Geometric image approximation

Adhemar Bultheel

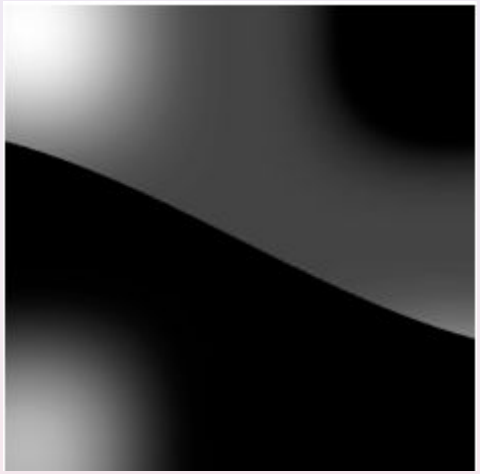
Department of Computer Science
K.U.Leuven

Wavelets and fractals
Esneux, April 26-28, 2010

Survey

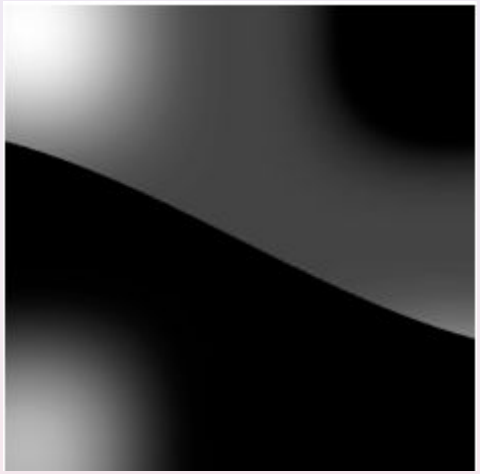
- Motivation and problem setting
- Normal offsets in 1D and 2D
- Decoration with polynomial wavelets
- tree pruning and encoding
- Experimental results

Horizon images, the class $\mathcal{H}$

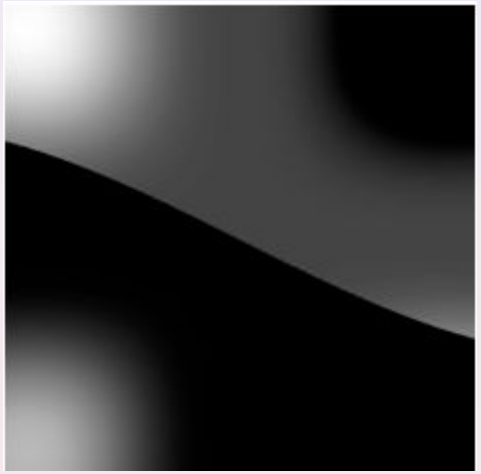

- $H^\alpha = \{c : [0,1] \to \mathbb{R} : |D^s c(x) - D^s c(x')| \leq C_\alpha |x - x'|^{\alpha - s}\}$, $s = \lfloor \alpha \rfloor$ (Hölder class).
- $\mathcal{PS}^{\alpha,\beta} = \{f : [0,1]^2 \to \mathbb{R}, f \in H^\beta, \text{ if } y \neq c(x),\ c \in H^\alpha\}$ (piecewise smooth) $\alpha, \beta \in (1,2]$
- $\mathcal{H}^\alpha = \{f : [0,1]^2 \to \{0,1\} :$ $f(x,y) = 1, \text{if } y \leq c(x), \text{ 0 otherwise}, \quad c \in H^\alpha\}$ (horizon class, $\alpha \in (1,2]$, most often $\alpha = 2$, then we write $\mathcal{H}$)

Horizon images, the class $\mathcal{H}$

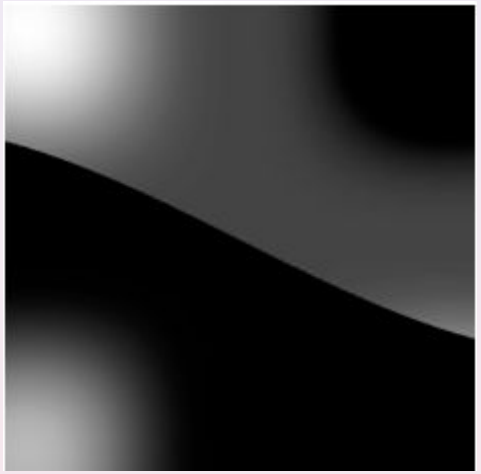

- $H^\alpha = \{c : [0,1] \to \mathbb{R} : |D^s c(x) - D^s c(x')| \leq C_\alpha |x - x'|^{\alpha - s}\}$, $s = \lfloor \alpha \rfloor$ (Hölder class).
- $\mathcal{PS}^{\alpha,\beta} = \{f : [0,1]^2 \to \mathbb{R}, f \in H^\beta, \text{ if } y \neq c(x), \ c \in H^\alpha\}$ (piecewise smooth) $\alpha, \beta \in (1,2]$
- $\mathcal{H}^\alpha = \{f : [0,1]^2 \to \{0,1\} :$ $f(x,y) = 1, \text{if } y \leq c(x), \text{ 0 otherwise}, \quad c \in H^\alpha\}$ (horizon class, $\alpha \in (1,2]$, most often $\alpha = 2$, then we write $\mathcal{H}$)

Horizon images, the class $\mathcal{H}$

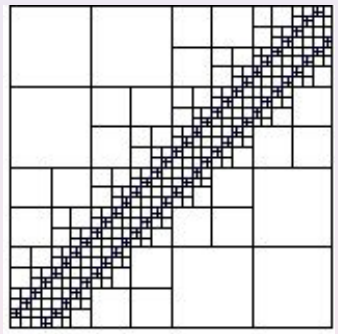

- $H^\alpha = \{c : [0,1] \to \mathbb{R} : |D^s c(x) - D^s c(x')| \leq C_\alpha |x - x'|^{\alpha - s}\}$, $s = \lfloor \alpha \rfloor$ (Hölder class).
- $\mathcal{PS}^{\alpha,\beta} = \{f : [0,1]^2 \to \mathbb{R}, f \in H^\beta, \text{ if } y \neq c(x), \ c \in H^\alpha\}$ (piecewise smooth) $\alpha, \beta \in (1,2]$
- $\mathcal{H}^\alpha = \{f : [0,1]^2 \to \{0,1\} :$ $f(x,y) = 1, \text{if } y \leq c(x), \text{ 0 otherwise}, \quad c \in H^\alpha\}$ (horizon class, $\alpha \in (1,2]$, most often $\alpha = 2$, then we write $\mathcal{H}$)

Horizon images, the class $\mathcal{H}$

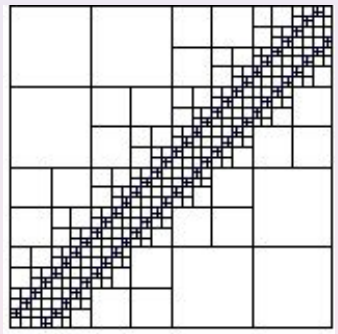

- $H^\alpha = \{c : [0,1] \to \mathbb{R} : |D^s c(x) - D^s c(x')| \leq C_\alpha |x - x'|^{\alpha - s}\}$, $s = \lfloor \alpha \rfloor$ (Hölder class).
- $\mathcal{PS}^{\alpha,\beta} = \{f : [0,1]^2 \to \mathbb{R}, f \in H^\beta, \text{ if } y \neq c(x), \ c \in H^\alpha\}$ (piecewise smooth) $\alpha, \beta \in (1,2]$
- $\mathcal{H}^\alpha = \{f : [0,1]^2 \to \{0,1\} :$ $f(x,y) = 1, \text{if } y \leq c(x), \text{ 0 otherwise}, \quad c \in H^\alpha\}$ (horizon class, $\alpha \in (1,2]$, most often $\alpha = 2$, then we write $\mathcal{H}$)

Problem with dyadic wavelet approximation



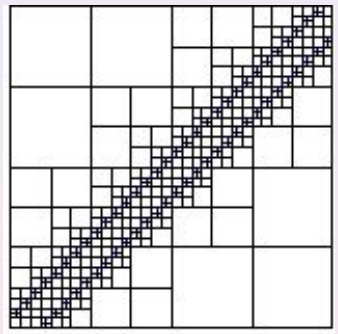
- Large number of dyadic squares of size 2^{-j} that intersect the curve $c(t)$.
- Slow decay of wavelet coefficients.
- Taking n largest wavelet coefficients gives approximant f_n and $\|f - f_n\|_2 = O(n^{-1/2})$
- while wavelet approximation of $c \in C^2$ decays like $O(n^{-2})$
- (Classical) wavelets are good for *point* singularities, but behave poorly on *line* singularities

Problem with dyadic wavelet approximation



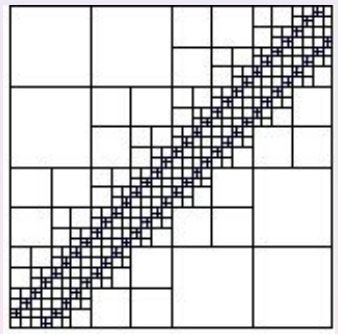
- Large number of dyadic squares of size 2^{-j} that intersect the curve $c(t)$.
- Slow decay of wavelet coefficients.
- Taking n largest wavelet coefficients gives approximant f_n and $\|f - f_n\|_2 = O(n^{-1/2})$
- while wavelet approximation of $c \in C^2$ decays like $O(n^{-2})$
- (Classical) wavelets are good for *point* singularities, but behave poorly on *line* singularities

Problem with dyadic wavelet approximation



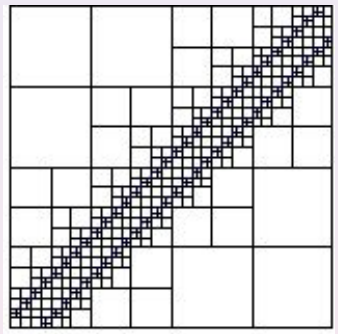
- Large number of dyadic squares of size 2^{-j} that intersect the curve $c(t)$.
- Slow decay of wavelet coefficients.
- Taking n largest wavelet coefficients gives approximant f_n and $\|f - f_n\|_2 = O(n^{-1/2})$
- while wavelet approximation of $c \in C^2$ decays like $O(n^{-2})$
- (Classical) wavelets are good for *point* singularities, but behave poorly on *line* singularities

Problem with dyadic wavelet approximation



- Large number of dyadic squares of size 2^{-j} that intersect the curve $c(t)$.
- Slow decay of wavelet coefficients.
- Taking n largest wavelet coefficients gives approximant f_n and $\|f - f_n\|_2 = O(n^{-1/2})$
- while wavelet approximation of $c \in C^2$ decays like $O(n^{-2})$
- (Classical) wavelets are good for *point* singularities, but behave poorly on *line* singularities

Problem with dyadic wavelet approximation



- Large number of dyadic squares of size 2^{-j} that intersect the curve $c(t)$.
- Slow decay of wavelet coefficients.
- Taking n largest wavelet coefficients gives approximant f_n and $\|f - f_n\|_2 = O(n^{-1/2})$
- while wavelet approximation of $c \in C^2$ decays like $O(n^{-2})$
- (Classical) wavelets are good for *point* singularities, but behave poorly on *line* singularities

Many alternatives

A multitude of techniques and -lets.
Optimize cvg rate of best approximation with n parameters in a class ($\alpha = 2$).

