ISPRS GEOSPATIAL WEEK 2019

From discrete to continuous levels of detail for managing nD-PointClouds

12-6-2019

Peter van Oosterom

Keynote at ISPRS geospatial week 13 June 2019, Enschede, Netherlands

Acknowledgements: based on joint work with Martijn Meijers, Edward Verbree, Haicheng Liu, Theo Tijssen, Oscar Martinez-Rubi, Mike Horhammer, Stella Psomadaki, Xuefeng Guan, Jippe van der Maaden, and Simon van Oosterom

Motivation

- point Clouds are large data sets (AHN2 640.000.000.000 points)
  → must be organized efficiently (clustering/indexing)

- just space and/or time is not sufficient
  - Works well for zoomed-in/detail queries
  - Does not work well for overview queries

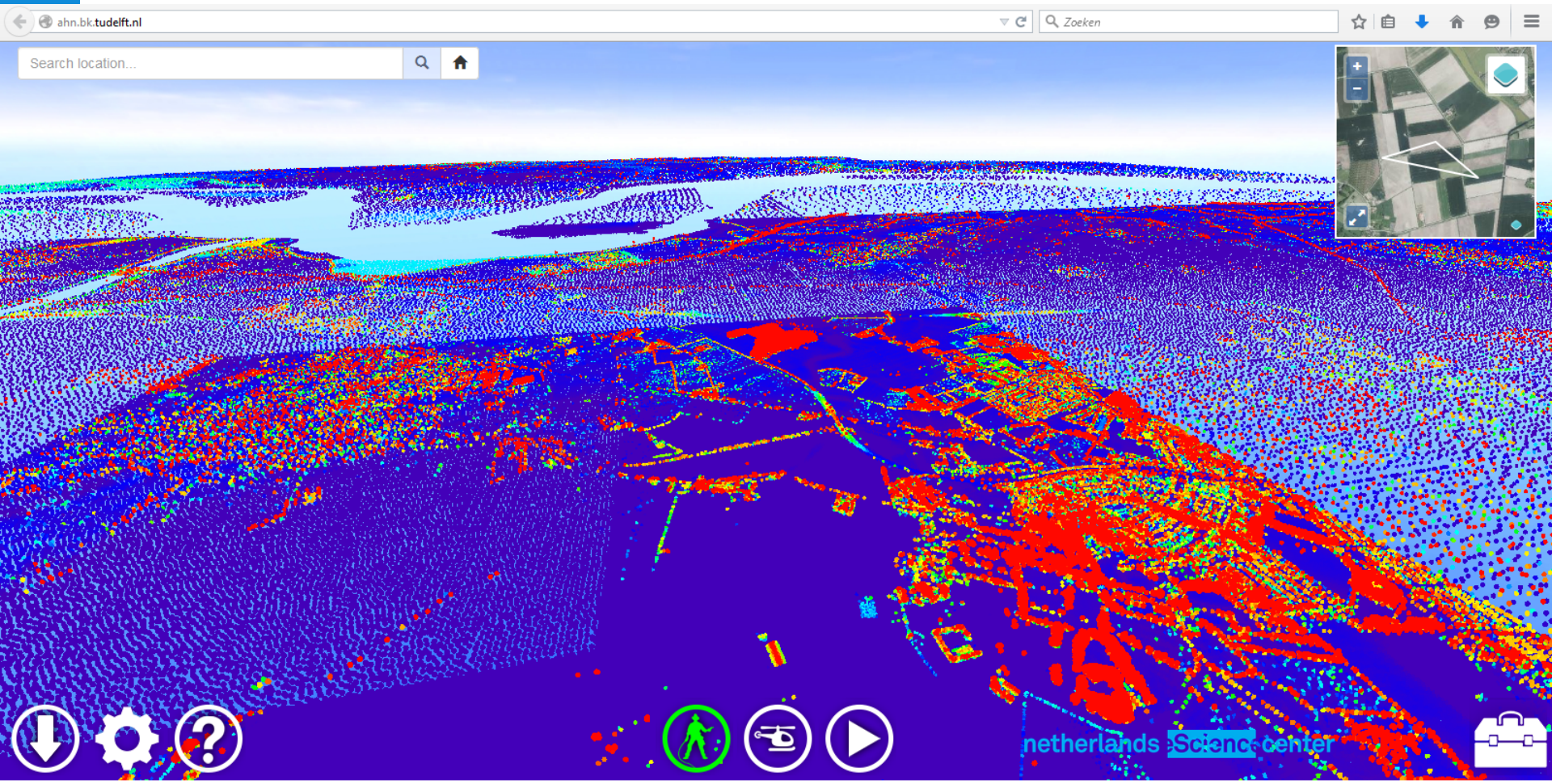
- therefore, need for levels of detail (LoD)
  as scale in vector maps and/or data pyramid in raster maps

- current state of the art is discrete LoDs
  e.g. octree levels as created by Potree (converter)

- discrete LoDs have drawbacks → continuous LoDs needed

Discrete LoDs are visible…

http://ahn2.pointclouds.nl: 640.000.000.000 points on-line 3D viewer

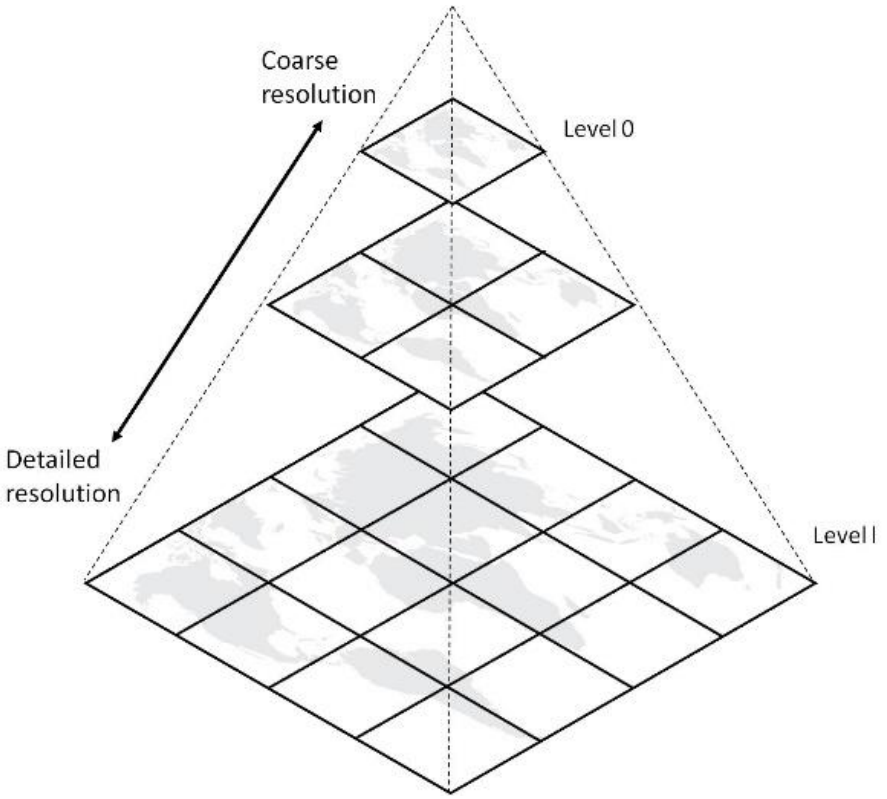


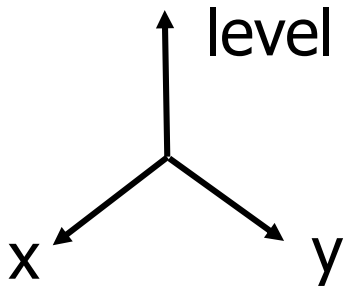
Overview

1. discrete Levels of Detail (scale levels)
2. vario-scale for point cloud data
3. post-processing to get of discrete levels
4. pre-processing to reduce effect (refined discrete levels)
5. continuous levels
6. conclusion

