New 3D Projection Transformation for Point Clouds

by

Alvaro Vázquez and Elisardo Antelo

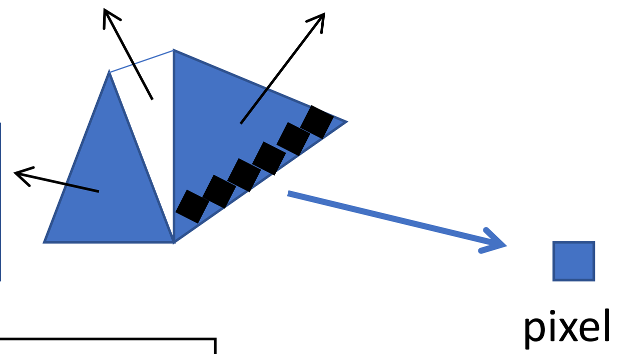
University of Santiago de Compostela

Spain

Polygon vs Point Rendering in 3D Computer Graphics (Animation)

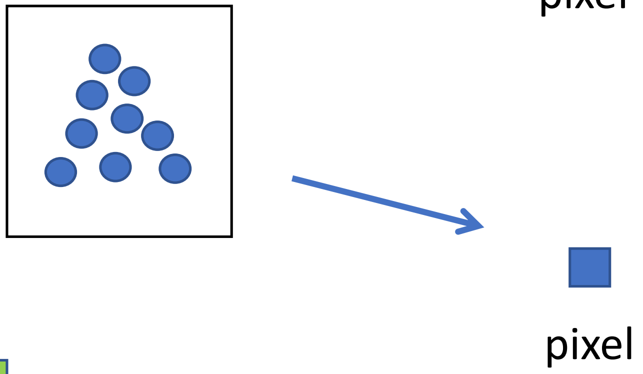
Conventional CG sytem: **polygoms for rendering**:

Fron-end: vertex
Back-end: pixel.



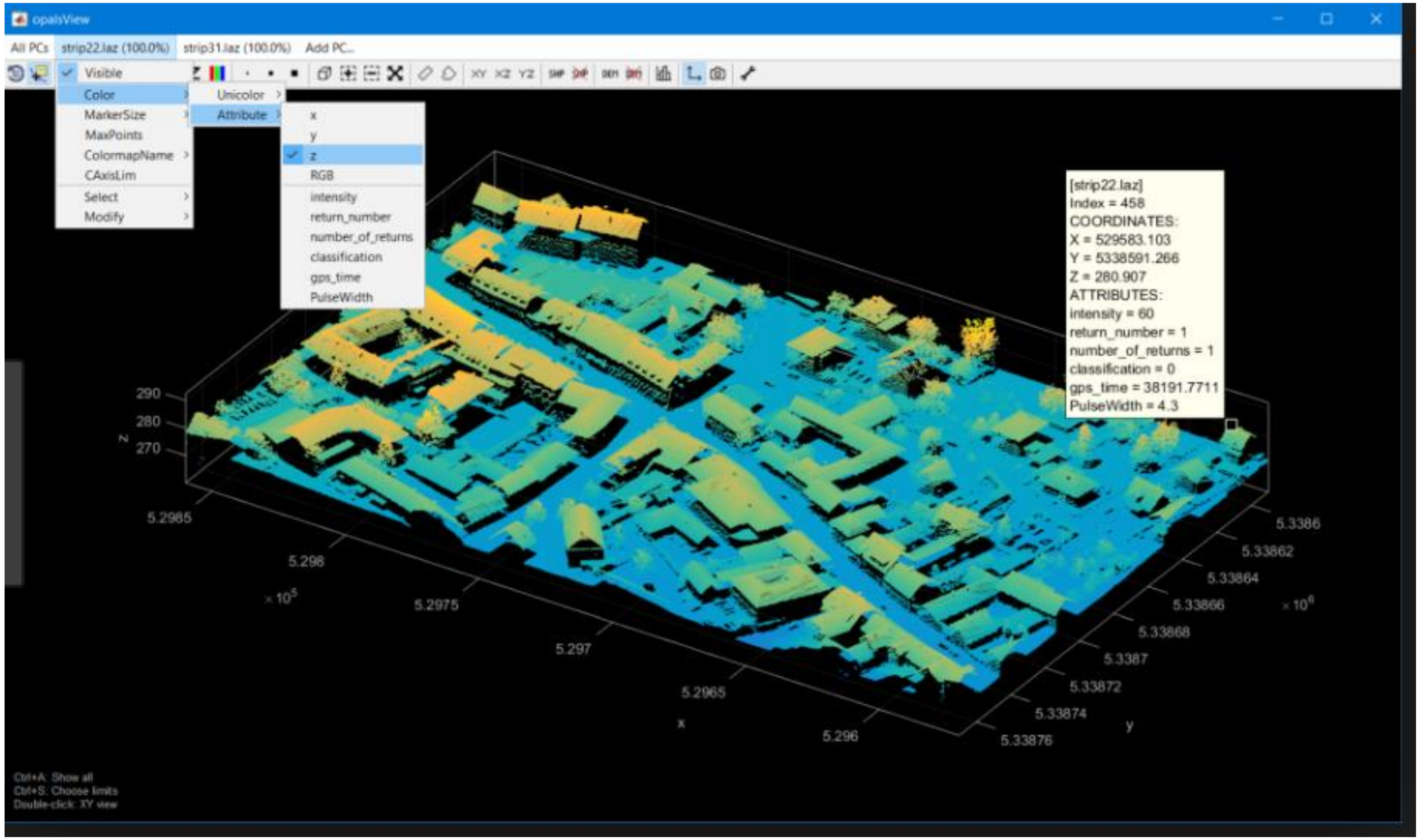
Alternative ⟶ Points

- For complex geometry
- Connectivity information is not necessary or not provided.
- Examples: 3D Scanning Systems to produce point clouds of real models.



