Schaalbare karakteristiekbehoudende codering van onregelmatige meshes
met ondersteuning voor interactieve interessegebieden

Scalable Feature-Preserving Irregular Mesh Coding
with Interactive Region-of-Interest Support

Jonas El Sayeh Khalil

Promotoren: prof. dr. P. Lambert, prof. dr. ir. A. Munteanu
Proefschrift ingediend tot het behalen van de graad van
Doctor in de ingenieurswetenschappen: computerwetenschappen

UNIVERSITEIT
GENT

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. K. De Bosschere
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2017 - 2018

ISBN 978-94-6355-115-1
NUR 965
Wettelijk depot: D/2018/10.500/33

# Leden van de examencommissie

Promotoren

*prof. dr. Peter Lambert*
Vakgroep Elektronica en Informatiesystemen
Universiteit Gent

*prof. dr. ir. Adrian Munteanu*
Vakgroep Elektronica and Informatica
Vrije Universiteit Brussel

Stemgerechtigde leden

*prof. dr. ir. Gert De Cooman (voorzitter)*
Vakgroep Elektronica en Informatiesystemen
Universiteit Gent

*dr. ir. Glenn Van Wallendael (secretaris)*
Vakgroep Elektronica en Informatiesystemen
Universiteit Gent

*prof. dr. Wesley De Neve*
Vakgroep Elektronica en Informatiesystemen
Universiteit Gent

*prof. dr. ir. Aleksandra Pizurica*
Vakgroep Telecommunicatie en Informatieverwerking
Universiteit Gent

*dr. Bart Pieters*
Graphine Software, Gent

*dr. ir. Sébastien Valette*
Institut National des Sciences Appliquées, Frankrijk

Interne verdediging: donderdag 19 april 2018
Publieke verdediging: woensdag 16 mei 2018

*Voor mijn moeder, Marleen Rousseeuw*
*1958 – 2011*

*Voor mijn vader, Abdel Gawad El Sayeh Khalil*
*1945 – 2017*

# Dankwoord

Hoe een leven kan veranderen in zes jaar tijd. Ik begon mijn onderzoek in de Zuiderpoort aan het toenmalige Multimedia Lab dat deel uitmaakte van het onderzoekscentrum IBBT (toen net hernoemd tot iMinds). Ondertussen begin ik te schrijven aan mijn dankwoord, hier in het Technologiepark (na een paar keer verhuizen), in het lab dat ondertussen IDLab heet, binnen het onderzoekscentrum imec. Na die zes jaar had ik toch graag enkele mensen bedankt.

Eerst en vooral wil ik prof. dr. ir. Rik Van de Walle bedanken. Na een stage en een masterproef kreeg ik binnen zijn lab de kans om verder onderzoek te doen. Een voorbeeld van ambitie om "u" tegen te zeggen, maar toch volstond "Rik" steeds als aanspreking. Verder wil ik ook mijn beide promotoren prof. dr. Peter Lambert en prof. dr. ir. Adrian Munteanu bedanken. De talrijke vergaderingen waren steeds diepgaand, uitputtend maar toch zo verrijkend. Zonder hun inzichten was dit werk nooit tot zijn huidige vorm gekomen.

Ook wil ik de leden van mijn examencommissie bedanken. De positieve evaluatie van dit werk had een zeer geruststellende werking. De beperkte, maar steeds terechte opmerkingen en suggesties hebben echter steeds het perfectionistisch kantje in mij geprikkeld; het kantje dat mij gedurende de voorbije dagen weinig rust gunde, zorgde ervoor dat dit proefschrift dankzij hun opmerkingen nog een stukje beter werd.

Verder wil ik ook mijn dichtste collega's binnen het officieuze *graphics lab* bedanken. Nadat ik van Bart, Charles en Aljosha veel heb geleerd tijdens mijn vroege dagen binnen het Multimedia Lab, bleven ook hun resultaten binnen *Graphine* inspirerend werken. Ondanks hun relatief korte verblijf bij ons, ben ik verder ook El-Hassan, Steven, Gaétan en Jelle dankbaar voor de steeds interessante samenwerking. Tenslotte heb ik met Ignace het langste een bureau gedeeld, en ik wil hem ook zeker bedanken voor de vele constructieve discussies die hielpen om mijn eigen inzichten te versterken, of om tot verrassende nieuwe inzichten te komen. Ook van mijn andere collega's heb ik veel bijgeleerd, zowel uit domeinen die nauw aansluiten bij dit werk als uit domeinen die een stuk verder weg liggen. Glenn, Sebastiaan en Jan in het bijzonder hebben mij als video-experten in mijn vroegere onderzoeksjaren goed op weg gezet. Bedankt!

*Mens sana in corpore sano*, een gezonde geest in een gezond lichaam, maar bij uitbreiding ook in een gezonde omgeving. Er zijn dus heel wat mensen die,

zonder het zelf te beseffen, zeker hebben bijgedragen tot de kwaliteit van dit werk en tot het behouden van mijn geestesgezondheid.

Ik kan dan al zeker alle ploeggenoten bij BBC Desselgem doorheen de jaren bedanken. Zo hield ik er een relatief gezond lichaam op na, dat soms wel eens tot uitersten gedreven werd (bedankt Bert). Niels, Kristof, Jens en Sander wil ik ook bedanken. Ondanks universitaire studies en ondanks dit onderzoek zijn we vrienden gebleven sinds het middelbaar, een vriendschap die mij zeer dierbaar is.

Mijn familie, waar ik steeds op kan rekenen, wil ik ook zeker bedanken. Nonkels, tantes, neven, nichten staan steeds voor mij klaar. Ondertussen kan ik daar ook een leuke schoonfamilie bij rekenen. Inderdaad, de belangrijkste personen in een mensenleven kan ik hier helaas niet meer vernoemen. Ma, pa, ook al kan ik jullie niet meer persoonlijk bedanken, ik besef maar al te goed dat ik hier niet stond zonder jullie. Jullie opvoeding heeft mij gevormd tot wie ik ben. Geduld en doorzettingsvermogen zijn maar enkele van de karaktertrekken die jullie mij meegegeven hebben en die doorheen de jaren zo waardevol gebleken zijn. Ma, je fierheid liet je zo vaak blijken, en je wist waarschijnlijk al veel eerder dan ikzelf dat ik wel mijn universitair diploma zou halen. Jammer dat je het nooit meer hebt mogen meemaken. Pa, ook jij verliet ons te vroeg. Zo trots als je was op de proclamatie van zes jaar geleden, die trots kan ik mij nu enkel maar inbeelden.

Karim en David, met wie ik hetzelfde El Sayeh-bloed deel, bedankt voor de bijna wekelijkse afleidingen, basketbalgesprekken, filosofische discussies, filmrecensies, toekomstdromen, en het steeds met raad en daad bijstaan. Hun blinde geloof in mijn kunnen was steeds een motivator om beter te doen.

En tenslotte wil ik Jolien bedanken voor ondertussen al meer dan drie jaar aan mooie momenten, en de zovele die we nog samen tegemoet gaan. Zij kent mij op mijn best, maar ondertussen zeker ook op mijn slechtst. Ze is dan ook als enige getuige geweest van welke uren ik soms wel eens durfde kloppen, hoeveel avonden ik toch nog bezig bleef en hoe ik soms iets teveel vrije tijd opofferde om stukken net dat tikkeltje beter te krijgen. Misschien had ze op sommige momenten meer aan onze poes *Pluuch* dan aan mij, maar omgekeerd heb ik altijd veel aan haar gehad! Bedankt om er steeds voor mij te zijn!

*Gent, mei 2018*
*Jonas El Sayeh Khalil*

# Table of Contents

# Glossary

**B**

**background** The region of a mesh which is not prioritized *(see also: Region-of-Interest)*.

**Bjøntegaard delta rate** Measure of the difference in rate, over a specified quality or distortion interval, between two PSNR-curves.

