NTNU

Innovation and Creativity

Framework for Polygonal Structures on Clusters

Leif Christian Larsen (leifchl@idi.ntnu.no)
Anne C. Elster (main supervisor),
Tore Fevang (co-supervisor, Schlumberger)

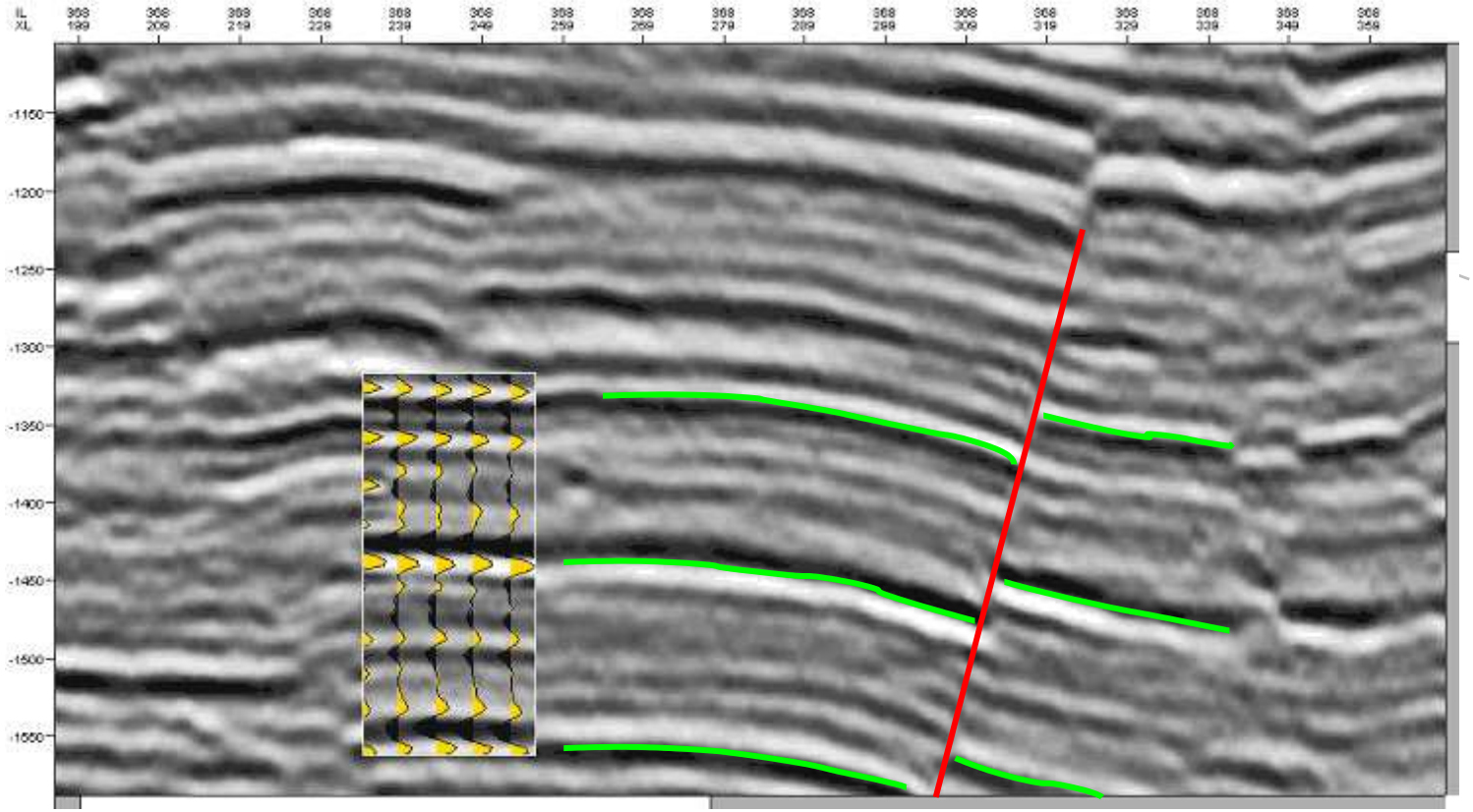
NTNU
Innovation and Creativity

Outline

- Motivation:
  Pillar Gridding for Seismic Fault Detection
- Main Problem I
  Efficient Voxelization
- Main Problem II
  Voxel transfer and caching, load-balancing

NTNU
Innovation and Creativity

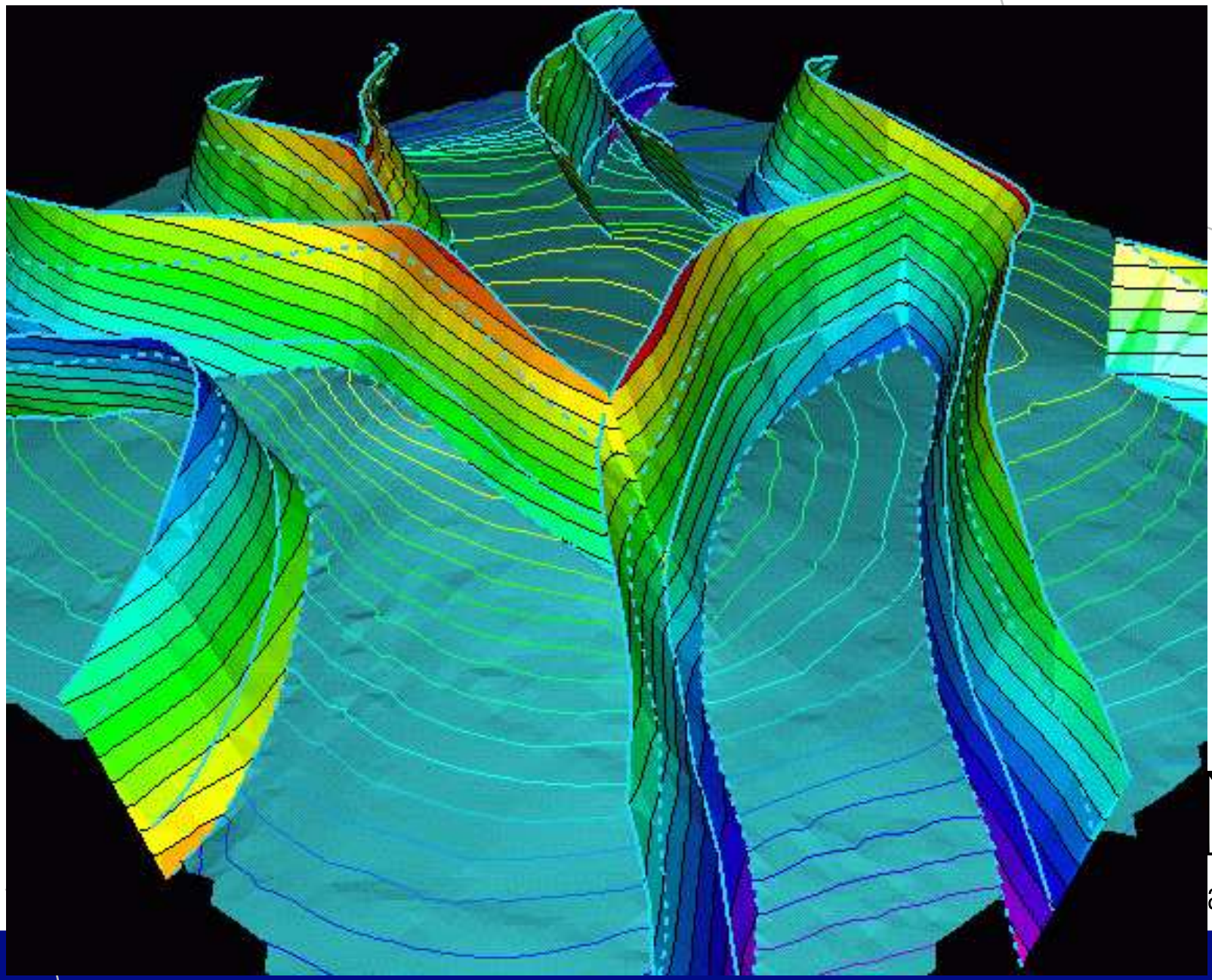
Motivation: Pillar Gridding



(Image courtesy Schlumberger)

