GPU Point List Generation through Histogram Pyramids

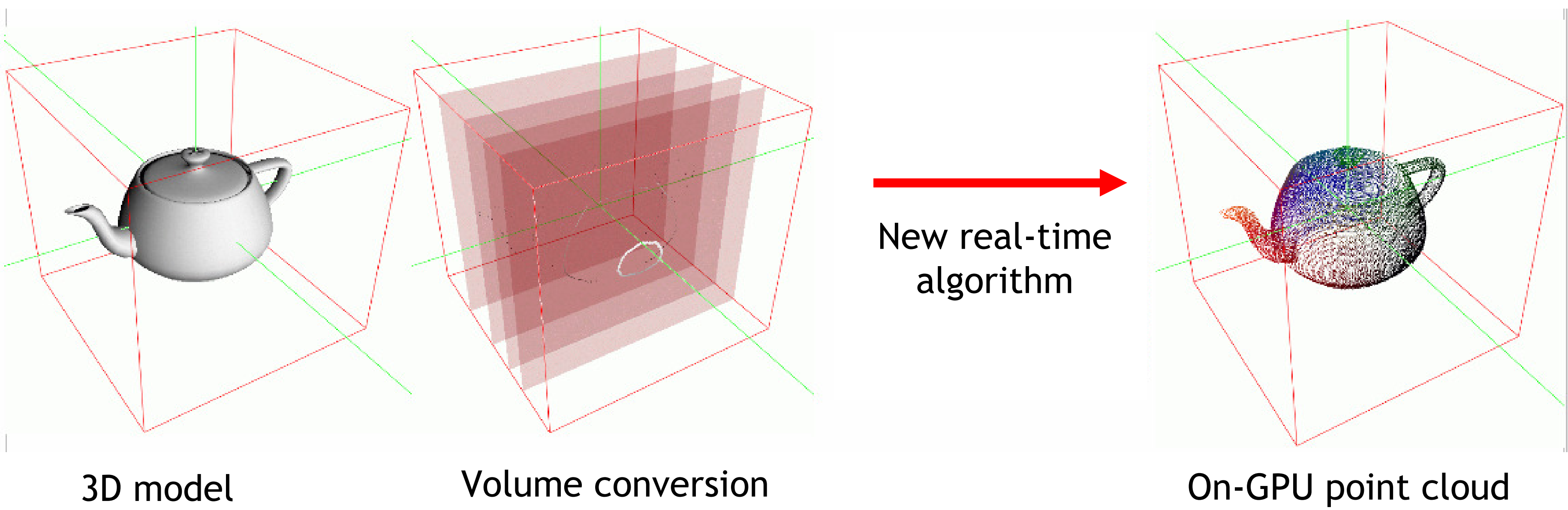
Gernot Ziegler, Art Tevs,
Christian Theobalt, Hans-Peter Seidel

Agenda

- Overall task
- Problems
- Solution principle
- Algorithm: Discriminator
- Algorithm: HistoPyramid Builder
- Algorithm: PointList Builder
- Applications
- Future Directions, Conclusion
- (Extra: Feature Detection, Geometry Generation, 3D Volume, QuadTree)

Overall task

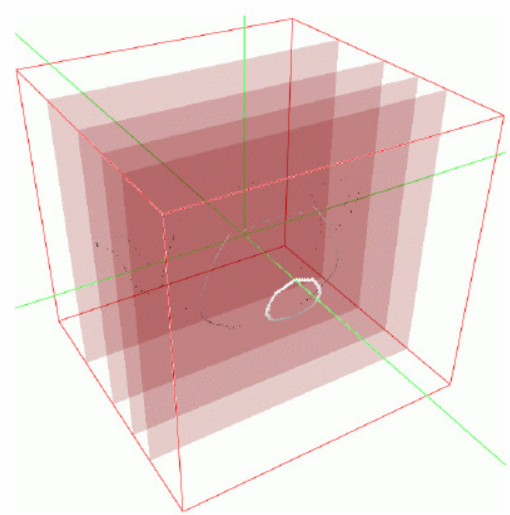
- Decompose a 3D model into a point cloud
- Vertices don't suffice, require minimum sampling density
- Make GPU render 3D model into 3D volume slices
- Problem: 256^3 points, many unfilled – find active ones !