**C**

**continuous Levels-of-Detail** A chain of levels of detail can be considered *continuous* if the difference between two subsequent LODs is a single vertex. In general, while this is per definition still discrte, one can consider such a chain as continuous if hundreds or thousands of levels are present, and the difference between two levels is nearly unnoticeable *(see also: Level-of-Detail)*.

**D**

**degree** The (vertex) degree: see valence.

**discrete Levels-of-Detail** A chain of levels of detail can be considered *discrete* if the difference between two subsequent LODs are multiple vertices. In general, one can consider such a chain as discrete if the amount of levels is limited, and the difference between two levels more clearly noticeable over the entire mesh *(see also: Level-of-Detail)*..

**F**

**face** The polygons that make up a polyhedron are called faces.

**P**

**pixel** Picture element. This represents a sample in a two-dimensional data set.

**Q**

**quantization** In digitizing a continuous signal, quantization is the process of limiting the signal values to a finite set of values. This does not suffice for a digital representation as a continuous signal consists of an infinite amount of signal values *(see also: sampling)*.

**R**

**Region-of-Interest** The region of a mesh which is prioritized *(see also: background)*.

**resolution** An indication of the order of magnitude of the smallest details that can be distinguished from one another. After sampling, this typically relates to the sampling density: a denser sampling results in a higher resolution as smaller details can be distinguished.

**S**

**sampling** When digitizing a continuous signal, sampling is the process of determining a finite amount of signal values. This does not suffice for a digital representation as each of the resulting values can still have any continuous value *(see also: quantization)*.

**V**

**valence** The number of neighbours a vertex has.

**vertex** Corner point of a polygon.

# List of Acronyms

**2D** two-dimensional
**3D** three-dimensional

**BD-rate** Bjøntegaard delta rate
**BG** background
**bpv** bits per vertex

**CABAC** context-adaptive binary arithmetic coding
**CAD** computer-aided design
**cLOD** continuous LOD

**dLOD** discrete LOD

**GPU** graphics processing unit

**LOD** level of detail

**MAD** mean absolute deviation

**PM** progressive mesh
**PSNR** peak signal-to-noise ratio

**RD** rate-distortion
**RDO** RD optimization
**RMS** root mean squared
**ROI** Region-of-Interest

**SAQ** successive approximation quantization
**SFWInCS** scalable feature-preserving wavelet-based irregular mesh coding system

**TD** triangle-distortion

# List of Symbols

**Symbols**

$\Delta^{\mathbf{r}}_{\mathbf{avg}}$ average rate difference

$\Delta^{\mathbf{t}}_{\mathbf{avg}}$ average triangle percentage difference

$\epsilon^p(v_j^{o_i})$ the vertex $v_j^{o_i}$, refined up to bit plane $p$

$\epsilon^p(w_{g,j-1}^i)$ the wavelet coefficient $w_{g,j-1}^i$ decoded up to bit plane $p$

$\gamma(v^e)$ the bijective operator which maps even vertices $v^e \in M_j^o$ to vertices $v \in M_{j-1}$

$\lambda_C$ connectivity leaf cell size

$\lambda_G$ geometry leaf cell size

$\mu(v)$ the operator which maps vertices $v \in M_j$ to template vertices $v^T \in M_j^T$

$\nu(v)$ valence/degree of vertex $v$

$\omega(v^o)$ the bijective operator which maps odd vertices $v^o \in M_j^o$ to wavelet coefficients $v \in W_{j-1}$

$\rho$ the percentage of decoded vertices when only the ROI is decoded

$\sigma(ROI_j^S)$ the propagation of an ROI in resolution $j$ to the lower resolution $j-1$

$\sigma^p(w)$ the significance of a wavelet coefficient $w$ with respect to the quantization threshold $\tau^p$

$\tau_{\mathbf{FP}}$ feature prediction threshold

$\tau_{\mathbf{MF}}$ multiple-feature threshold

**D**

$D_j^{(p)}$ the distortion of the odd vertices of resolution $j$ when refined up to bitplane $p$

$D_{(j,t)}^{(p)}$ the distortion of the odd vertices of tile $t$ of resolution $j$ when refined up to bitplane $p$

**M**

$M$ an original mesh

$M_j^e$ the even vertices of mesh $M$ at resolution $j$

$M_{enc}$ the binary storage of a mesh $M$

$M_{\alpha \leq j}$ *a* partial approximation of $M$, with the resolution varying across the mesh surface without surpassing $j$

$M_j^o$ the odd vertices of mesh $M$ at resolution $j$

$\widetilde{M}_j^o$ the set of *predicted* odd vertices of mesh $M$ at resolution $j$, i.e., the vertices *before* refinement

$\widetilde{M}_{j+1}$ the mesh obtained after upsampling $M_j$ but before refining the odd vertices

$M_j^T$ the template mesh corresponding to $M_j$

$M^{\mathbf{TF}}$ a transformed representation of $M$

**N**

$n_t$ number of triangles

$n_v$ number of vertices

**Q**

$Q$ the number of quantization bits, i.e., the number of bit planes after quantization

**R**

$R$ the number of resolutions, i.e., the resolutions are denoted by $M_0, M_1, \ldots, M_{R-1} = M$

$R_j^{(p)}$ the rate required to refine the odd vertices of resolution $j$ up to bitplane $p$

$R_{(j,t)}^{(p)}$ the rate required to refine the odd vertices of tile $t$ of resolution $j$ up to bitplane $p$

$R_j^{\mathbf{conn}}$ the rate required for upsampling resolution $j$ (i.e., decoding the connectivity layer without refining the geometry)

$R_{(j,t)}^{\mathbf{conn}}$ the rate required for upsampling tile $t$ of resolution $j$ (i.e., decoding the connectivity layer without refining the geometry)

$R_j^{\mathbf{geom}}$ the rate required for refining the odd vertices of resolution $j$

$R^{\mathbf{geom}}_{(j,t)}$ the rate required for refining the odd vertices of tile $t$ at resolution $j$

$ROI^S_j$ the ROI in $M_j$, i.e. in the *spatial* domain

$ROI^S_{k|j}$ the minimal ROI of $M_k$ in order to ensure that $ROI^S_j$, with $j > k$ can be properly defined

$ROI^W_j$ the ROI in $W_j$ of the *wavelet* domain

**T**

$\widetilde{T}^{\mathbf{x}}_j$ the $\mathbf{x}^{\text{th}}$ tile of $M_j$ after *partitioning*, not necessarily decodable

$\widehat{T}^{\mathbf{x}}_k$ the tile at resolution $k > j$ which encompasses samples affected by $\widetilde{T}^{\mathbf{x}}_j$

$\check{T}^{\mathbf{x}}_k$ the tile at resolution $k > j$ which encompasses samples affecting $\widehat{T}^{\mathbf{x}}_{r_{\max}}$

$T^{\mathbf{x}}_j$ the $\mathbf{x}^{\text{th}}$ (final) tile of $M_j$

$T(.)$ the operator which maps vertices to tiles

$\mathcal{T}^{\mathbf{req}}_j$ the set of tiles required for decoding $ROI^W_j$

**W**

$W^B_j$ the wavelet subband $W_j$ after boosting the coefficients in $ROI^W_j$

$WT^{-1}$ the inverse wavelet transform, i.e., $WT^{-1}(M_j, W_j)$ transforms lower-resolution mesh $M_j$ with wavelet subband $W_j$ to $M_{j+1}$

# List of Figures

# List of Tables

# Nederlandstalige samenvatting –Dutch Summary–

Binnen de digitale media groeit het aanbod aan digitaal 3D materiaal aan een enorme snelheid. Waar we enkele decennia geleden bij de term "3D" voornamelijk aan videogames en gespecialiseerde CAD-software dachten, is 3D vandaag alomtegenwoordig, mede dankzij de ontwikkeling van 3D-printers en de opkomst van *augmented* en *virtual reality*. Augmented reality *augmenteert* of verrijkt de reële wereld met virtuele objecten die met de nieuwste *smart glasses* gezien kunnen worden, terwijl virtual reality elke toepassing omvat waarin een virtuele wereld geschapen wordt. Denk bijvoorbeeld aan trainingssimulatoren, het digitaal bewaren van cultureel erfgoed, of het ontwerp van jouw toekomstig droomhuis door een architect. Naast de groeiende kwantiteit van 3D materiaal wordt ook de kwaliteit van elk *3D-model* van een object steeds hoger, enerzijds door nauwkeurigere scanapparatuur, anderzijds door krachtigere modelleersoftware die almaar geavanceerdere technieken toelaat om modellen tot in de fijnste details te bewerken.