NTNU
Innovation and Creativity

Motivation: Pillar Gridding



Motivation: Pillar Gridding

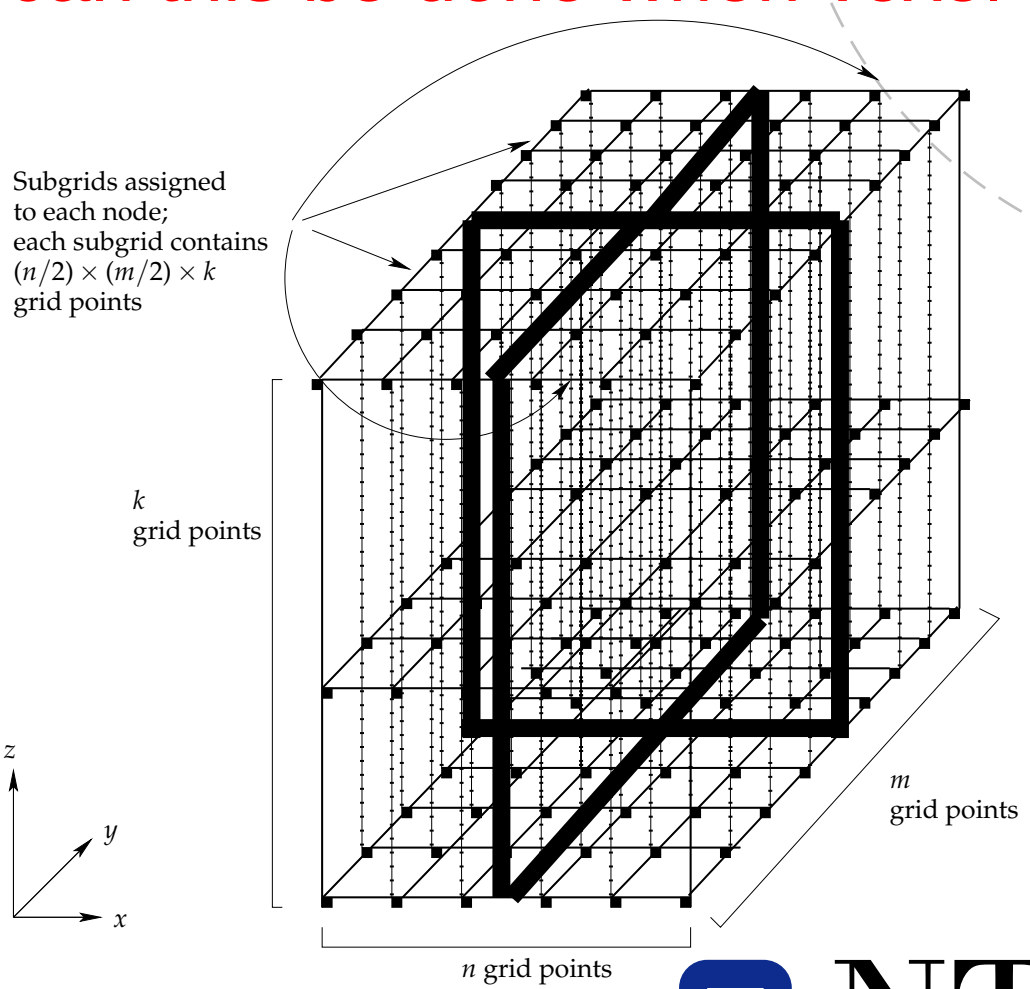
Pillar Gridding Workflow

1. A geologist marks fault pillars or fault sticks
2. Insert additional fault pillars between interpreted pillars, and connect pillars by a polygon mesh.
3. Finally, polygon is *voxelized* to do computations (e.g. filtering) over voxels
4. Subsequently, pillars/polygon vertices can be moved

NTNU
Innovation and Creativity

Motivation: Pillar Gridding

Problem: How can this be done when voxel data



is distributed?

NTNU
Innovation and Creativity

Motivation: Pillar Gridding

Need efficient interface to:

- Add polygons
- Remove polygons/vertices
- Move vertices
- Create/destroy meshes
- Voxelize all faces on all nodes

NTNU
Innovation and Creativity

Contributions

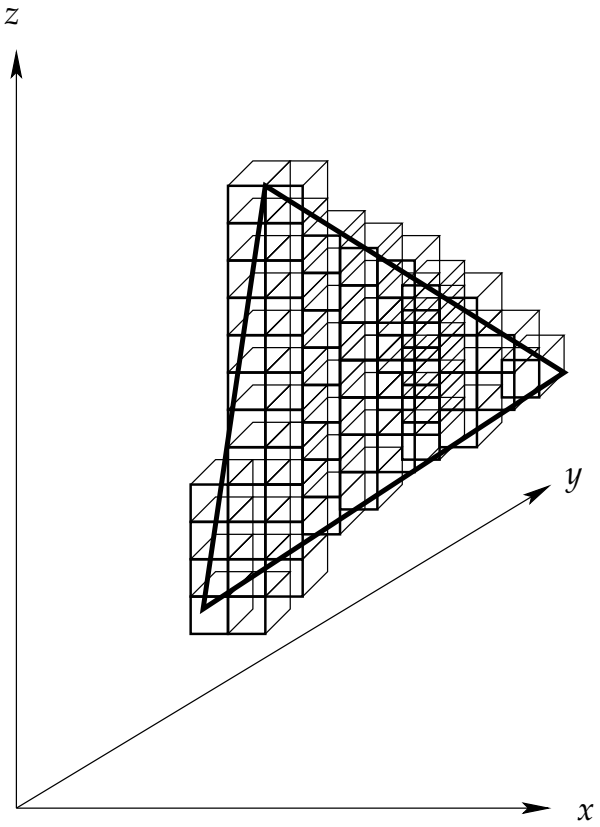
Contributions:

- New parallel polygon voxelization algorithm
- Introduce efficient caching/transfer strategies for the operations on the previous slide
- Introduce three algorithms for load-balancing

NTNU
Innovation and Creativity

Problem I: Voxelization

Voxelization of a polygon/triangle



NTNU
Innovation and Creativity

Voxelization — Introduction

- Voxelization is the process of discretizing a 3D object with 3D voxels.
- Rasterization is the same process and is implemented in hardware on GPUs, but discretizes only to 2D pixels.

Voxelization — Introduction

- Traditionally, visualization has focused at going the other direction (from voxels to polygons) through the Marching Cubes algorithm
- Therefore: Small research community

NTNU
Innovation and Creativity

Voxelization — Introduction II

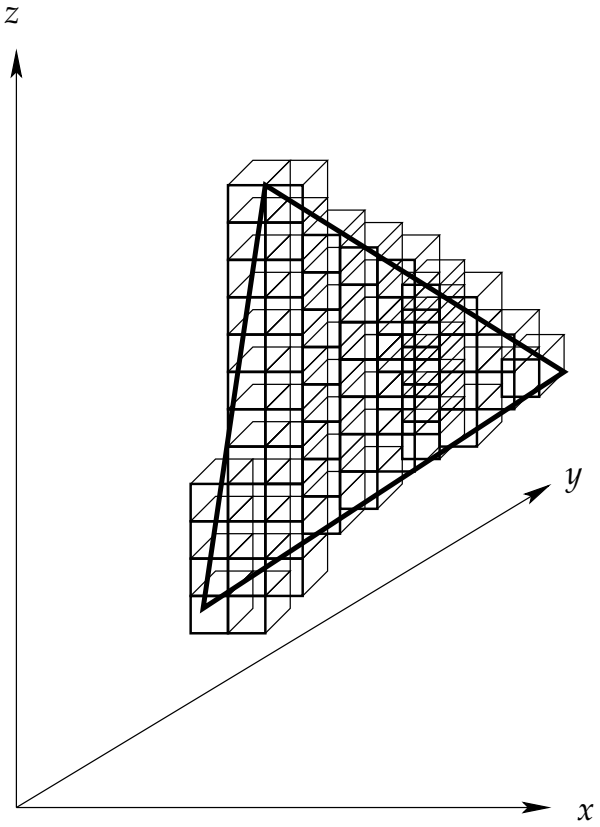
- Voxelization has applications in:
  - Certain volume-rendering algorithms in visualization
  - 3D screens (which were first suggested in 1912, but are yet to come)
  - A few other applications: seismologic interpretation for representing faults and horizons, and medical applications for representing radiation therapy beam surfaces intersecting a human body