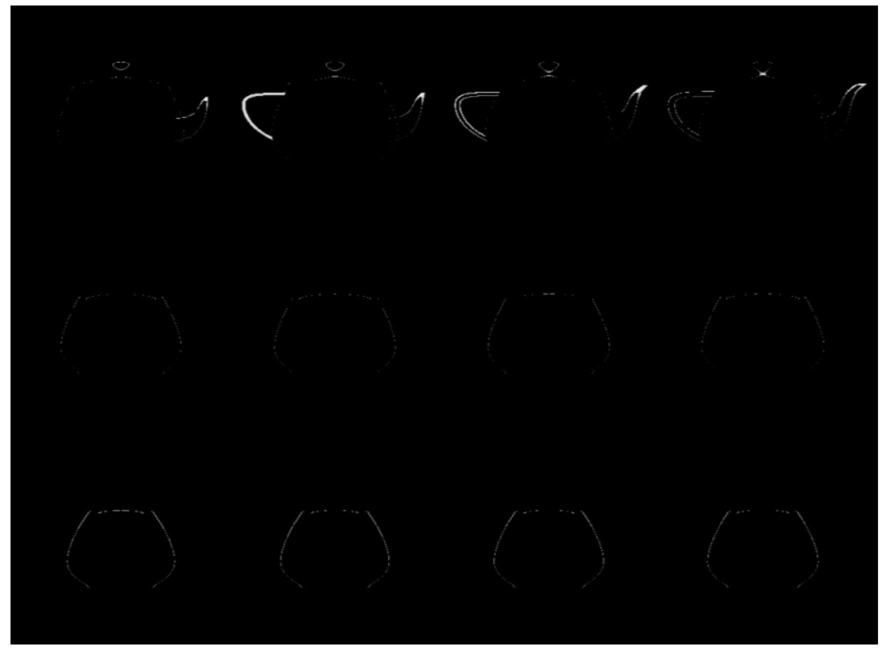
http://tinyurl.com/y5d8sh

Side problem: 3D to 2D mapping

- NVidia GPUs cannot render into 3D volume textures.
- **Solution: Create a 2D mapping scheme for 3D volume.**



3D Volume

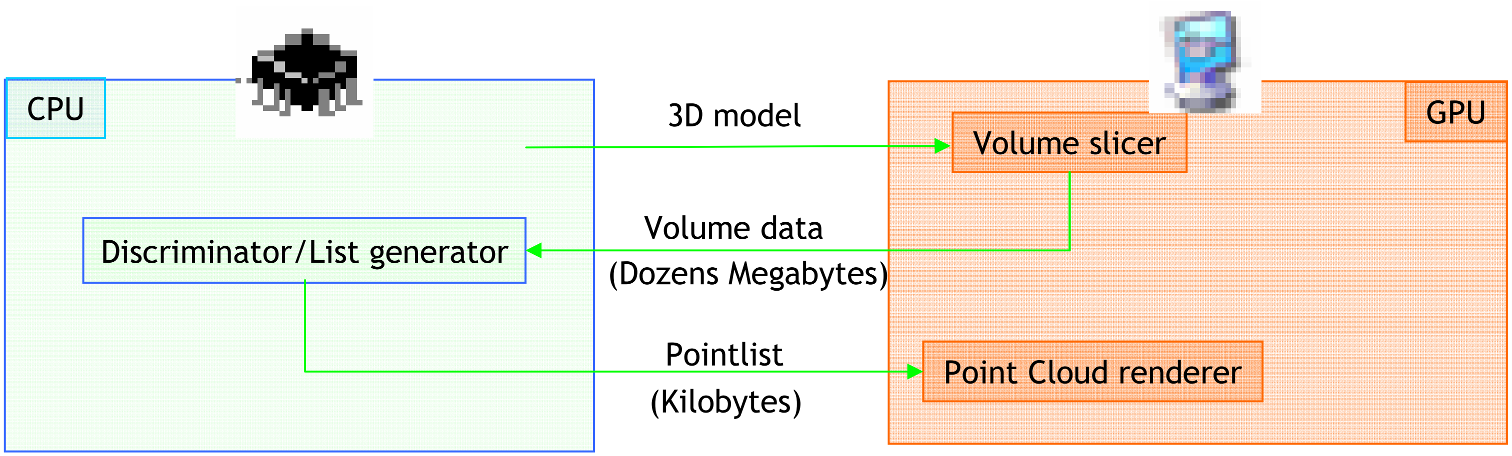


"Lattice": 2D mapping of 3D volume

- **Side effect: Following issues become a 2D problem.**

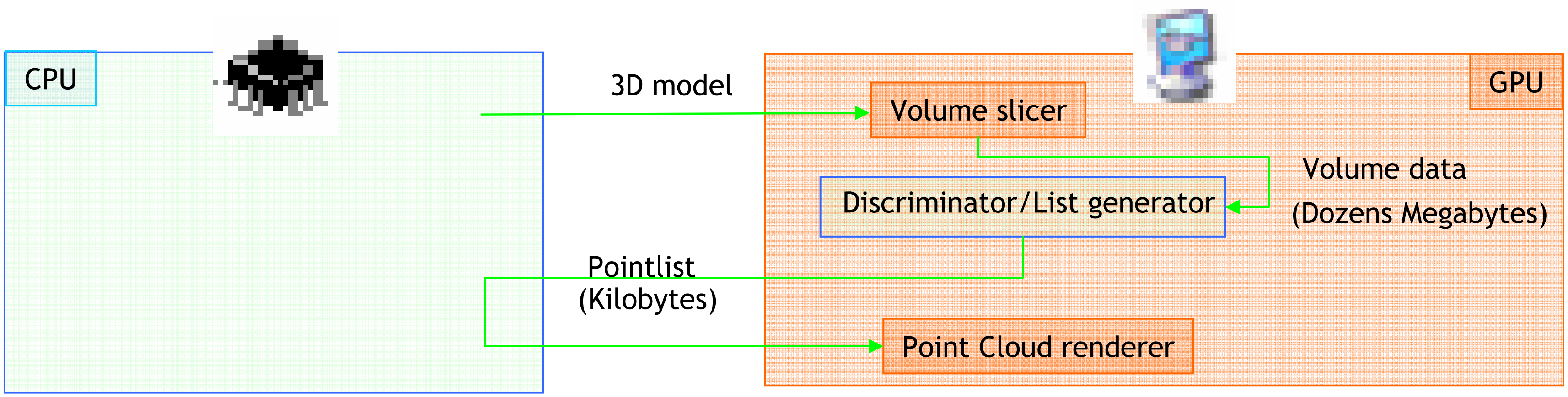
Current problem

- 3D volume is on GPU, find active voxels there
- Require dynamically growing list of point coordinates…
- **But: Bus transfers are expensive**.