OPALS: A Framework for Airborne Laser Scanning Data Analysis (TU Wien)
- High precision topographic data adquisition

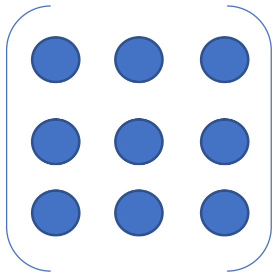
Example of Point Cloud: use supercomputers to generate digital suface models



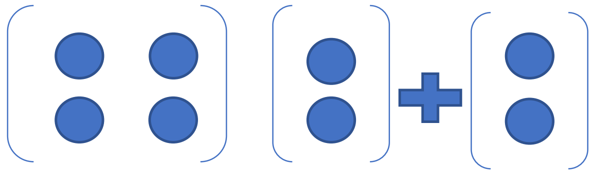
```
[strip22.laz]
Index = 458
COORDINATES:
X = 529583.103
Y = 5338591.266
Z = 280.907
ATTRIBUTES:
intensity = 60
return_number = 1
number_of_returns = 1
classification = 0
gps_time = 38191.7711
PulseWidth = 4.3
```

Point Rendering: From 2D Homogenenous Coordinates to Screen Coordinates

2 D Homogeneous Coordinates obtained applying a **3x3 transformations** (**Rotation + translation+ Scaling**): (x,y,z) [Composition EASY]



If **2x2 Rotation** + **traslation** are used – compositions can not be done - and computation is **highly inefficient**: R(2x2) + T



To obtain **screen coordinates (2D coordinates) (u,v)**:

Perpective correction (x,y,z): scale (x,y) by the factor 1/z

Point Rendering Processing Model and Data Structure

Scene: cubic resolution areas with a grid of **sxsxs**. Regular **data structure** organized as an **octree** (recursiverly transversed)

Nodes in the higher level of the octree build by subsampling by a factor of two.

The recursion process of the octree selects all nodes that are in the view fustrum.

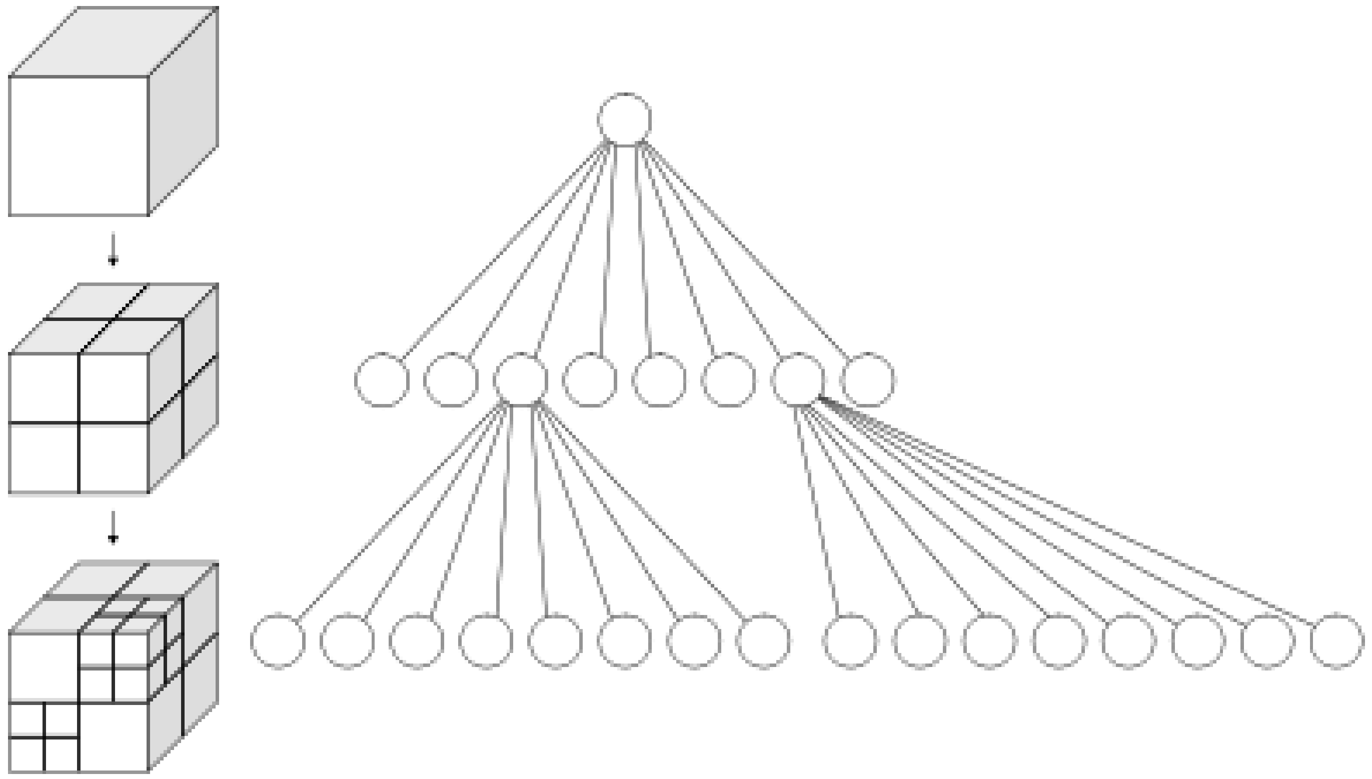
The proyection of a **sxsxs** sample grid (with points) should assure one sample per pixel on the screen.