- Ridgelets (Donoho)
- Curvelets/Contourlets (Candès, Donoho) $O(n^{-2} \log n)$
- Wedgelets (Donoho) $O(n^{-2}) + \delta$ (δ angular resolution)
- Dictionaries (e.g. Basis pursuit and matching pursuit)
- Bandelets (Mallat) (wavelets adapted to geometric contents)
- Domain partitioning (edge detection and segmentation)
- Binary space partitioning (e.g. geometric wavelets)
- Adaptive thinning (adaptive thinning of triangular mesh)

Many alternatives

A multitude of techniques and -lets.
Optimize cvg rate of best approximation with n parameters in a class ($\alpha = 2$).

- Ridgelets (Donoho)
- Curvelets/Contourlets (Candès, Donoho) $O(n^{-2} \log n)$
- Wedgelets (Donoho) $O(n^{-2}) + \delta$ (δ angular resolution)
- Dictionaries (e.g. Basis pursuit and matching pursuit)
- Bandelets (Mallat) (wavelets adapted to geometric contents)
- Domain partitioning (edge detection and segmentation)
- Binary space partitioning (e.g. geometric wavelets)
- Adaptive thinning (adaptive thinning of triangular mesh)

Many alternatives

A multitude of techniques and -lets.
Optimize cvg rate of best approximation with n parameters in a class ($\alpha = 2$).

- Ridgelets (Donoho)
- Curvelets/Contourlets (Candès, Donoho) $O(n^{-2} \log n)$
- Wedgelets (Donoho) $O(n^{-2}) + \delta$ (δ angular resolution)
- Dictionaries (e.g. Basis pursuit and matching pursuit)
- Bandelets (Mallat) (wavelets adapted to geometric contents)
- Domain partitioning (edge detection and segmentation)
- Binary space partitioning (e.g. geometric wavelets)
- Adaptive thinning (adaptive thinning of triangular mesh)

Many alternatives

A multitude of techniques and -lets.
Optimize cvg rate of best approximation with n parameters in a class ($\alpha = 2$).

- Ridgelets (Donoho)
- Curvelets/Contourlets (Candès, Donoho) $O(n^{-2} \log n)$
- Wedgelets (Donoho) $O(n^{-2}) + \delta$ (δ angular resolution)
- Dictionaries (e.g. Basis pursuit and matching pursuit)
- Bandelets (Mallat) (wavelets adapted to geometric contents)
- Domain partitioning (edge detection and segmentation)
- Binary space partitioning (e.g. geometric wavelets)
- Adaptive thinning (adaptive thinning of triangular mesh)

Many alternatives

A multitude of techniques and -lets.
Optimize cvg rate of best approximation with n parameters in a class ($\alpha = 2$).

- Ridgelets (Donoho)
- Curvelets/Contourlets (Candès, Donoho) $O(n^{-2} \log n)$
- Wedgelets (Donoho) $O(n^{-2}) + \delta$ (δ angular resolution)
- Dictionaries (e.g. Basis pursuit and matching pursuit)
- Bandelets (Mallat) (wavelets adapted to geometric contents)
- Domain partitioning (edge detection and segmentation)
- Binary space partitioning (e.g. geometric wavelets)
- Adaptive thinning (adaptive thinning of triangular mesh)

Many alternatives

A multitude of techniques and -lets.
Optimize cvg rate of best approximation with n parameters in a class ($\alpha = 2$).

- Ridgelets (Donoho)
- Curvelets/Contourlets (Candès, Donoho) $O(n^{-2} \log n)$
- Wedgelets (Donoho) $O(n^{-2}) + \delta$ (δ angular resolution)
- Dictionaries (e.g. Basis pursuit and matching pursuit)
- Bandelets (Mallat) (wavelets adapted to geometric contents)
- Domain partitioning (edge detection and segmentation)
- Binary space partitioning (e.g. geometric wavelets)
- Adaptive thinning (adaptive thinning of triangular mesh)

Many alternatives

A multitude of techniques and -lets.
Optimize cvg rate of best approximation with n parameters in a class ($\alpha = 2$).

- Ridgelets (Donoho)
- Curvelets/Contourlets (Candès, Donoho) $O(n^{-2} \log n)$
- Wedgelets (Donoho) $O(n^{-2}) + \delta$ (δ angular resolution)
- Dictionaries (e.g. Basis pursuit and matching pursuit)
- Bandelets (Mallat) (wavelets adapted to geometric contents)
- Domain partitioning (edge detection and segmentation)
- Binary space partitioning (e.g. geometric wavelets)
- Adaptive thinning (adaptive thinning of triangular mesh)

Many alternatives

A multitude of techniques and -lets.
Optimize cvg rate of best approximation with n parameters in a class ($\alpha = 2$).

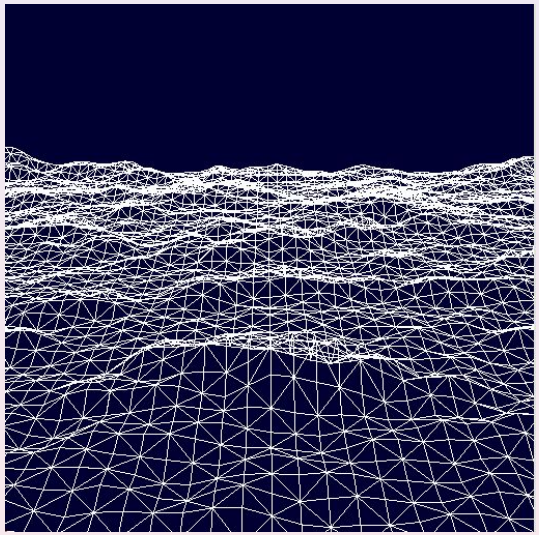
- Ridgelets (Donoho)
- Curvelets/Contourlets (Candès, Donoho) $O(n^{-2} \log n)$
- Wedgelets (Donoho) $O(n^{-2}) + \delta$ (δ angular resolution)
- Dictionaries (e.g. Basis pursuit and matching pursuit)
- Bandelets (Mallat) (wavelets adapted to geometric contents)
- Domain partitioning (edge detection and segmentation)
- Binary space partitioning (e.g. geometric wavelets)
- Adaptive thinning (adaptive thinning of triangular mesh)

Many alternatives

A multitude of techniques and -lets.
Optimize cvg rate of best approximation with n parameters in a class ($\alpha = 2$).

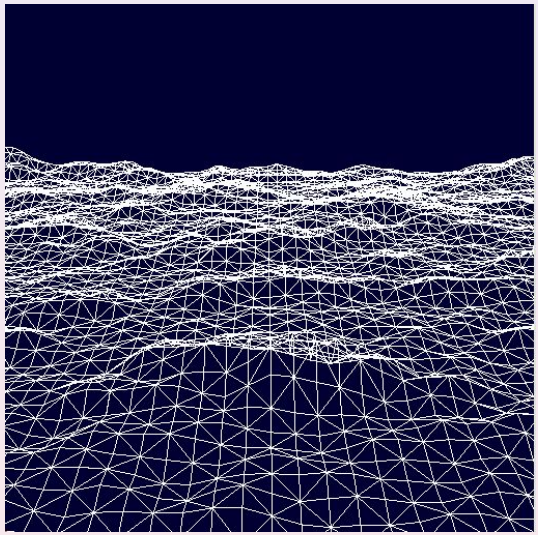
- Ridgelets (Donoho)
- Curvelets/Contourlets (Candès, Donoho) $O(n^{-2} \log n)$
- Wedgelets (Donoho) $O(n^{-2}) + \delta$ (δ angular resolution)
- Dictionaries (e.g. Basis pursuit and matching pursuit)
- Bandelets (Mallat) (wavelets adapted to geometric contents)
- Domain partitioning (edge detection and segmentation)
- Binary space partitioning (e.g. geometric wavelets)
- Adaptive thinning (adaptive thinning of triangular mesh)

Some terminology



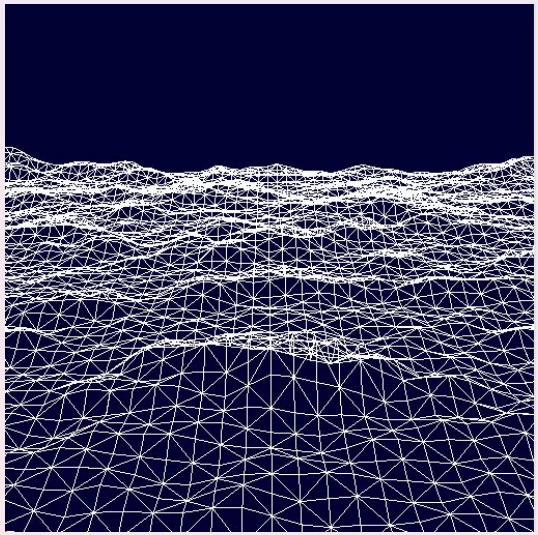
- A method with origins in computer graphics
- Pixel values = z-coordinate, then image = object
- triangulation of image = triangular mesh on the object
- However in CG the objects are smooth

Some terminology



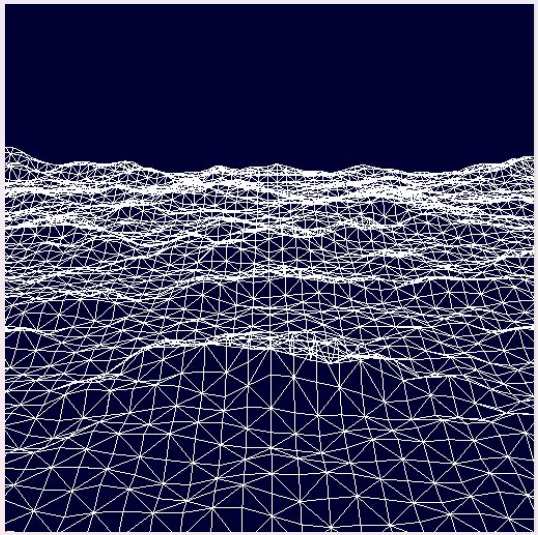
- A method with origins in computer graphics
- Pixel values = z-coordinate, then image = object
- triangulation of image = triangular mesh on the object
- However in CG the objects are smooth