Scale as dimension

- less obvious than space or time
- data pyramids
  (Level of Detail/ Multi-scale)
- well-known from raster data
  (and vector tiles)
- results in discrete
  number of levels (multi-scale)
- level could be considered
  as additional dimension

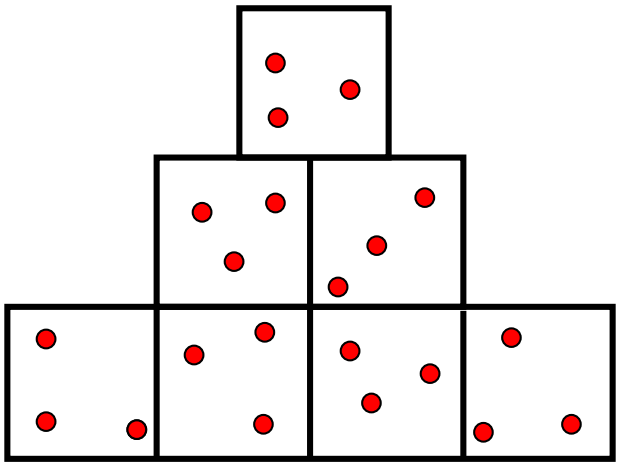




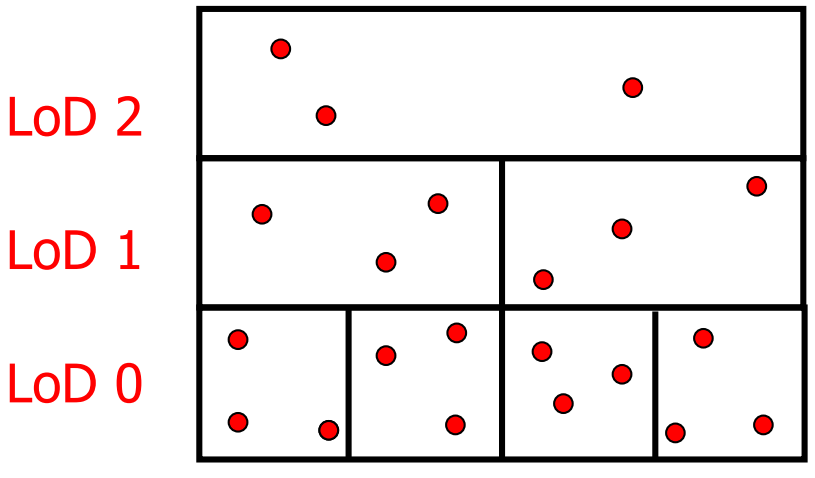
Point cloud data pyramid

- overview queries just want top-subset
- detailed queries part of bottom-subset
- organize in data pyramid

2D schematic view, data blocks….



stretched over domain



density
low

high

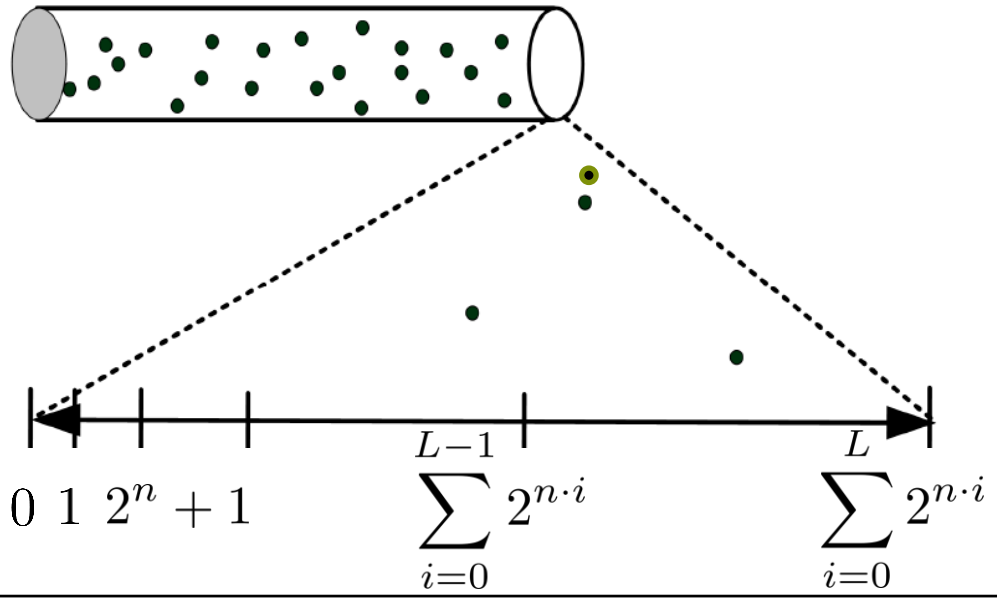
every next higher level, density 2^n times less: 1D, n=1 → 2 times less;
2D, n=2 → 4 times less; 3D, n=3 → 8 times less

Random LOD
ideal distribution among levels

- N_l is number of points in nD space on level l (l=0: top, l=L bottom)
- $N_0 = 2^{n*0}$, $N_1 = 2^{n*1}$, …… , $N_L = 2^{n*L}$
- the $L+1$ ranges of the levels $(0, 1, \ldots L)$ are defined as

$$[0,1],[1,2^n+1],[2^n+1,2^{2n}+2^n+1],\ldots,[\sum_{i=0}^{L-1}2^{n\cdot i},\sum_{i=0}^{L}2^{n\cdot i}]$$

- uniform random sampling used to distribute point over the level ($L+1$ ranges):
- implemented in SFClib (fast C++ parallel)

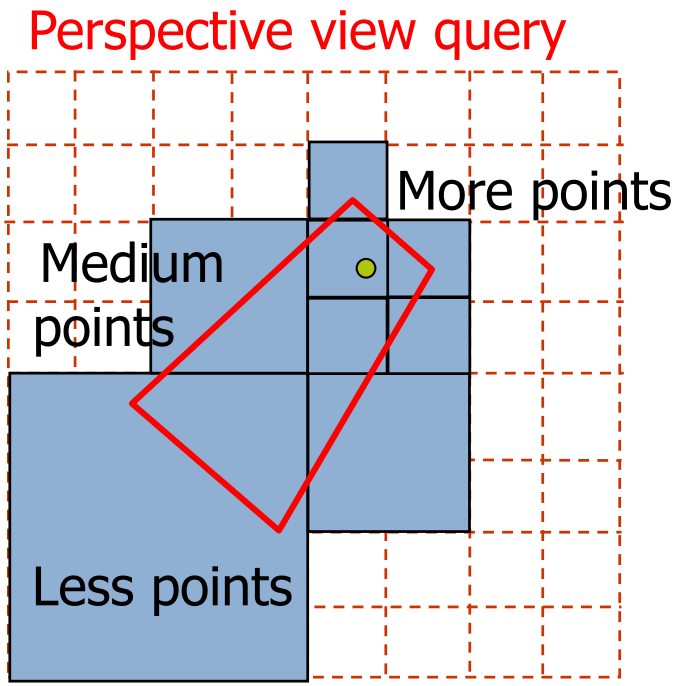


Data pyramid/multi-scale

- allows fast spatial searching including LoD selection

- the further away from viewer the lesser points selected (i.e. the higher level blocks/points)

- drawback:
  discrete number of levels are visible (suboptimal)