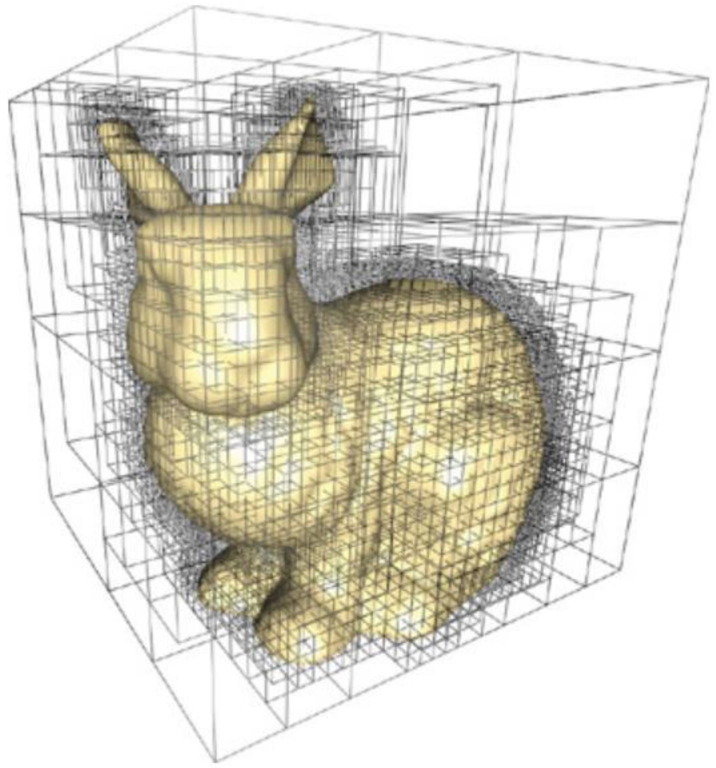
If this is not verified (more pixels), the node level is rejected and the children is analyzed (more resolution is necessary) ➡ **at least one point per pixel or more**

Point Rendering Processing Model and **Data Structure**

s x s x s (grid – s parameter)

Regular **data structure** organized as an **octree** (recursively transversed)





https://en.wikipedia.org/wiki/Octree

https://developer.nvidia.com

Simplified Perspective Correction: Computation of 1/z

As said before: (x,y,z) 2D homogeneous coordinates in camera space after composite transformations

Compute screen coordinates (u,v) from camera coordinates (x,y,z), perspective correction is performed:

$$(\ u, v \ ) = (\ (x, y) \ ) \cdot (1/z)$$

$$f(x) \approx f(x_0) + (x - x_0) \left.\frac{\partial f(x)}{\partial x}\right|_{x=x_0} + \cdots + \frac{(x-x_0)^n}{n!} \left.\frac{\partial^n f(x)}{\partial x^n}\right|_{x=x_0}$$

1) Compute (1/zr) in **Reference Points** Pr (full precision) – and copute its square $(1/zr)^2$.

2) Compute **linear approximation** of 1/z for points with **distance up to D** from Pr: USE MAC INSTEAD RECIPROCAL

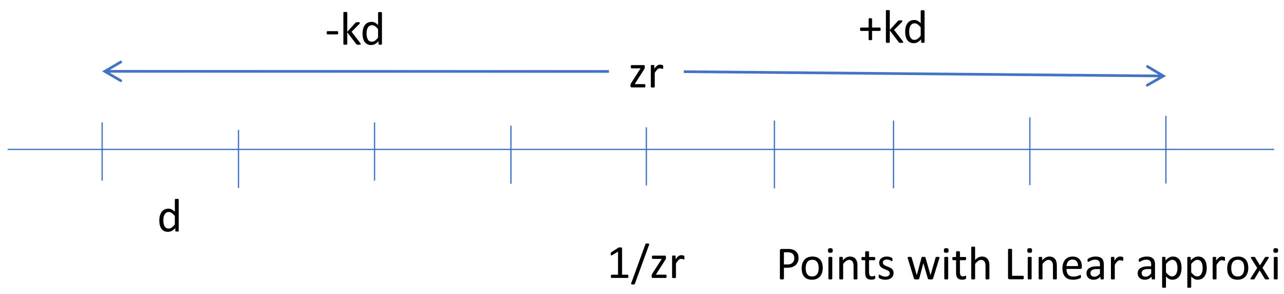
$$\frac{1}{z} = \frac{1}{z_r} - \left(\frac{1}{z_r}\right)^2 (z - z_r) = 2 \ \left(\frac{1}{z_r}\right) - z \ \left(\frac{1}{z_r}\right)^2$$

Simplified Perspective Correction: Computation of 1/z

- Trade-off among **computation error** and number of points per reference points (Pr) that use the **linear approximation (just one MAC)**.