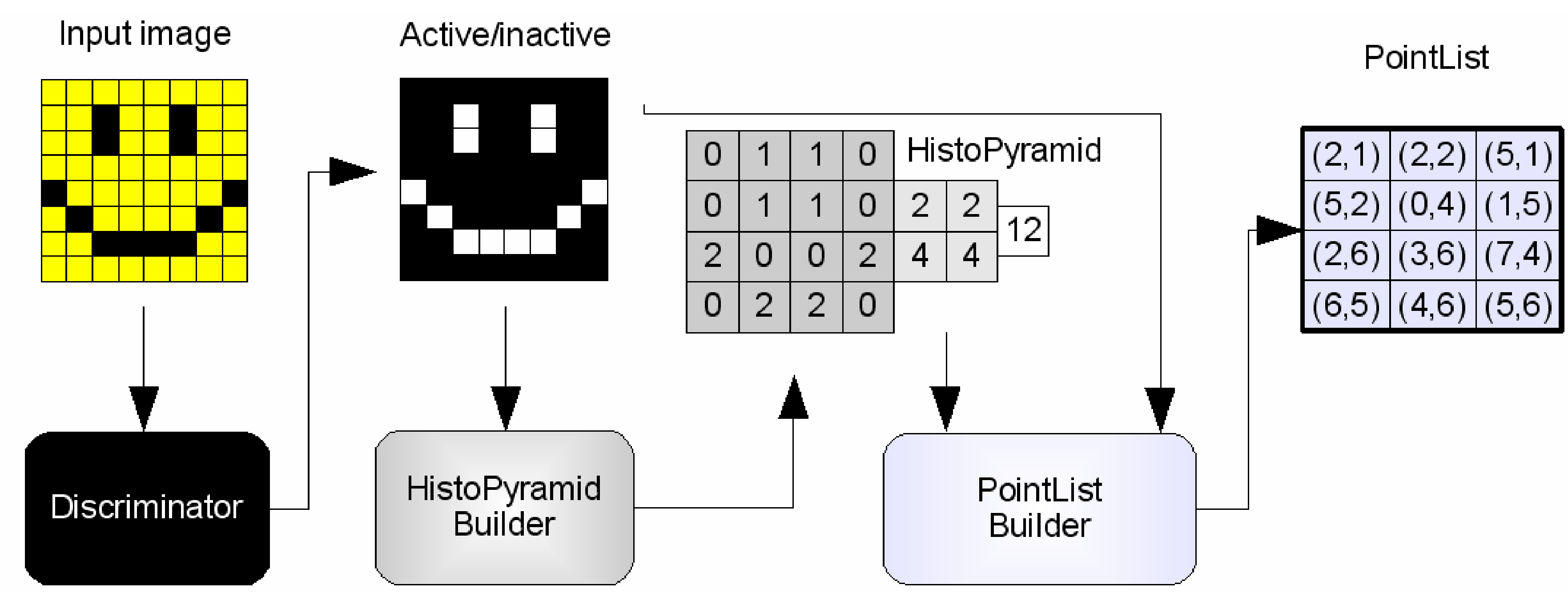
Our solution

- A 2D/3D algorithm which runs on Shader Model 3.0 *)
- See it as data compaction problem (Cell = Pixel/Voxel)
  Empty cells are "useless air", only interesting cells remain.
- Data compaction on stream processors active area of research (Horn et al, GPU Gems 2; Sengupta et al, Edge Workshop 2006)