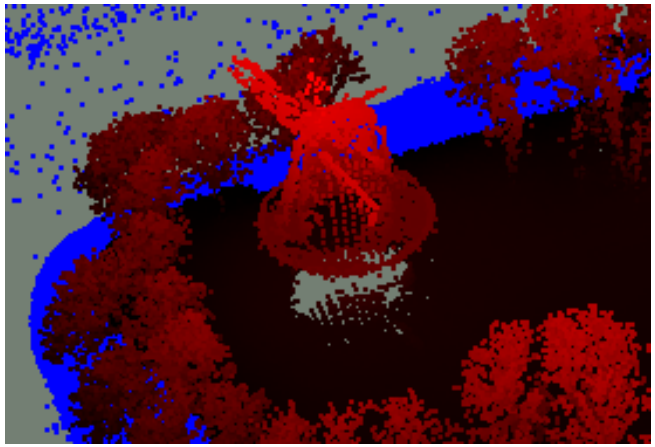
Overview

1. discrete Levels of Detail (scale levels)
2. vario-scale for point cloud data
3. post-processing to get of discrete levels
4. pre-processing to reduce effect (refined discrete levels)
5. continuous levels
6. conclusion

Vario-scale for point cloud data

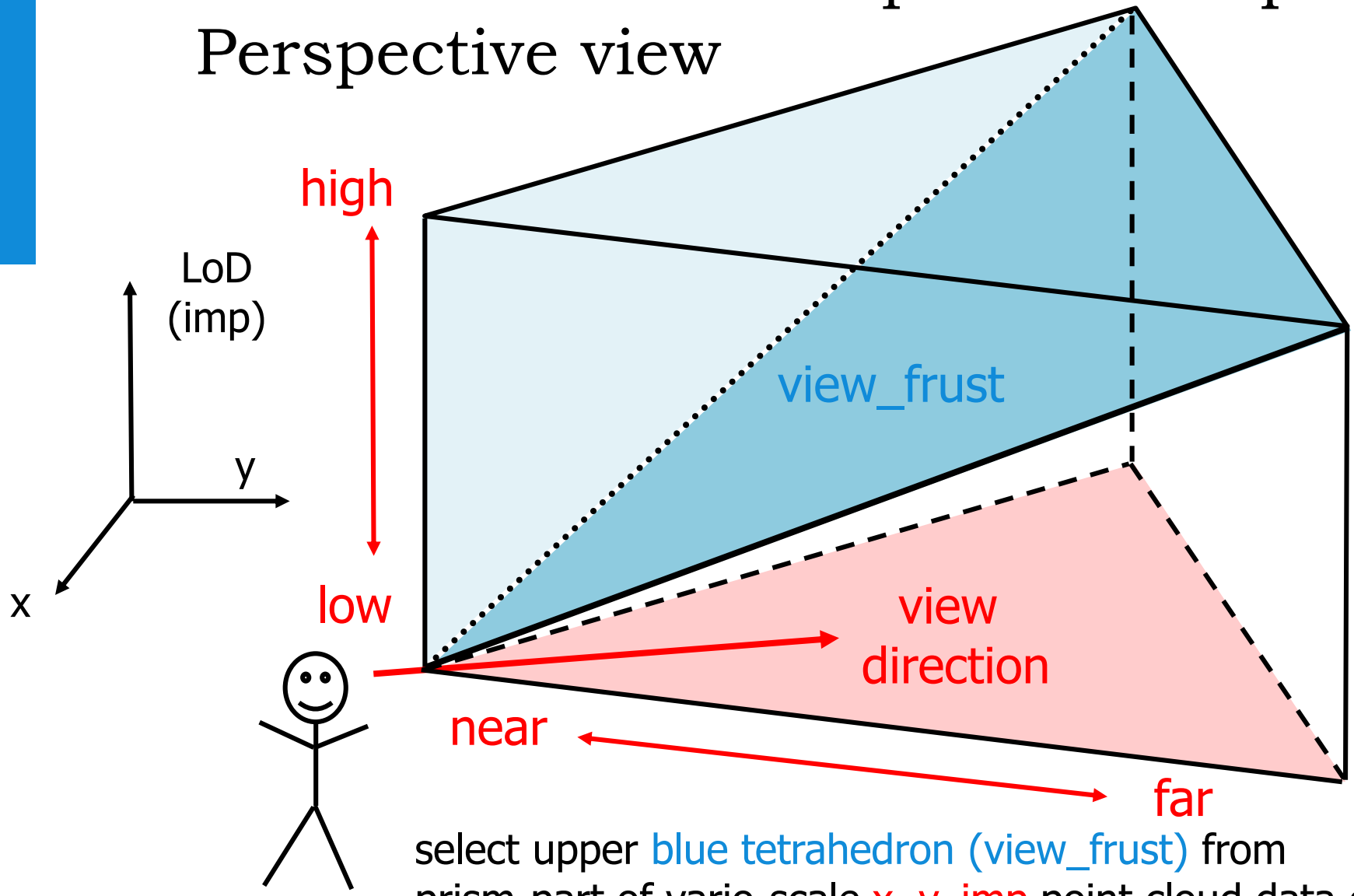
- lesson from vario-scale research: add one continuous dimension to the geometry to represent scale
  (2D data vario-scale represented by 3D geometry)

- apply this to point cloud data
  1. compute the imp value per point
  2. add this as dimension, either
     x, y, imp (z and others attributes) or
     x, y, z, imp (and others as attributes)
  3. Cluster/index the 3D or 4D point
  4. Define perspective view selections,
     view frustum with one more dimension:
     the further, the higher imp's



nD-PC trans-dimension space-scale query

Perspective view



select upper blue tetrahedron (view_frust) from prism-part of vario-scale x, y, imp point cloud data cube

Normal view frustum selection and streaming based on importance

- view frustum selection (pseudo code)

```
select point
from point_cloud
where overlaps (point, view_frust)
```

- ordered on importance for streaming add `order by imp desc;` (or distance from tilted plane)

Delta queries for moving and zoom in/out (in VR/AR environments)

- select and send new points (pseudo SQL):
  `point in new_frust and point not in old_frust`
- find and drop old points:
  `point in old_frust and not in new_frust`
- note this works for both
  1. changing view position x,y(,z)
  2. zooming in or out ('view from above', imp-dimension)
- optional to work at point or block granularity
  (in selection and server-client communication)

Overview

1. discrete Levels of Detail (scale levels)
2. vario-scale for point cloud data
3. post-processing to get of discrete levels
4. pre-processing to reduce effect (refined discrete levels)
5. continuous levels
6. conclusion

Getting rid of discrete level postprocessing (1/2)