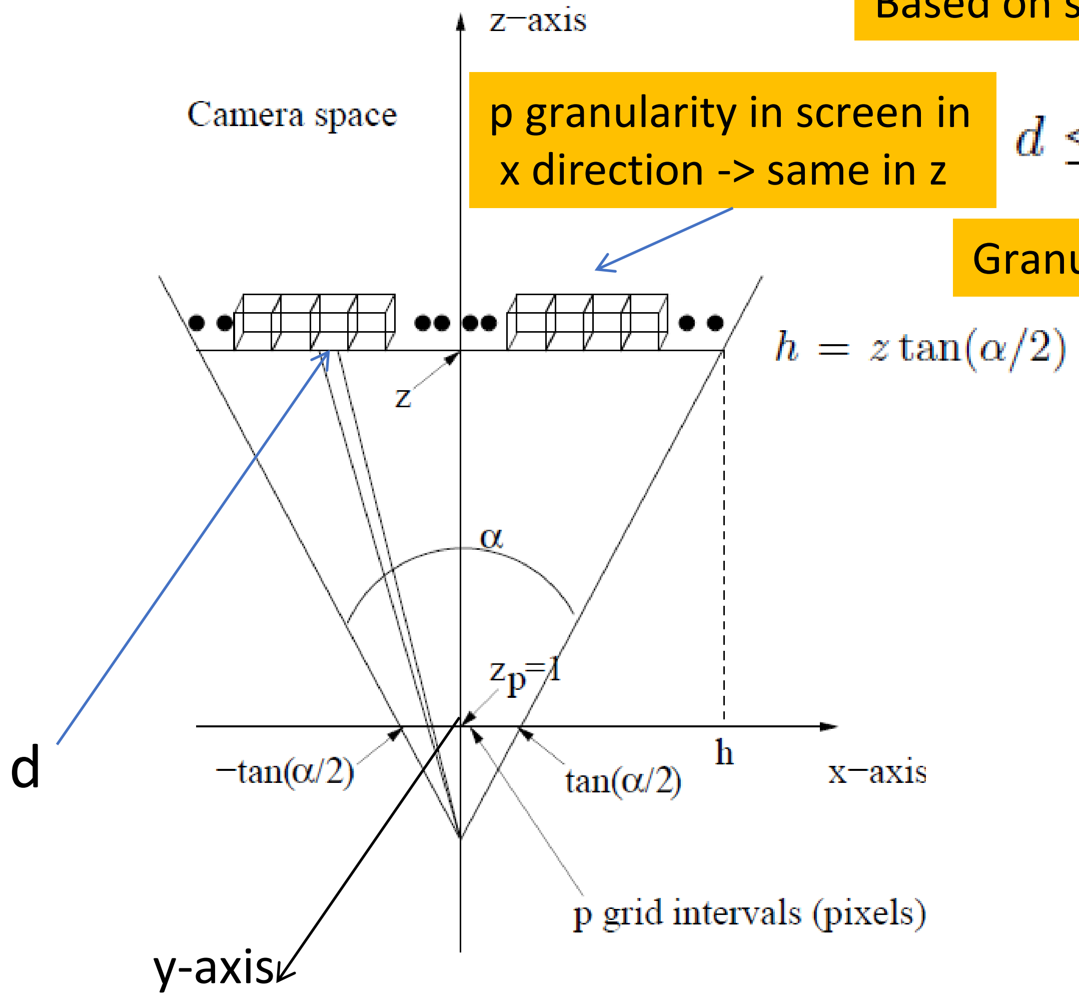
- More error implies more points per reference point using a linear approximation (just one MAC), and less reference points with full precision of reciprocal.

- Relate the error of computaton of 1/z using the linear approximation, **with distante D** to the reference point (1/zr)



Points with Linear approximation computation

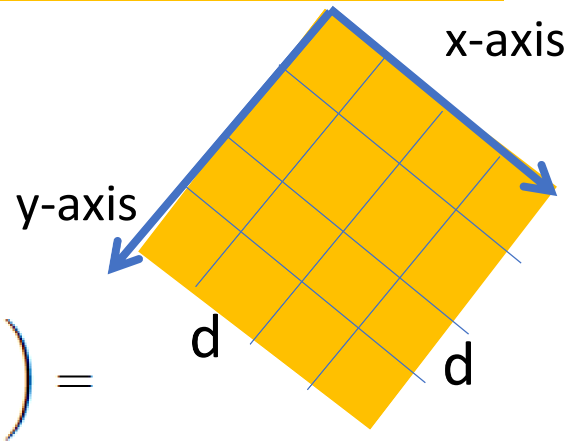
Computation Model: granularity d


z−axis
Camera space
p granularity in screen in x direction -> same in z
h = z tan(α/2)
z
α
zp=1
−tan(α/2)
tan(α/2)
h
x−axis
p grid intervals (pixels)
d
y-axis

Based on simple geometry: **d** upper bound of the granularity of the cube

$$d \leq \frac{z\tan(\alpha/2)}{p/2} = \frac{2z\tan(\alpha/2)}{p}$$

Granularity in x -> granularity in z



$$\frac{1}{z} = \frac{1}{z_r \pm kd} = \frac{1}{z_r}\left(\frac{1}{1 \pm kd/z_r}\right) =$$

$$\frac{1}{z_r}\left(1 \mp \frac{kd}{z_r} + \left(\frac{kd}{z_r}\right)^2 \mp \ldots\right)$$