*) SM 2.0 possible, but cumbersome

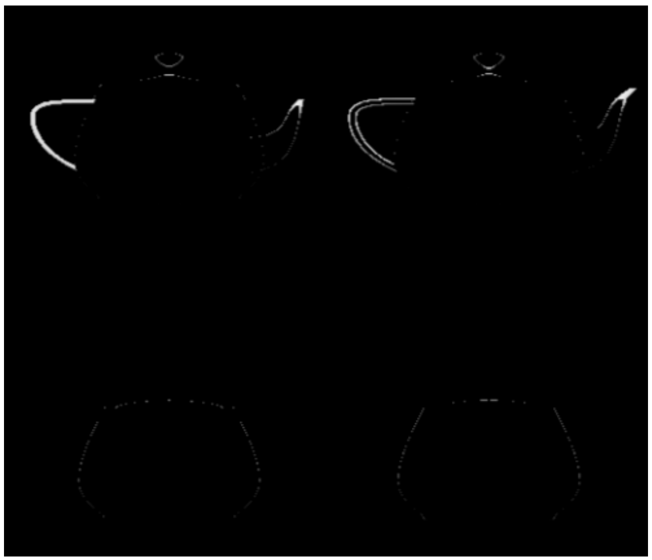
Overview, data compaction



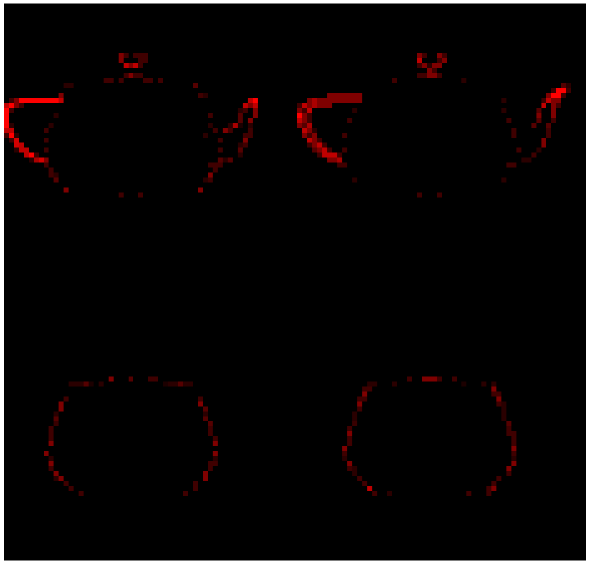
http://tinyurl.com/y5d8sh

Algorithm: Discriminator

- Tells active cells from inactive/empty ones.
- Easy criterion in our case: Alpha = 1.



Data input

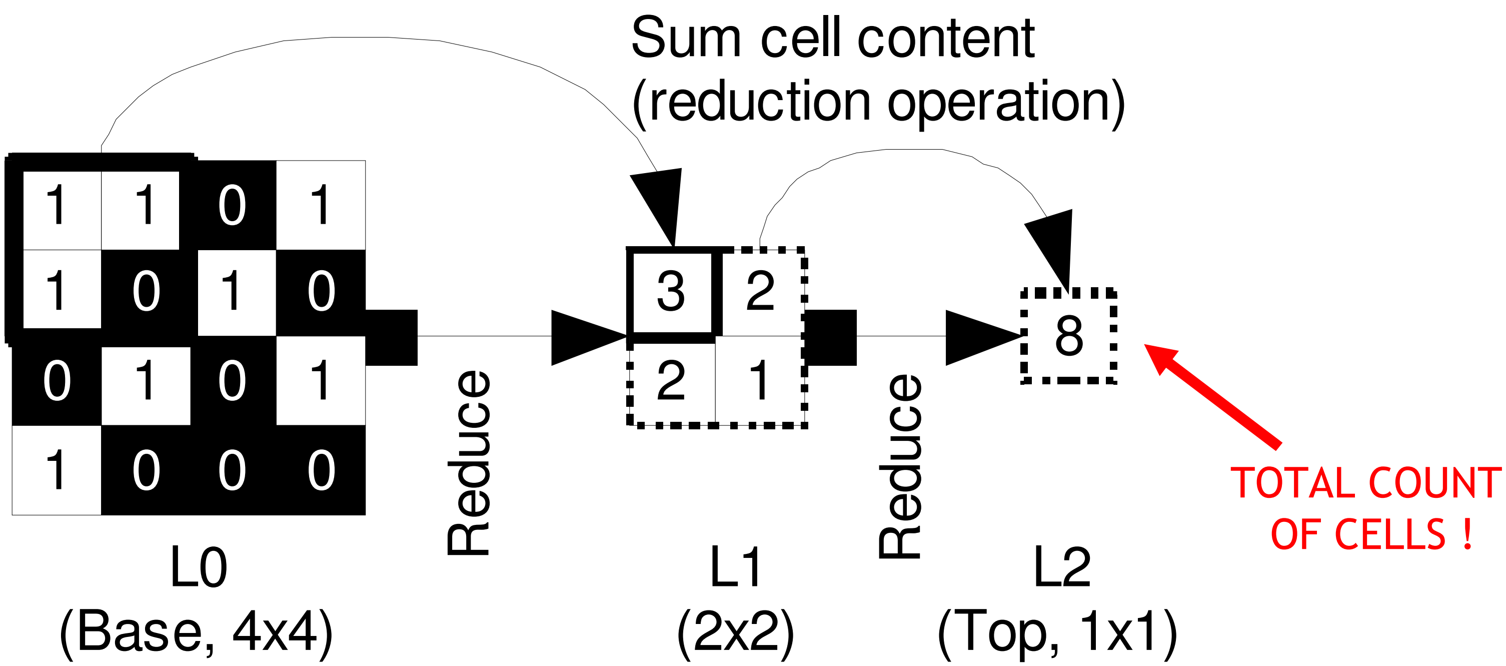


Base level of HistoPyramid

- More complicated discriminators in the paper!