Real-Time Continuous Level of Detail Rendering of Point Clouds

Markus Schütz*
TU Wien

Katharina Krösl†
TU Wien,
VRVis Forschungs-GmbH

Michael Wimmer‡
TU Wien

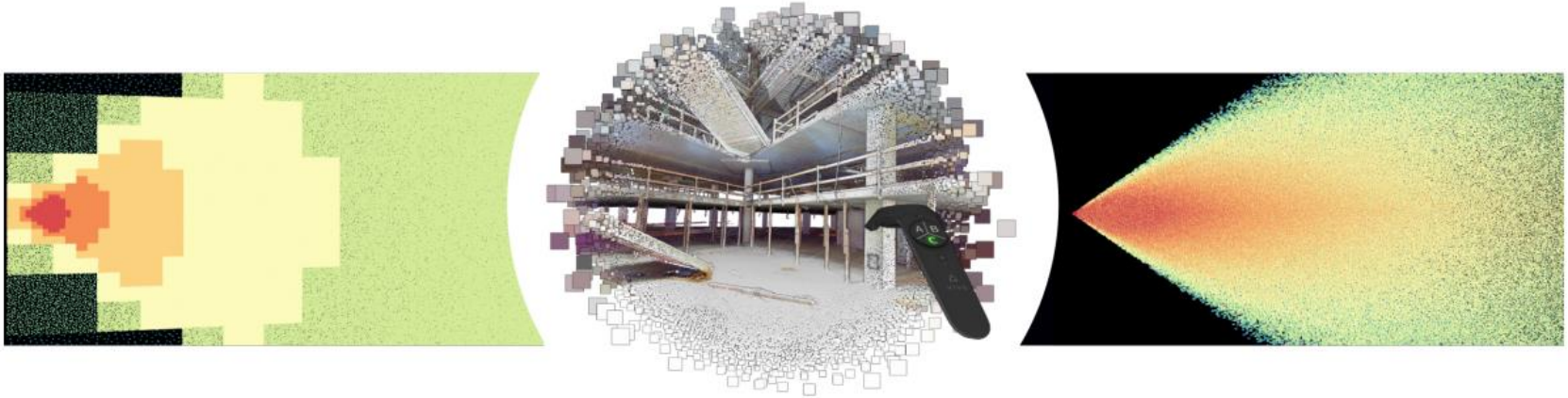
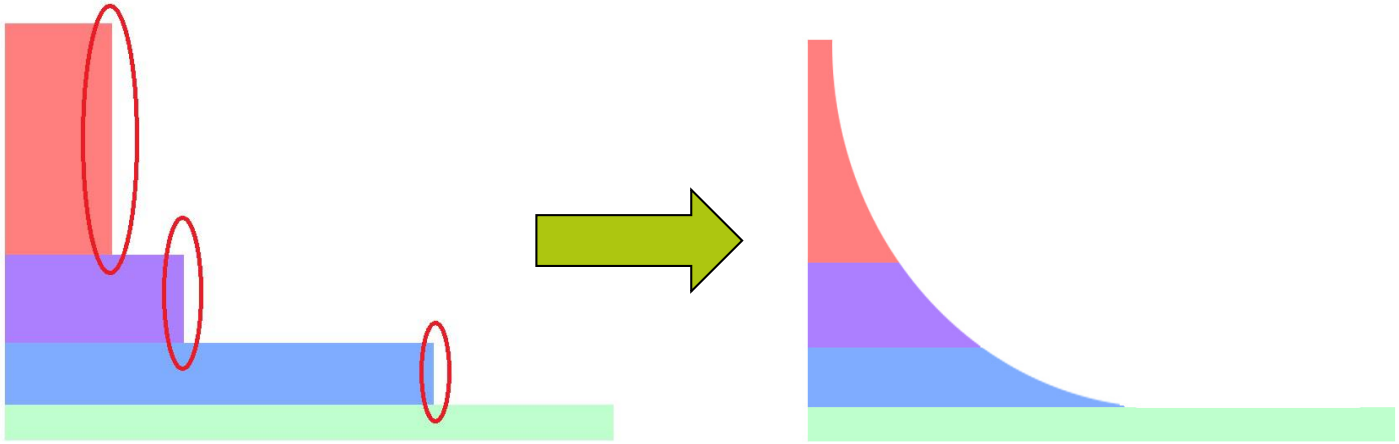


Figure 1: Left: Discrete LOD structure with sudden jumps in density and popping artifacts under motion. Middle: Screenshot of our method in virtual reality. No visible seams across different levels of detail due to a continuous reduction in density, and also continuously reduced density towards the periphery to reduce geometric complexity where less is needed. Endeavor point cloud courtesy of NVIDIA [4]. Right: Continuous transition from one LOD to another.

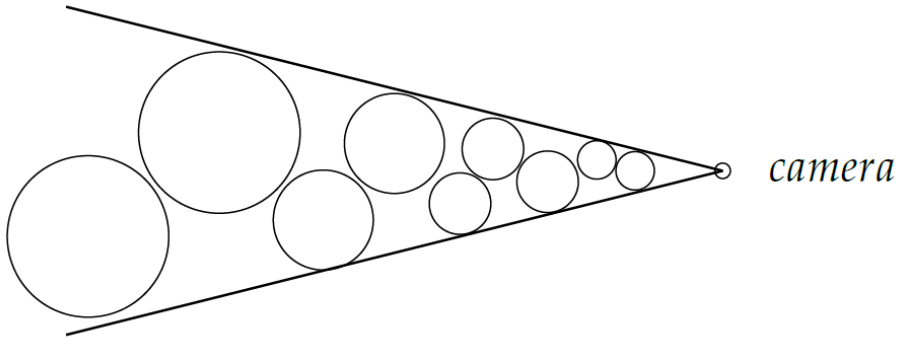
- paper at IEEE VR 2019 (postprocessing at GPU, from 86.000.000 to 5.000.000 points every 5 to 6 frames → 90 fps left+right = 180 fps)

Getting rid of discrete level postprocessing (2/2)

- MSc Geomatics thesis 'Vario-scale visualization of the AHN2 point cloud' by Jippe van der Maaden (TU Delft, April 2019).



- assign radius to points based on distance to camera
- select points with empty circle/sphere
- remove 70-90% of points