Consider linear approximaton: (the same in camera space)

$$\frac{1}{z} \approx \frac{1}{z_r}\left(1 \mp \frac{kd}{z_r}\right)$$

Bound of Error:

$$\Delta \leq \frac{1}{z_r}\left(\frac{kd}{z_r}\right)^2\left(1 + 2\left(\frac{kd}{z_r}\right)\right)$$

Bound on **k** (2k grid points use the same zr)

$$d \leq \frac{2z_{min}\tan(\alpha/2)}{p}$$

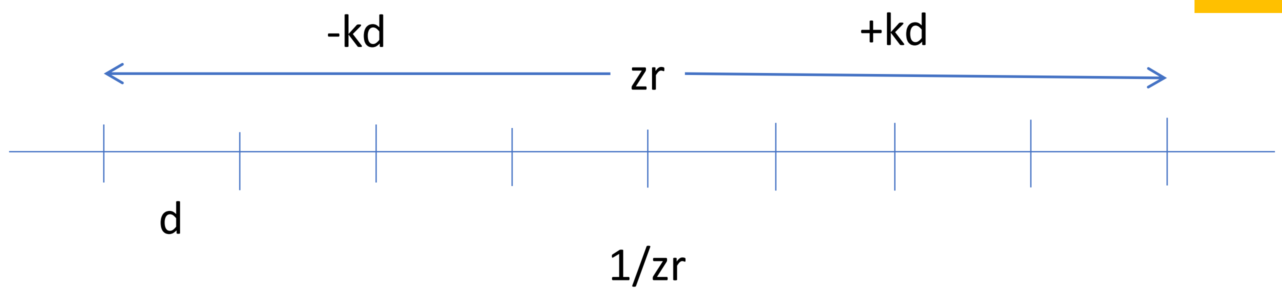
Number of pixels

Lower bound of granularity of proyected pixels

$$\text{Error in } x/z \leq f \times \left(\frac{2\tan(\alpha/2)}{p}\right) = f \times \delta$$

Bound of error in the **proyected pixel** (to compute (u,v)):
Example f=0.1 means 10% error

This bound of error allows to make linear approximation of 1/z in a bound of 2k grid points (zr-kd, zr+kd).

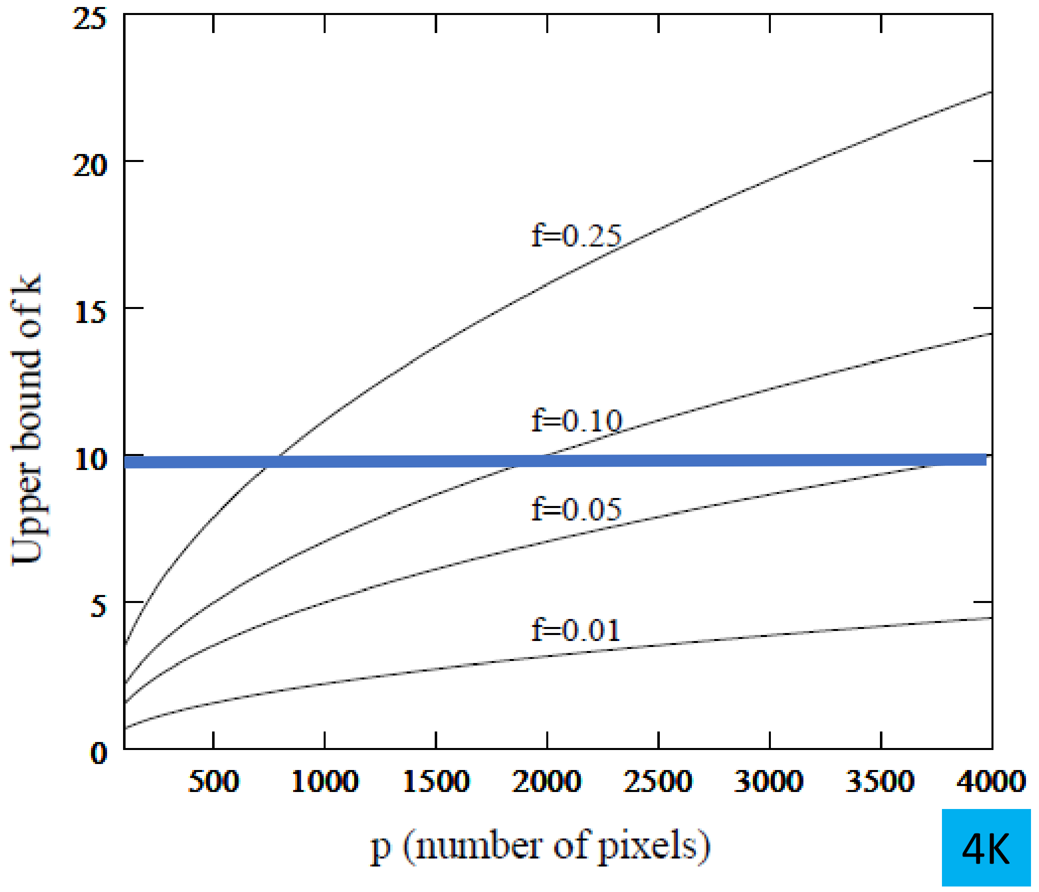


After some work on different bounds, we obtain a bound for k

$$\boxed{k \leq \sqrt{\frac{pf}{2}}}$$

$$\frac{1}{z} \approx \frac{1}{z_r}\left(1 \mp \frac{kd}{z_r}\right)$$

Upper Bound of **k** (**2k** grid points use the same **zr** for linear approximation)