Some terminology



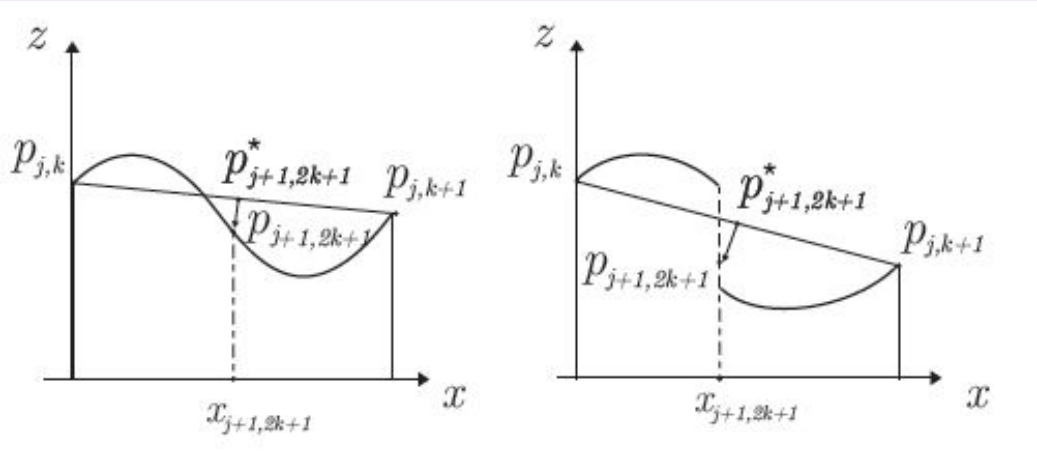
- A method with origins in computer graphics
- Pixel values = z-coordinate, then image = object
- triangulation of image = triangular mesh on the object
- However in CG the objects are smooth

Some terminology



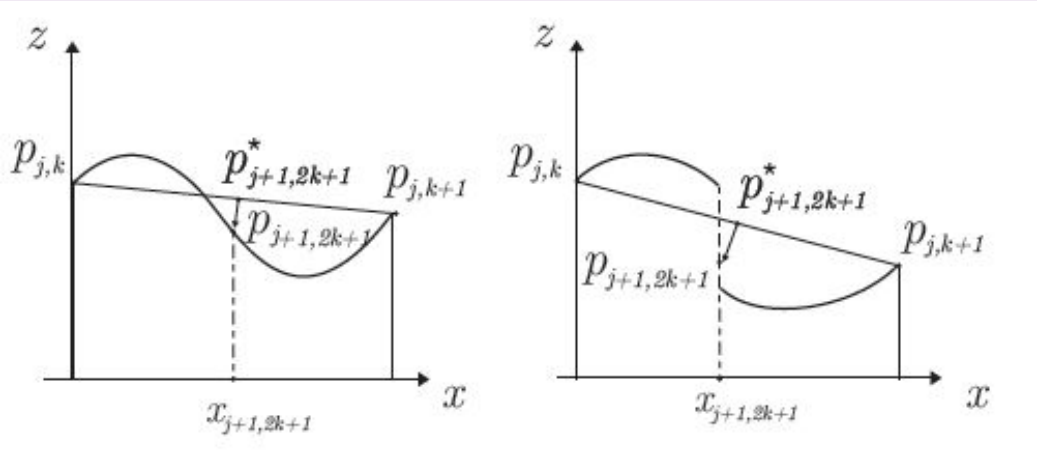
- A method with origins in computer graphics
- Pixel values = z-coordinate, then image = object
- triangulation of image = triangular mesh on the object
- However in CG the objects are smooth

Normal offsets in 1D



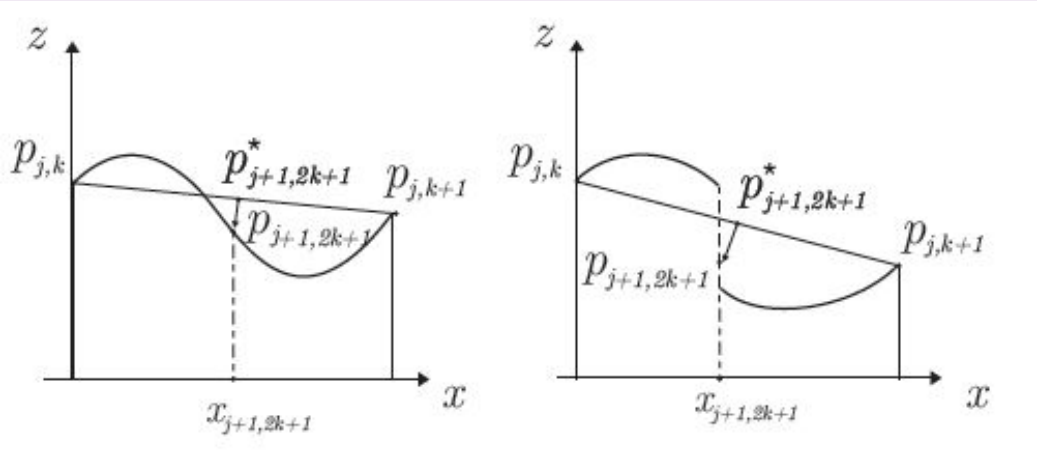
- normal bisector
- piercing point $\rightarrow$ normal offset
- defines refinement
- discontinuity rapidly detected
- vertical vs. normal offset

Normal offsets in 1D



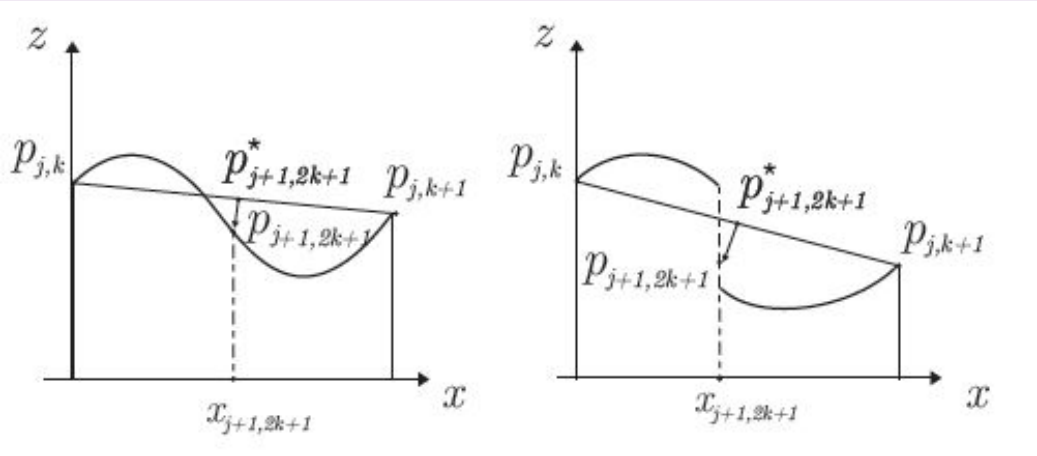
- normal bisector
- piercing point $\rightarrow$ normal offset
- defines refinement
- discontinuity rapidly detected
- vertical vs. normal offset

Normal offsets in 1D



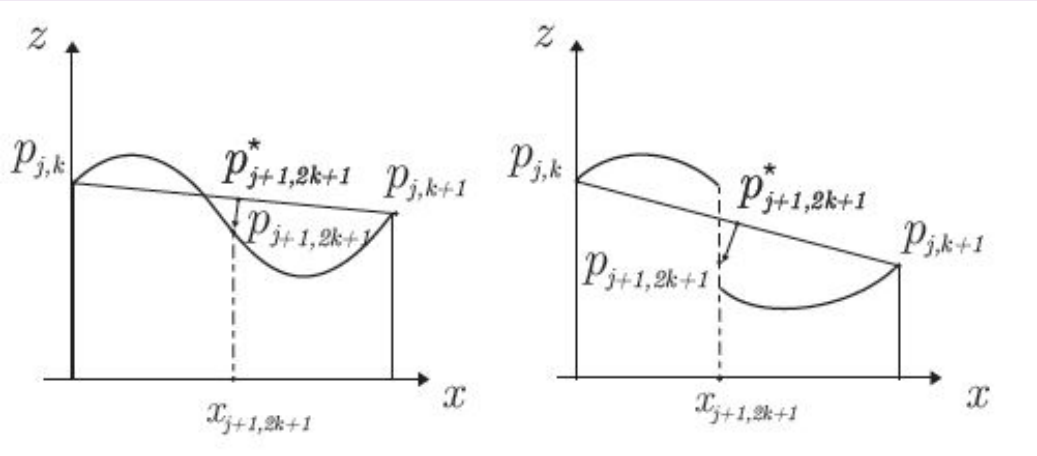
- normal bisector
- piercing point $\rightarrow$ normal offset
- defines refinement
- discontinuity rapidly detected
- vertical vs. normal offset

Normal offsets in 1D



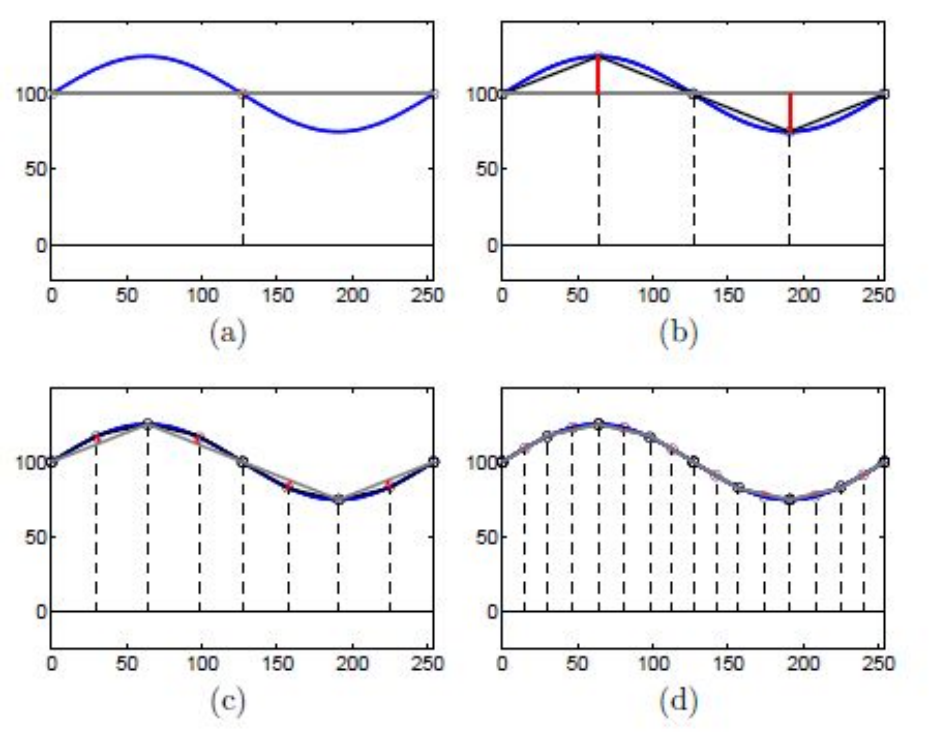
- normal bisector
- piercing point $\rightarrow$ normal offset
- defines refinement
- discontinuity rapidly detected
- vertical vs. normal offset