NTNU
Innovation and Creativity

Voxelization — Algorithms

- Most voxelization research focuses on visualization or voxelization of non-planar objects. Not relevant for our purposes
- Original voxelization algorithm by A. Kaufman in 1988 is one of the few voxelization techniques applicable to our problem

Kaufman's Algorithm



NTNU
Innovation and Creativity

Voxelization — Algorithms

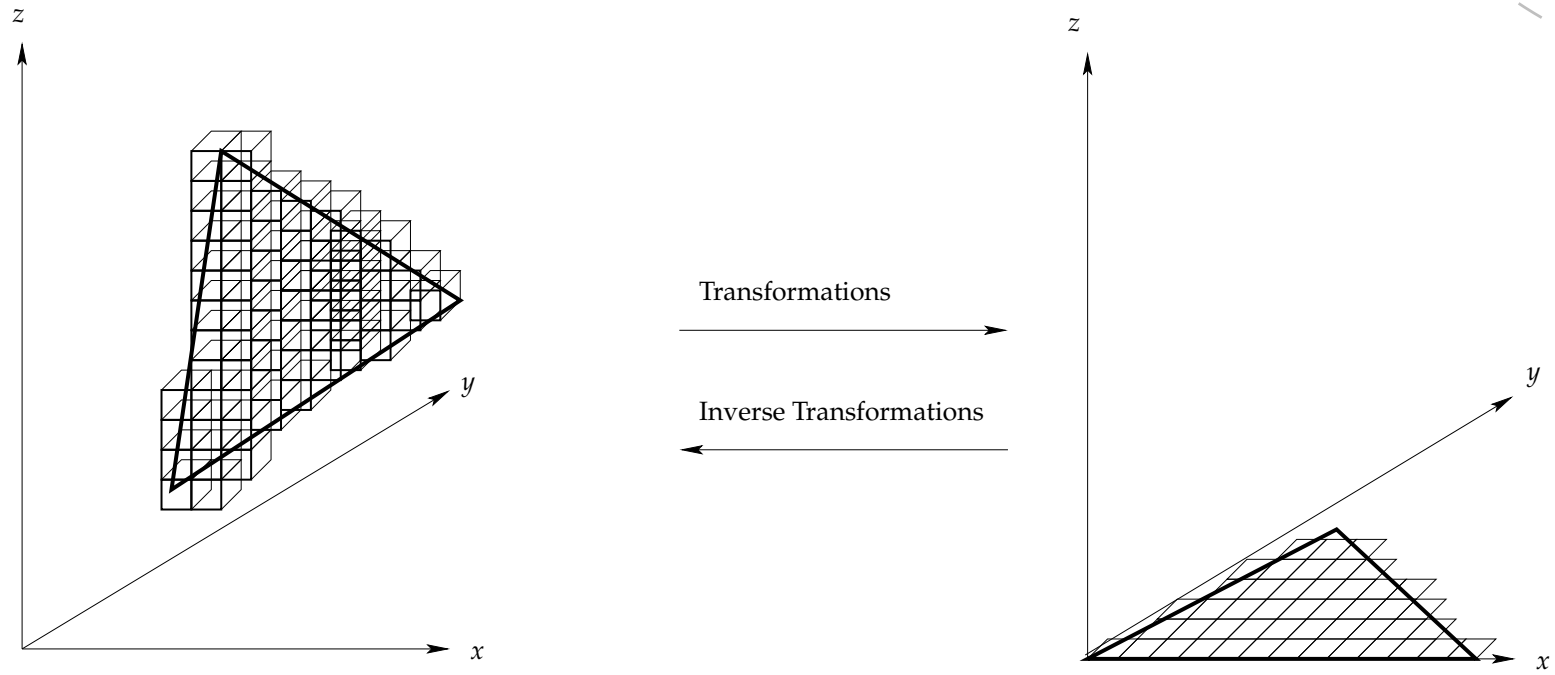
- Voxelization is just an extension of rasterization. Thus we should be able to exploit GPU hardware to do voxelization.

Voxelization — Algorithms

- Some previous attempts have been made on doing voxelization on GPU.
  - Previous approaches focus on visualization apps

Voxelization — Algorithms

- Idea: Transform the voxelization problem to a rasterization problem



Voxelization — Algorithms

1. Translate one vertex to the origin.
2. Rotate vertices such that the longest edge is in $z = 0$ plane.
3. Rotate about z-axis such that longest edge is aligned with x-axis.
4. Rotate final vertex about x-axis such that it too is in the $z = 0$ plane.
5. VS Kaufman: Trade more computation for less branches

NTNU
Innovation and Creativity

Voxelization — Algorithms

Three *quaternions* represent the rotations

$$q_i = (\mathbf{v}, s) \tag{1}$$

Inverse quaternion q_i^{-1}: simply negate $\mathbf{v}$
Quaternions are combined by the formula

$$\eta_1 * q_2 = (s_1\mathbf{v}_2 + s_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2, s_1 s_2 - \mathbf{v}_1 \bullet \mathbf{v}_2) \tag{2}$$

NTNU
Innovation and Creativity

Voxelization — Algorithms

Corresponding 4×4 rotation matrix

$$\begin{bmatrix} s^2 + v_x^2 - v_y^2 - v_z^2 & 2v_xv_y - 2sv_z & 2sv_y + 2v_xv_z & 0 \\ 2sv_z + 2v_xv_y & s^2 - v_x^2 + v_y^2 - v_z^2 & 2v_yv_z - 2sv_x & 0 \\ 2v_xv_z - 2sv_y & 2sv_x + 2v_yv_z & s^2 - v_x^2 - v_y^2 + v_z^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

NTNU
Innovation and Creativity

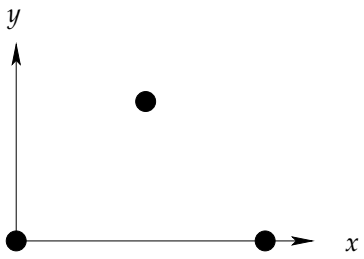
Voxelization Algorithm

Therefore:

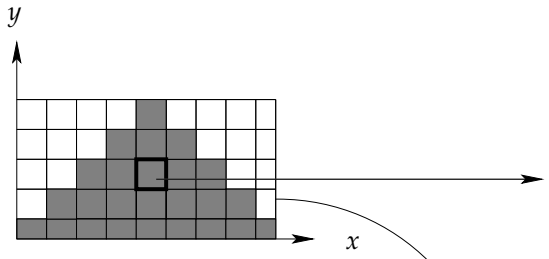
1. Rotate initial polygon vertices. From this we obtain the combined quaternion q
2. Apply rasterization to the transformed vertices.
3. For each rasterized coordinate found, multiply by rotation matrix obtained from q^{-1}.

NTNU
Innovation and Creativity

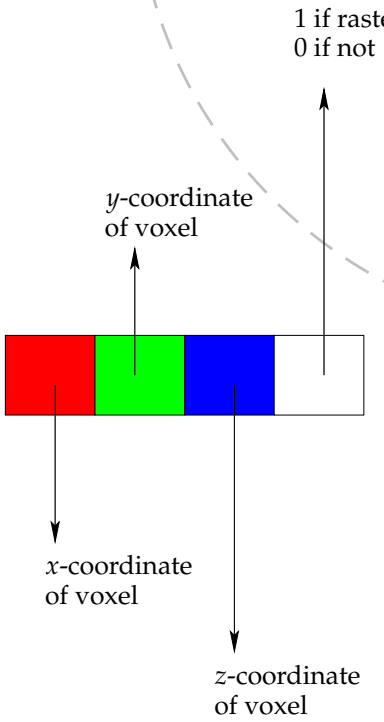
Voxelization Algorithm



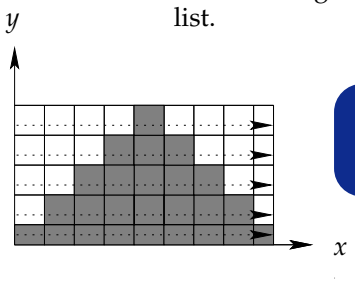
Step 1
Vertices passed to vertex processors



Step 2
Rasterization and fragment processor program execution. Each rasterized coordinate is multiplied by the inverse rotation/translation matrix, and the result is written to GPU texture memory.



