15.05.19

Application Image Compression (2D)

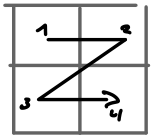
Method: Input grayscale image

1. Construct complete quadtree bottom up;
   Propergate sum, min and max of the pixel values.

2. Pruning: Prune subtree (i.e. replace by a leaf),
   if $\max(\text{block}) - \min(\text{block}) \leq \theta$ ← user defined threshold

3. Represent all pixels contained in a leaf
   by one grayscale value $= \frac{1}{2}(\min + \max)$ or .. $= \frac{1}{n} \cdot \text{sum}$
   ← #pixels in block

4. Coding grayscale values:



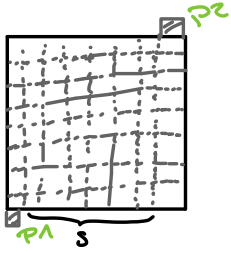
a) Traverse quadtree with DFS in z-order

b) $\text{code}(a, p, s, \theta) = \begin{cases} [(a-p) \cdot \frac{s}{\theta}], & \text{if } s < \theta \\ [a-p], & \text{if } s > \theta \end{cases}$

with a = grayscale in leaf, p = value of predictor
s = side length of leaf

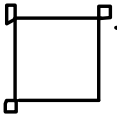
Note: code $\in [-255, 255]$, assumins $a \in [0, 255]$

Choose Predictor $p = \frac{1}{2}(p_1 + p_2)$



Note: p_1, p_2 are always known during step 4, b/c of z-order traversal

Note: p_1, p_2 must be encoded values! (b/c these only known in decoder)

Either prediction  → gradiat for square

5. Generate Treecode

Traverse Qtree in z-order, output 1 for inner nodes, output 0 for leaves, encoded and quantised grayscale value.

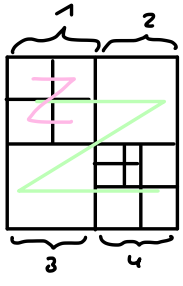
Quantization scheme (simple): $0 \rightarrow 0$

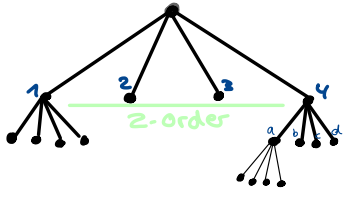
$-1 \rightarrow 100$ $\quad 1 \rightarrow 101$

$-2 \rightarrow 1100$ $\quad 2 \rightarrow 1101$

$-3 \rightarrow 11100$ $\quad 3 \rightarrow 11101$

(unary representation)





Treecode: 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0

root 1 | leaves of 1 | 1 2 3 4 a | b c d

Decompression

1. Reconstruct quadtree from treecode

2. Reconstruct grayscale values:

$$a = \begin{cases} code \cdot \frac{s}{\theta} + 12, & s < \theta \\ code + 12, & s > \theta \end{cases}$$

↳ already known, b/c p_1, p_2 are already reconstruct

3. Optional smoothing

Results

Faster than JPEG
e.g. at 1 bit/pixel → 2x faster
Similar RMSE

$$= \sqrt{\frac{1}{n} \sum_{i,j} (I[i,j] - D[i,j])^2}$$

(I: orig., D: decomp.)

Iso surfaces

Definitions:

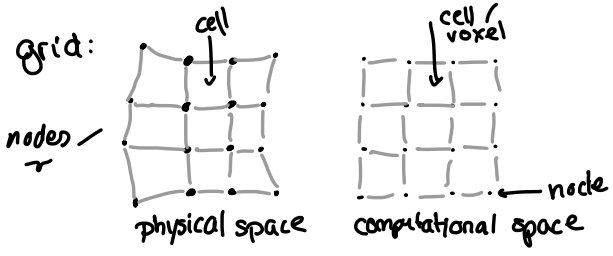
A fct $f: \mathbb{R}^3 \to \mathbb{R}$ is called scalar field. (function)

The isosurface over scalar field f with isovalue τ is all points in space

$$S = \{x \in \mathbb{R}^3 \mid f(x) = \tau\}$$

(example: isolating in molecules)

Curvilinear grid:



Implementation:

as 3D array:
where $F[i][j][k]$ stores position x of node, and scalar value $f(x)$