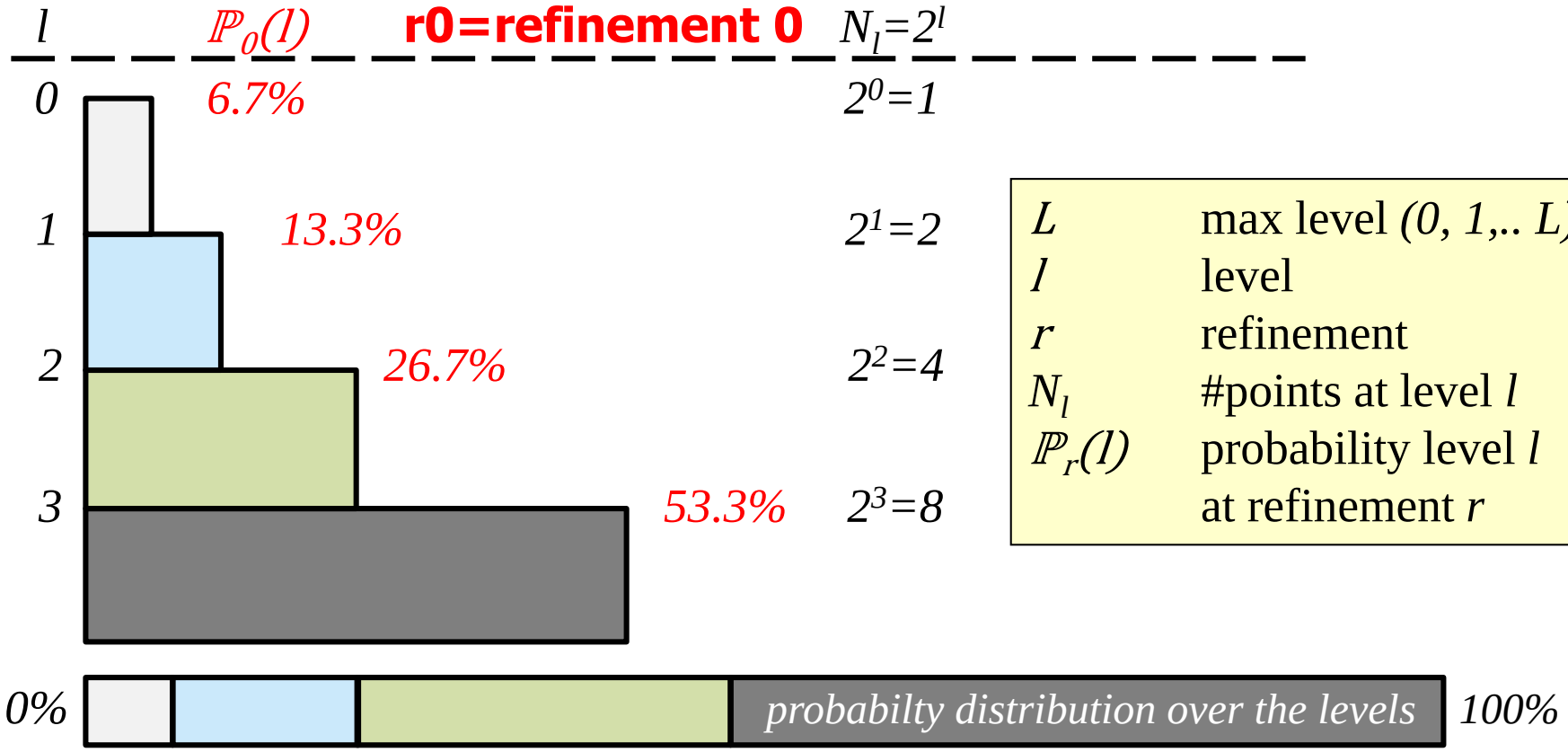


Overview

1. discrete Levels of Detail (scale levels)
2. vario-scale for point cloud data
3. post-processing to get of discrete levels
4. pre-processing to reduce effect (refined discrete levels)
5. continuous levels
6. conclusion

Ideal distribution refined LoD r0

- 1D example, duplicate data (per dimension) for next level: $N_l=2^l$
- same principle as vector, raster LoD's (and quadtree/octree approach)



L	max level $(0, 1,.. L)$
l	level
r	refinement
N_l	#points at level l
$\mathbb{P}_r(l)$	probability level l at refinement r

example with L=3 (4 levels)

Mathematics behind $P_0(l)$

- probability point x assigned to level l at refinement 0

$$\mathbb{P}_0[x \to l] = \frac{2^l}{\mathrm{Tot}_0}$$

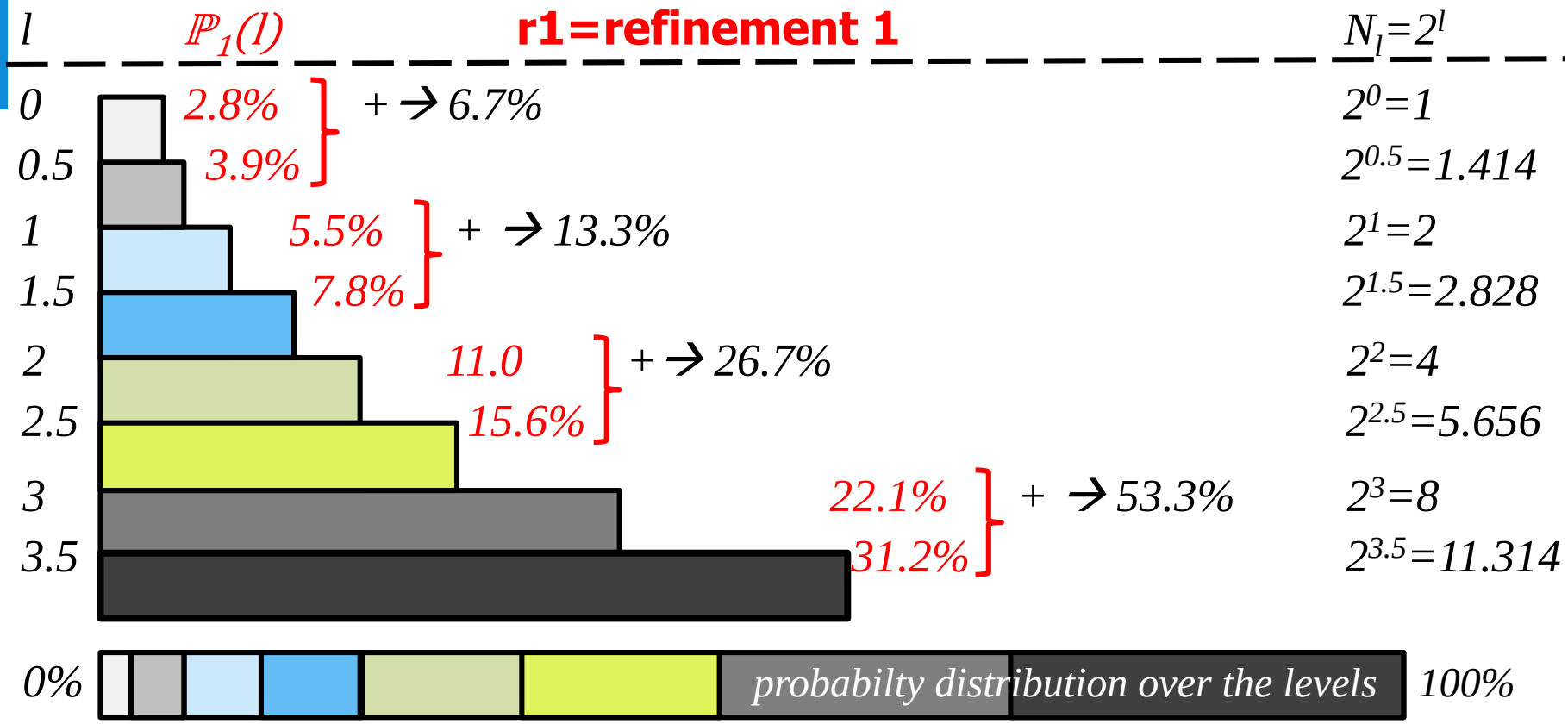
- total number of points at refinement 0 summed over all levels

$$\mathrm{Tot}_0 = \sum_{l=0}^{L} 2^l$$

- note: this is the 1D case
  if nD, then replace the 2 in the above formulas by 2^n

Ideal distribution refined LoD r1

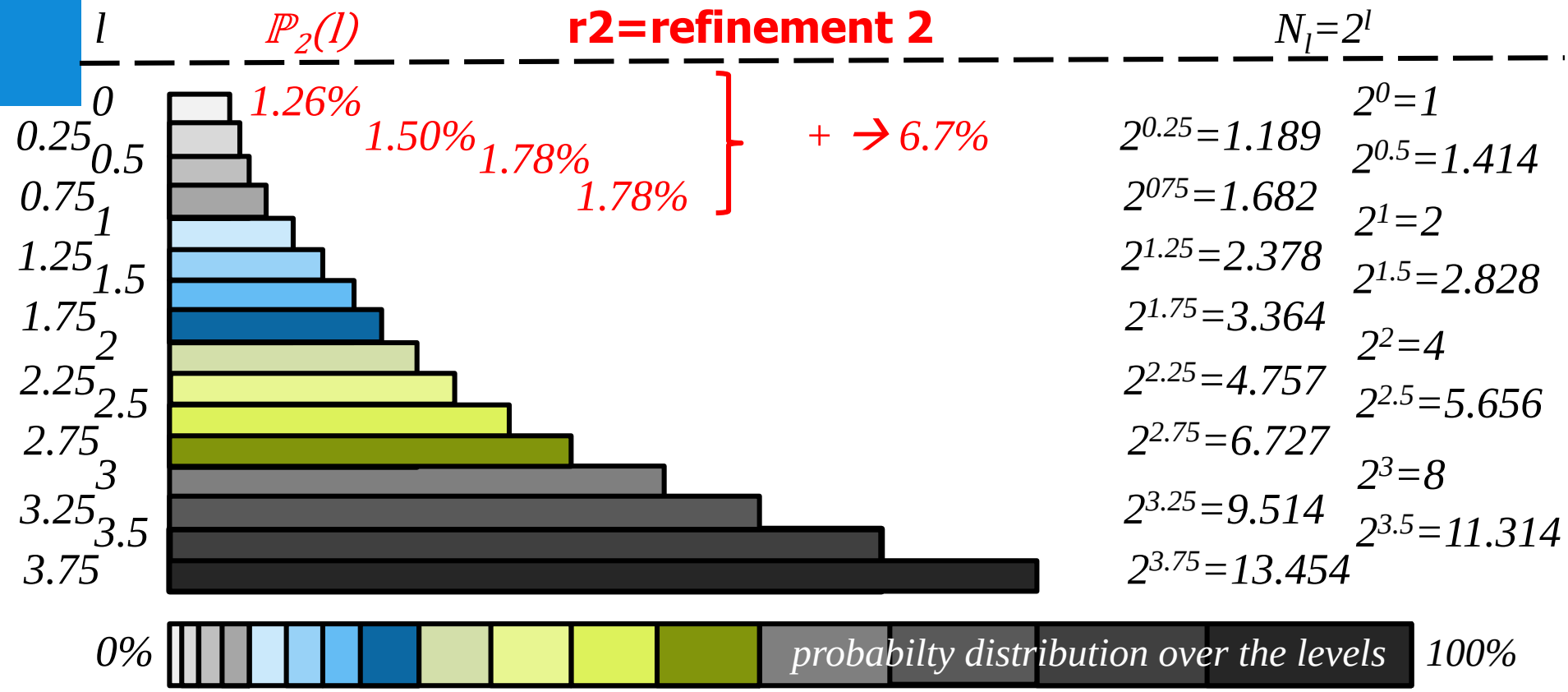
- next to integer levels also compute ideal data amount for half levels
- use exactly same for formula: $N_l=2^l$