Helaas volstaat de groei in rekenkracht en geheugen in computers niet om interactief met dergelijke modellen te werken. Daarenboven worden onze toestellen zelf steeds gevarieerder: we moeten vandaag rekening houden met allerhande computers, gaande van desktop PCs en laptops tot tablets en smartphones die elk verschillend presteren. Verder moeten we ook rekening houden met verschillende opslag- en transmissiemogelijkheden, van lokale opslag en externe media tot cloudopslag die toegankelijk is via Wi-Fi of mobiele netwerken. Enkel modellen opslaan die volstaan in *elk* mogelijk scenario leidt tot ondermaatse kwaliteit; immers, op TV wil ik ook betere kwaliteit dan wat over een mobiel netwerk naar mijn smartphone gestreamd wordt. Een model voorzien per mogelijk denkbare configuratie, daarentegen, is onhoudbaar. Er is nood aan schaalbare representaties.

De problemen zelf worden in Hoofdstuk 2 diepgaander behandeld. Om een 3D-model digitaal te representeren wordt vaak gebruik gemaakt van een zogenaamde *polygonmesh*, een datastructuur bestaande uit *vertices* die 3D-punten in de ruimte voorstellen, en polygonen die de vertices met elkaar verbinden. Zijn alle polygonen driehoeken, dan wordt er gesproken over *driehoeksmeshes*. Deze laatste worden vaak gebruikt om virtuele scenes interactief te visualiseren; dit proefschrift hanteert eveneens driehoeksmeshes, of kortweg *meshes*. Deze meshes ontstaan na twee belangrijke processen die te vinden zijn in elke analoog-naar-

digitaalconversie: *samplen* (of bemonsteren) en *kwantisatie*. Het samplen van een object levert een discreet aantal samples of punten op het oppervlak van het object op; hoe meer samples, hoe meer we weten over ons model maar hoe meer opslag we ook nodig hebben. Elk van deze samples wordt een vertex van de mesh, en elk van deze vertices wordt na kwantisatie voorgesteld door een benadering met een eindige nauwkeurigheid. In tegenstelling tot traditionelere multimediadomeinen moeten de relaties tussen verschillende samples expliciet gekend zijn; deze connectiviteit wordt bepaald door de driehoeken die gevormd worden tussen de verschillende vertices. Samen vormen deze driehoeken een benadering van het oppervlak van het originele object. In Hoofdstuk 2 wordt dieper ingegaan op het samplen en kwantiseren van modellen. Hierbij wordt ook prijs-kwaliteit in rekening gebracht: hoeveel bits kan ik spenderen en wat is dan de kwaliteit van mijn 3D-model?

We zullen zien dat deze representatie niet schaalt naar grotere 3D-modellen. We hebben nood aan compressie en onderzoeken in dit proefschrift *verliesloze* compressie, wat toelaat om het originele digitale model feilloos te reconstrueren. Technieken om dit te bekomen worden ook behandeld in Hoofdstuk 2, waarbij vormen van schaalbaarheid onder de loep genomen worden. Schaalbaarheid moet toelaten dat één enkele representatie volstaat om het gevisualiseerde model te laten schalen naargelang de vereisten van applicaties en de beperkingen van systemen en netwerken. Hier kunnen drie belangrijke vormen van schaalbaarheid bekeken worden: (1) resolutieschaalbaarheid, (2) kwaliteitsschaalbaarheid, en (3) spatiale schaalbaarheid met interessegebieden (*regions of interest, ROIs*). Het schalen van de resolutie bepaalt hoeveel vertices gebruikt worden om een model te representeren, het schalen van de kwaliteit bepaalt de nauwkeurigheid waarmee vertices gepositioneerd worden, en het encoderen en decoderen van interessegebieden laat toe om specifieke gebieden uit een grotere mesh te selecteren en enkel deze gebieden te verwerken. Hoofdstukken 3 en 4 behandelen de ontworpen representatie en encodering die aan al deze schaalbaarheidsvormen voldoet.

Het basisontwerp van de voorgestelde representatie en encodering wordt besproken in het eerste deel van Hoofdstuk 3, wat leidt tot een resolutieschaalbaar systeem. Hierin wordt de wavelet-transformatie beschouwd, die ervoor zorgt dat een model op een efficiënte manier wordt ontbonden in enerzijds een model met een lagere resolutie en anderzijds informatie over de details die ervoor zorgen dat het originele hoge-resolutiemodel kan gereconstrueerd worden. Deze details worden de waveletcoëfficiënten genoemd, in een zogenaamde *waveletsubband* per resolutie. Een belangrijke keuze is dat er gewerkt wordt met *onregelmatige* meshes: hierbij volgt de connectiviteit van de vertices geen vaste structuur. Dit zorgt enerzijds voor een duurdere compressie, maar anderzijds zorgt dit ervoor dat modellen kwaliteitsvol gerepresenteerd kunnen worden met minder vertices en driehoeken. De voorgestelde wavelet-transformatie is zodanig ontworpen dat belangrijke geometrische kenmerken, of dus, “scherpe randen”, behouden worden in lagere resoluties, zonder expliciet aan te geven vanaf wanneer een rand als belangrijk of scherp moet worden gezien. Dit zorgt er opnieuw voor

dat modellen kwaliteitsvol gerepresenteerd blijven met minder vertices, en dus bij lagere resoluties. De resulterende waveletcoëfficiënten kunnen compact voorgesteld worden in zogenaamde *octrees*. Dit zijn abstracte datastructuren die elke ruimte per stap onderverdelen in acht kleinere ruimtes door eenmaal te splitsen per dimensie. Zodoende worden waveletcoëfficiënten die dichter bij elkaar liggen en sterker gecorreleerd zijn met elkaar, samen geëncodeerd om een betere compressie te bekomen. Het gebruik van onregelmatige meshes levert resultaten op die competitief zijn met geavanceerde technieken in de literatuur en waarbij modellen met veel geometrische kenmerken aan een beduidend hogere kwaliteit weergegeven worden bij lagere resoluties. Bovenop klassieke *rate-distortie*-evaluaties waar de distortie onderzocht wordt in functie van de gespendeerde *bitrate* (d.i., het aantal bits per vertex), wordt een nieuwe evaluatie voorgesteld waarbij de distortie uitgezet wordt in functie van het percentage van gereconstrueerde vertices. Dit geeft een indicatie van het geheugengebruik van schaalbare representaties in interactieve toepassingen, *na* decodering, waarbij de voorgestelde representatie beduidend betere kwaliteit levert met minder geheugen. Bijgevolg bevestigen de resultaten dat enerzijds een betere kwaliteit bekomen wordt met minder vertices en dus geheugenvereisten voor interactieve toepassingen reduceert, terwijl anderzijds de hogere prijs per vertex gecompenseerd wordt door het lagere aantal vereiste vertices.