Normal offsets in 1D



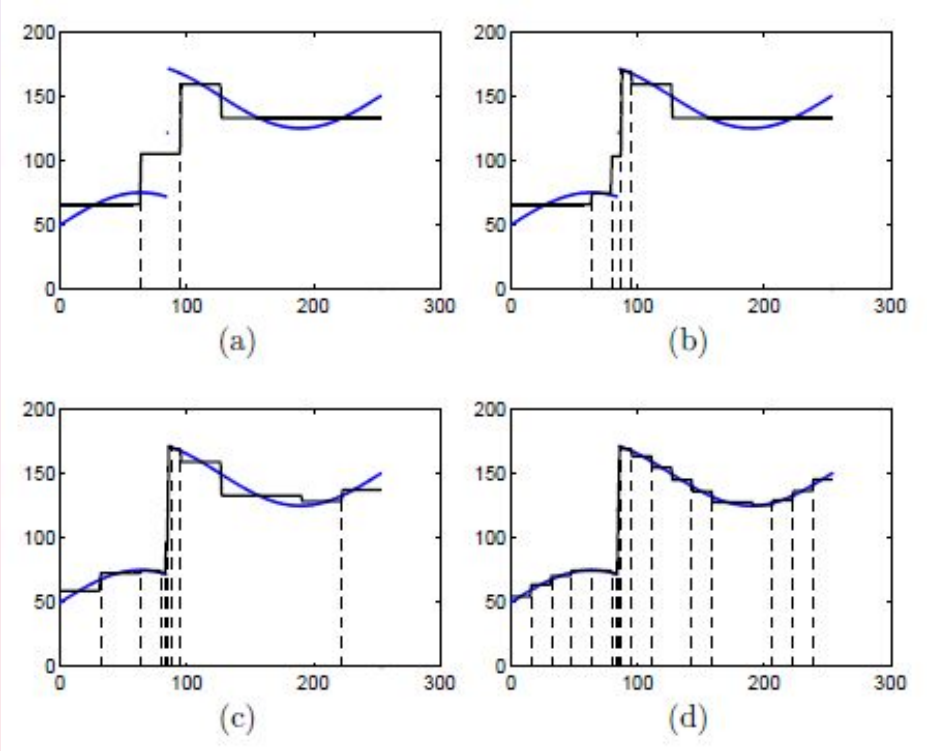
- normal bisector
- piercing point $\rightarrow$ normal offset
- defines refinement
- discontinuity rapidly detected
- vertical vs. normal offset

Normal offsets for a smooth curve



Normal offsets generate a regular grid on a smooth function.

Dyadic subdivision and singularity



Dyadic subdivision needs 'infinitely many' steps to locate the singularity.

Normal offsets in 2D

- Not a normal in the centers of the triangles of the object mesh
- A normal bisector for every edge in that mesh in a vertical plane though that edge.
- In that plane you do the 1D-case, with projections of the piercing points giving a subdivision point on every edge of the triangulation

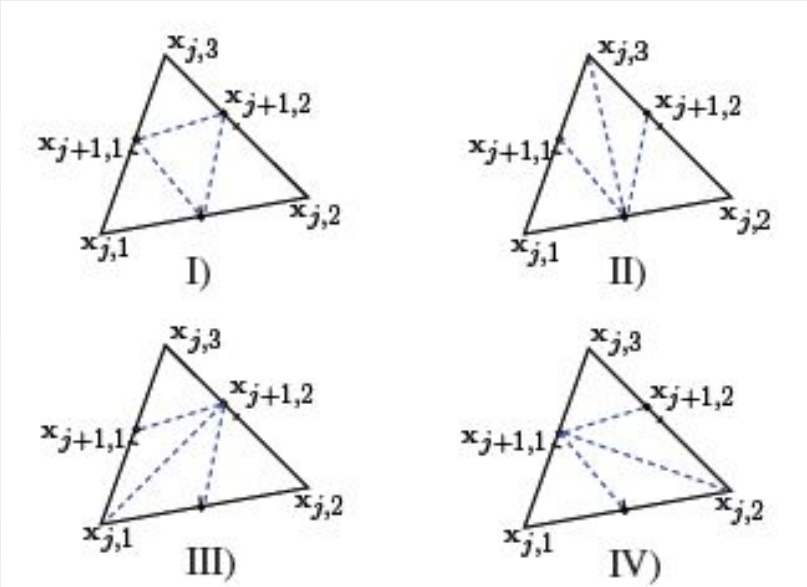
Normal offsets in 2D

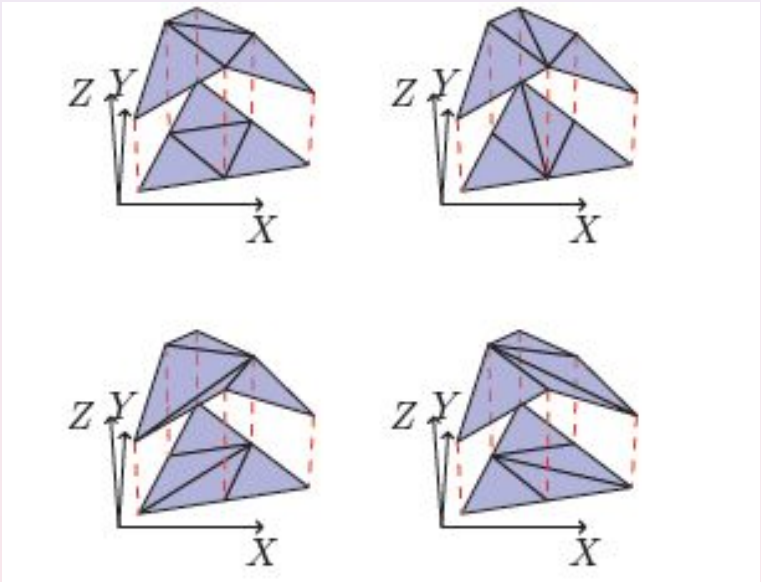
- Not a normal in the centers of the triangles of the object mesh
- A normal bisector for every edge in that mesh in a vertical plane though that edge.
- In that plane you do the 1D-case, with projections of the piercing points giving a subdivision point on every edge of the triangulation

Normal offsets in 2D

- Not a normal in the centers of the triangles of the object mesh
- A normal bisector for every edge in that mesh in a vertical plane though that edge.
- In that plane you do the 1D-case, with projections of the piercing points giving a subdivision point on every edge of the triangulation

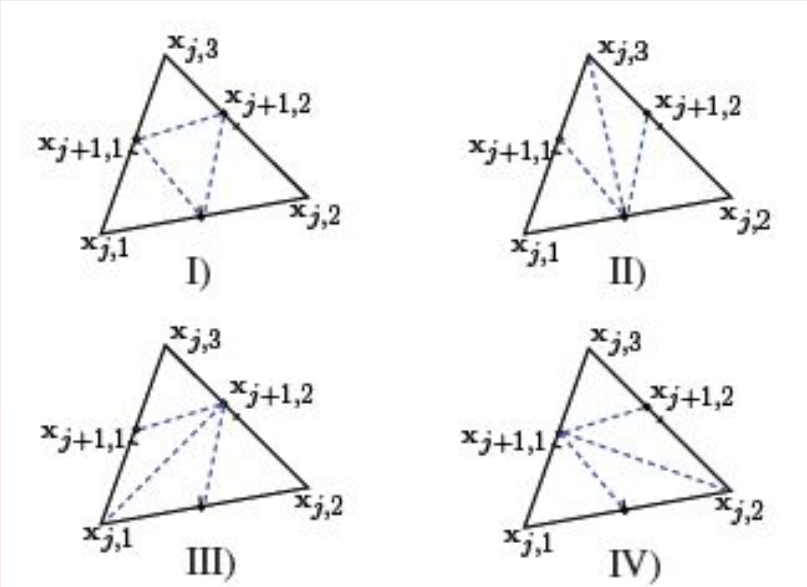
Possible splits

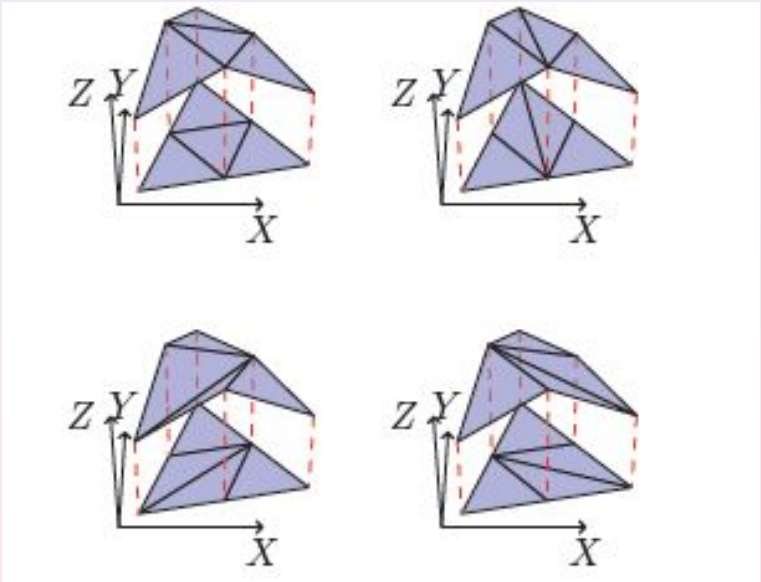




- Need to take the best possible split into 4 finer triangles.
- Take the one that gives the smallest polynomial approximation error