Step 3
Entire output texture downloaded to CPU, which finds elements which have alpha component = 1, and then reads x,y,z coordinate of voxel from R, G, B-components The resulting coordinates are put into a list.



NTNU
Innovation and Creativity

Preliminary Results

In general: GPU Speedup only when polygons are about 10^5; about 2

- Arithmetic complexity of algorithm is low
- Post-processing which must be done on CPU ruins GPU speedup.
- Using multple cores gives a slowdown on Intel systems, but not on Njord (speedup = 2, max at 8 nodes, decreases for 16/32)

NTNU
Innovation and Creativity

Now, let’s talk about something completely different

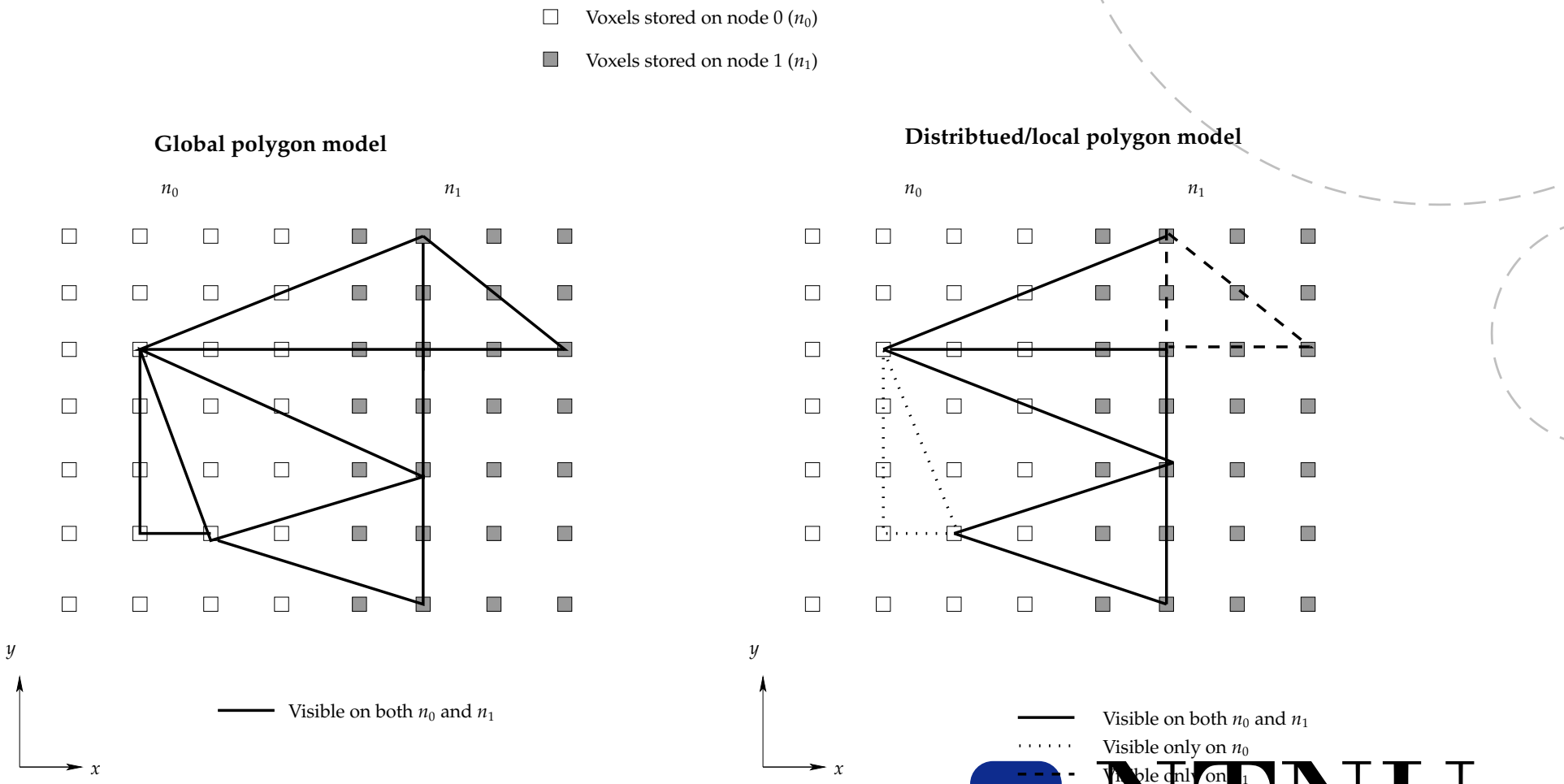
Framework Operations

Need support for the following operations:

- Create/destroy meshes
- Add/remove polygons
- Voxelize all polygons on all nodes (Extract-All Operation)
- Move vertices of polygons

Framework

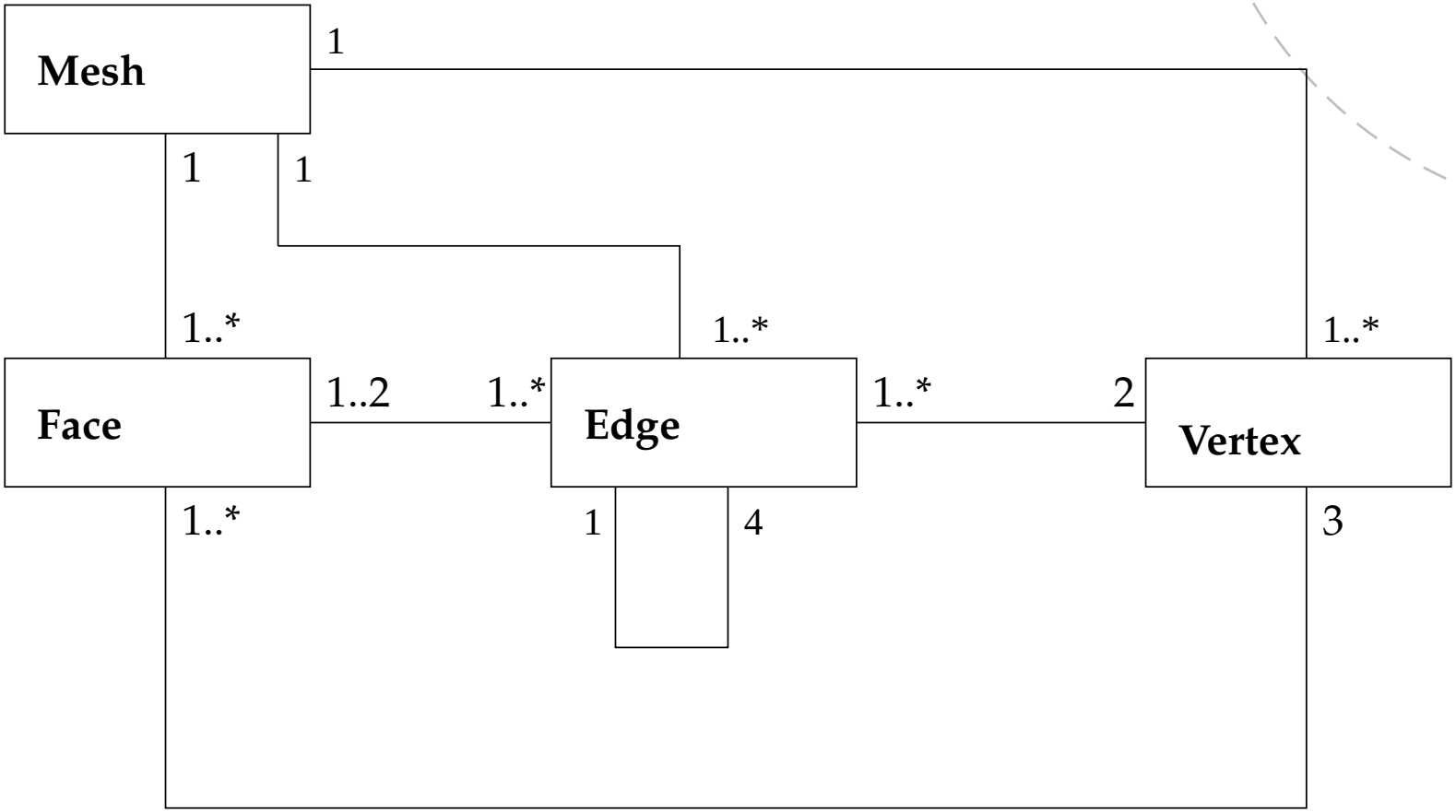
Decision: Keep polygon model global



NTNU
Innovation and Creativity

Framework

Use a variant of Baumgart's winged-edge model



NTNU
Innovation and Creativity

Framework

- Idea: Defer all communication to when voxelizing all polygons on all nodes.
- Keeps add/remove/move.. operations quick