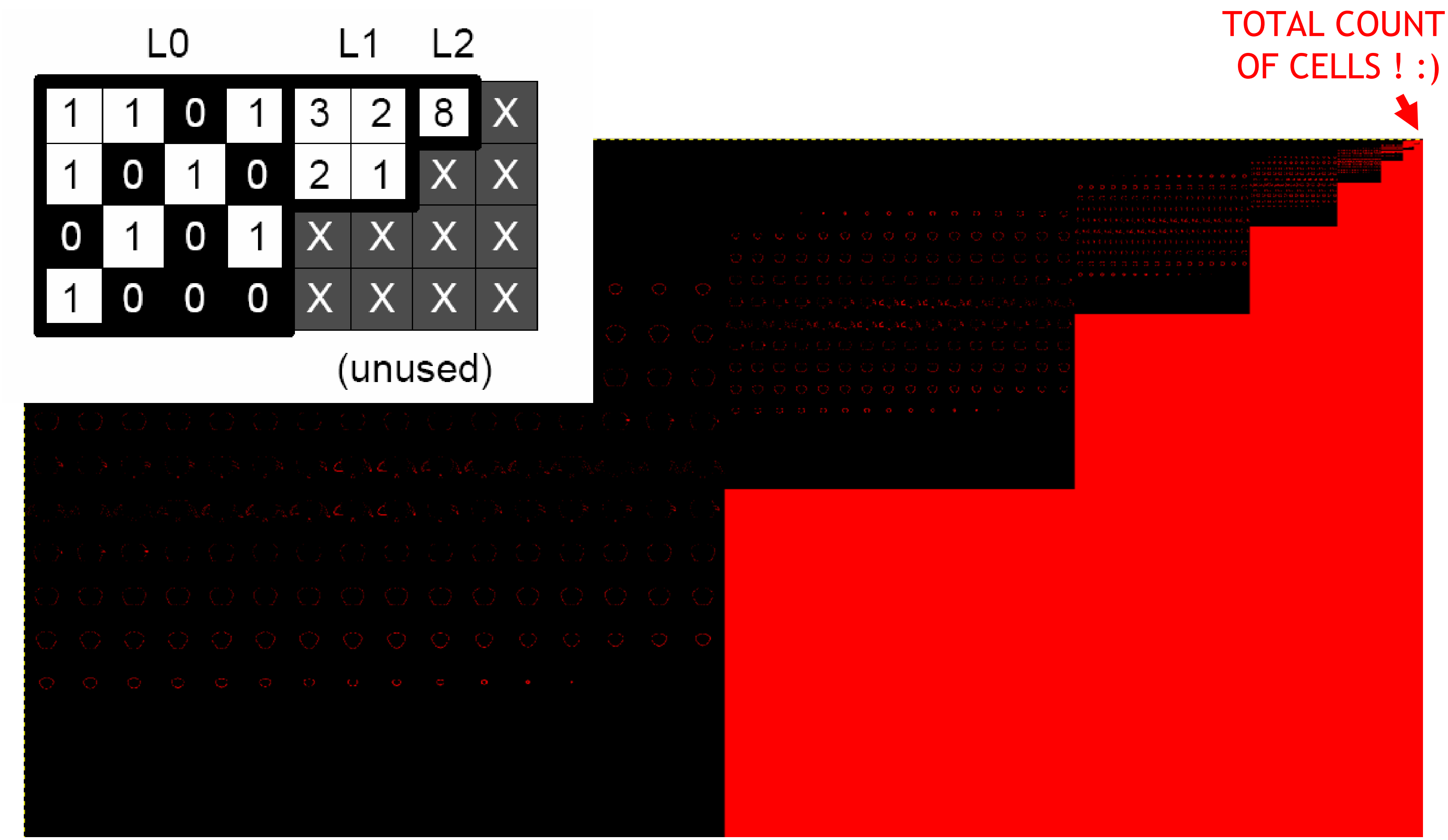
Algorithm: HistoPyramid Builder

- Hierarchically sums up active element count.
- Similar to mip-mapping (w/o averaging).