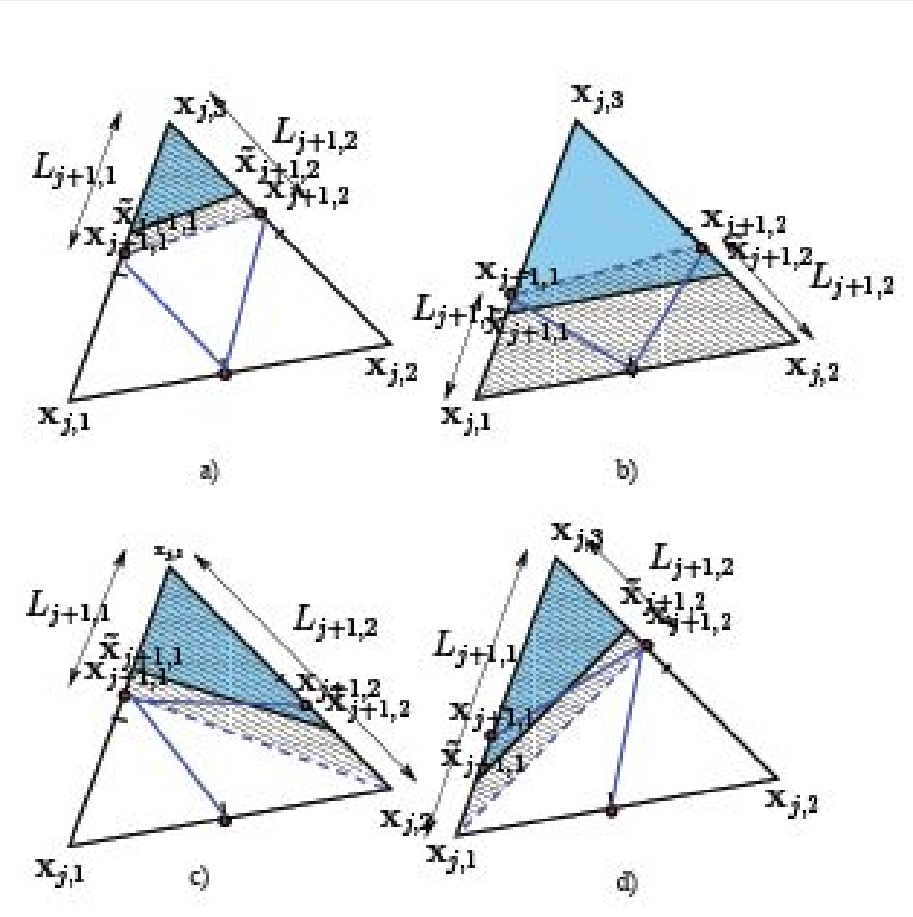
Possible splits





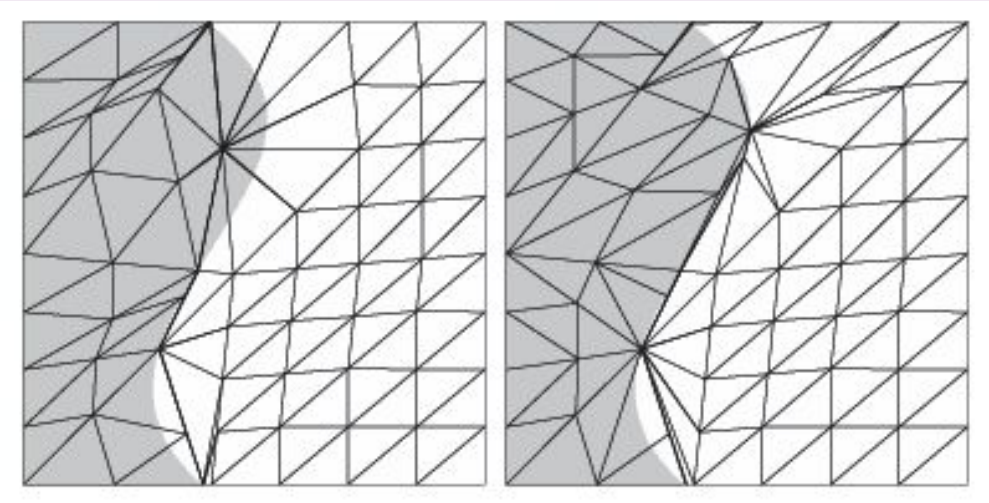
- Need to take the best possible split into 4 finer triangles.
- Take the one that gives the smallest polynomial approximation error

Possible splits



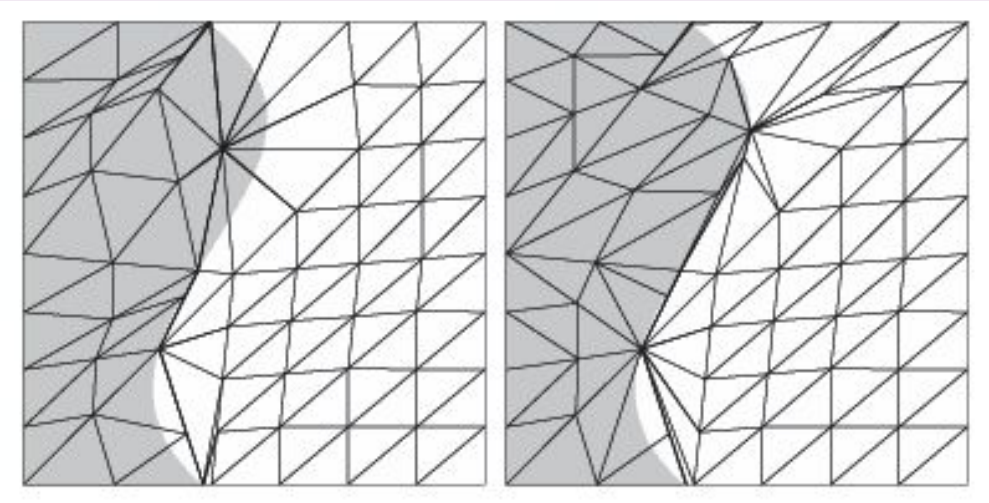
- class $\mathcal{H}$
- Rule of thumb: choose the one giving the least possible intersections with the discontinuity.
- Gray triangles need subdivision.

Possible splits



- Left: regular subdivision; Right: adaptive subdivision
- Right has less intersections with singularity contour

Possible splits



- Left: regular subdivision; Right: adaptive subdivision
- Right has less intersections with singularity contour

Update: polynomial wavelet

- Interpolating mesh gives poor approximation
- P_Δ= best polynomial approximation on triangle Δ
  If Δ is a subtriangle of Δ', then the geometric wavelet is $\psi_\Delta = P_\Delta - P_{\Delta'}$ restricted to Δ. [Dekel & Leviatan]
- We define $\psi_\Delta = P_\Delta - Q_\Delta$ where $Q_\Delta \in \Pi_2$ is an interpolating polynomial (comes for free).
  The wavelets are the detail info to be added to the object mesh.

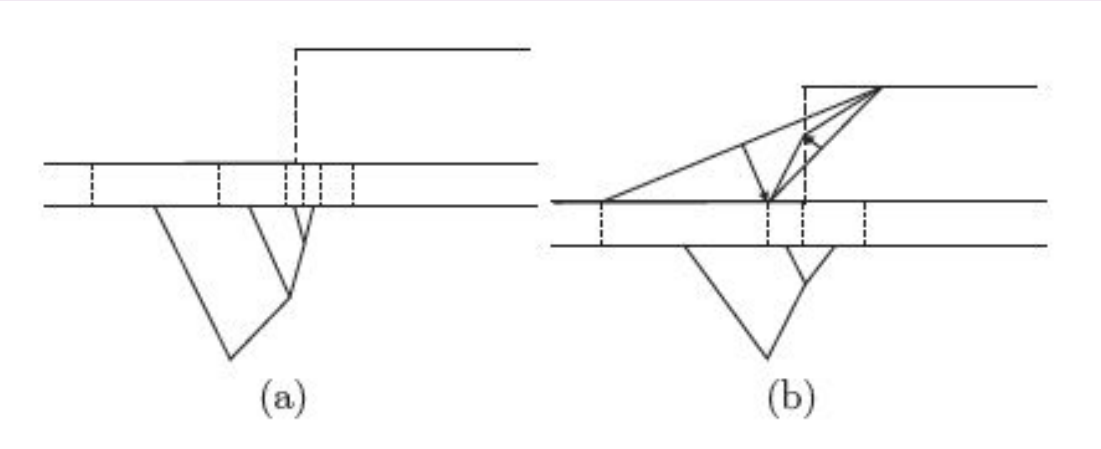
Update: polynomial wavelet

- Interpolating mesh gives poor approximation
- P_Δ= best polynomial approximation on triangle Δ
  If Δ is a subtriangle of Δ', then the geometric wavelet is $\psi_\Delta = P_\Delta - P_{\Delta'}$ restricted to Δ. [Dekel & Leviatan]
- We define $\psi_\Delta = P_\Delta - Q_\Delta$ where $Q_\Delta \in \Pi_2$ is an interpolating polynomial (comes for free).
  The wavelets are the detail info to be added to the object mesh.

Update: polynomial wavelet

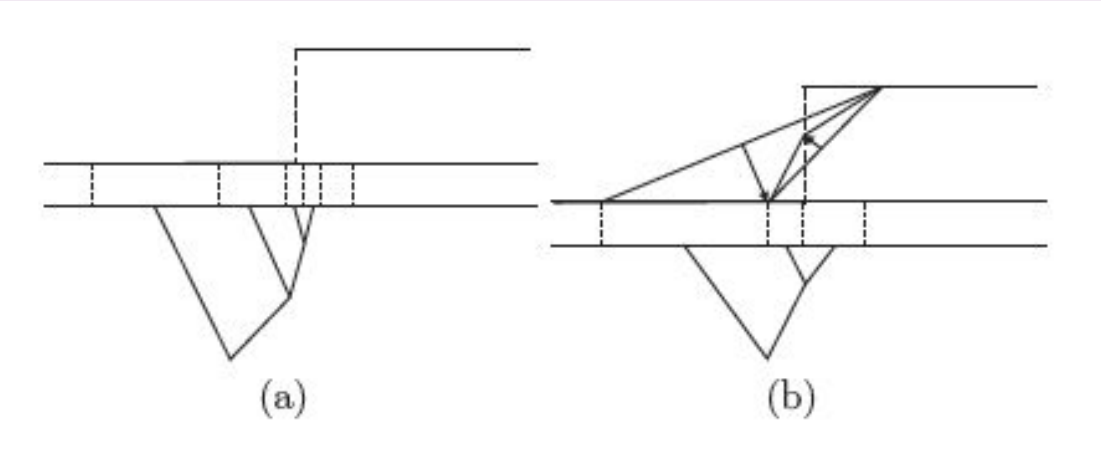
- Interpolating mesh gives poor approximation
- $P_\Delta =$ best polynomial approximation on triangle Δ
  If Δ is a subtriangle of Δ', then the geometric wavelet is $\psi_\Delta = P_\Delta - P_{\Delta'}$ restricted to Δ. [Dekel & Leviatan]
- We define $\psi_\Delta = P_\Delta - Q_\Delta$ where $Q_\Delta \in \Pi_2$ is an interpolating polynomial (comes for free).
  The wavelets are the detail info to be added to the object mesh.