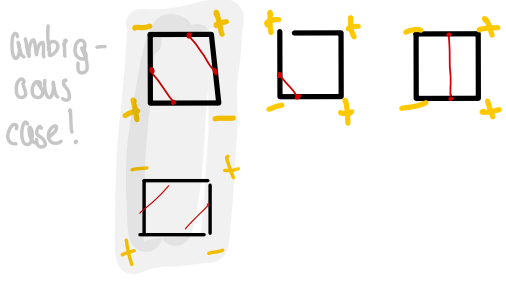
Isosurface S_τ over f with isovalue τ on computational grid is a surface where for all voxel intersecting S_τ:

$$\exists v: f(v_1) < \tau \wedge \exists v_2 : f(v_1) > \tau,$$

where $\{v_1, \dots, v_2\}$ are nodes of the voxel

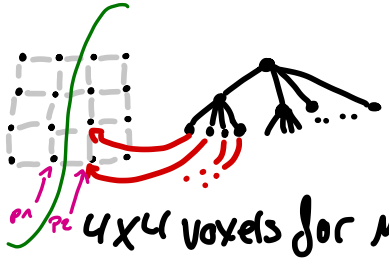
Marching cubes (first algo) 2D:

Consider all voxels (any order)
calculate signs at its nodes → ⊕, ⊖
triangulate voxel by LUT



Min - Max Octree:

Complete octree,
where leaves point to voxels, e.g. pointer to lower left node F[i][i][k].



4x4 voxels for Min-Max Octree

Leaves store min and max of scalar values over their nodes

Inner nodes store min/max over children

Observe:

isosurface crosses a voxel region of octree node v
$\Leftrightarrow \min(v) \leq \tau \leq \max v$

(quick talk in class ↓)

→ algorithm: construct isosurface
top - down
check above criterion → stop if not

Practical optimization:

Maintain hash table for edges,
store vertex on an edge in the hashtable, who calculated for the first time

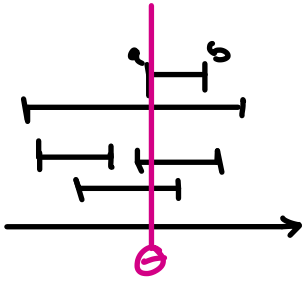
Position of vertex on edge $p_1 p_2$ is $lerp(p_1, p_2) = \frac{f(p_1)}{f(p_1)+f(p_2)} p_1 + \frac{f(p_2)}{f(p_1)+f(p_2)} p_2$

Digression: Stabbing Query

Problem: 1-dim case

Given N intervalls $[a_i, b_i] \subseteq \mathbb{R}$
θ = query point

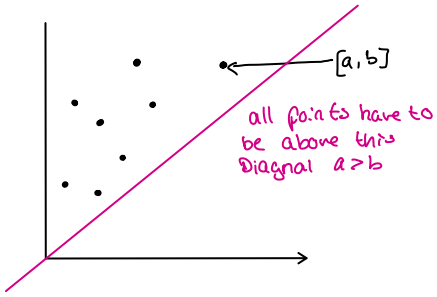
Wanted: all intervalls with $\theta \in [a_i, b_i]$



Standard algo: intrival tree

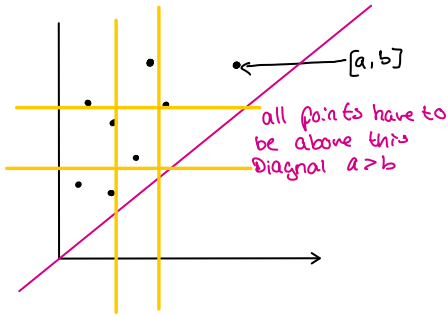
The Span Space.

Consider $[a,b]$ as points $(a,b) \in \mathbb{R}^2$



Superimpose a lattice of $L \times L$ cells, such that

1. points are approx uniformly distrib. among cells
2. Diagonal intersects cells diagonally, (or not at all) $\Rightarrow$ side length of cells along $a=b$ must be equal



For each row i in lattice store 2 lists:

1) $L_i^a = \{$ all pts in row i, in cols $1, \dots, (i-1)$, sorted by coord $a\}$