Framework — Definitions

We use some definitions:

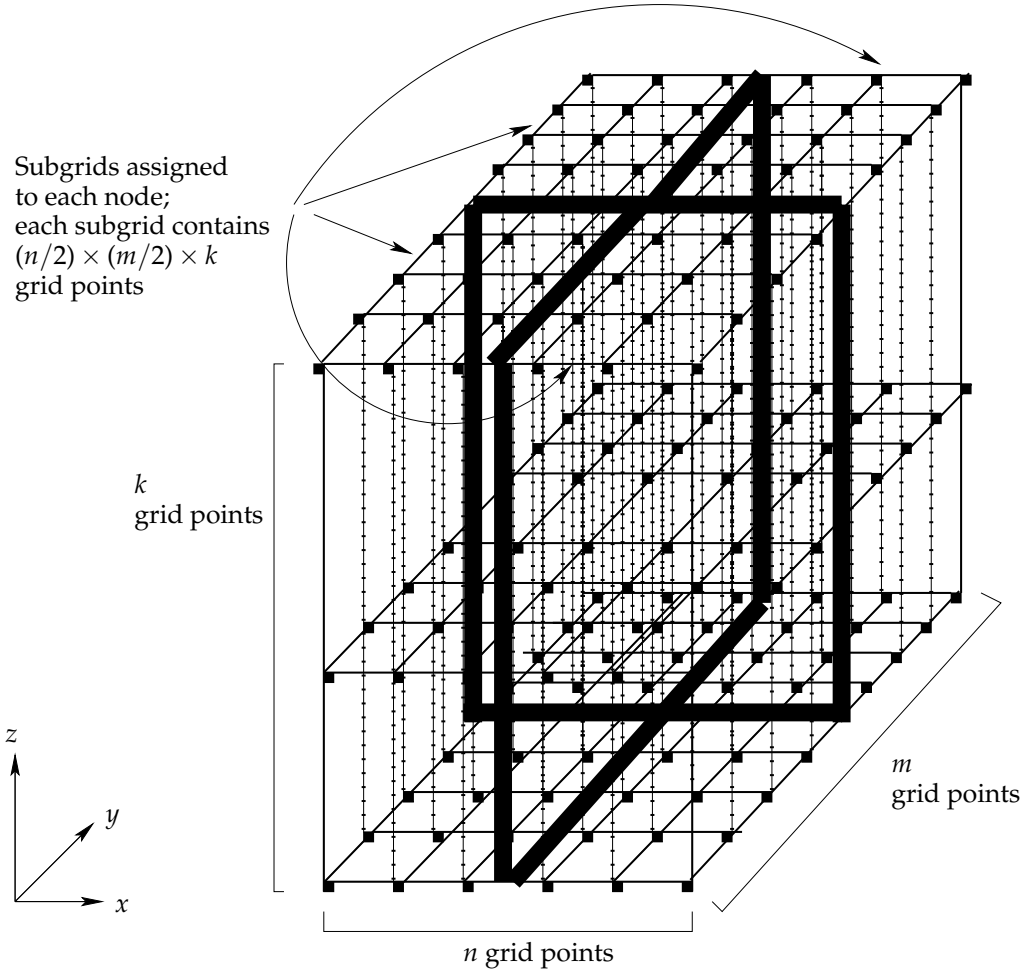
- *Centroid* $\bar{c}$ of a triangle:

$$\bar{c} = \frac{1}{3}\left(\sum_{i=1}^{3} v_{i,x}, \sum_{i=1}^{3} v_{i,y}, \sum_{i=1}^{3} v_{i,z}\right) \tag{4}$$

- *Coordinate space* C_i *of a node*: Each node i stores a subset C_i of the 3D space $N \times M \times K$. With $p = ab$ processes, $|C_i| = |N|/a \times |M|/b \times |K|$.

NTNU
Innovation and Creativity

Coordinate Spaces

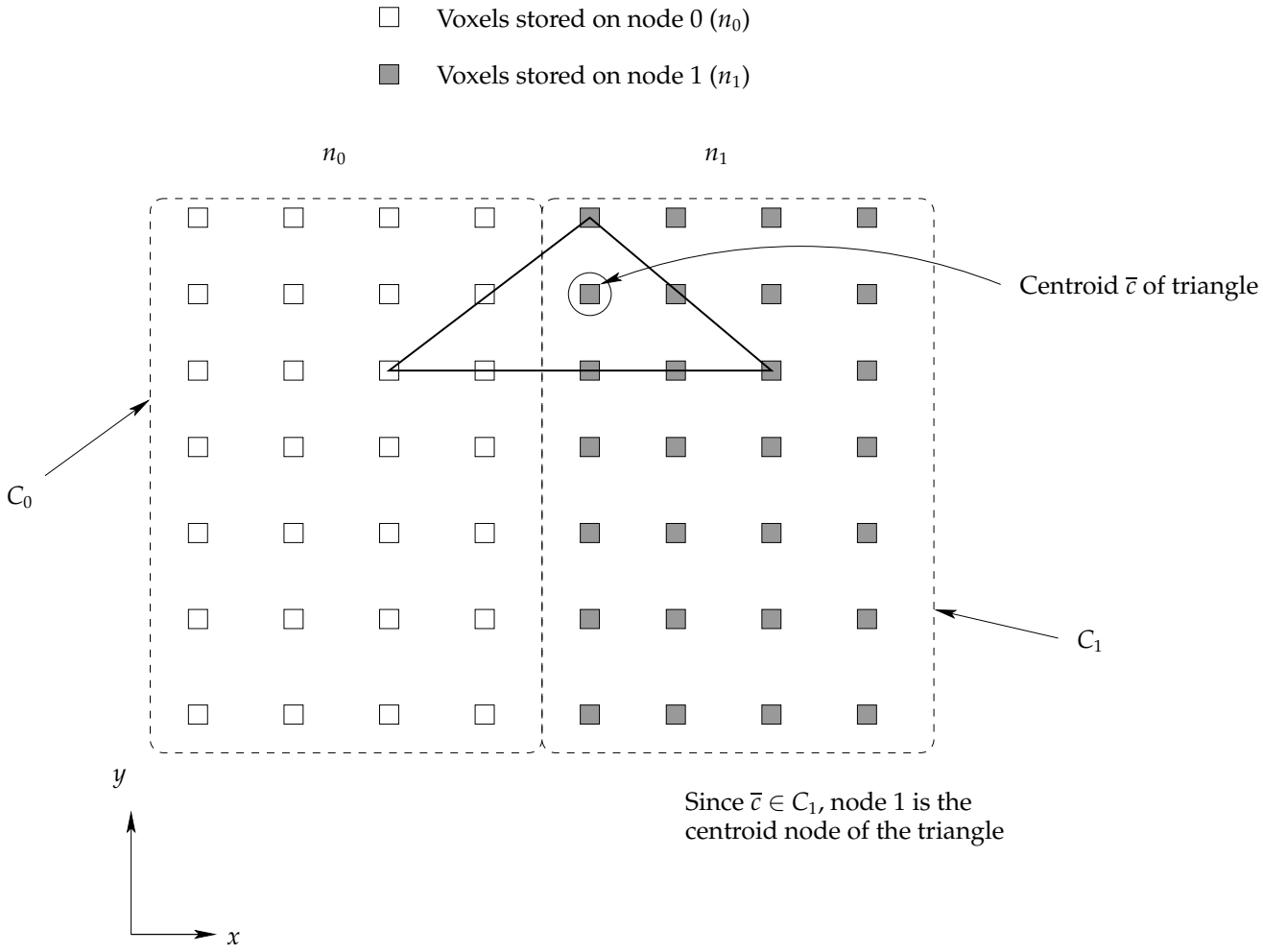


NTNU
Innovation and Creativity

Framework — Definitions

- Define the *centroid node* of a triangle T with centroid $\bar{c}$ to be the node i such that $\bar{c} \in C_i$.