For 4K Resolution and f=5% (fraction of pixel error)

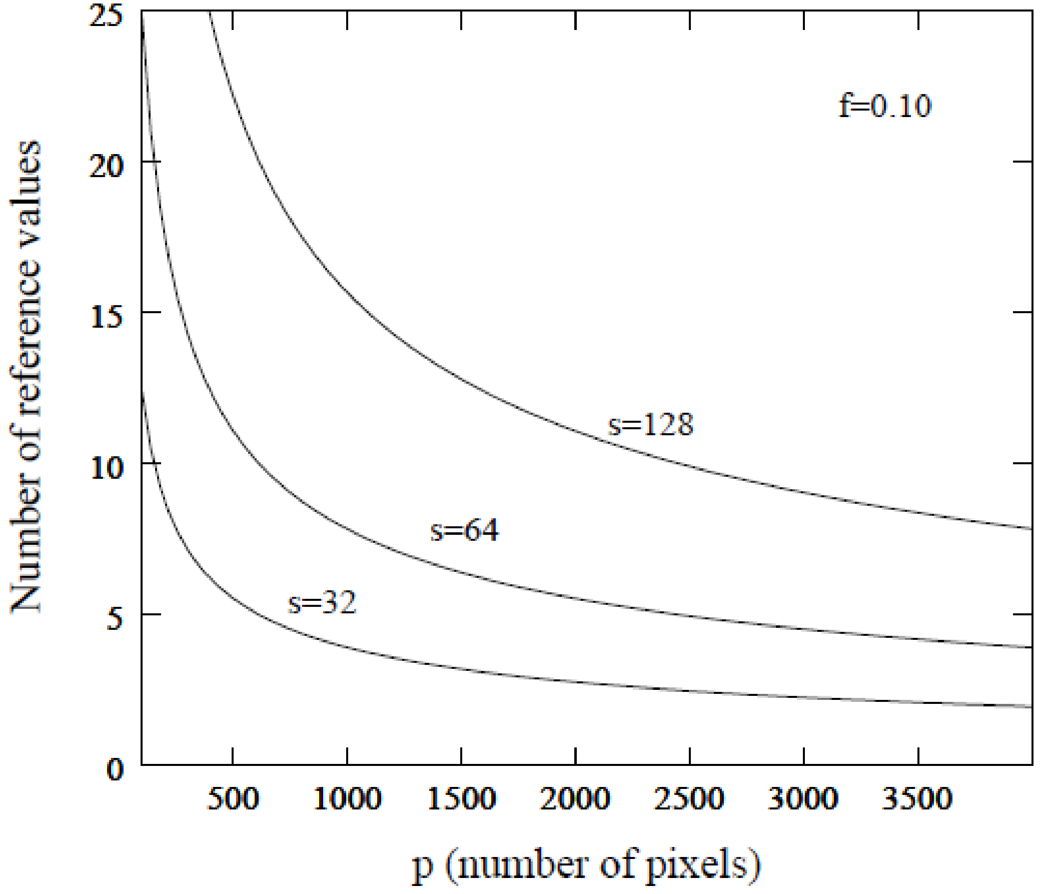
UPPER BOUND OF k=10

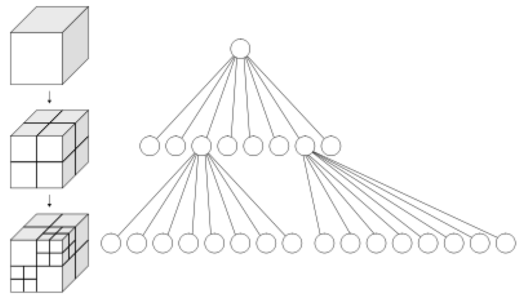
Interval of z convered by a reference point zr is **+-10d**

$$\frac{1}{z} = \frac{1}{z_r} - \left(\frac{1}{z_r}\right)^2 (z - z_r) = 2\left(\frac{1}{z_r}\right) - z\left(\frac{1}{z_r}\right)^2$$