summing *r1* probabilities *l=0* and *l=0.5* totals to *r0* probability for *l=0*

Ideal distribution refined LoD r2

- next refinement, also compute for quarter levels
- use exactly same for formula: $N_l = 2^l$



note: Ideal distribution is maintained (check by summing first 4 levels)

Mathematics behind $P_r(l)$

- probability point x is assigned to level l at refinement r

$$\mathbb{P}_r[x \to l] = \frac{2^l}{\mathrm{Tot}_r}$$

- total number of points at refinement r summed over all levels

$$\mathrm{Tot}_r = \mathrm{Tot}_0 \prod_{i=1}^{r} (1 + 2^{1/2^i})$$

- note: this is the 1D case

The discrete dimension level corresponds to data density

- assume:
  - N = number of points in dataset
  - E^n = size of spatial domain in nD case (data cube, equal size all dims)
- then overall data density: $D = N/E^n$
- discrete probability function (nD case) at refinement r:

$$\mathbb{P}_{r,n}[x \to l] = \frac{2^{n \cdot l}}{\mathrm{Tot}_{r,n}}$$

- density at discrete level l at refinement r

$$D_{r,n}(l) = \mathbb{P}_{r,n}(l) \cdot N/E^n$$

→ direct linear relation between level and density!

Overview

1. discrete Levels of Detail (scale levels)
2. vario-scale for point cloud data
3. post-processing to get of discrete levels
4. pre-processing to reduce effect (refined discrete levels)
5. continuous levels
6. conclusion

Getting rid of (refined) discrete levels → real continuous levels, 1D case

- for ideal continuous function over levels: $r \to \infty$

$$f(l) = \frac{2^l \ln 2}{2^{L+1} - 1}$$ for l between 0 and $L+1$

- this function has Cumulative Distribution Function (CDF):

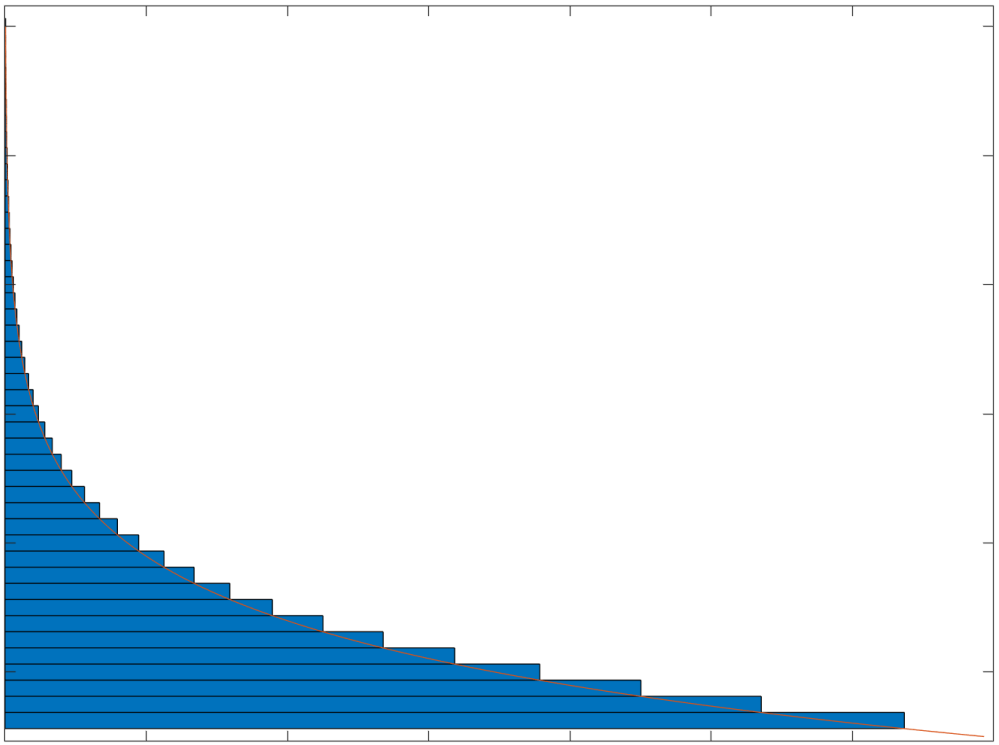
$$F(l) = \frac{2^l - 1}{2^{L+1} - 1}$$ for l between 0 and $L+1$

- using random generator U (uniform between 0 and 1) to generate level l (between 0 and $L+1$) for next point:

$$l = F^{-1}(U) = \frac{\ln((2^{L+1} - 1)U + 1)}{\ln 2}$$

Getting rid of discrete levels implementation

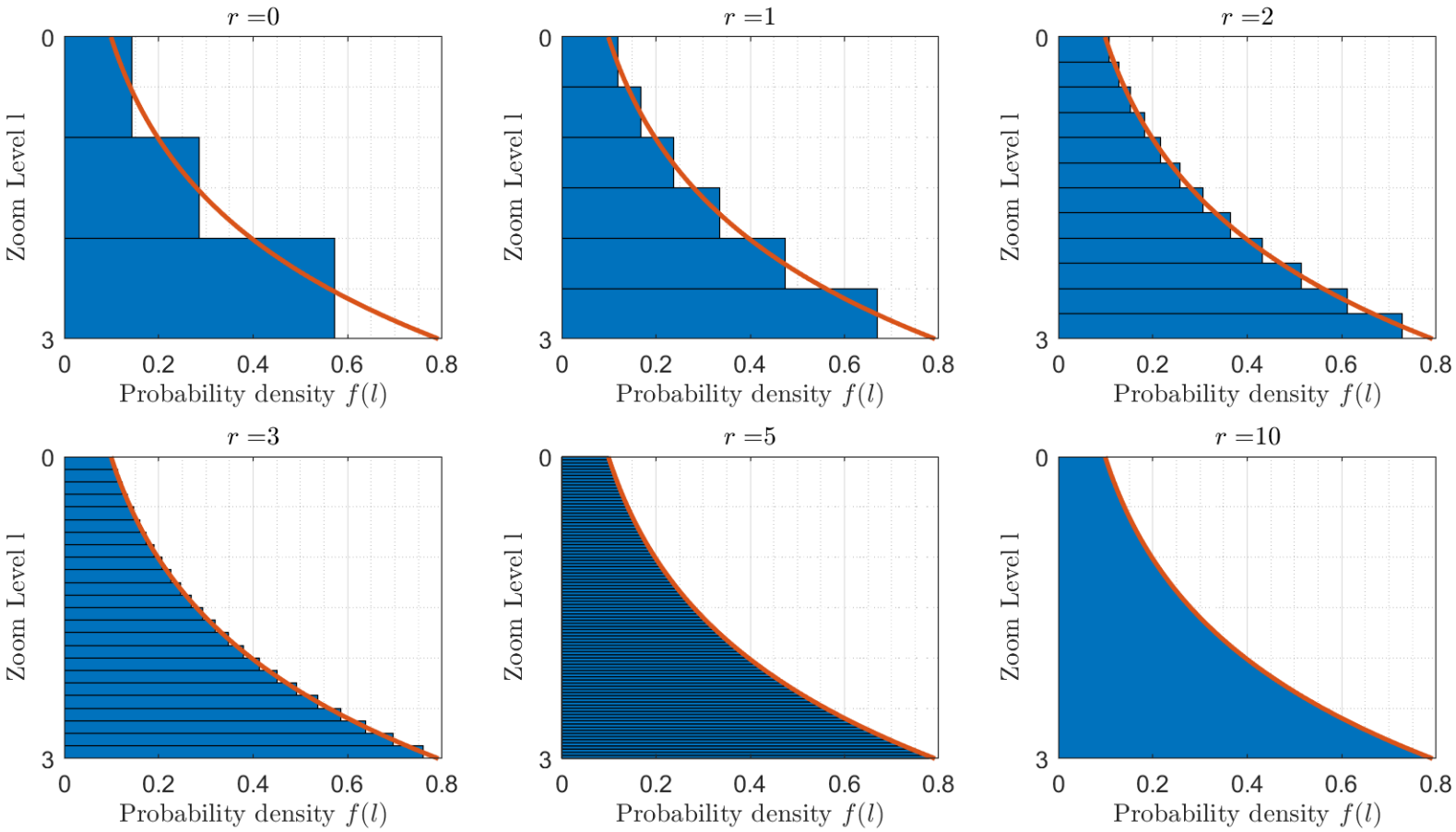
- refined ideal discrete probability and continuous function, 1D case (Simon van Oosterom, Matlab)
- example L=10 (11 levels: 0..10), R=2 refinements 22=4 sublevels:



Matlab: refined discrete levels

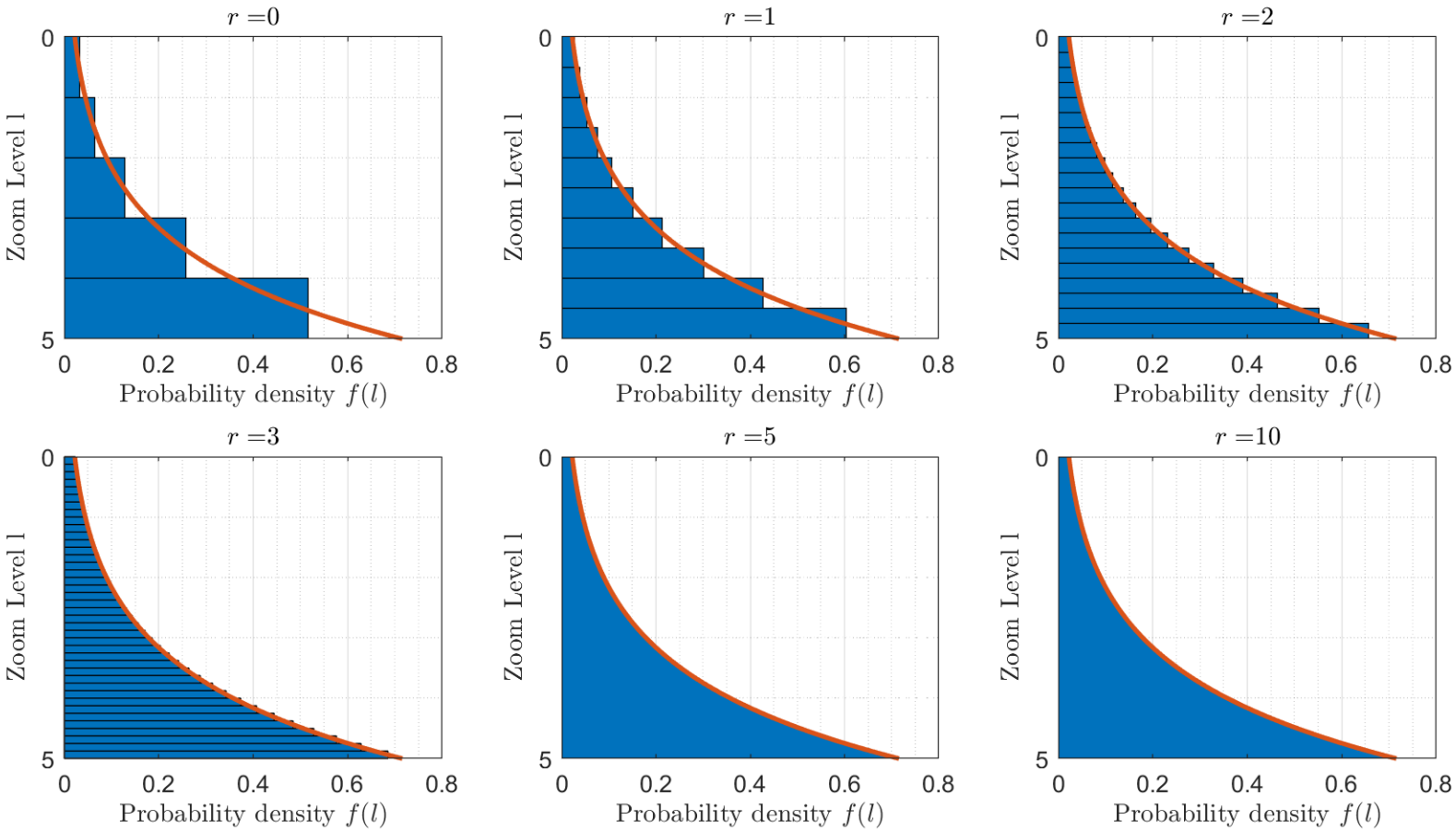
```
L = 2;
R = [0,1,2,3,5,10];

/* plot, loop over R and call function Prob(r,L) */

function [l,P] = Prob(r,L)
    l = 0:0.5^r:L+1-0.5^r;  /* loop beg:step:end */
    P = (2^l)/Tot(r,L);
end

function tot = Tot(r,L)          /* recursive def */
    if r == 0
       tot = 2^(L+1)-1;
    else
       tot = (1+2^(1/2^r))*Tot(r-1,L);
    end
end
```

Refined discrete levels (L=2: 0, 1, 2)