In het tweede deel van Hoofdstuk 3 wordt kwaliteitsschaalbaarheid dieper onderzocht. Een fundamenteel probleem om de kwaliteit van de waveletcoëfficiënten dynamisch te laten schalen, is het gesynchroniseerd houden van de encoder en decoder: de decoder moet met dezelfde informatie kunnen decoderen zoals waarmee de encoder zijn data encodeert. Er ontstaan bijgevolg afwijkingen als een decoder een nieuwe resolutie aanvat vooraleer de waveletcoëfficiënten van de voorgaande subband aan een maximale kwaliteit gereconstrueerd zijn. Om desalniettemin kwaliteitsschaalbaarheid toe te laten, wordt het gebruik van een sjabloonmesh of *templatemesh* onderzocht. In plaats van de eigenlijke mesh te gebruiken in het encodeerproces, wordt deze templatemesh gebruikt die zowel aan de encoderkant als aan de decoderkant bekomen kan worden zonder waarden van de waveletcoëfficiënten te gebruiken. Het gebruik van deze templatemesh laat toe om datablokken in een vrijwel arbitraire volgorde op te slaan of te versturen. De interessantste toepassing die hierdoor mogelijk wordt, is rate-distortie-optimalisatie (RD-optimalisatie): op elk moment in het encodeerproces kan de encoder analyseren in hoeverre het opslaan van een nieuwe resolutie of juist het verbeteren van een bestaande subband de globale kwaliteit van een model het sterkst laat toenemen met een minimaal aantal bits. De resultaten tonen duidelijk aan dat het gebruik van een templatemesh, die kwaliteitsschaalbaarheid en bijgevolg RD-optimalisatie toelaat, geen significante meerkost introduceert. Deze optimalisatie heeft ook een duidelijk waarneembaar effect in lage en middelmatige bitrates; met andere woorden, voor elke rate kan de beste kwaliteit verkregen worden, wat vooral bij lage resoluties een significant verschil oplevert.

Resolutie- en kwaliteitsschaalbaarheid schalen een model globaal. Dit volstaat

zolang een model in zijn geheel gevisualiseerd wordt. Hedendaagse meshes zijn echter zo gedetailleerd dat de fijnste details te klein zijn om waar te nemen wanneer deze modellen volledig gevisualiseerd worden. Pas wanneer we dichter bij het model gaan kijken komen hogere resoluties tot hun recht, maar dan wordt een hoge rate en veel rekentijd gespendeerd aan delen van de mesh die niet op het scherm getoond worden. Hoofdstuk 4 onderzoekt *Region-of-Interest-* of ROI-codering: hoe kan het concept van interessegebieden in rekening gebracht worden om de resolutie en kwaliteit van een mesh spatiaal adaptief te laten variëren? In het eerste deel van dit hoofdstuk wordt dit probleem aangekaart vanuit het standpunt van de encoder. Onafhankelijk van welk deel de decoder zal opvragen, kan de encoder specifieke gebieden prioriteren. Zo kunnen details in het gezicht van een menselijk virtueel karakter voorrang krijgen op details in zijn kleren, of kan de voorkant van virtueel monument voorrang krijgen op de achterkant. De aanpak die voorgesteld wordt in dit hoofdstuk, maakt gebruik van het versterken en verzwakken van waveletcoëfficiënten: de coëfficiënten die nodig zijn om details in de ROI te reconstrueren worden versterkt vooraleer deze geëncodeerd worden. Hierdoor zullen deze details vroeger in de datastroom terug te vinden zijn. Dankzij de kwaliteitsschaalbaarheid die in het voorafgaande hoofdstuk bekomen wordt, kan de datastroom zodanig opgesteld worden dat de ROI van *elke* resolutie perfect gereconstrueerd wordt vooraleer de overige achtergrondgebieden verwerkt worden. Deze aanpak laat zowel de wavelet-transformatie als de codeeroperaties ongewijzigd. Echter, een onaangepaste transmissie, die de achtergrondgebieden opschaalt en verder ongemoeid laat, resulteert in verminderde kwaliteit door opstapelende fouten per resolutie. Om dit aan te pakken, wordt handig gebruik gemaakt van het feit dat onregelmatige meshes gebruikt worden, namelijk door het toevoegen van vertices – wat per resolutie normaal over de volledige mesh gebeurt – te beperken tot de interessegebieden. In plaats van een groot aantal vertices te behouden en deze glad te strijken in achtergrondgebieden, wordt hetzelfde effect bekomen door er de resolutie lager te houden. Experimenten met grote interessegebieden die een half model omvatten en als *visueel verliesloos* kunnen beschouwd worden, leveren een globale winst op wanneer enkel de ROI gedecodeerd wordt, ondanks een beperkte meerkost. Het hanteren van kleinere interessegebieden resulteert in grotere bitbesparingen zonder hierbij de visuele kwaliteit zichtbaar te reduceren.

De meest interessante toepassingen van interessegebieden liggen aan de decoderkant, wat verder in het tweede deel van Hoofdstuk 4 besproken wordt. Het doel van ROI-decoderen is dat de gevisualiseerde mesh perfect aangepast kan worden aan de noden van de applicatie. Dit kan een dokter zijn die een manueel bepaalde regio van een medisch beeld in detail wil kunnen bekijken, of dit kan een interactieve visualisatie zijn waarbij modellen slechts deels ingeladen worden, rekening houdend met waar een gebruiker zich in de virtuele wereld bevindt. Om dit mogelijk te maken, zijn enkele wijzigingen nodig aan de originele aanpak. Waar de ROI's aan de *encoderkant* zodanig gekozen konden worden zodat het decoderen van enkel deze gebieden voldoende informatie biedt om alle ROI's in hogere resoluties te reconstrueren, kan deze garantie niet gemaakt worden voor

ROI-decodering: aangezien de decoder (per definitie) nog geen weet heeft van hogere resoluties, noch van welke ROI's bepaald zullen worden in die hogere resoluties, is de kans steeds groot dat het reconstrueren van de ROI op een bepaalde resolutie zal zorgen dat lagere resoluties opnieuw gereconstrueerd moeten worden om voldoende informatie te voorzien. Deze adaptieve inverse transformatie laat toe om meshes te genereren die perfect aangepast zijn aan de noden van de applicatie. Voor een echt schaalbaar systeem volstaat dit echter niet. Hoewel meshes aangepast zijn om het grafische geheugen optimaal te benutten, moeten alle waveletcoëfficiënten gekend zijn en moet alle data bijgevolg gedecodeerd worden. Om dit te vermijden, moet de informatie in het geëncodeerde bestand willekeurig toegankelijk zijn. De mate waarin deze informatie toegankelijk is, is sterk toepassingsgericht: in het ene extreme geval kan elke waveletcoëfficiënt individueel opgevraagd worden maar dan is er van compressie geen sprake meer, in het andere extreme geval worden alle coëfficiënten zo optimaal mogelijk samen geëncodeerd maar dan is het willekeurig opvragen van coëfficiënten onmogelijk. In dit proefschrift wordt voorgesteld om de informatie in drie dimensies te tegelen, waarbij elke tegel nu onafhankelijk van naburige tegels verwerkt kan worden. Bijgevolg kan deze betegeling per resolutie optimaal aangepast worden worden aan de densiteit van de vertices. Daarenboven laat een dergelijke betegeling van de geëncodeerde data toe om RD-optimalisatie lokaal uit te voeren: tegels kunnen zodanig opgeslagen worden dat de regio's die de kwaliteit het sterkst laten toenemen met het minimum aan bits eerst geëncodeerd worden. Om ROI-codering te evalueren worden twee gevallen beschouwd: enerzijds worden maximale bruikbare interessegebieden bekeken, anderzijds worden minimaal mogelijke interessegebieden gebruikt. Uit de resultaten blijkt dat de adaptieve inverse transformatie zoals verwacht het aantal vertices en driehoeken in de mesh perfect kan aanpassen aan de gevraagde ROI's, enkel aan de rand van deze gebieden zijn extra vertices nodig om te garanderen dat de ROI's zelf zonder afwijkingen gereconstrueerd worden. Bij grotere meshes wordt het aandeel van deze extra vertices verwaarloosbaar. Het decoderen van tegels toont dat een groter aantal kleinere tegels toelaat om een groot aantal bits uit te sparen, ondanks een grotere meerkost om ROI-codering toe te laten. De resultaten tonen daarenboven aan dat deze meerkost daalt voor groter wordende meshes, terwijl ook de relatieve kost om één punt te decoderen (en bijgevolg alle vertices die in diezelfde tegel geëncodeerd werden) daalt.

