Efficient gather operations using histogram pyramids

Christopher Dyken

Based on the VMV 2006 presentation
"GPU Point List Generation through Histogram Pyramids"
by G. Ziegler, A. Tevs, C. Theobalt, and H-P. Seidel

UNIVERSITETET
I OSLO

CMA Centre of Mathematics for Applications

Problem

We have a huge $N \times N$ set of data, and we only want to continue computation on a small subset.

- The histogram pyramid orders the input data in a set of buckets, and extracting the elements of each bucket is fast.
- Construction is done in $\log_2(\frac{1}{2}N)$ passes.
- Extraction of an element is done in $\log_2(\frac{1}{2}N)$ texture lookups.
- Each output element can be outputed more than once.
- **Without any data transfer from GPU to CPU**

Thus,

$\Longrightarrow$ Point cloud generation

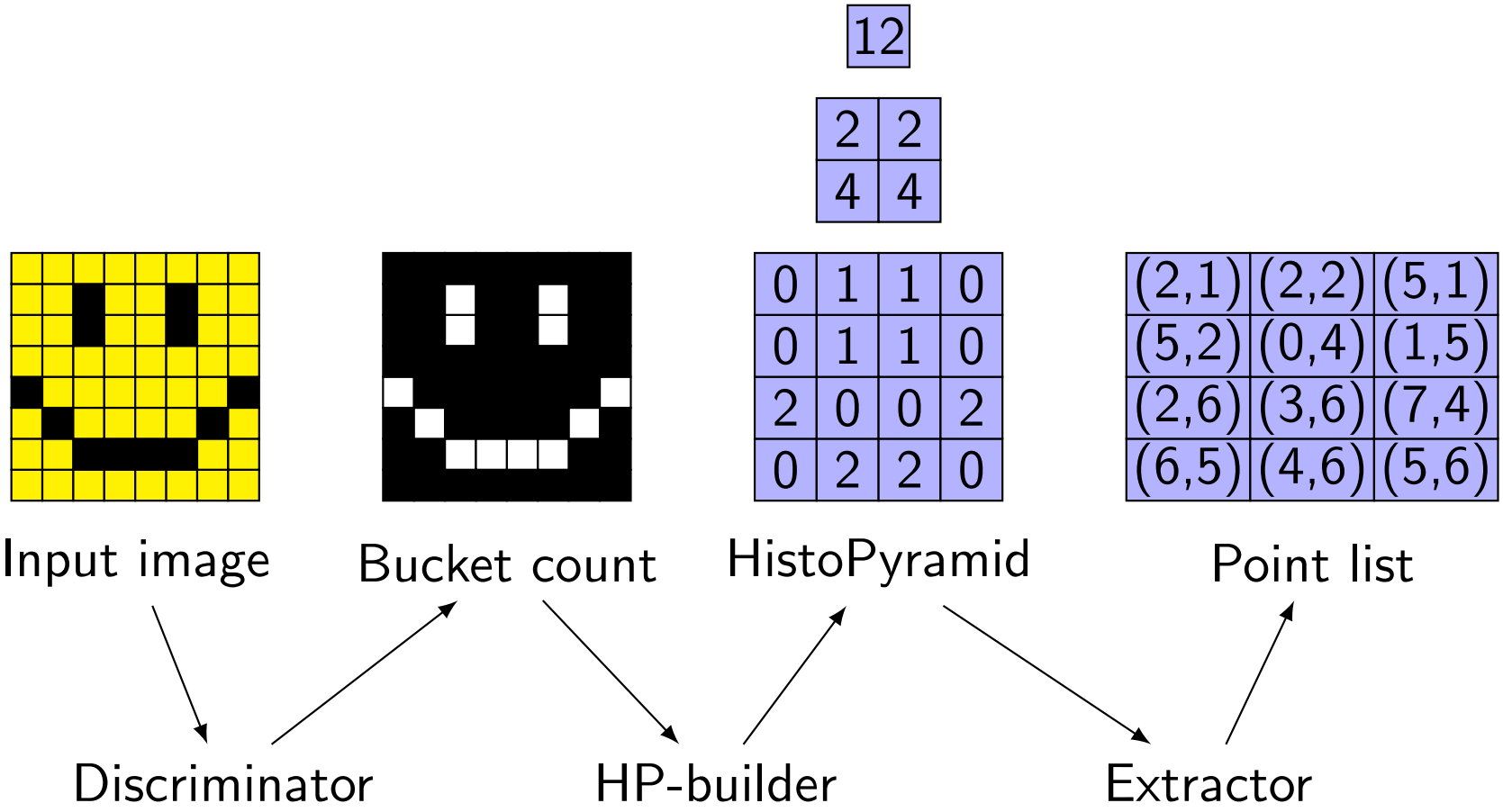
$\Longrightarrow$ Compaction of intermediate results

$\Longrightarrow$ Sparse matrix extraction

$\Longrightarrow$ Emulate GS-type operations on non-GS hardware.