http://tinyurl.com/y5d8sh

Algorithm: HistoPyramid Builder's output

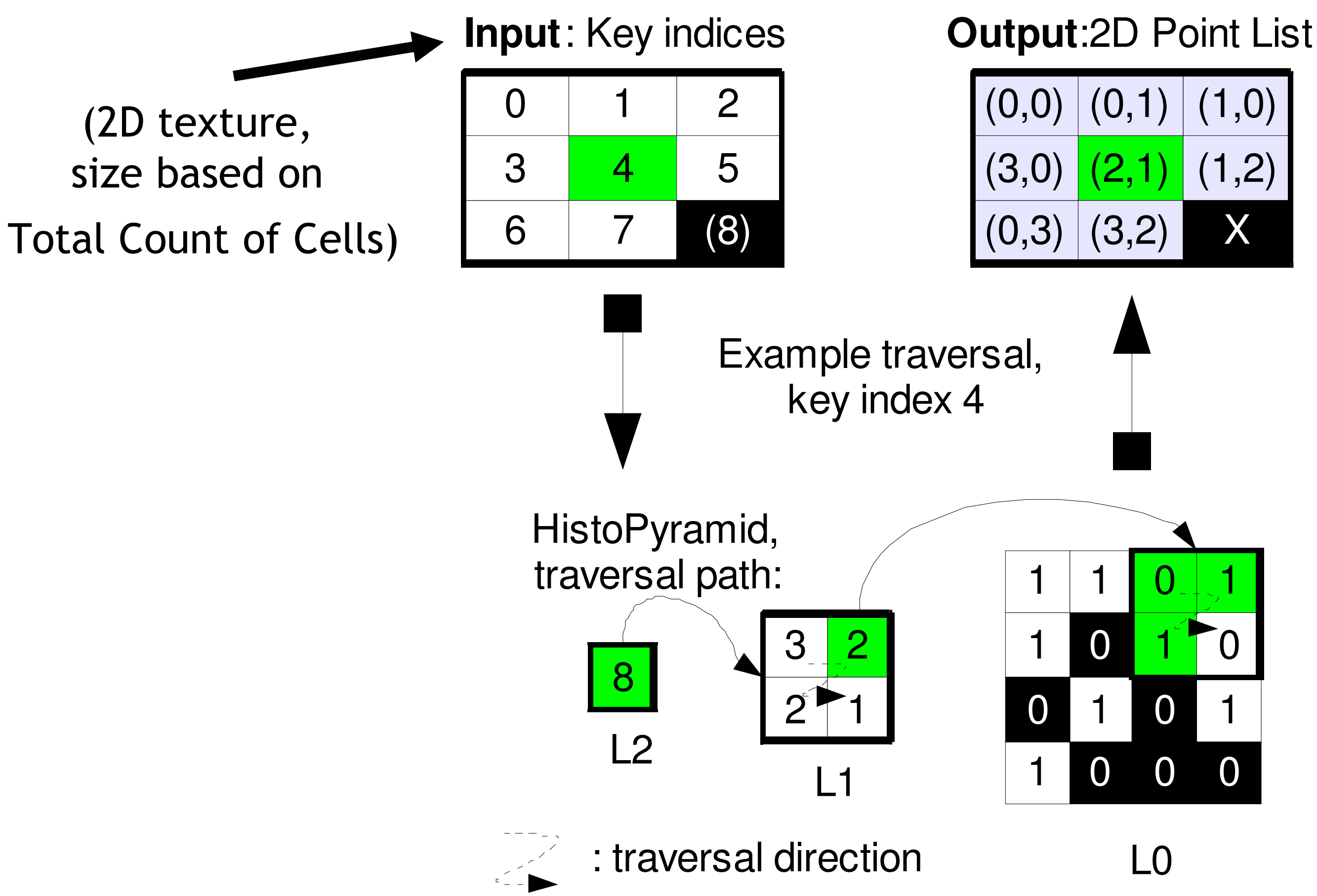


Histopyramid for Teapot

http://tinyurl.com/y5d8sh

Algorithm: PointList Builder

- Traverses HistoPyramid top-down to generate PointList.

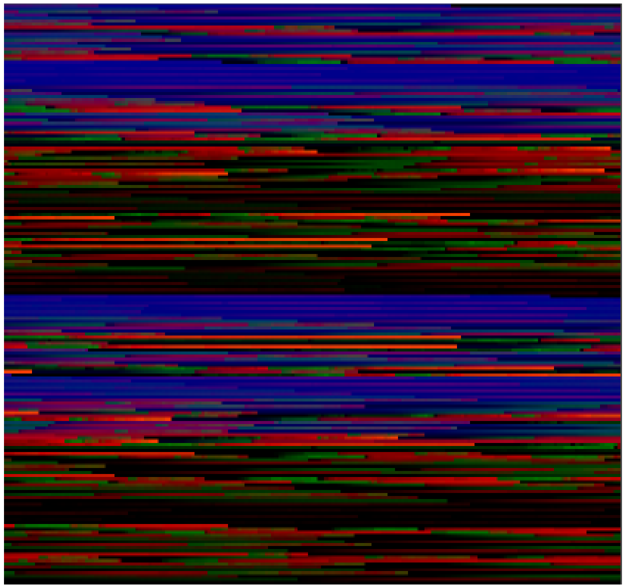


http://tinyurl.com/y5d8sh

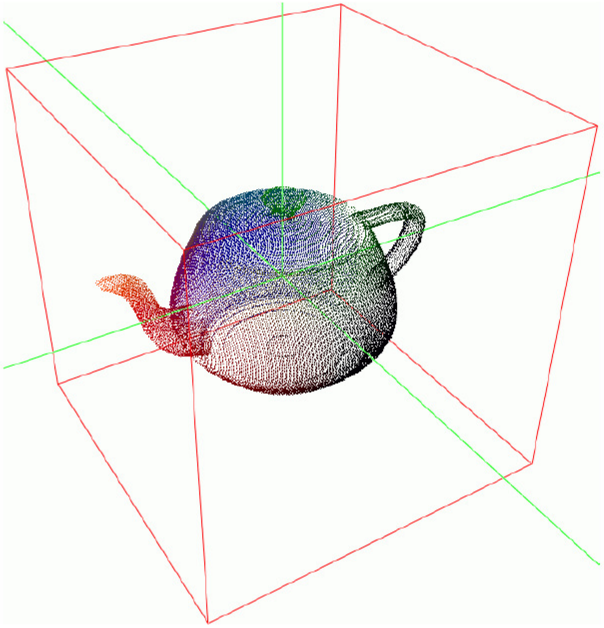
Algorithm: PointList Builder's output

max planck institut informatik

- A 2D texture listing active cells' 3D point coordinates.
- (2D to 3D mapping happens in PointList Builder)



PointList for Teapot.
false colors: 3D positions



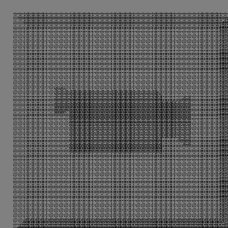
Example image,
PointCloud renderer

Done !