Het resultaat is een representatie en encodering voor 3D-modellen die schaalt volgens de noden van applicaties en volgens de beperkingen van systemen. Via een wavelet-transformatie die geometrisch kenmerkende eigenschappen behoudt tot in lagere resoluties, wordt een model opgesplitst in een lage-resolutiebenadering en een aantal waveletsubbanden die de details beschrijven op een beter comprimeerbare manier. Deze resolutieschaalbare codec laat eveneens kwaliteitsschaalbaarheid toe door gebruik te maken van een templatemesh die bekomen wordt zonder waveletcoëfficiënten, en zo de encodering van de representatie scheidt; dit vermijdt dat waveletcoëfficiënten volledig moeten gereconstrueerd worden vooraleer een hogere resolutie te

decoderen. De combinatie van beiden laat een rate-distortie-geoptimaliseerde opslag en transmissie toe. Om te voorkomen dat een volledig model verwerkt moet worden als maar een deel ervan vereist is, is het concept van ROI's geïntroduceerd. Aan de encoderkant kan dit ingevoegd worden zonder veel aanpassingen; aan de decoderkant is een adaptieve inverse transformatie nodig. Om ROI-decodering efficiënt toe te laten, is een betegelde encodering voorgesteld om willekeurige toegang in de datastroom toe te laten. De resulterende representatie en encodering voor 3D-modellen laat alle vormen van schaalbaarheid toe.

# English Summary

The amount of 3D content within digital media is growing at a steady pace. A few decades ago, the term "3D" mainly brought video games and specialized CAD software to mind; today 3D is ubiquitous due to the advent of 3D printers and the rise of augmented reality (AR) and virtual reality (VR) applications. AR augments the real world with virtual objects to be seen with the newest smart glasses, whereas VR encompasses each application where an entire virtual world is created. Training simulators, the preservation of digital heritage and an architect's design of your future dream house are just a few examples. In addition to the growing quantity of 3D content, the quality of each 3D model is increasing because of more accurate scanning devices and because of more powerful modeling tools which allow for editing even the finest details.

Unfortunately, the rates at which the performance and memory of computers increase do not suffice for interactively handling such models. Additionally, our devices are becoming increasingly diversified: today we need to consider digital devices from commodity desktop PCs and laptops to tablets and smartphones, each with its own technical specifications. Furthermore, we also need to consider several ways for storage and transmission, from local and external USB storage to cloud storage which is only accessible through a WiFi or mobile connection. Providing a single model which suffices for all use-cases results in reduced quality; after all, on my TV I prefer a better quality than what is streamed to my mobile device. Conversely, providing a model for each possible configuration is unsustainable. There is a need for a scalable representation which can accommodate for all storage, transmission and rendering limitations.

The actual challenges are discussed in Chapter 2. To represent a 3D model digitally, *polygon meshes* are often used. A polygon mesh is a data structure which represents 3D points as *vertices*, and which connects these vertices using polygons. If all polygons are triangles, this is called a *triangle mesh*. The latter is often used for interactive visualization of virtual scenes; triangle meshes will also be used in this dissertation, and will be referred to shortly as *meshes*. Such meshes emerge after two important processes occurring in each analogue-to-digital conversion: *sampling* and *quantization*. Sampling an object results in a discrete amount of samples or points on the surface of the object; more available samples means more information is known about the surface, but also more storage is required. Each of these samples becomes a vertex of the mesh, and each of these vertices is represented by an approximation with a finite precision after quantization. Contrary to more traditional multimedia domains, the relationships between the

samples need to be known explicitly; this connectivity information is given by the triangles which are formed between the vertices. These triangles approximate the surface of the original object. Chapter 2 goes deeper into sampling and quantization. Also, the quality for any given cost is considered: how many bits can be spent and what quality can then be obtained?

We observe that this representation does not scale for larger 3D models. There is a need for compression, and this dissertation investigates *lossless* compression which allows for accurately reconstructing the original digital model. Approaches to accomplish this are also discussed in Chapter 2, where forms of scalability are investigated. A scalable representation needs to be able to provide sufficient quality depending on the requirements of applications and the limitations of systems and networks. Three important forms of scalability are considered: (1) resolution scalability, (2) quality scalability, and (3) spatial scalability with *regions of interest (ROIs)*. Scaling the resolution determines the amount of samples used for representing a model, scaling the quality determines the accuracy of positioning the vertices, and encoding and decoding ROIs allows for selecting specific regions out of a larger mesh and only processing these regions. Chapters 3 and 4 discuss the designed representation and encoding which meet all these scalability considerations.

The base design of the proposed representation and encoding is discussed in the first part of Chapter 3, leading to a resolution-scalable system. A wavelet-based transform is discussed, which efficiently represents a high-resolution mesh by a lower-resolution approximation together with the detail information which allows for reconstructing the original mesh. These details are called wavelet coefficients, collected in a *wavelet subband* per resolution. Contrary to many approaches in literature, *irregular* meshes are considered in this dissertation. The vertices of such meshes do not follow any regular structure, which results in a more expensive compression on the one hand, but on the other hand allows for improved quality using fewer vertices and triangles. The proposed mesh codec preserves geometric features, or "sharp edges", by design. Features are preserved implicitly, without any explicit indication of which edges should be considered as features. Such feature-preservation again improves the quality of the lower-resolution approximations using fewer vertices and triangles. The resulting wavelet coefficients are compactly represented using so-called *octrees*. These are abstract data structures which iteratively subdivide each cell into eight smaller cells by splitting once in every dimension. Hence, wavelet coefficients which are located nearby and which are more strongly correlated, are encoded together to improve the compression performance. The results show the trade-off between more expensive irregular meshes and better quality using fewer vertices: despite a higher cost per vertex, fewer vertices are required to obtain a similar quality compared to semi-regular mesh coding systems. This results in competitive coding performance, and even improves upon the state of the art when encoding feature-rich models, especially at lower resolutions. In addition to the classical *rate-distortion (RD)* evaluation where the distortion is evaluated w.r.t. the spent *bit rate* (i.e., the amount of bits per vertex), a novel evaluation measure is proposed,

showing the distortion in function of the percentage of reconstructed vertices. This gives an indication of the memory footprint for interactive applications, *after* having decoded the models. In this regard, the proposed representation using irregular meshes clearly outperforms the state of the art.

In the second part of Chapter 3, quality scalability is investigated. A fundamental issue for adaptively scaling the quality of the wavelet coefficients is keeping the encoder and decoder synchronized: the decoder needs to use the same information as was used by the encoder to encode the data. Consequently, distortions due to decoding drift occurs if a decoder processes a higher resolution before fully reconstructing all lower-resolution subbands. To allow for quality scalability nonetheless, the use of template meshes is proposed. Instead of using the real mesh in the encoding process, a template mesh is employed at both the encoder and decoder side which is obtained without information about wavelet coefficients. This approach allows for a nearly arbitrary storage and transmission order of the data blocks. With both resolution and quality scalability available, the most interesting application is *RD optimization (RDO)*: at each moment during the encoding process, the encoder can analyze what information improves the global quality most optimally at a minimal rate, either the transmission of a new resolution or the refinement of an existing subband. The results clearly show that the use of template meshes, which allows for quality scalability and which unlocks RDO, does not introduce a significant rate penalty. Furthermore, the optimization has an observable effect at low and midrange bit rates; in other words, the system allows for decoding the most optimal quality at each rate, with most noticeable improvements at low resolutions.