$\Longrightarrow$ Maybe reduce the workload of the geometry shader.

Overview


12
2 2
4 4
0 1 1 0
0 1 1 0
2 0 0 2
0 2 2 0
(2,1) (2,2) (5,1)
(5,2) (0,4) (1,5)
(2,6) (3,6) (7,4)
(6,5) (4,6) (5,6)
Input image
Bucket count
HistoPyramid
Point list
Discriminator
HP-builder
Extractor

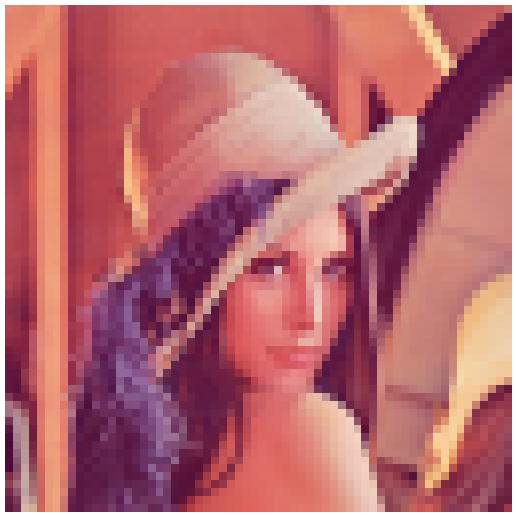
Discriminator

- For each input element, classify and output bucket and bucket count
  - MRT on NV40 allows 4xRGBA=16 buckets, G80 8xRGBA=32 buckets.
  - Count is number of output elements this texture position should have for a particular bucket.

  $\Longrightarrow$ Buckets can be overlapping!
- Often **one** class and **binary** count (on/off)

Example discriminator: edge extraction

- Classify texture positions as edge/non-edge:
  - Apply Laplace filter
  - Threshold output