Tree pruning



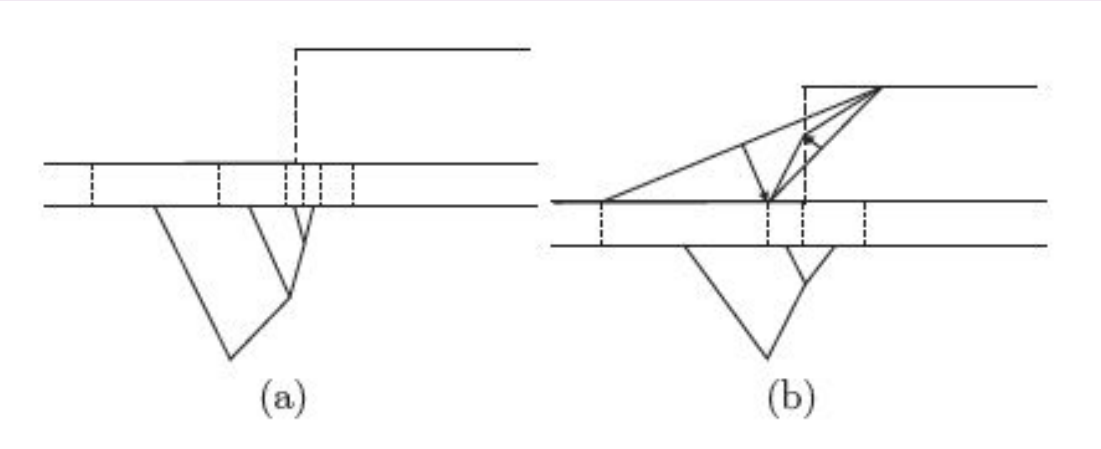
- Successive subtriangulations = graph
- Nodes = subdivision (geometric) data + detail info (wavelets)
- Pruning = cut away the less important branches

Tree pruning



- Successive subtriangulations = graph
- Nodes = subdivision (geometric) data + detail info (wavelets)
- Pruning = cut away the less important branches

Tree pruning



- Successive subtriangulations = graph
- Nodes = subdivision (geometric) data + detail info (wavelets)
- Pruning = cut away the less important branches

Tree pruning

$$\min \|f - \tilde{f}_n\|, \quad \tilde{f}_n \leftarrow \sum_{t \in \ell(S)} Q_{\Delta(t)} + \sum_{t \in \ell(S)} \psi_{\Delta(t)}$$

- S subtree
- t nodes of tree
- $\ell(S)$ leaves of tree S
- $\Delta(t)$ triangle at node t
- $Q_{\Delta(t)}$ (linear) interpolating polynomial on $\Delta(t)$
- $\psi_{\Delta(t)}$ polynomial (least squares) wavelet on $\Delta(t)$
- Note: there can be a wavelet at every node but generated from leave-wavelets

Optimal tree pruning (no $\psi_{\Delta(t)}$)

Define

- $R(S)$ monotonically increasing bit-rate functional on (sub)tree S
- $D(S)$ monotonically decreasing distortion functional on (sub)tree S

Prune S such that $\min_S D(S) + \lambda(R(S) - R_{\text{budget}})$,
(λ regularization parameter)

i.e., $\min_{\tilde{f}} D(\tilde{f})$ such that $R(\tilde{f}) \leq R_{\text{budget}}$[1]

- Optimal S for given λ by top-down pruning in linear time (in $N = \#\text{nodes}$).
- Iteration on λ to approach R_{budget}
- Complexity $O(N \log N)$

[1] Chou, Lookabaugh, Gray (1989)

Optimal tree pruning (no $\psi_{\Delta(t)}$)

Define

- $R(S)$ monotonically increasing bit-rate functional on (sub)tree S
- $D(S)$ monotonically decreasing distortion functional on (sub)tree S

Prune S such that $\min_S D(S) + \lambda(R(S) - R_{\text{budget}})$,
(λ regularization parameter)

i.e., $\min_{\tilde{f}} D(\tilde{f})$ such that $R(\tilde{f}) \leq R_{\text{budget}}$[1]

- Optimal S for given λ by top-down pruning in linear time (in $N = \#\text{nodes}$).
- Iteration on λ to approach R_{budget}
- Complexity $O(N \log N)$

[1] Chou, Lookabaugh, Gray (1989)

Optimal tree pruning (no $\psi_{\Delta(t)}$)

Define

- $R(S)$ monotonically increasing bit-rate functional on (sub)tree S
- $D(S)$ monotonically decreasing distortion functional on (sub)tree S

Prune S such that $\min_S D(S) + \lambda(R(S) - R_{\text{budget}})$,
(λ regularization parameter)

i.e., $\min_{\tilde{f}} D(\tilde{f})$ such that $R(\tilde{f}) \leq R_{\text{budget}}$[1]

- Optimal S for given λ by top-down pruning in linear time (in $N = \#\text{nodes}$).
- Iteration on λ to approach R_{budget}
- Complexity $O(N \log N)$

[1] Chou, Lookabaugh, Gray (1989)

Optimal tree pruning (with normal offsets, wavelets and quantization)

Recall a wavelet at every node, but only encoded in the leaves.

- Approximation error on $\Delta(t)$:
  $D_a(t) = \|f - (Q_{\Delta(t)} + \psi_{\Delta(t)})\|_2, \quad \forall t \in S$
- For subtree S_t rooted at t:
  $D_a(S_t) = \sum_{l \in \ell(S_t)} D_a(l) - D_a(t)$
- Quantization error on $\Delta(t)$: $D_q(t)$
- For subtree S_t rooted at t: $D_q(S_t) = \sum_{l \in \ell(S_t)} D_q(l) - D_q(t)$
- Refinement cost function $R_\theta(t)$ (normal offsets + tesselation)
  $R_\theta(S_t) = \sum_{t \in S \setminus \ell(S)} R_\theta(t)$
- Quantization cost: $R_q(S_t) = \sum_{l \in \ell(S_t)} R_q(l) - R_q(t)$
- Total cost for S_t: $R(S_t) = R_\theta(S_t) + R_q(S_t)$.

Optimal tree pruning (with normal offsets, wavelets and quantization)

Recall a wavelet at every node, but only encoded in the leaves.

- Approximation error on $\Delta(t)$:
  $D_a(t) = \|f - (Q_{\Delta(t)} + \psi_{\Delta(t)})\|_2, \quad \forall t \in S$
- For subtree S_t rooted at t:
  $D_a(S_t) = \sum_{l \in \ell(S_t)} D_a(l) - D_a(t)$
- Quantization error on $\Delta(t)$: $D_q(t)$
- For subtree S_t rooted at t: $D_q(S_t) = \sum_{l \in \ell(S_t)} D_q(l) - D_q(t)$
- Refinement cost function $R_\theta(t)$ (normal offsets + tesselation)
  $R_\theta(S_t) = \sum_{t \in S \setminus \ell(S)} R_\theta(t)$
- Quantization cost: $R_q(S_t) = \sum_{l \in \ell(S_t)} R_q(l) - R_q(t)$
- Total cost for S_t: $R(S_t) = R_\theta(S_t) + R_q(S_t)$.

Optimal tree pruning (with normal offsets, wavelets and quantization)

Recall a wavelet at every node, but only encoded in the leaves.

- Approximation error on $\Delta(t)$:
  $D_a(t) = \|f - (Q_{\Delta(t)} + \psi_{\Delta(t)})\|_2, \quad \forall t \in S$
- For subtree S_t rooted at t:
  $D_a(S_t) = \sum_{l \in \ell(S_t)} D_a(l) - D_a(t)$
- Quantization error on $\Delta(t)$: $D_q(t)$
- For subtree S_t rooted at t: $D_q(S_t) = \sum_{l \in \ell(S_t)} D_q(l) - D_q(t)$
- Refinement cost function $R_\theta(t)$ (normal offsets + tesselation)
  $R_\theta(S_t) = \sum_{t \in S \setminus \ell(S)} R_\theta(t)$
- Quantization cost: $R_q(S_t) = \sum_{l \in \ell(S_t)} R_q(l) - R_q(t)$
- Total cost for S_t: $R(S_t) = R_\theta(S_t) + R_q(S_t)$.

Optimal tree pruning (with normal offsets, wavelets and quantization)

Recall a wavelet at every node, but only encoded in the leaves.

- Approximation error on $\Delta(t)$:
  $D_a(t) = \|f - (Q_{\Delta(t)} + \psi_{\Delta(t)})\|_2, \quad \forall t \in S$
- For subtree S_t rooted at t:
  $D_a(S_t) = \sum_{l \in \ell(S_t)} D_a(l) - D_a(t)$
- Quantization error on $\Delta(t)$: $D_q(t)$
- For subtree S_t rooted at t: $D_q(S_t) = \sum_{l \in \ell(S_t)} D_q(l) - D_q(t)$
- Refinement cost function $R_\theta(t)$ (normal offsets + tesselation)
  $R_\theta(S_t) = \sum_{t \in S \setminus \ell(S)} R_\theta(t)$
- Quantization cost: $R_q(S_t) = \sum_{l \in \ell(S_t)} R_q(l) - R_q(t)$
- Total cost for S_t: $R(S_t) = R_\theta(S_t) + R_q(S_t)$.

Optimal tree pruning (with normal offsets, wavelets and quantization)

Recall a wavelet at every node, but only encoded in the leaves.

- Approximation error on $\Delta(t)$:
  $D_a(t) = \|f - (Q_{\Delta(t)} + \psi_{\Delta(t)})\|_2, \quad \forall t \in S$
- For subtree S_t rooted at t:
  $D_a(S_t) = \sum_{l \in \ell(S_t)} D_a(l) - D_a(t)$
- Quantization error on $\Delta(t)$: $D_q(t)$
- For subtree S_t rooted at t: $D_q(S_t) = \sum_{l \in \ell(S_t)} D_q(l) - D_q(t)$
- Refinement cost function $R_\theta(t)$ (normal offsets + tesselation)
  $R_\theta(S_t) = \sum_{t \in S \setminus \ell(S)} R_\theta(t)$
- Quantization cost: $R_q(S_t) = \sum_{l \in \ell(S_t)} R_q(l) - R_q(t)$
- Total cost for S_t: $R(S_t) = R_\theta(S_t) + R_q(S_t)$.

Optimal tree pruning (with normal offsets, wavelets and quantization)

Recall a wavelet at every node, but only encoded in the leaves.

- Approximation error on $\Delta(t)$:
  $D_a(t) = \|f - (Q_{\Delta(t)} + \psi_{\Delta(t)})\|_2, \quad \forall t \in S$
- For subtree S_t rooted at t:
  $D_a(S_t) = \sum_{l \in \ell(S_t)} D_a(l) - D_a(t)$
- Quantization error on $\Delta(t)$: $D_q(t)$
- For subtree S_t rooted at t: $D_q(S_t) = \sum_{l \in \ell(S_t)} D_q(l) - D_q(t)$
- Refinement cost function $R_\theta(t)$ (normal offsets + tesselation)
  $R_\theta(S_t) = \sum_{t \in S \setminus \ell(S)} R_\theta(t)$
- Quantization cost: $R_q(S_t) = \sum_{l \in \ell(S_t)} R_q(l) - R_q(t)$
- Total cost for S_t: $R(S_t) = R_\theta(S_t) + R_q(S_t)$.

Optimal tree pruning (with normal offsets, wavelets and quantization)

Recall a wavelet at every node, but only encoded in the leaves.