Resolution and quality scalability operate on entire models. This suffices as long as models are visualized entirely. Currently, however, meshes have such high levels of detail that the finest details cannot be observed when visualizing entire models. These fine details are only observable when displaying these models from nearby, but then high rates and long computation times are spent on regions that are not being displayed. Chapter 4 therefore investigates *Region-of-Interest (ROI)* coding: how can the concept of regions of interest be used to vary the resolution and quality in a spatially adaptive way? The first part of this chapter tackles the problem from the point-of-view of an encoder. Independent of any decoding, an encoder can prioritize specific regions of a model. The details in the face of a virtual character can be prioritized over details in his clothes, or the front side of a monument can be prioritized over the back side. The approach suggested in this chapter uses boosted wavelet coefficients: the coefficients which are required for accurately reconstructing the ROI per resolution are boosted before encoding, which ensures that this information is found earlier within the data stream. This approach leaves the wavelet transform and the encoding process unaltered. Furthermore, by exploiting quality scalability, the data stream can be constructed such that the ROI of *each* resolution is perfectly reconstructed before any background region is processed. However, increasing the resolution in background regions without refining these vertices results in deterioration due to prediction errors accumulating per resolution. To remedy this, the addition of

vertices – which occurs over the entire mesh – is limited to those vertices within the ROIs: instead of preserving a large amount of vertices and smoothing the background regions, smoothness is obtained by preserving a lower resolution in the background. Experiments with large ROIs determined by all areas visible from a given position, which on average cover half of a model and which are considered *visually* lossless, show a global rate decrease when only decoding the ROI, despite a limited rate penalty for lossless decoding. Employing smaller ROIs results in larger rate savings without visually observable quality deterioration.

The most interesting application of ROIs is at the decoder side, which is discussed further in the second part of Chapter 4. The goal of ROI decoding is to adapt the visualized mesh perfectly to the needs of the application, whether this is a doctor who needs to manually determine the part of a medical image to display in more detail, or an interactive visualization whereby models are only partly fetched based on the point-of-view of the user within the virtual world. To allow for such ROI decoding, some modifications to the system as introduced in Chapter 3 are required. Whereas encoder-side ROIs are determined such that solely decoding these regions suffices for decoding all ROIs, such guarantees cannot be made for decoder-side ROI coding. As a decoder (by definition) has no knowledge of higher resolutions nor of which ROIs will be required at these resolutions, there is a significant probability that the reconstruction of an ROI at a specific resolution will require re-evaluating lower-resolution ROIs to ensure sufficient information is available. This results in an adaptive inverse transform which allows for reconstructing meshes that are perfectly adapted for interactive visualization, making optimal use of the graphics memory. However, this does not suffice for a truly scalable system as all data still needs to be decoded before performing the transform using a limited selection of wavelet coefficients. A random access strategy is proposed to reduce the amount of decoded data. The granularity of random access is very application-specific: at one extreme case, each coefficient is individually accessible but this allows little compression; at the other extreme, all wavelet coefficients are optimally encoded together which disallows random access. This dissertation proposes to tile the data in three dimensions, where each tile can be processed individually. Consequently these tiles can optimally adapt to the vertex densities. Additionally, such tiling of the encoded data allows for performing RDO locally, storing tiles such that the regions are stored first which improve the quality most optimally using a minimal rate. To evaluate ROI coding, two cases are considered: on the one hand the maximally useful regions and on the other hand the smallest possible regions are investigated. The results show that the adaptive inverse transformation perfectly adapts the amount of vertices and triangles to the requested ROIs. Only near the borders of these regions, additional vertices are required to ensure that the ROIs are perfectly reconstructed. These additional vertices become irrelevant for larger meshes. Furthermore, experimenting with tile-based decoding shows that a larger amount of smaller tiles allows for saving a significant amount of bits despite a larger rate penalty to allow for ROI coding. The results also show that the penalty reduces for larger meshes, as well as the relative cost for decoding a single vertex

– and consequently each vertex which is encoded within the same tile.

The result is a representation and encoding system for 3D models which scales according to the needs of the applications and the limitations of the systems. By employing a wavelet transform which preserves geometric features down to the lowest resolutions, a model is split into a low-resolution approximation and a set of wavelet subbands which describe detail information in a compressible way. This resolution scalable codec also allows for quality scalability by employing a template mesh for the encoding. This template mesh decouples the encoding from the representation, allowing for decoding higher resolutions prior to fully decoding all lower-resolution subbands. Combining both resolution and quality scalability furthermore allows for a rate-distortion-optimized storage and transmission order. To avoid processing entire models if only a small part is required, the concept of ROIs is introduced. At the encoder side, this can be employed while leaving the transform and encoding process unaltered. For ROI decoding, a tiled encoding is proposed to allow for random access into the data stream. The resulting representation and coding system for 3D models hence allows for all required forms of scalability.

# 1
# Introduction

Ever since the dawn of mankind, we have been fascinated with modeling our world. We want to understand structure, we need plans and simulations, we do storytelling or we are merely interested for the sake of art. Examples are found everywhere, from the cave paintings of the stone age, over the designs of Leonardo Da Vinci, to the LEGO cars and Barbie doll houses of our own childhood.

With the advent of the digital era, all such models are being converted from analogue to digital media. Additionally, media no longer needs to be stored locally thanks to the online presence of many of our devices. We no longer use audio tapes, and even our MP3 collection is becoming obsolete now that more and more of us have a *Spotify* account. Our camera rolls have been swapped with SD cards, but we increasingly take pictures using our phones which upload these to any online platform. Our television cabinets are no longer filled with video cassettes and we no longer need to go outside to rent DVDs now that we can enjoy *Netflix*. Similarly, by now physical scale models are exchanged for digital models stored in the cloud.

Three-dimensional (3D) content has long been associated with computer-animated films, video games and specialized computer-aided design (CAD) software. Nowadays, 3D content is omnipresent, with applications ranging from augmented reality and virtual reality to far outside the entertainment sector, such as medical imagery, online virtual shopping, architectural heritage, scientific data visualization, training simulators, remote sensing, 3D printing, geographic information systems, etc. All these domains benefit from advancements in the representation and encoding of 3D content. The main challenges that arise are (1) how to *manage this content* which grows in amount and quality, (2) how to

properly *process and edit* such highly detailed content, and (3) how to efficiently display such complex data. In addition to considering the increasing data sizes, obtained through the use of high-end scanning devices or designed using ever-more capable modeling tools, the proposed solutions need to consider the wide spectrum of devices ranging from high-end desktop PCs to low-end mobile devices, with a very diversified range of capabilities and a very broad range of interconnecting networks.

## 1.1 Representation of 3D Content

The concept of "3D" is very general, indicating merely that the data describes "*three independent dimensions*". For instance, audio only has *one dimension*: audio data describes how air pressure changes over time. A picture is *two-dimensional (2D)*: color values change in both the horizontal and vertical direction. Combining both a time dimension and two spatial dimensions, a video is *three-dimensional (3D)*, describing how a picture changes over time. Alternatively, this dissertation covers 3D content which handles the three *spatial* dimensions we observe in our world and which we can informally identify as depth, width and height. Conventionally, these dimensions are indicated by $x$, $y$ and $z$. This is illustrated in Figure 1.1, which shows two points $\mathbf{p}_1$ and $\mathbf{p}_2$ decomposed into their $x$, $y$ and $z$ components.