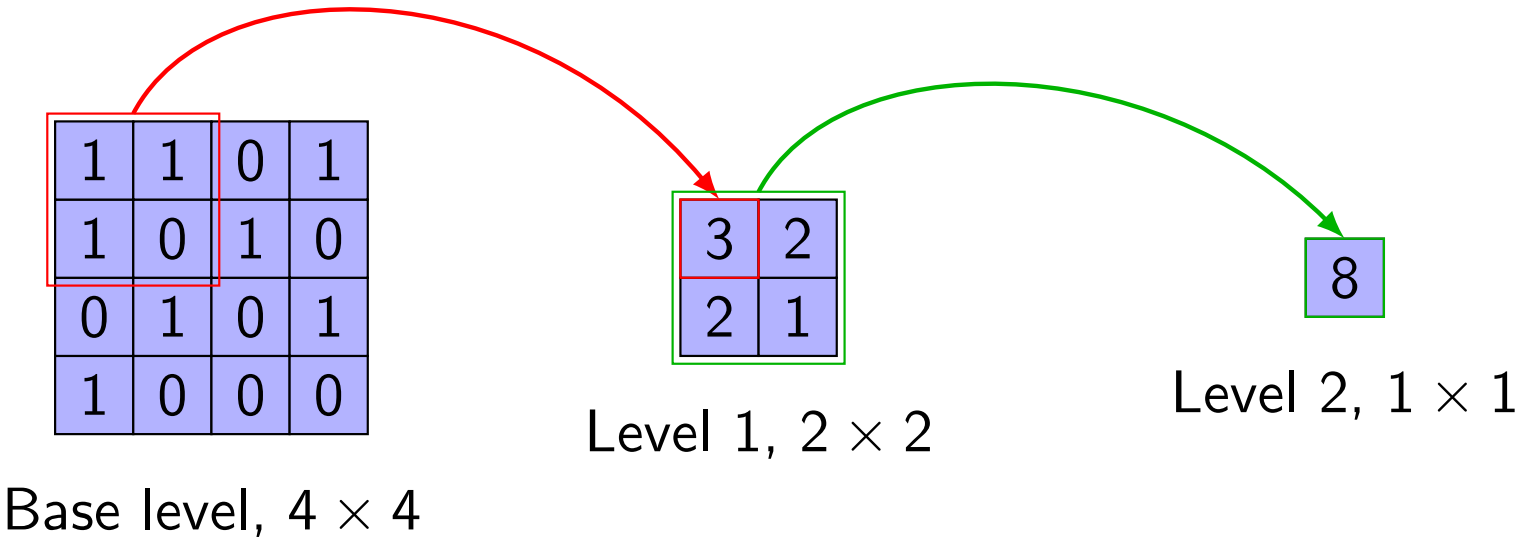
Input data

Histopyramid base level

HistoPyramid builder

- Build pyramid layer-by-layer bottom up.
- Each cell corresponds to the number of elements in the sub-pyramid below.

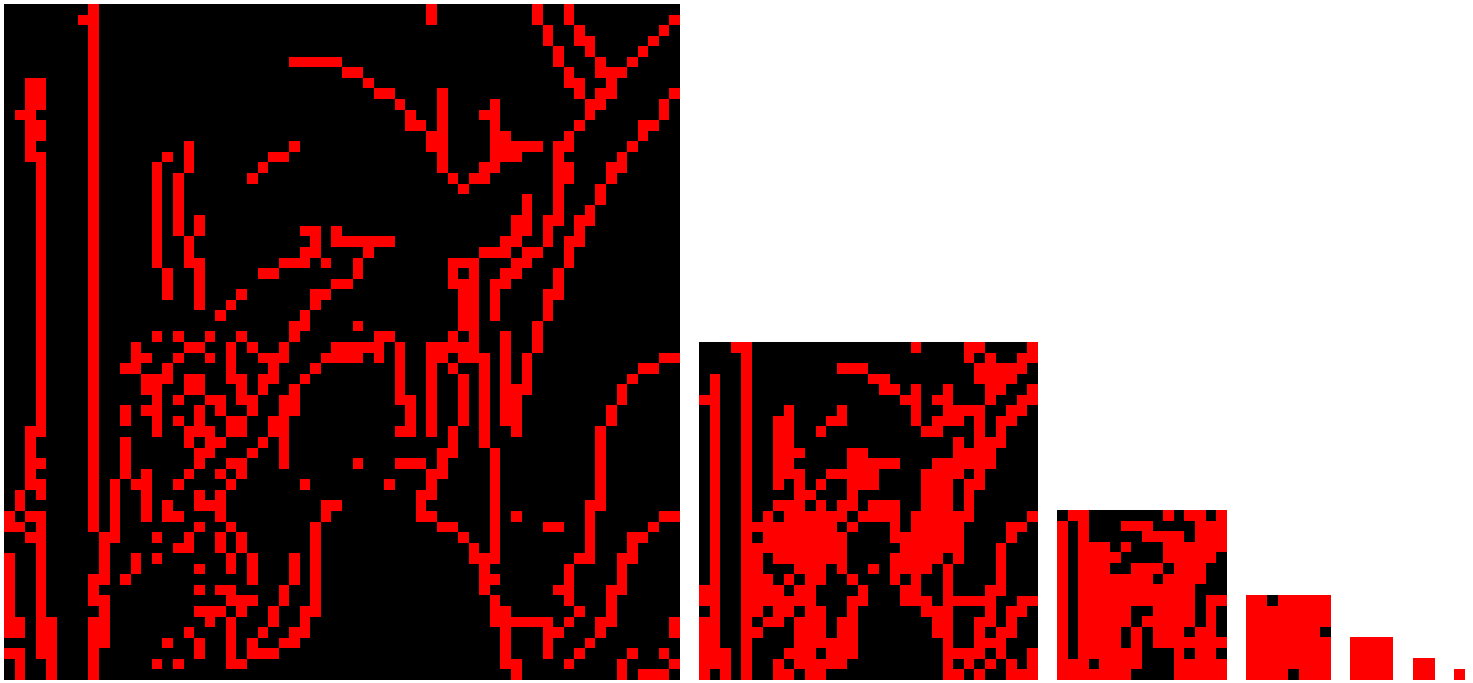
$\Longrightarrow$ Mipmap-generation without averaging.



- Top element contains the total number of cells in the pyramid.