$$\mathrm{k} \leq \sqrt{\frac{pf}{2}}$$

Upper Bound of Reference Points (r) per Cube





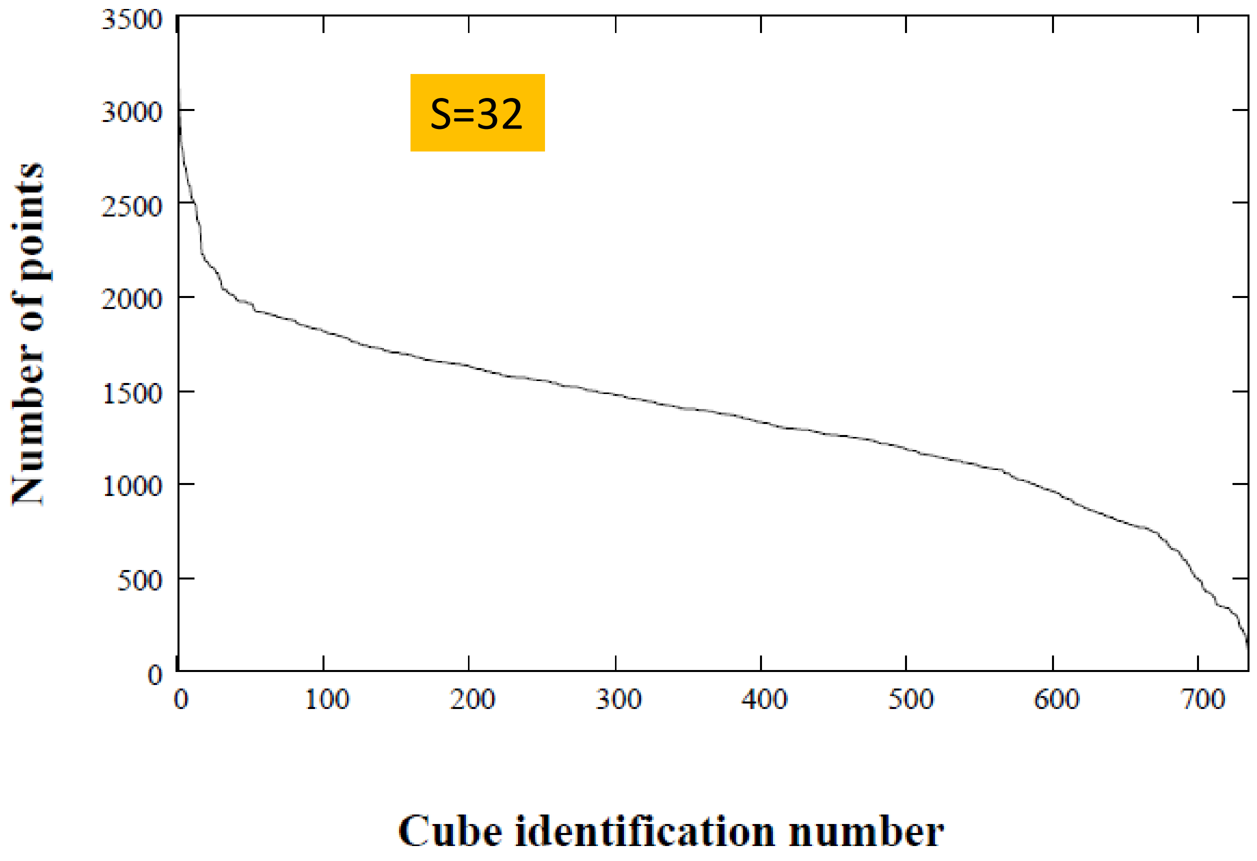
$$r = \left\lceil \frac{\sqrt{3}\ s\ d}{2\ k_m\ d} \right\rceil = \left\lceil \frac{\sqrt{3}\ s}{2\ \left\lfloor \sqrt{\frac{f\ p}{2}} \right\rfloor} \right\rceil \approx \sqrt{\frac{3}{2\ f\ p}}\ s$$

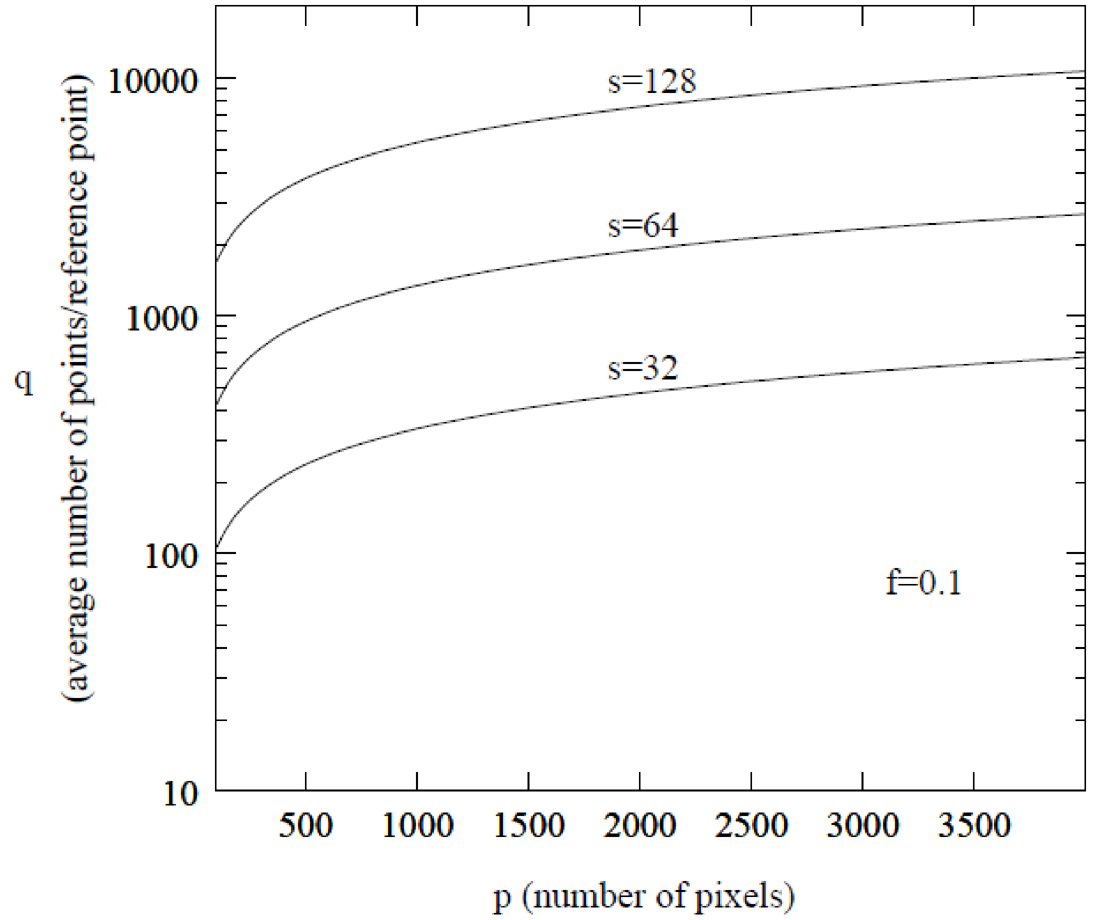
sxsxs cubic resolution areas (grid) in samples organized in **octrees**.