Coordinate Spaces



Framework — Definitions

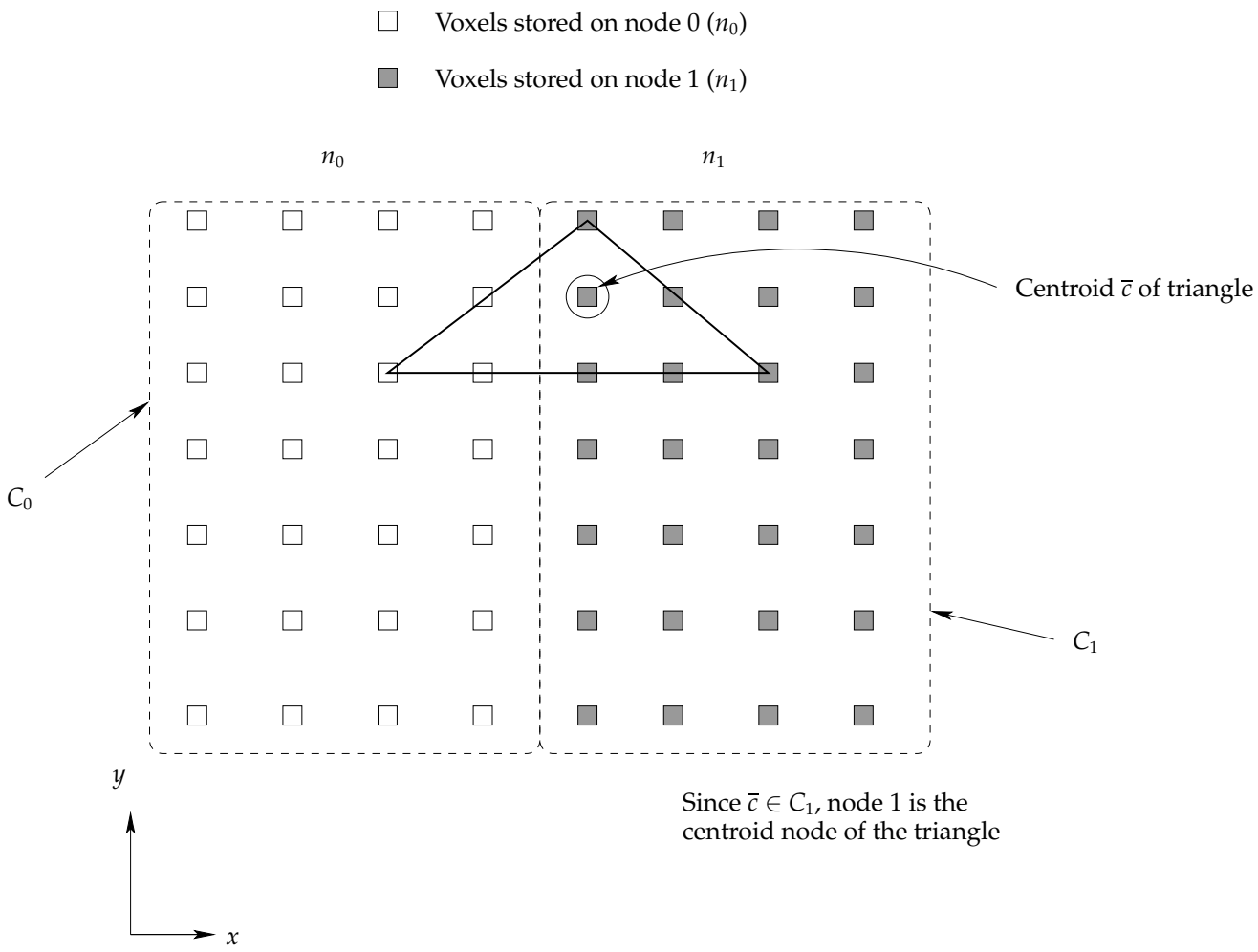
- Let F be a face. Then let $V^*(F)$ be the set of coordinates in F's voxelization.
- Define a function $V(p)$ which associates which a coordinate $p = (x, y, z)$ a user-defined, application-dependent value, typically single 32-bit float
- For each face F, all $V(p)$ values for $p \in V^*(F)$ must be copied/available to/on a single node in order to do computations

NTNU
Innovation and Creativity

Framework — Definitions

- Define the *responsible node* of a triangle T the node at which all $V(p)$ for $p \in V^*(T)$ values are available after doing the Extract-All operation.
- For now, assume that the responsible node is equal to the centroid node for all triangles.

Coordinate Spaces



NTNU
Innovation and Creativity

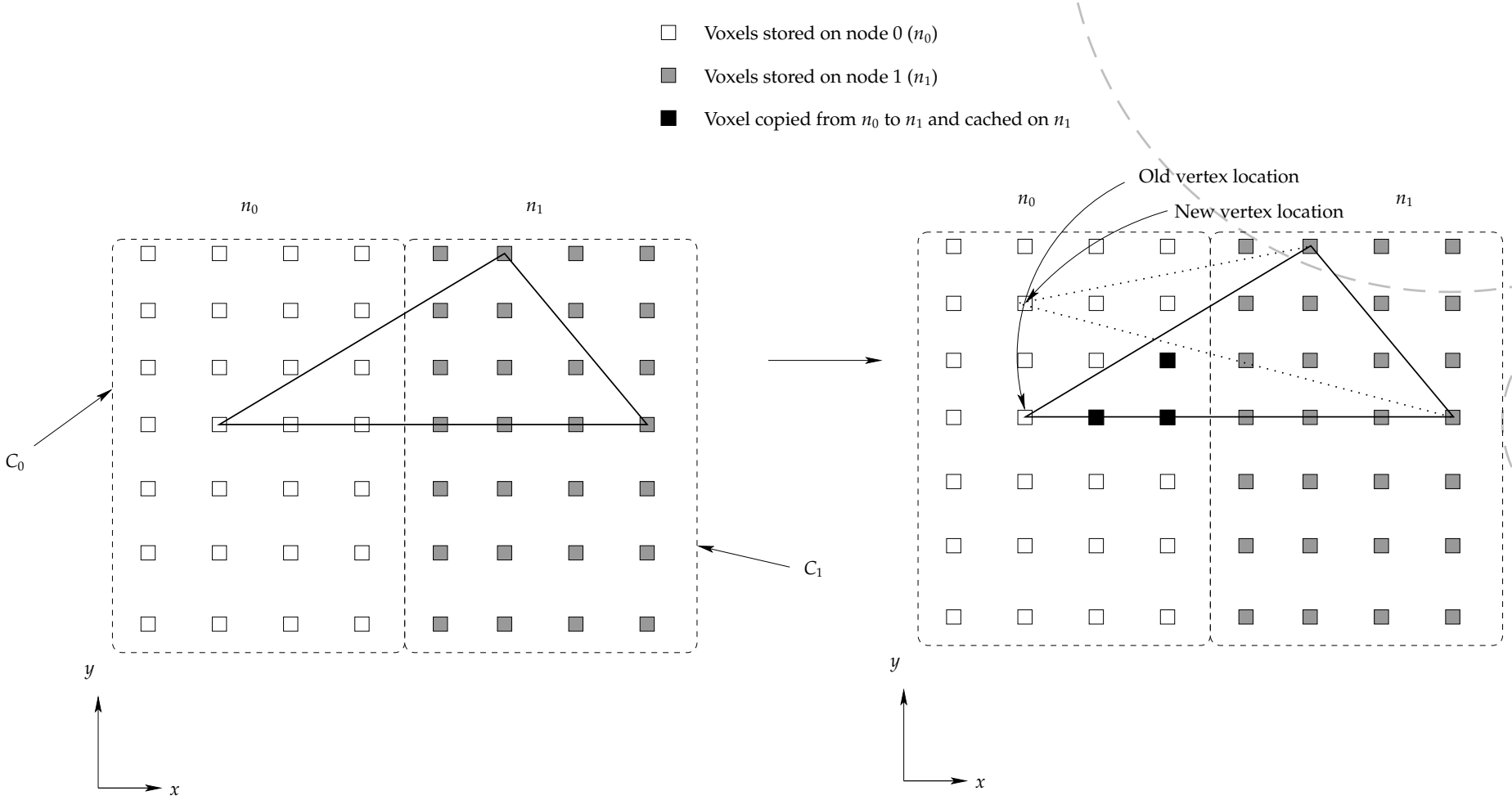
Caching Strategy I

On each node:

1. Voxelize all faces that node is responsible for.
2. Request missing $V(p)$ values and surrounding $k \geq 0$ $V(p)$ values from other nodes, receive requests from other nodes.
3. Transfer/receive voxels to/from other nodes, caching the voxels.
4. Do computation.

NTNU
Innovation and Creativity

Caching Strategy I



Caching Strategy II

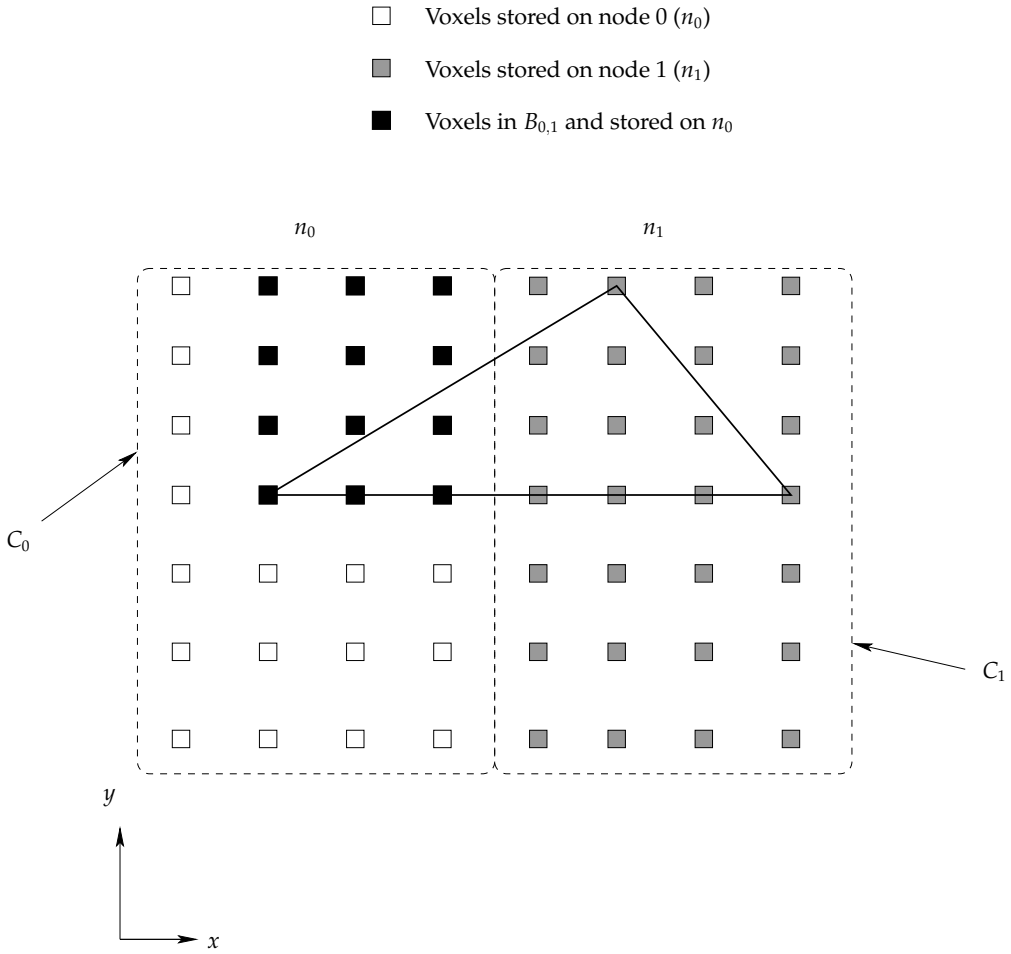
On each node:

1. For all faces, check if the bounding volume of the face intersects the coordinate space of the present node *and* has a different responsible node.
2. Transmit those voxels to the other node, if not already transmitted before.
3. (As nonblocking call in Step 2 proceeds) Voxelize all polygons node is responsible for.
4. Do computation when nonblocking call is complete.

NTNU

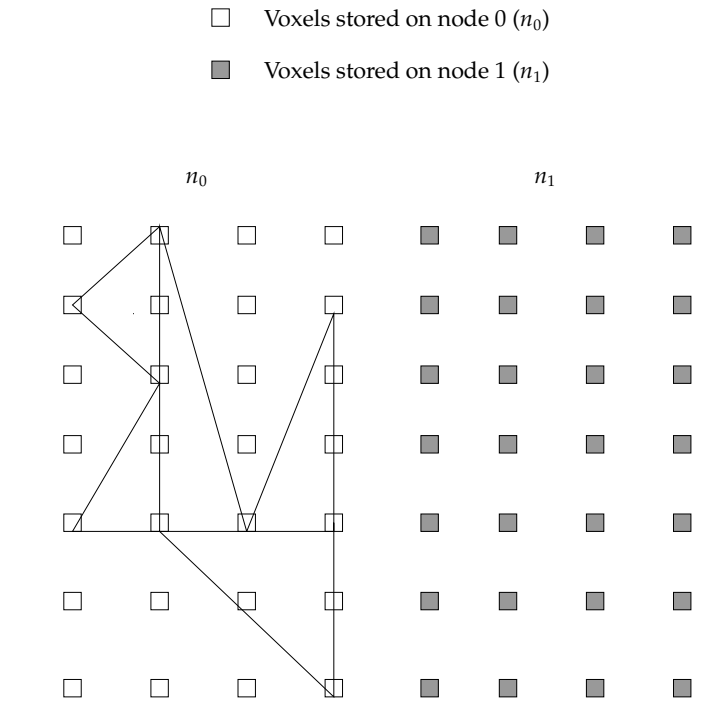
Innovation and Creativity

Caching Strategy II

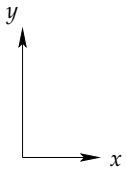


Load-Balancing

Question: Should the responsible node =



centriod node?



NTNU
Innovation and Creativity

Load-Balancing

- Let $A(F)$ be the area of a set of faces F, $\overline{A}$ = area of all faces/# of nodes, F^i = set of faces node i is responsible for
- Node i is overloaded if