- Approximation error on $\Delta(t)$:
  $D_a(t) = \|f - (Q_{\Delta(t)} + \psi_{\Delta(t)})\|_2, \quad \forall t \in S$
- For subtree S_t rooted at t:
  $D_a(S_t) = \sum_{l \in \ell(S_t)} D_a(l) - D_a(t)$
- Quantization error on $\Delta(t)$: $D_q(t)$
- For subtree S_t rooted at t: $D_q(S_t) = \sum_{l \in \ell(S_t)} D_q(l) - D_q(t)$
- Refinement cost function $R_\theta(t)$ (normal offsets + tesselation)
  $R_\theta(S_t) = \sum_{t \in S \setminus \ell(S)} R_\theta(t)$
- Quantization cost: $R_q(S_t) = \sum_{l \in \ell(S_t)} R_q(l) - R_q(t)$
- Total cost for S_t: $R(S_t) = R_\theta(S_t) + R_q(S_t)$.

Optimal tree pruning

Now we have a bit-rate function $R(S)$ and a distortion function $D_a(S) + D_q(S)$

- Prune over nested subtrees $\cup_{t' \in \ell(S_t)} \Delta(t') = \Delta(t)$
- A one-bit decoration indicator whether or not to add a wavelet at node t
- functions no longer linear or monotone $\rightarrow$ optimal pruning algorithm needs adaptation
- Wavelets represented by Bernstein polynomials $\psi_\Delta(x, y) = \sum_{|i|=1} b_i B_i(z, y)$

Optimal tree pruning

Now we have a bit-rate function $R(S)$ and a distortion function $D_a(S) + D_q(S)$

- Prune over nested subtrees $\cup_{t' \in \ell(S_t)} \Delta(t') = \Delta(t)$
- A one-bit decoration indicator whether or not to add a wavelet at node t
- functions no longer linear or monotone $\rightarrow$ optimal pruning algorithm needs adaptation
- Wavelets represented by Bernstein polynomials $\psi_\Delta(x, y) = \sum_{|i|=1} b_i B_i(z, y)$

Optimal tree pruning

Now we have a bit-rate function $R(S)$ and a distortion function $D_a(S) + D_q(S)$

- Prune over nested subtrees $\cup_{t' \in \ell(S_t)} \Delta(t') = \Delta(t)$
- A one-bit decoration indicator whether or not to add a wavelet at node t
- functions no longer linear or monotone $\rightarrow$ optimal pruning algorithm needs adaptation
- Wavelets represented by Bernstein polynomials $\psi_\Delta(x, y) = \sum_{|i|=1} b_i B_i(z, y)$

Optimal tree pruning

Now we have a bit-rate function $R(S)$ and a distortion function $D_a(S) + D_q(S)$

- Prune over nested subtrees $\cup_{t' \in \ell(S_t)} \Delta(t') = \Delta(t)$
- A one-bit decoration indicator whether or not to add a wavelet at node t
- functions no longer linear or monotone $\rightarrow$ optimal pruning algorithm needs adaptation
- Wavelets represented by Bernstein polynomials $\psi_\Delta(x, y) = \sum_{|i|=1} b_i B_i(z, y)$

Optimal tree pruning

Now we have a bit-rate function $R(S)$ and a distortion function $D_a(S) + D_q(S)$

- Prune over nested subtrees $\cup_{t' \in \ell(S_t)} \Delta(t') = \Delta(t)$
- A one-bit decoration indicator whether or not to add a wavelet at node t
- functions no longer linear or monotone $\rightarrow$ optimal pruning algorithm needs adaptation
- Wavelets represented by Bernstein polynomials $\psi_\Delta(x, y) = \sum_{|i|=1} b_i B_i(z, y)$

Encoding

What needs encoding?

- tree structure of S
- the partition of each triangle (points on edges)
- the type of partitioning for each triangle
- coefficients of polynomials ψ_Δ for leaf nodes

Encoding

What needs encoding?

- tree structure of S
- the partition of each triangle (points on edges)
- the type of partitioning for each triangle
- coefficients of polynomials ψ_Δ for leaf nodes

Encoding

What needs encoding?

- tree structure of S
- the partition of each triangle (points on edges)
- the type of partitioning for each triangle
- coefficients of polynomials ψ_Δ for leaf nodes

Encoding

What needs encoding?

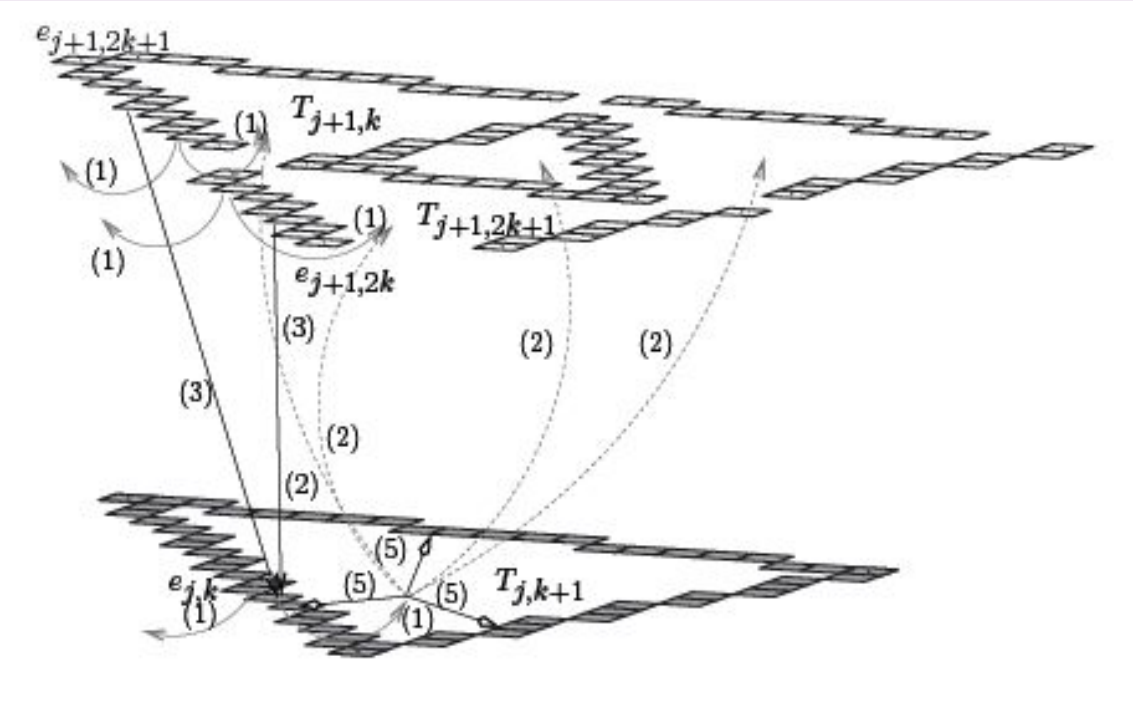
- tree structure of S
- the partition of each triangle (points on edges)
- the type of partitioning for each triangle
- coefficients of polynomials ψ_Δ for leaf nodes

Encoding

What needs encoding?

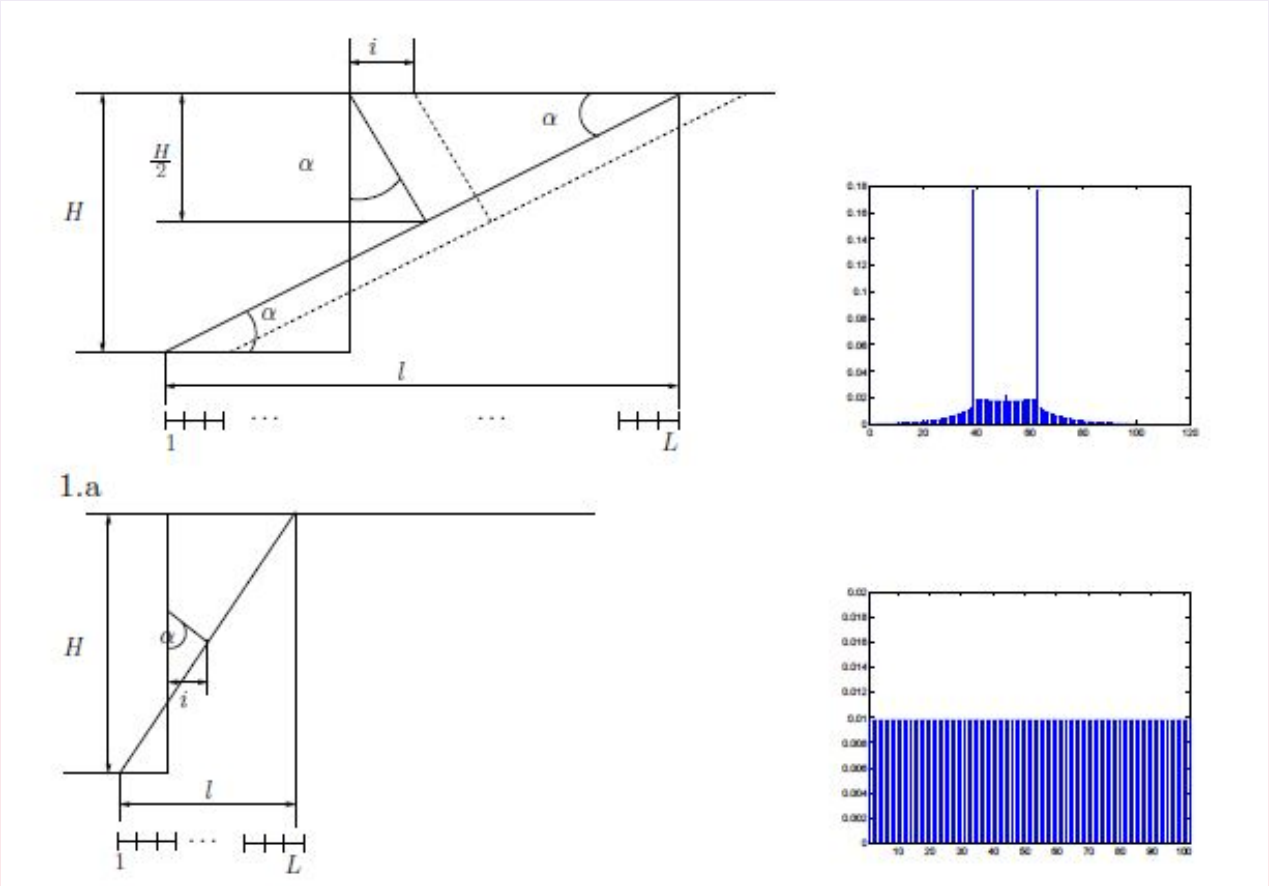
- tree structure of S
- the partition of each triangle (points on edges)
- the type of partitioning for each triangle
- coefficients of polynomials ψ_Δ for leaf nodes