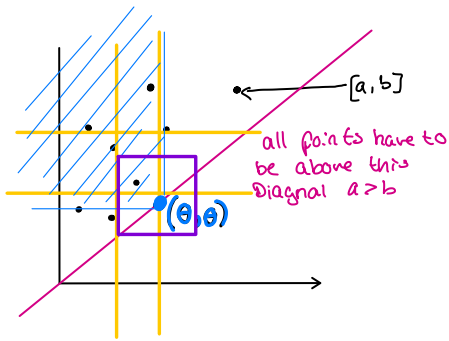
2) $L_i^b = \{$ all pts in row i, cols $1, \dots, (i-a)$, sorted by b descending$\}$

3) For lattice cells on Diagonal:

options: a) recursively geomete lattice
b) use interval tree, or just simple list

Algorithm:

find lattice cell (l,l) that contains (θ,θ)



for rows $i = l+1, \dots$:
traverse L_i^a up to $L_i^a[j] > \theta$

image

in row l:
traverse L_l^b up to $L_l^b[j] < \theta$

image

for cell (l,l):
recurse into lattice stored with this cell (option a)

For option b:
consider all points in (l,l), or traverse interval tree

Complexity:

Assumption: each lattice cell contains $\frac{2n}{L^2}$ points (intervals).

$$T(n) = O(k) + O(L) + T\left(\frac{2n}{L^2}\right)$$

- $O(k)$: inner body of finest loop + second loop
- $O(L)$: first loop
- $T\left(\frac{2n}{L^2}\right)$: diagonal

$$\Rightarrow T(n) = O(L \cdot \log_{L^2} n + k); \text{ e.g. choose } L = \log n \Rightarrow T(n) = O\left(\frac{\log^2 n}{\log \underbrace{\log n}_{L}} + k\right)$$

Isosurface over time-varying fields

Given: N 3D scalar fields for time $t_i \in [t_0, \dots, t_{N-1}]$
(write $t = i$)

Trivial solution: one octree / span space per time step:

Def: $\min_t(v) =$ min over all nodes of voxel v

$\max_t(v) =$ max $-\!\!-$

$$\min_i^j(v) = \min_{t \in [i,j]} \{\min_t(v)\}$$

$$\max_i^j(v) = \max_{t \in [i,j]} \{\max_t(v)\}$$

("temporal min/max of a voxel")

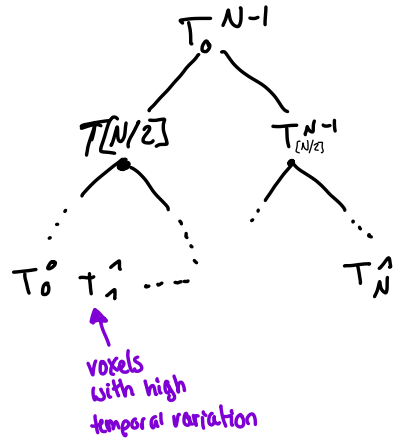
Observation: as $t = i, \dots, j$, the point $(\min_t(v), \max_t(v))$ moved around in span space

Def: voxel v has low temporal variation over time interval $[i,j]$
$\Leftrightarrow$
all points $(\min_t(v), \max_t(v))$ lie in at most 2×2 lattice cells in span space

Temporal index tree (TI tree):

T_i^j contains all voxels $v \in V$ with low temporal variation over time interval $[i,j]$, and which are not stored at one of the parents of T_i^j.

$V(T_i^j) = \{$ voxels with small variations $\}$



Construction

Start with all voxels V and T_0^U

build lattice in span space over V and time interval $[0, N-1]$

check "low variations" for all $v \in V$,

if true $\rightarrow$ add v to $V(T_0^N)$

recursion for $T_0^{N/2-1}$, $T_{N/2}^N$ with $V - V(T_0^{N-1})$.

build octree for all voxels $v \in V(T_i^j)$ with

$V = (\min_i^j(V), \max_i^j(V))$ (no longer complete octree)

Isosurface generation:

Given t, Θ

1) Traversee TI tree from $T_0^N, \ldots, T_t^t$.
   For all nodes T_a^b on path: $a \leq t \leq b$.

2) For all T_a^b search associated octree for voxels
   v with $\min_a^b(v) \leq \Theta \leq \max_a^b(v)$ (just like for "static octree")

3) Filter voxels by testing $\min_t(v) \leq \Theta \leq \max_t(v)$

← Question:
when/how are "bad" voxels returned here EXAM