Example histogram pyramid

- Red cells denote non-zero count



Pointlist builder

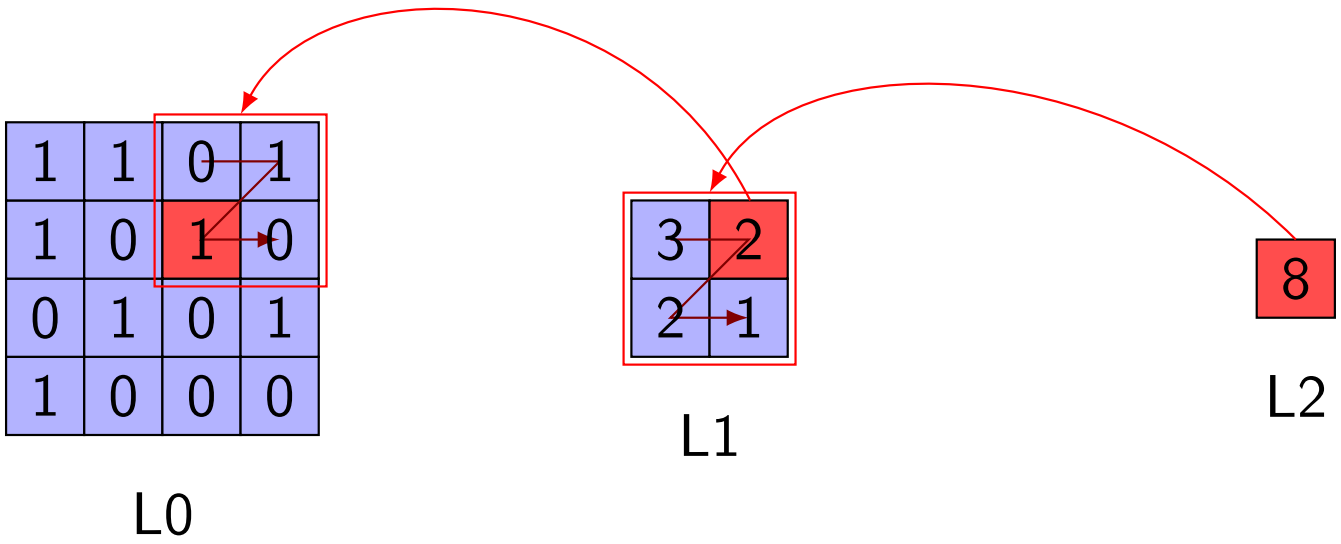
- Given a key index, traverse the histopyramid top-down to find the corresponding texture position

Output: Point list

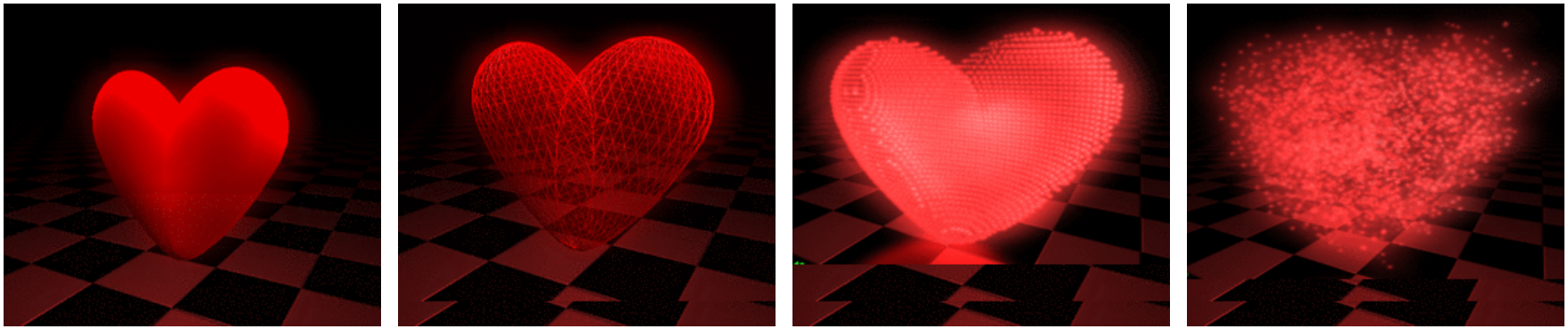
(0,0)	(0,1)	(1,0)
(3,0)	(2,1)	(1,2)
(0,3)	(3,2)	×

Input: Key indices

0	1	2
3	4	5
6	8	×



Applications: Point list generation of 3D volumes[1]