Results

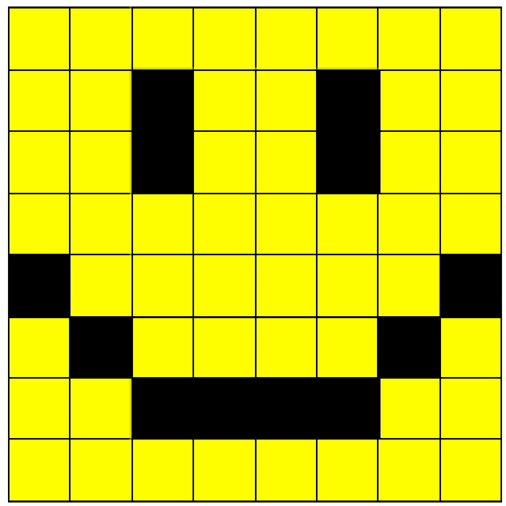
- Result: Real-time particle cloud (Art Tevs)

- Result: 3D Volume Analysis
  (Vector Field Contours, Tom Annen et al)



Future Directions

- Feature detection on GPU
- Geometry generation (SM 3.0 marching cubes)
- Real-time compression
  (with e.g. wavelet analysis)
- Conversion to SM4.0, and of course:
  "Showdown" with SM4 Geometry Shaders ;)
- Other uses for HistoPyramid
  (complete: Region Quadtrees/Octrees on GPU)
- Your ideas ? Discussions welcome !



Thank you !

http://tinyurl.com/y5d8sh

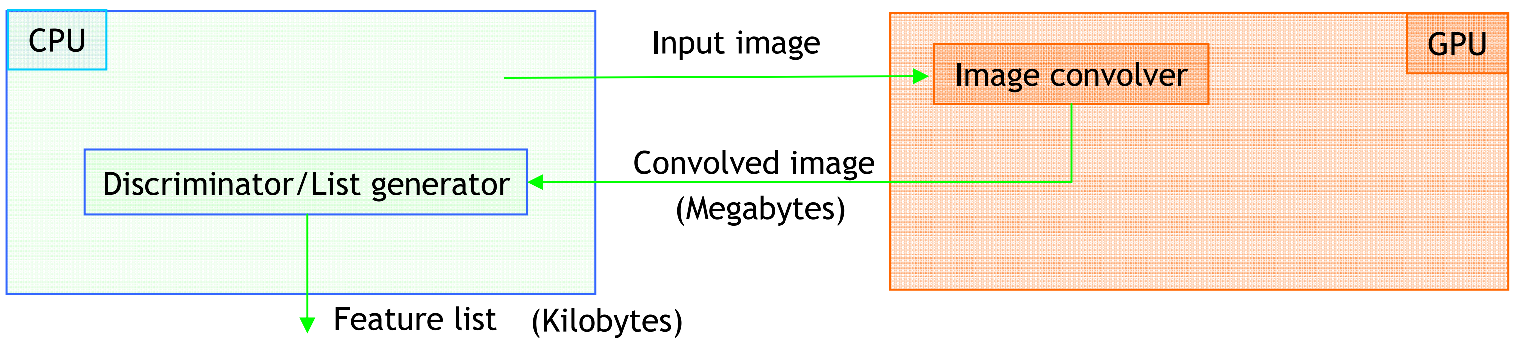
HistoPyramids vs. Geometry Shaders

- SM4.0 GPUs now have one option to create lists: Geometry Shaders.
- Used to delete and create geometry on the GPU.
- Still unclear if HistoPyramids become obsolete:
- Option 1: Route the whole data input through the geometry shader and let it “weed out” the empty cells.
  Vertex shader applied to millions of points ?
- Option 2: Let geometry shader traverse the data input, and generate geometry for all found cells.
  Can geometry shader create more than 1024 points?

Problems in GPU Feature detection

max planck institut informatik

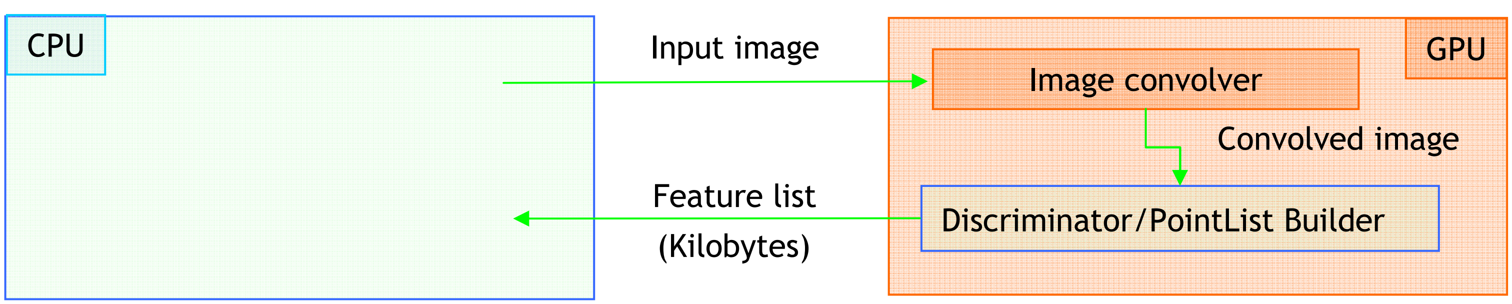
- Image analysis: GPU can convolve images.
- Feature points isolation: e.g. thresholding, requires dynamically growing list of point coordinates…
- GPU as a stream processor cannot generate this, must download to CPU.
- Bus transfers are expensive, hence:
  Speed advantage only for complex convolvements.