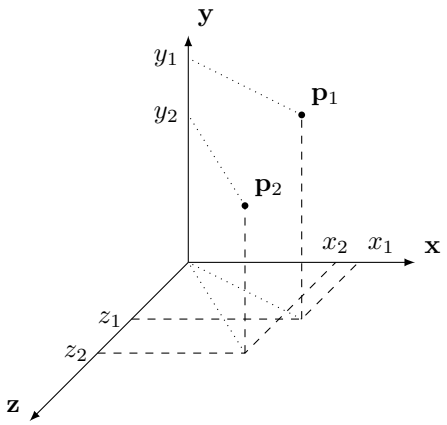
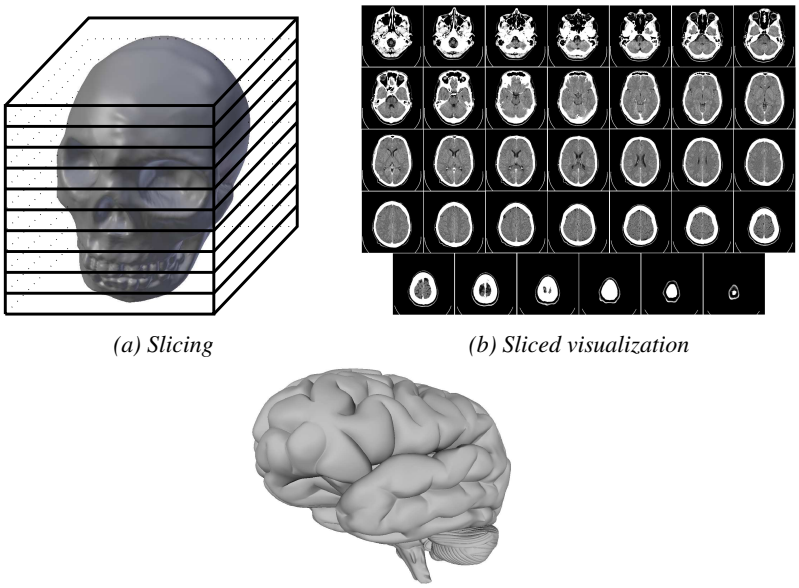



*Figure 1.1: Points in 3D.*

### 1.1.1 Volumetric Data

In general, 3D *data* defines a value for every point within a volume. Consider the following mathematical example. Choose a fixed point $\mathbf{c}_0 = (x_0, y_0, z_0)$; the value



*(a) Slicing* *(b) Sliced visualization*

*(c) Isosurface visualization*

*Figure 1.2: Volumetric data visualization. A conventional approach to visualize volumetric data is by slicing (a) and depicting the resulting 2D slices (b). Alternatively, an isosurface (c) can represent all points having a specific X-ray absorption. However, while a set of slices can be perfectly displayed on a 2D medium, this is not possible for a set of isosurfaces. For instance, although the isosurface itself represents both the front and back of this brain model, both cannot be displayed together.*

at each point $\mathbf{p} = (x, y, z)$ can for instance be described by a distance function

$$F(\mathbf{p}) = F(x, y, z) = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}, \qquad (1.1)$$

signifying that the value at each point $\mathbf{p}$, i.e. $F(\mathbf{p})$, is the distance from $\mathbf{p}$ to $\mathbf{c}_0$.

Real-world data often does not allow itself to be formalized easily in such mathematical formulae. Consider, for instance, the data obtained during a CT scan. At each point a scalar value represents the amount of X-ray absorption, and these scalar values are visualized as grayscale values ranging from black to white.

A fundamental problem with 3D *visualization* is that we are limited to two dimensions, be it on paper or on a display. A conventional approach to visualize 3D volumetric data is to "*slice*" the data, similar to how one would look inside a physical object, only less destructive. Figure 1.2a depicts this approach for a human skull, and the resulting slices are shown in Figure 1.2b. Alternatively, such data is often visualized as short video sequences, displaying all slices in rapid succession.

### 1.1.2 Isosurfaces

A different approach to represent volumetric 3D data on a planar 2D display, such as the pages of this dissertation, is to consider all points which have specific values. This approach is well-known in 2D, where isolines connect all points in a plane which have the same specific value, forming lines *embedded* in the 2D space. Familiar examples include isohypses on an elevation map and isobars on a weather map.

Similarly, volumetric data can be visualized using *isosurfaces*. Given a specific value, all points which have this exact value form a continuous surface embedded in the 3D space. Consider again the volumetric data given in Equation 1.1. Choosing a positive value $r$, the set of all points $(x, y, z)$ for which $F(x, y, z) = r$ form a sphere, as the distance from each of these points to $\mathbf{c}_0$ is exactly $r$. This is known as an *implicit surface*, defined by $F(x, y, z) = r$ or $F(x, y, z) - r = 0$.

Observe that such a surface is in fact a 2D construction: one can only traverse a surface in two independent directions. Hence, it can be defined by two parameters and the corresponding representation is known as a *parametric surface*. For instance, the sphere is formally known by:

$$\begin{aligned} \mathcal{S} : [0, 2\pi] \times [0, \pi] \subset \mathbb{R}^2 &\to \mathbb{R}^3; \\ (\phi, \theta) &\mapsto \mathcal{S}(\phi, \theta) = (x, y, z), \\ \text{with } &\begin{cases} x = x_0 + r\cos(\phi)\sin(\theta) \\ y = y_0 + r\sin(\phi)\sin(\theta) \\ z = z_0 + r\cos(\theta) \end{cases} . \end{aligned} \tag{1.2}$$

Each couple $(\phi, \theta)$, maybe better known as *longitude* and *latitude*, corresponds to a point $(x, y, z)$ on the sphere with radius $r$ around the point $\mathbf{c} = (x_0, y_0, z_0)$. This alternative definition of the sphere formed by $F(x, y, z) = r$ clearly illustrates the embedding of a 2D surface (i.e., there are two independent dimensions, $\phi$ and $\theta$) in the 3D space (i.e., values have three components, $x$, $y$ and $z$).

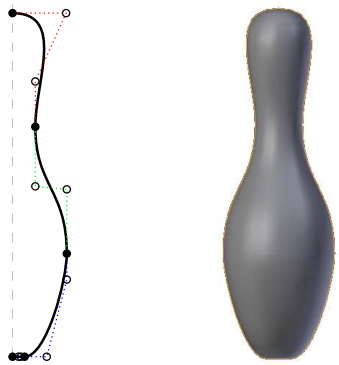
For the scanned data of Figure 1.2, an appropriately chosen value could result in the isosurface shown in Figure 1.2c. Observe that such a surface is still embedded in 3D, giving rise again to the fundamental issue of how to visualize 3D content on a 2D medium. Fortunately, contrary to *volumetric* data where we need to see *inside* a volume, we humans have been visualizing 3D *surfaces* for centuries: apart from occlusions due to projecting on a 2D plane, the human brain can be tricked into seeing a third dimension by making use of proper shading. Nonetheless, for a full observation of the surface either a 3D printed model is required, or it has to be viewed in an interactive 3D environment.

This dissertation will handle the processing of 3D surfaces. While *isosurfaces* are often related to volumetric data, many applications merely require a representation of the physical surfaces of objects. Consider in this case the

volumetric data given by $D(x, y, z)$, the distance of a point $(x, y, z)$ to the surface. The isosurface is then the implicit surface $D(x, y, z) = 0$.

### 1.1.3 Digital Surfaces: the 3D Mesh

Surfaces can be represented digitally in several ways. In theoretical cases, 3D data can ideally be represented using mathematical equations such as Equation 1.2; for instance, any sphere can be represented using only four numbers, i.e., a radius and three numbers for the center. Such parametrized descriptions are great for expressing perfect planes, spheres, cones or tori, and with a little bit more effort these descriptions can describe combinations of (possibly transformed versions of) such primitives. However, for arbitrary surfaces, an *exact representation* quickly become cumbersome.