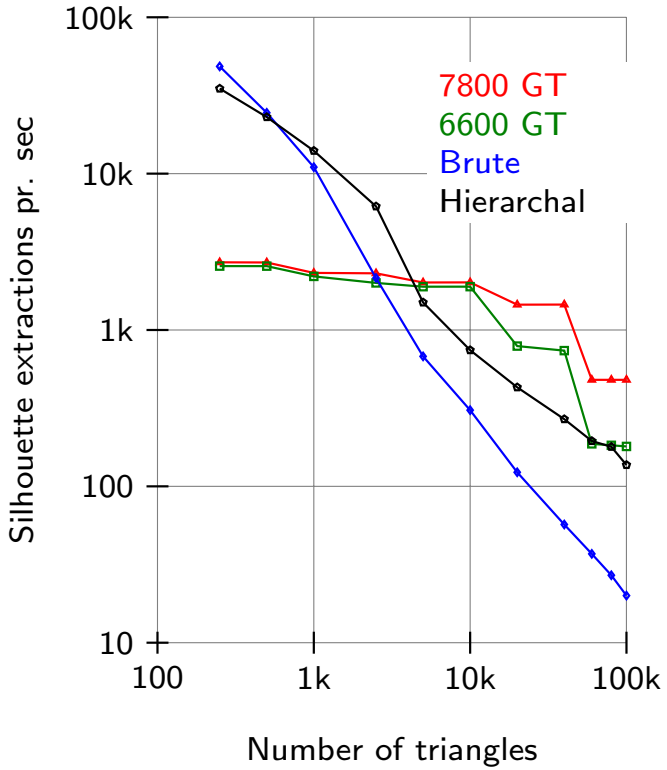
Creates directly a compacted list of points from a 3D-volume, entirely on the GPU.

[1] Ziegler, Tevs, Theobalt, and Seidel 2006

Applications: Silhouette extraction of 3D-meshes[2]

We let one texel represent an edge.

- Check if edge is on silhouette or not (predicate)
- Build histopyramid
- Read back the set of silhouette edges to CPU

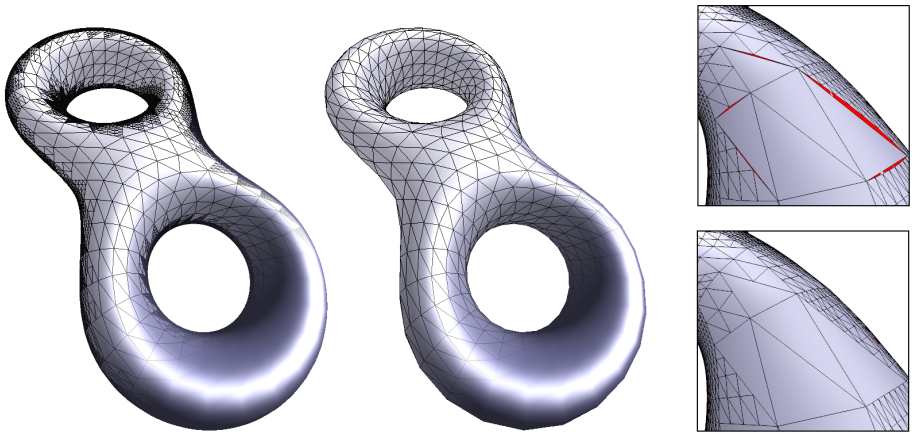


$\Longrightarrow$ GPU beats hierarchal CPU around 7-8k triangles

$\Longrightarrow$ This is not particularly computationally expensive, probably more savings for heavier calculations.