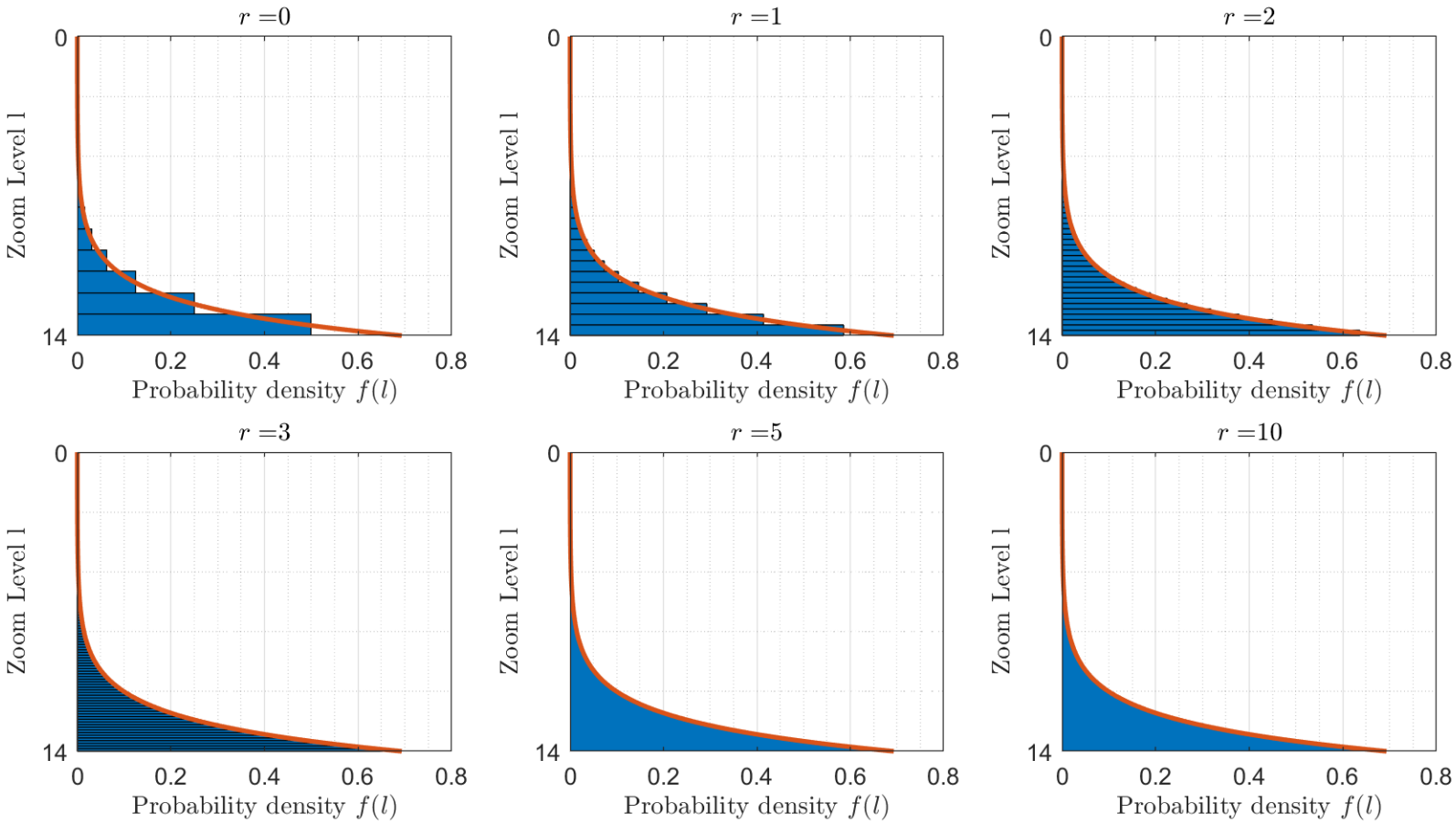


blue bars: refined discrete, red curve: continuous function

Refined discrete levels (L=4: 0, 1, ..4)



Refined discrete levels (L=13: 0, 1, …13) → 14 AHN2 levels



Getting rid of (refined) discrete levels → real continuous levels, nD case

- for ideal continuous function over levels (nD):

$$f(l,n) = \frac{2^{(n-1)l}(n-1)\ln 2}{2^{(n-1)(L+1)} - 1}$$

for l between 0 and $L+1$ and n number of dimensions

- this function has Cumulative Distribution Function (CDF):

$$F(l,n) = \frac{2^{(n-1)l} - 1}{2^{(n-1)(L+1)} - 1}$$

for l between 0 and $L+1$ and n number of dimensions

- using random generator U (uniform between 0 and 1) to generate level l (between 0 and $L+1$) for next point in nD space:

$$l = \frac{\ln((2^{(n-1)(L+1)} - 1)U + 1)}{(n-1)\ln 2}$$

The continuous dimension level also corresponds to data density

- 'thickness' of a continuous level is 0 → density at that level is also 0 therefor cumulative density, summed from top (level 0) to level l

- use Cumulative Distribution Function (CDF) for nD case:

$$F(l,n) = \frac{2^{(n-1)l} - 1}{2^{(n-1)(L+1)} - 1}$$

with l between 0 and $L+1$ and n number of dimensions

- Cumulative Density (CD) at continuous level l for nD case:

$$CD(l,n) = F(l,n) \cdot N/E^n$$

with N total points in dataset E^n size spatial domain nD case

→ Direct linear relation between continuous level - cumulative density!

The influence of L (#levels -1)

- discrete integer levels as inspiration for a good distribution
- in 2D about 80% of the data is at lowest level
  (and in 3D about 90% at lowest level)
  → Let's roughly say all data are at lowest level
- total data set (say 640.000.000.000), #points per block (say 10.000)
  → need 64.000.000 blocks
- in 2D with L=13 we can host $4^{13} = 67.108.864$ blocks at level 13
  (the lowest of the 14 levels named level 0, level 1, ..., level 13)
  → enough and indeed the depth of the current AHN2 octree
- in general L needed in nD case: $L = \lceil \frac{\ln(\#\text{ blocks})}{\ln(2^n)} \rceil$ (for 2D: $2^n = 4$)

But, does L really matter?

- as long as L is large enough...
- for 2D, 10.000 points/block, L=31
  46.116.860.184.273.879.040.000 ($>10^{22}$)
  points can be stored at lowest level
- foes it matter when fewer points are stored and their l dimension is computed with assuming L=31 (instead of L=13 in case of AHN2)?
  - probably not, just remember that least important points (the majority) get level value l between 31 and 32
  - needed when creating a proper space-scale (xyz + l) predicate in the where-clause of the query

Overview

1. discrete levels of detail (scale levels)
2. vario-scale for point cloud data
3. post-processing to get of discrete levels
4. pre-processing to reduce effect (refined discrete levels)
5. continuous levels
6. conclusion

Conclusion

- nD-PointClouds as 3rd representation: direct use (storage, analysis, visualization) or conversation to vector or raster

- nD-PointClouds better for vario-scale than
  - raster: factor 2 data pyramid, discrete, 'redundant"
  - vector: continuous, but hard to keep topology

- ideal point distribution over LoD via random sampling
  - keeps relative point density, may me relevant
  - can be done in parallel and very efficient

- other than random is possible, but hard to get ideal distribution (1. distance/density based, 2. semantics based)

- mapping from level to expected density is possible
  → allows for gradual perspective views (and/or focus views)

Future work

- implement nD-PointCloud with ideal continuous level dimension (for AHN, use xy+l to organize data)
- store nD-PointCloud in database using Space Filling Curve (cluster/index) at point and/or block level
- create/adapt 3D interactive nD-PointCloud web-viewer
  - stable continuous zoom (same point not on/off when zooming in) as supported with our method and
  - perspective/focus view (required density based on view distance)
- larger data sets; e.g. dense matched high resolution areal images NL nationwide dataset expects 60 trillion 60.000.000.000.000 points
- temporal point cloud, also time dimension in organization
- more dimensions in organizations (and same number of points) the more sparse some parts of the nD space gets → use nD-histogram