Encoding



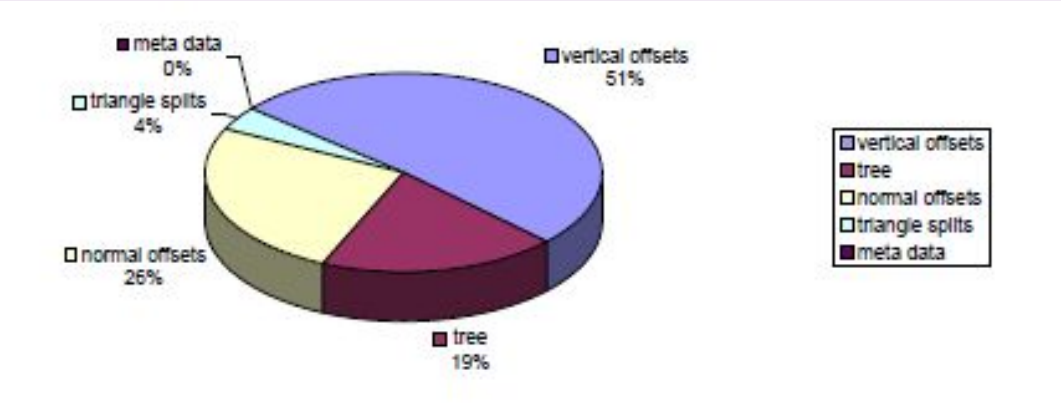
- Problems with digital images

Encoding



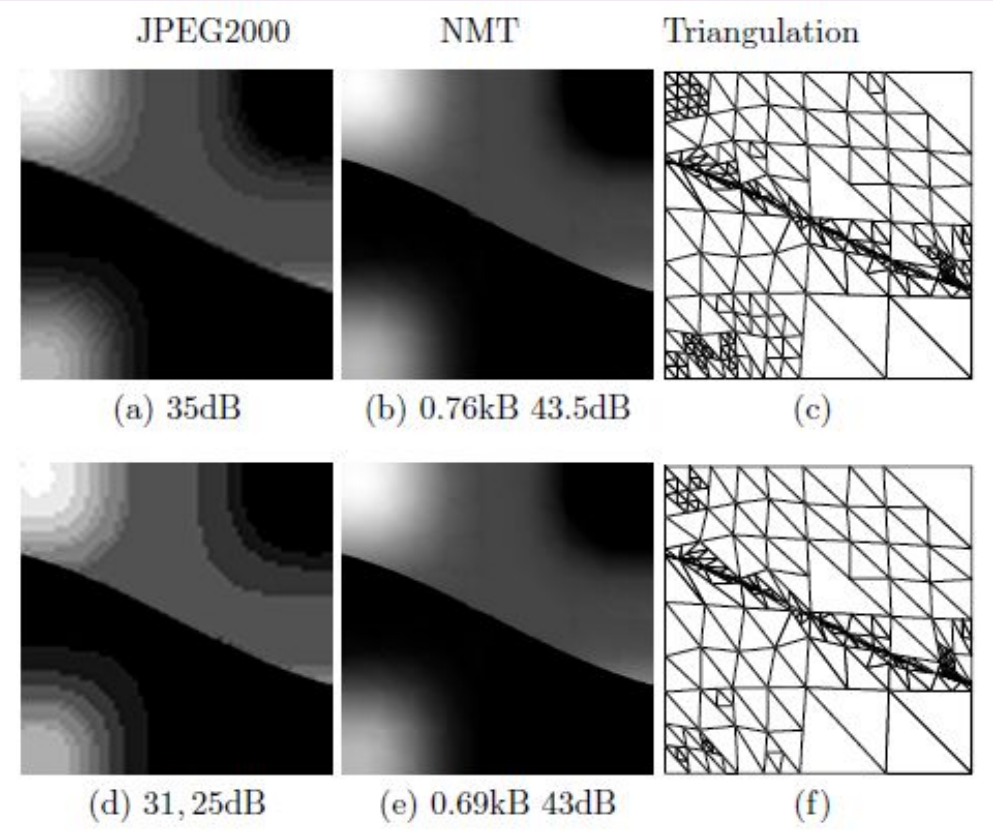
- Statistics of offsets for uniform distribution of singularity to assign bits and bins.

Encoding

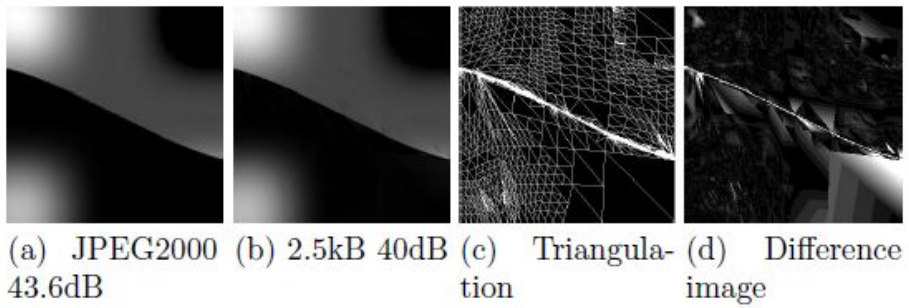


- 51% wavelets
- 26% normal offsets
- 19% tree
- 4% triangular splits
- meta data

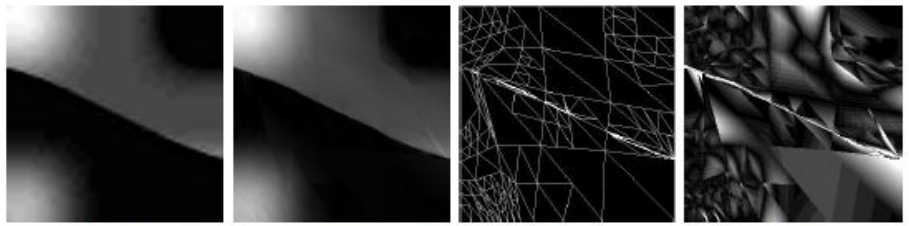
Example


JPEG2000
NMT
Triangulation
(a) 35dB
(b) 0.76kB 43.5dB
(c)
(d) 31, 25dB
(e) 0.69kB 43dB
(f)

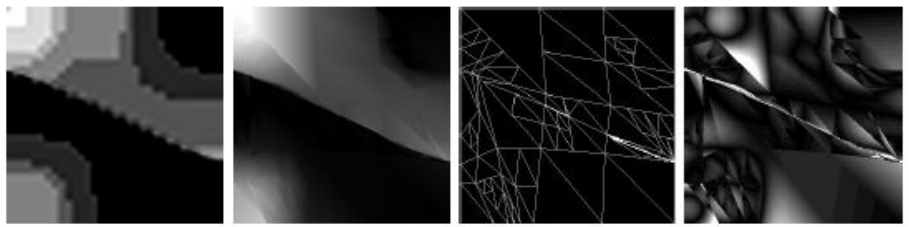
Example



(a) JPEG2000 43.6dB (b) 2.5kB 40dB (c) Triangulation (d) Difference image

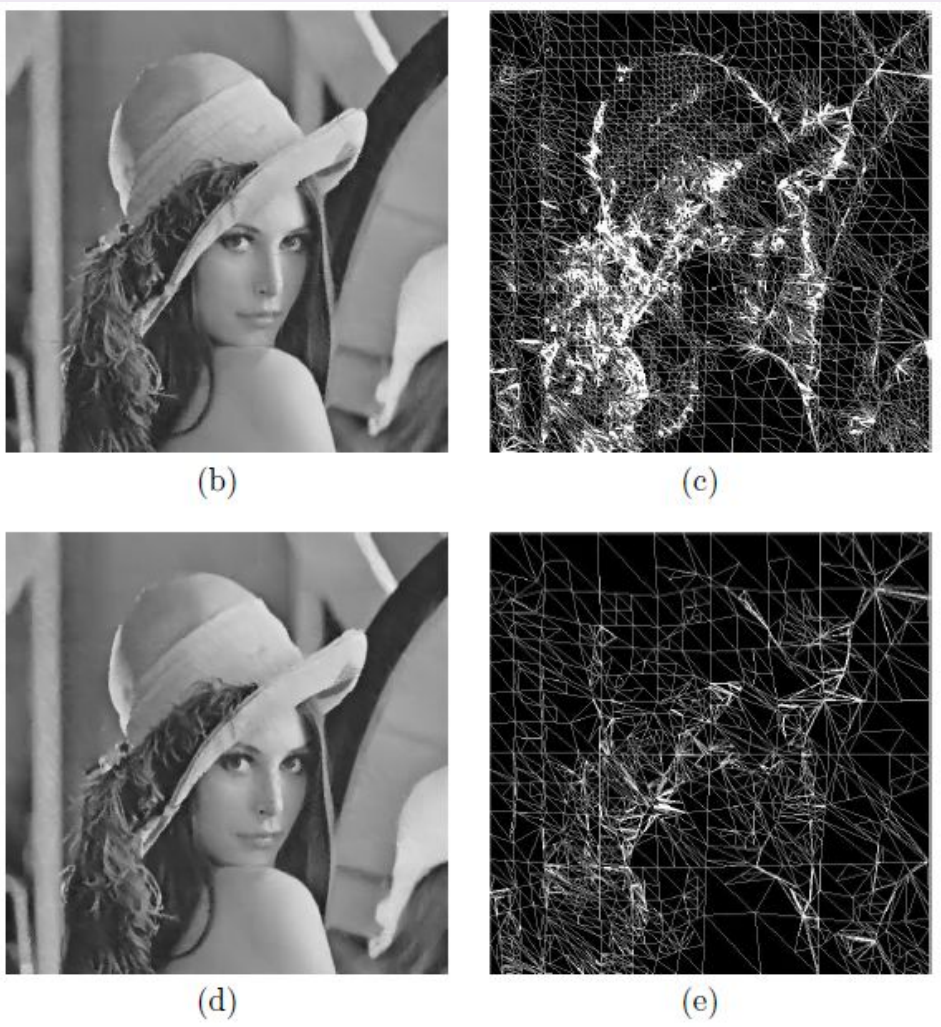


(e) JPEG2000 37dB (f) 0.6kB 36dB (g) Triangulation (h) Difference image



(i) JPEG2000 24dB (j) 0.34kB 33dB (k) Triangulation (l) Difference image

Pruning effect



(b) (c)

(d) (e)

Reference


W. Van Aerschot, M. Jansen, and A. Bultheel.
Normal mesh based geometrical image compression.
Image and Vision Computing, 27(4):459–468, 2009.


W. Van Aerschot, M. Jansen, E. Vanraes, and A. Bultheel.
Image compression using normal mesh techniques.
In A. Cohen, J.-L. Merrien, and L.L. Schumaker, editors, *Curves and Surface Fitting, Avignon 2006*, pages 266–275, Brentwood, 2007. Nashboro Press.


Ward Van Aerschot.
Multiscale Geometric Image Approximation.
PhD thesis, K.U.Leuven, September 2009.