Reference Point: zr

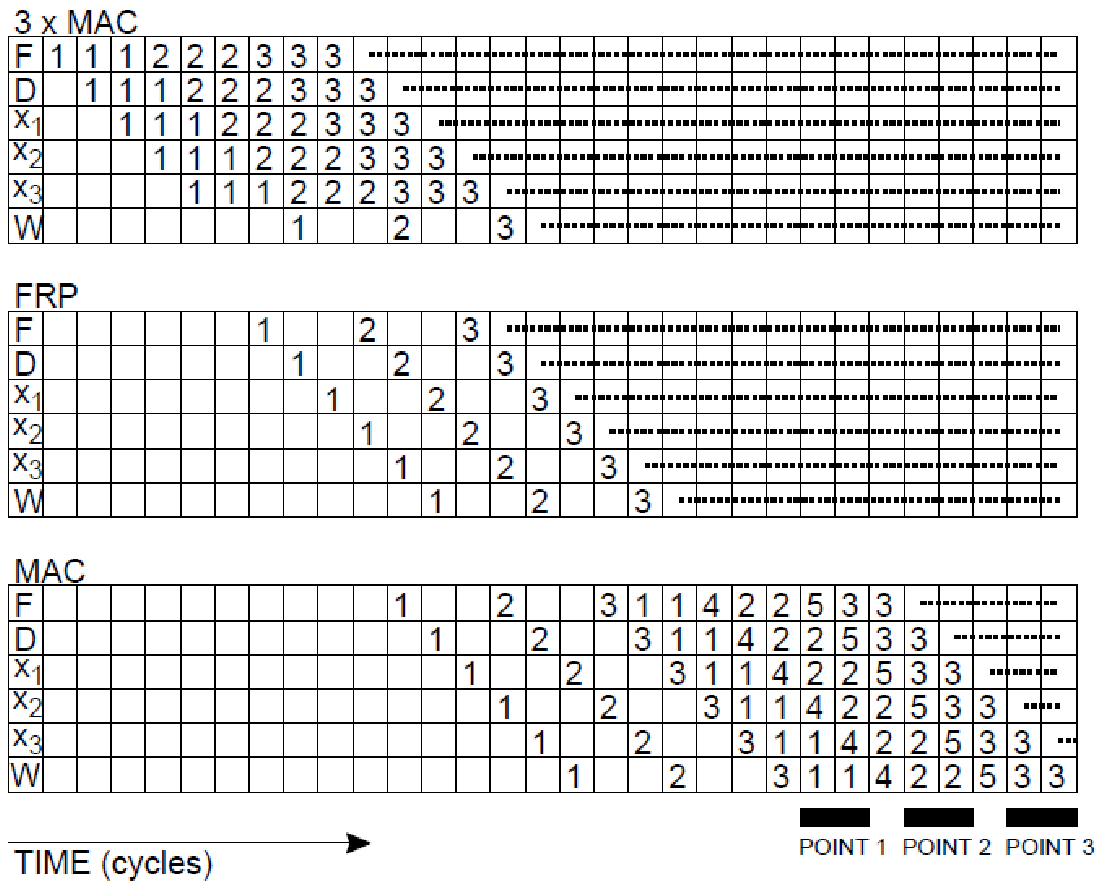
f: error of proyected pixel.

Average Number of Points per Reference Point (standard model)





Computational Model



X_1,X_2,X_3: Execution Stages

$$\frac{1}{z} = \frac{1}{z_r} - \left(\frac{1}{z_r}\right)^2 (z - z_r) = 2\left(\frac{1}{z_r}\right) - z\left(\frac{1}{z_r}\right)^2$$

- Transformation Step: done with 3FMACs
- Determination of the reference value from z coordinate using instruction FRP (basically usung table look-up).
- Compute 1/z with smal error: computation of **S+Tz** with **S=1/R[i]** and **T=-(2/R[i])^2**; S and T precomputed for all points of the reference value (Table look-ups)
- Multiplication of **computed reciprocal** by **(x,y)** requiring a FMAC.

Current Graphics hardware (5-6K SP-FPU) extended with 1.5K FRP units:
500 Points/cycle at the cost of 10% of pixel error

Conclusions

- Hardware for 3D Proyection Transformation using points as primitives.
- The 3D proyection requires a reciprocal operation on each point.
- This implies a lot reciprocal operations, and reciprocal is not an efficient operation.
- We reduced the number of reciprocal operations, introducing MAC instead reciprocal.
- The cost of this reduction in computing complexity is some controlable pixel error.
- We related all the important parameters for the design such as: number of pixels, maximum level of pixel error, cubic resolution areas (grid).
- We also showed that current graphics hardware extended with some specificic hardware could achieve a high throughput for point rendering if some low pixel error is alowed (10%).