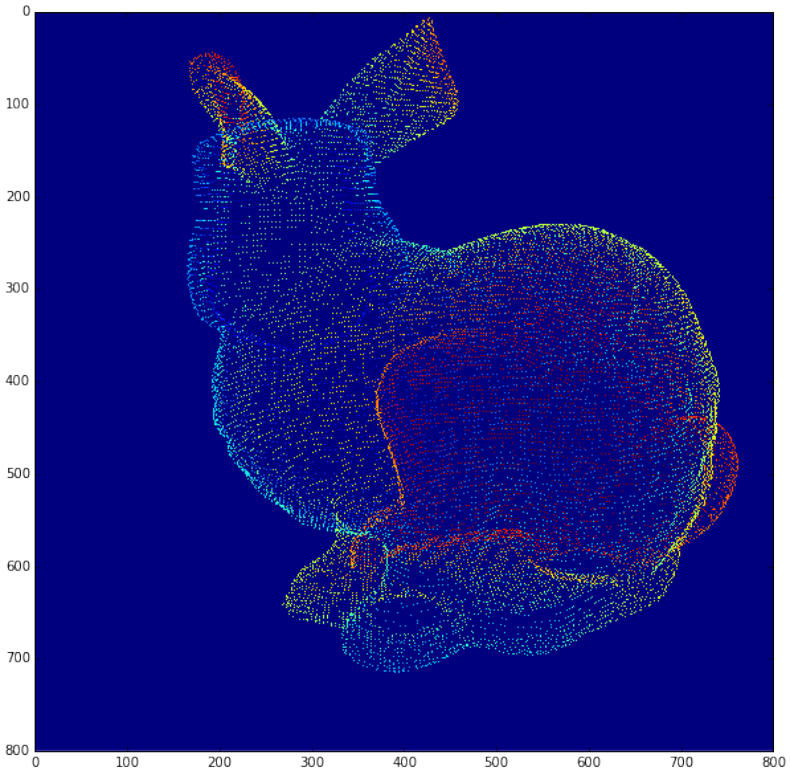
a certain scale

preserve shapes

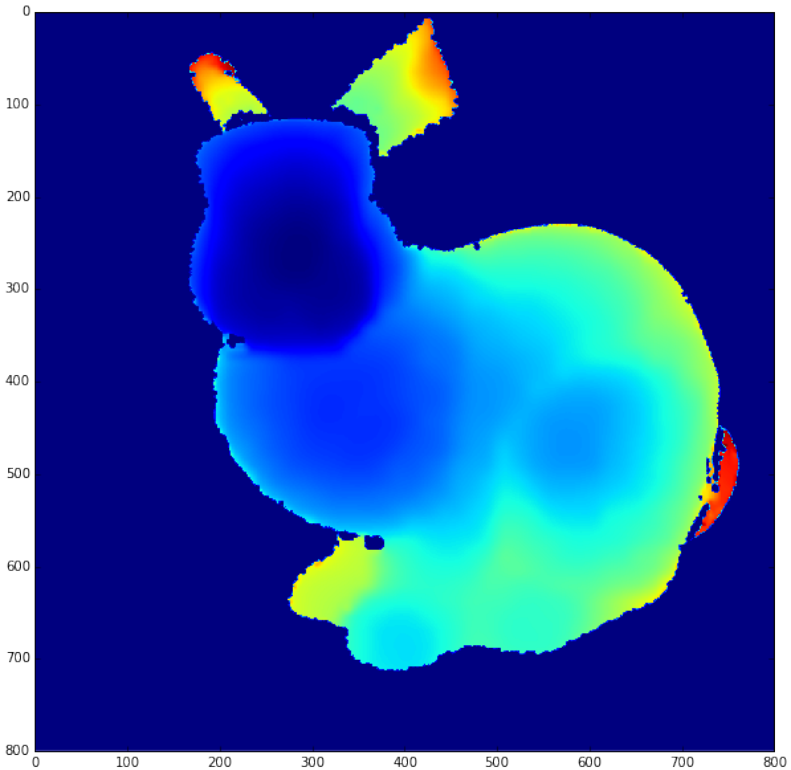
merging

iPython Implementation

Results obtained:



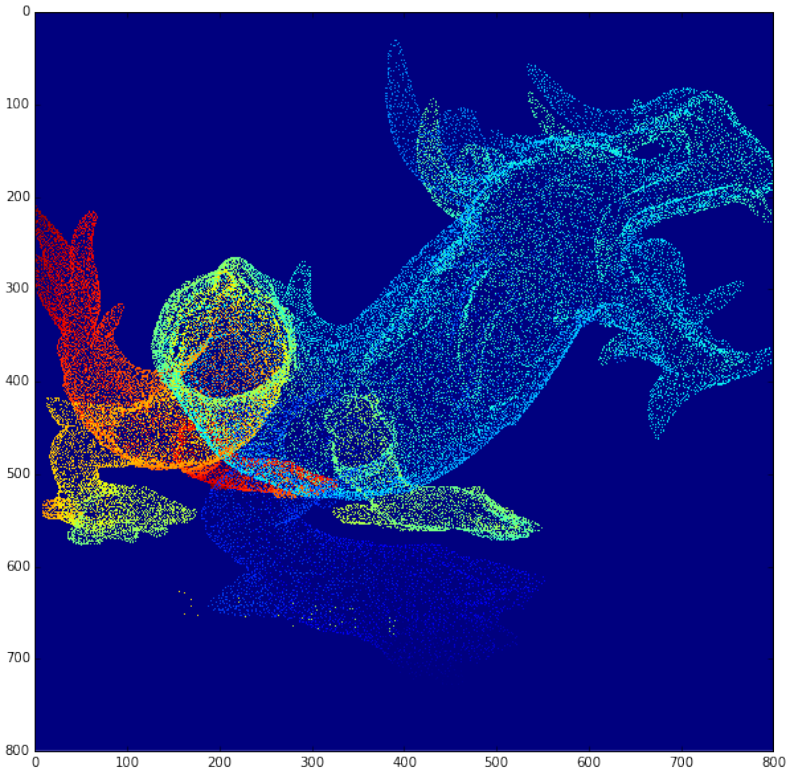
Original sparse projection



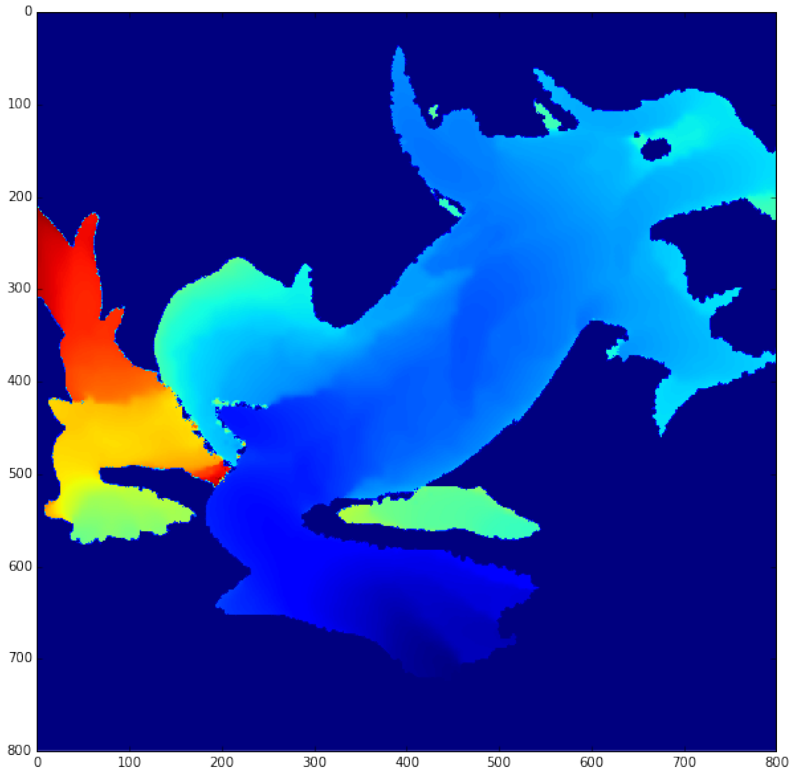
Result of superresolution algorithm

iPython Implementation

Results obtained:



Original sparse projection



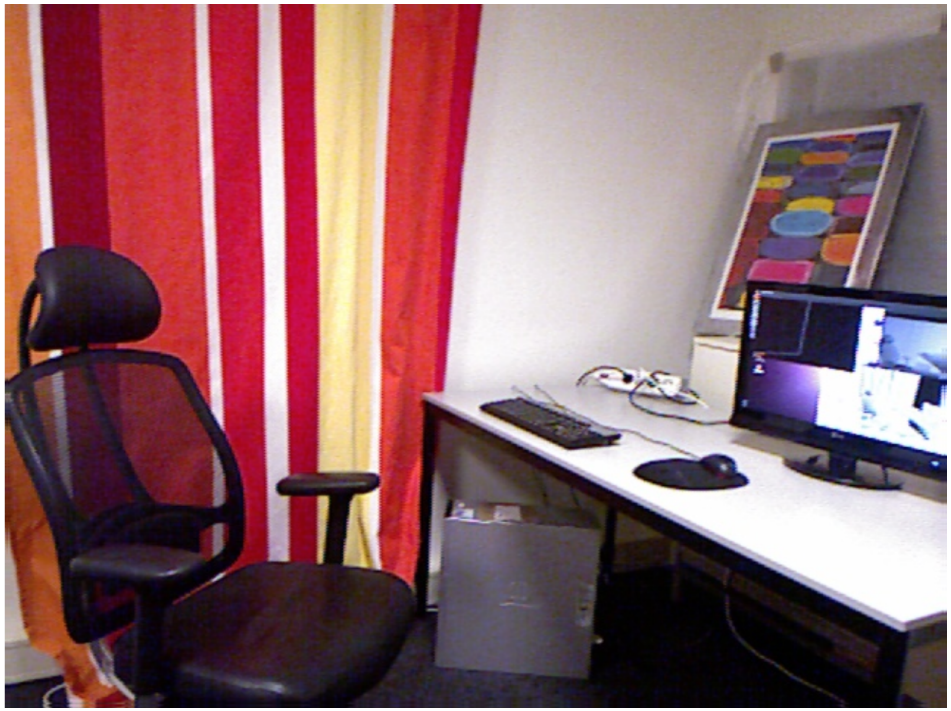
Result of superresolution algorithm

iPython Implementation

Demo

Application: Range-Color Image Alignment

Dense range image
Output of algorithm



Color photograph
Slightly different view angle

Application: Range-Color Image Alignment

- Detect edges in both photo and range image
- Correspondences of photo & range edge pixels
  $\rightarrow$ Initially: closest on other image
- Have 2D-3D pairs (upon backprojection)
- Solve Perspective-to-Point equation

$$\arg\min_{\mathbf{M}} \sum_i (\mathbf{PM}\vec{r_i} - \vec{c_i})^2$$

  $\rightarrow$ find better camera pose $\mathbf{M}$
- Project using new $\mathbf{M}$
- Apply superresolution, fill holes
- Repeat until $\mathbf{M}$ converges