[2] Dyken, Reimers, and Seland 2006

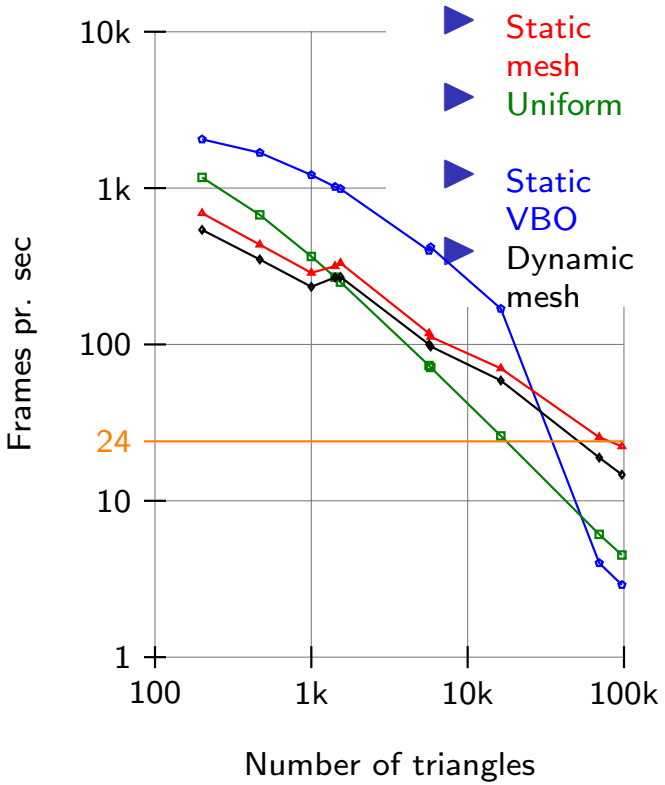
Applications: Adaptive tessellation[3]



- Use histopyramids to generate compacted lists of patches that should be refined.

$\Longrightarrow$ Beats uniform refinement at 2k patches.

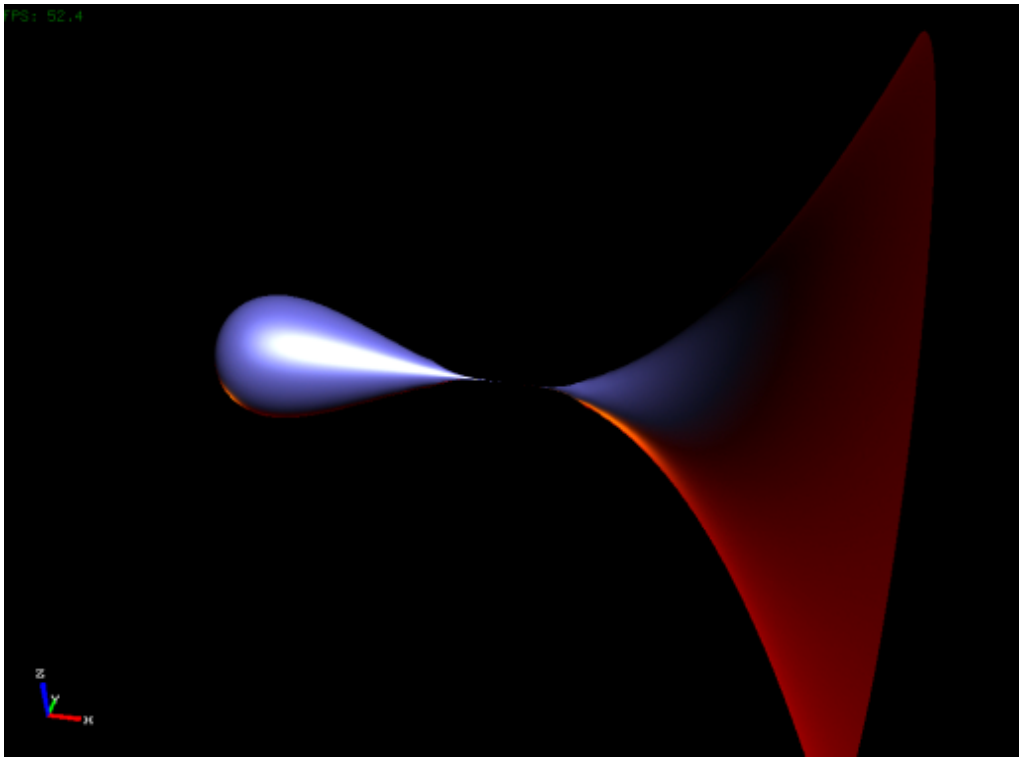
$\Longrightarrow$ Beats static VBOs for huge meshes.



[3] Dyken, Reimers, and Seland 2006

Applications: Marching cubes[4]

Create an arbitrary number of points by using bucket count $\neq 1$.



Get 50-60 fps marching a $64 \times 64 \times 64$ volume on SM3-cards.

[4] Dyken, current research

References:

- G. Ziegler, A. Tevs, C. Tehobalt, H.-P. Seidel, "GPU Point List Generation through Histogram Pyramids", Tech. Rep. MPI-I-2006-4-002, Max-Planck-Institut für Informatik, 2006.
- C. Dyken, J. Seland, and M.Reimers, "Real Time Silhouette Refinement using Graphics Hardware", submitted to Computer Graphics Forum, 2006