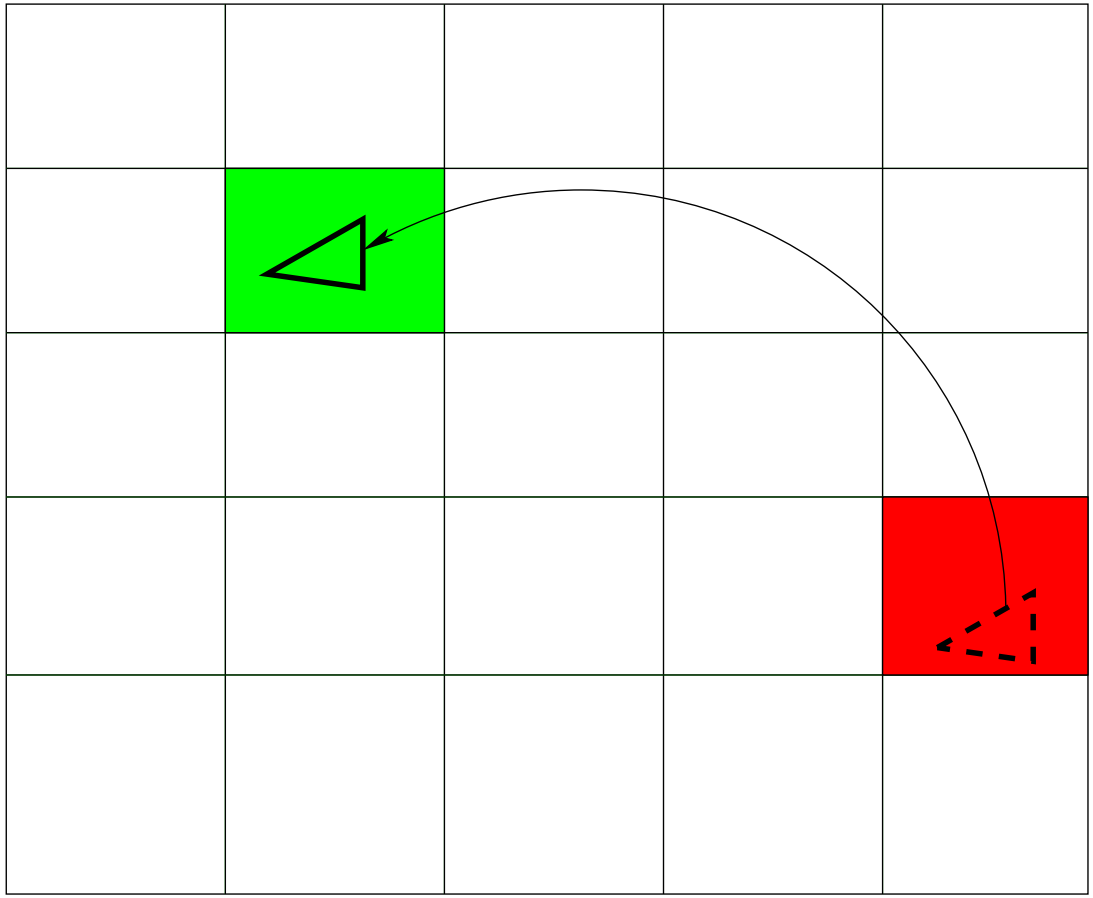
$$L(i) = A(F^i) - \overline{A} \tag{5}$$

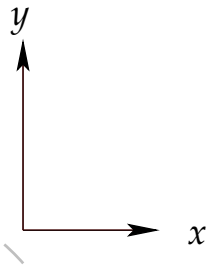
is > 0. Generally use a threshold $\delta \geq 0$ so that i is considered overloaded only if $L(i) > \delta$.

Load-Balancing

- All LB algorithms execute w/ no communication.
- Simplifies, minimizes overhead, and all nodes know which node is responsible for all faces.
- Potential disadvantage when scaling to 10,000+++ processors

Load-Balancing Strategy I: Global



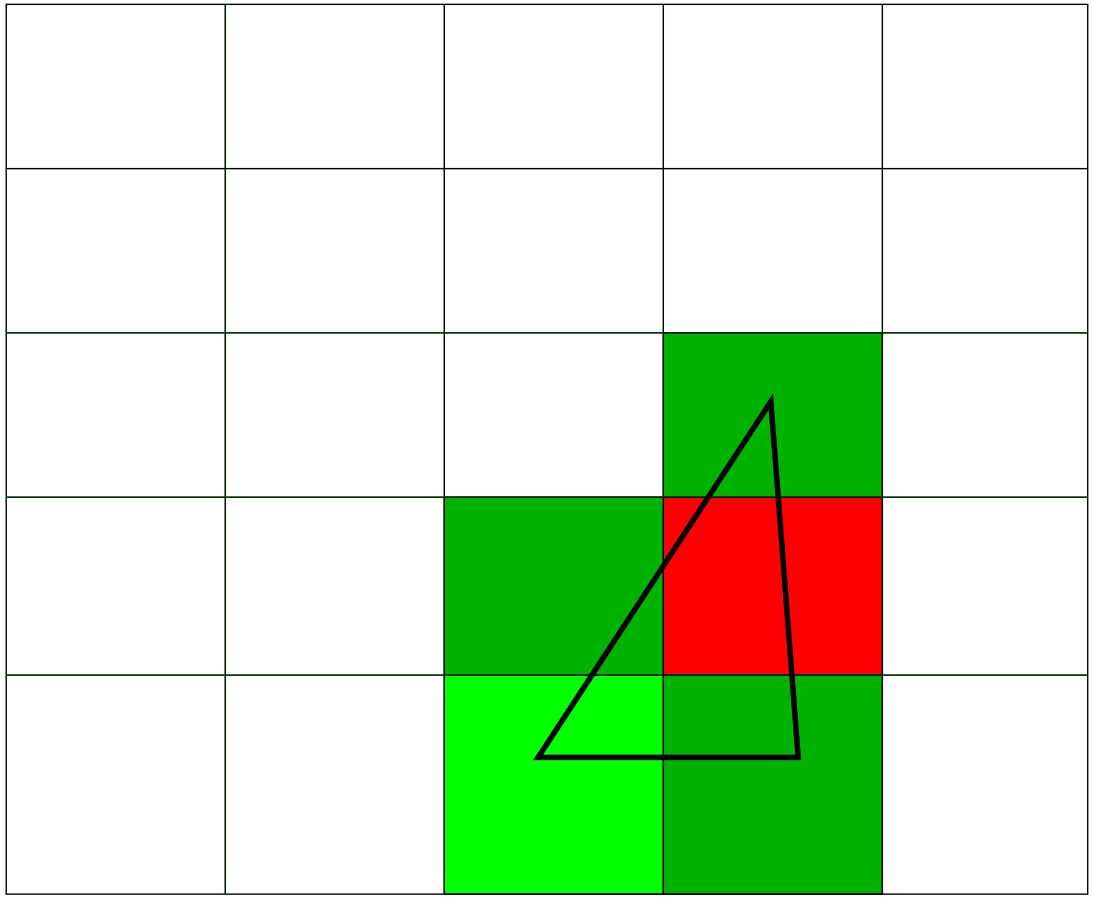


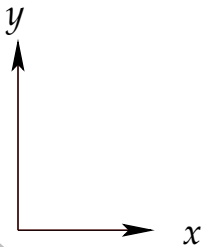
Overloaded node to which a triangle is added

The most underloaded node, to which the triangle from the overloaded node is assigned

NTNU
Innovation and Creativity

Load-Balancing Strategy II: Local





Overloaded centroid node

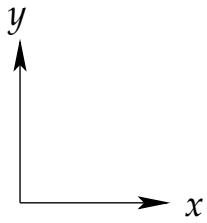
The most underloaded node of the nodes whose coordinate space the face intersects, and which is assigned to be responsible for the face

Other nodes whose coordinate spaces intersect the triangle

NTNU
Innovation and Creativity

Load-Balancing Strategy III: Manhattan

7	6	5	4	3	4	5	5
6	5	4	3	2	3	4	5
5	4	3	2	1	2	3	4
4	3	2	1	0	1	2	3
5	4	3	2	1	2	3	4
6	5	4	3	2	3	4	5



NTNU
Innovation and Creativity

Summary

- New parallel voxelization algorithm
- Caching/transfer strategies for the optimizing parallel voxelization operation
  - Strategy 1: Extract only required voxels (+ more if using greater block size)
  - Strategy 2: Based on bounding volumes, less data transferred + voxelization/transfer in parallel
- Load-balancing: global, local, Manhattan

NTNU
Innovation and Creativity

Future Work — Voxelization

- Compare to Kaufman's algorithm.
- Extend algorithm to not only voxelize triangles but entire meshes.

Future Work — Framework

- Further LB investigation.
  - How often should LB be done?
  - New LB algorithms which explicitly consider how many voxels have already been cached at target node
  - Make LB algorithms fully dynamic

Future Work — Framework

- Investigate parameters and optimal cache/LB strategies for various:
  - data set size, average polygon size, average distance moved when moving a vertex, number of nodes, number of voxelization threads on Njord, different workloads (we consider voxelize-move-voxelize-move.....), different LB thresholds

NTNU
Innovation and Creativity

Thank you for your attention!

NTNU
Innovation and Creativity