Speeding up GPU Feature detection

max planck institut informatik

- We introduce a 2D/3D solution which runs on Shader Model 3.0.
- Make it a data compaction problem (Cell = Pixel/Voxel)
  Uninteresting cells are “useless air”,
  only interesting cells (feature points) remain
- Data compaction on stream processors
  active area of research (Horn, GPU Gems 2)
- Hence:



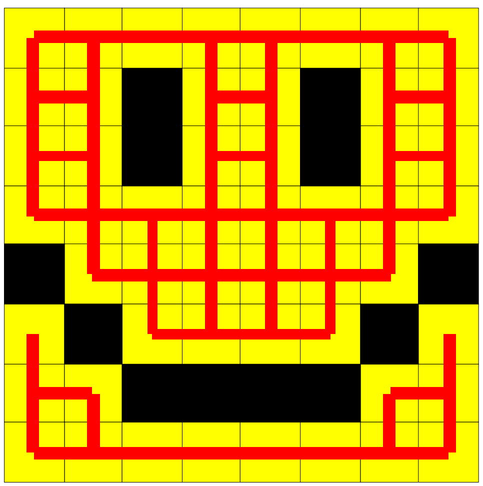
http://tinyurl.com/y5d8sh

Sketch: 3D HistoPyramid

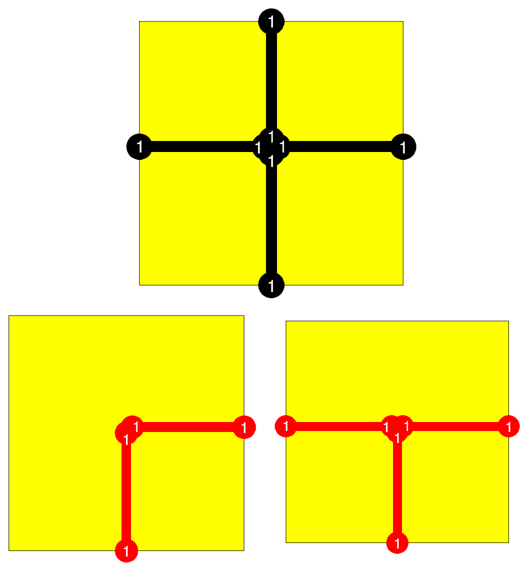
- Each pyramid level is a 3D volume.
- Like in mipmapped 3D volumes, the volumes are halved in size for every level.
- Thus, every time, 8 cells get reduced into one.
- PointList Builder accesses 3D textures one by one.
- Implementation obstacle:
  - Cannot easily generate 3D volume mipmaps on NV40 architecture without internal copying (Missing render-to-3Dtexture support).
  - The alternative, laying out several 3D volumes in one 2D texture, is very cumbersome and trashes cache behaviour.

Sketch: Geometry generation

- At base level, the discriminator calculates $n_{x,y}$, the number of intended vertices at each cell (x,y) from neighbor information (think: Marching Cubes from 3D volume).
- HistoPyramid will sum up the count of all intended vertices.



Intended line geometry



Discriminator:
Vertex creation template &
Example, $n_{x,y}$, first cells

4	6	4	4	4	4	6	4
6	6	0	6	6	0	6	6
6	6	0	6	6	0	6	6
4	8	6	8	8	6	8	4
0	4	8	8	8	8	4	0
2	0	4	6	6	4	0	2
6	4	0	0	0	0	4	6
4	6	4	4	4	4	6	4

HistoPyramid base level

http://tinyurl.com/y5d8sh

Sketch: Geometry generation

- GeometryBuilder, derived from PointList Builder, will end up parallelly ($n_{x,y}$ times) at cell (x,y) during top-down traversal.
  But it will know this (as it knows the initial index for cell (x,y)), and thus create the intended *[1.. $n_{x,y}$]*-th vertex from a lookup table.

4	6	4	4	4	4	6	4
6	6	0	6	6	0	6	6
6	6	0	6	6	0	6	6
4	8	6	8	8	6	8	4
0	4	8	8	8	8	4	0
2	0	4	6	6	4	0	2
6	4	0	0	0	0	4	6
4	6	4	4	4	4	6	4

22	14	14	22
24	20	20	24
6	26	26	6
20	8	8	20

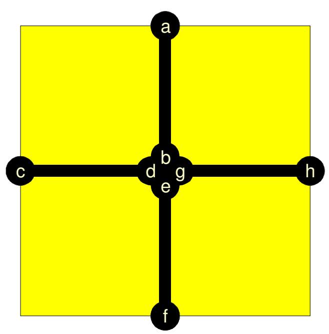
80	80
60	60

280

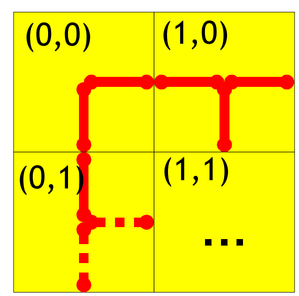
Created HistoPyramid

(0,0,e)	(0,0,f)	(0,0,g)	(0,0,h)
(1,0,c)	(1,0,d)	(1,0,e)	(1,0,f)
(1,0,g)	(1,0,h)	(1,1,a)	(1,1,b)
(1,1,..)	...		

GeometryBuilder's vertex output list



GeometryBuilder's vertex creation order



Above list as geometry

Real-time QuadTree analysis

- Vision task: Detect regions with common features
- "Common feature": Low variance around average color
- Usually very time-consuming, by far not real-time
- Our Answer:
  - Extension of GPU point list algorithm
  - Only create simple region geometry, region quadtrees
  - Typical analysis time: 640x480 in 15 ms
- Useful for:
  - Color grouping
  - Motion vector clustering
  - Acceleration of grid calculations

Results

ffkvadrat: Real-time quadtree analysis

Video frame, Edge analysis, several results

(Footage by GusGus, *Call of The Wild*)