*(a) Bézier curve* *(b) Resulting surface*

*Figure 1.3: Bézier curve and surface. This example shows how a bowling pin is modeled using a Bézier curve. While this representation results in perfectly smooth curves, it is too complex for real-time, interactive applications.*

For approximating arbitrary surfaces, well-known approaches are built on *Bézier surfaces* and *B-splines*. Such approximating surfaces can be designed arbitrarily accurate by defining sufficient control points. These surfaces are popular representations in CAD applications, but are overly complex for *interactive* visualization. Figure 1.3 shows an example. A bowling pin can be represented by cubic Bézier curves as shown in Figure 1.3a. By spinning this curve around a fixed center axis, the surface shown in Figure 1.3b is obtained.

Instead of relying on the curvature defined by the tangential vectors formed by control points, one can also consider a *linear interpolation* between control points, resulting in a *polygon mesh*. Such a mesh is defined by the control points or *vertices*, and the linear interpolation between these vertices which forms polygons.

Newell's *Utah teapot* (Figure 1.4) is a famous 3D mesh from the early computer graphics days.

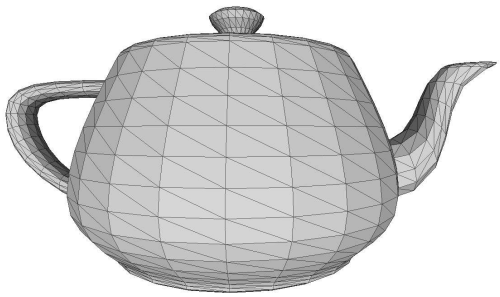
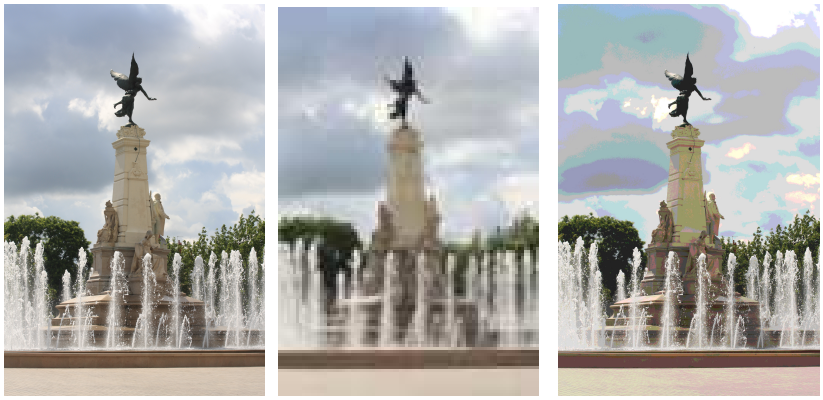


*Figure 1.4: Example mesh:* Utah teapot. *This is an example of a 3D mesh. It represents the surface of a teapot, and approximates this surface by defining vertices, i.e., the points where multiple lines cross, and faces, i.e., the triangles formed between the vertices. Observe that, while the geometry information as such is given by the vertices, the connectivity information is equally important in defining the approximated surface. Imagine what the surface would look like if vertices at the tip of the spout were connected with top of the lid; while the geometry information remains unchanged, a new surface would be defined.*

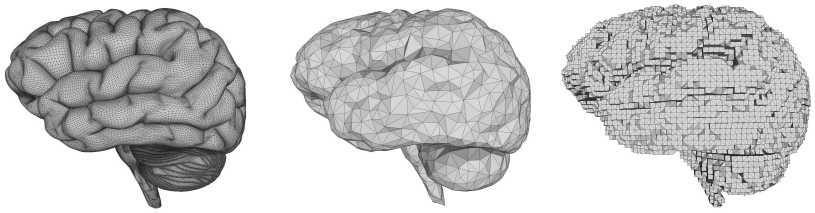
Similar to other multimedia domains, the distortion associated with a digital representation is related to the resolution determined by the sampling density on the one hand, and the quality of the samples on the other hand. Let us compare with image coding. Firstly, a digital image *samples* a picture, resulting for instance in a resolution of $3\,888 \times 2\,592$ *picture elements* or *pixels*, meaning there are only $3\,888$ columns of pixels, over $2\,592$ rows. When reducing the sampling density to $128 \times 86$ pixels, fine detail is lost, as is illustrated by Figure 1.5a and its lower resolution approximation in Figure 1.5b. Secondly, the color value of each sample or pixel is obtained by mixing three color components, namely red, green and blue. Each color value can be *quantized* to, for instance, $8$ bits per color component. This allows for $2^{8\times3} \approx 17$ million possible color values. Reducing the amount of quantization bits to $3$ bits per component as shown in Figure 1.5c shows *banding* artifacts: the colors can no longer vary smoothly, resulting in possibly large color jumps between neighboring pixels.

For meshes, the distortion is similarly determined by the vertex density. For instance, the brain surface depicted in Figure 1.2c is represented by a high-resolution mesh as shown in Figure 1.6a. The amount of vertices relates to the approximation accuracy. The denser sampling, as shown in Figure 1.6a, preserves



*(a) Original image* *(b) Low resolution* *(c) Coarse quantization*

*Figure 1.5: Image resolution and quantization. In (a), an original high-resolution and finely-quantized image is displayed. (b) shows the resulting image at a much lower resolution, i.e., using fewer pixels, while (c) shows the resulting image at a much coarser quantization, i.e., with the same number of pixels but with a lower color quality.*



*(a) High resolution:* 294 012 *vertices* *(b) Low resolution:* 3 941 *vertices* *(c) Coarse quantization:* 294 012 *vertices*

*Figure 1.6: Mesh resolution and quantization. (a) shows an originally digitized version of the brain isosurface depicted in Figure 1.2c. A lower-resolution mesh is shown in (b): observe that the use of fewer vertices no longer preserves finer detail. (c) shows a high-resolution mesh where the vertex positions have a coarser quantization. Observe that fine detail is again lost.*

fine detail better compared to the sparser sampling shown in Figure 1.6b. This is referred to as the resolution of a mesh. Quantization additionally affects the accuracy of the vertex locations and results in banding artifacts similar to what is observed in image coding, where the $x$, $y$ or $z$ value of neighboring vertices either stays fixed, or jumps by a significant amount. This is depicted in Figure 1.6c.

**Functions on 3D Surfaces** At this point, the mesh approximates the geometry of an object, by combining vertices (i.e., geometry information which samples

the actual geometry) and faces (i.e., connectivity information which defines the remaining geometry by interpolation).

Taking these surfaces one step further, functions can be defined *over* a surface. For instance, one could describe the average yearly temperature for each point on the surface of planet Earth. Useful values for describing the surface of an object often include appearance attributes, encompassing color information, reflectance, transparency, etc. After sampling, such information is typically provided either via specific vertex attributes in addition to the required geometric coordinates, or via several texture maps which are indexed by $(u, v)$ coordinates given per vertex. Either the specific attributes or the $(u, v)$ coordinates per vertex are then interpolated over the faces to obtain values over the entire surface.

In this dissertation, only geometric information per vertex is considered. The techniques proposed in this thesis can be extended to other attributes, which will require taking into consideration other specific attribute-related decision criteria. Additionally, only *triangular* meshes are considered, i.e., meshes for which all faces are triangles. For real-time rendering, this mesh representation is encountered most often; any polygon mesh can be converted to a triangle mesh without altering its geometry.

## 1.2 Need for Scalable Representations

A typical rendering system can look as depicted in Figure 1.7. A server, which can be located remotely in the cloud, nearby within your local network, or even locally on a DVD or USB disk, holds the mesh data. This data either needs to be copied to a local disk or uploaded immediately to memory for processing. Eventually, the data is uploaded to a graphics processing unit (GPU) for interactive rendering. GPUs are perfectly equipped to process triangular meshes, where a triangle mesh is conventionally represented as a list of vertices together with a list of triangles, each composed of three indices into the list of vertices. Nonetheless, the last several decades have shown that this representation does not suffice. Using modern modeling software and acquisition hardware, models with millions and even billions of vertices are no exception. For instance, in the *Digital Michelangelo Project*[1] at Stanford University, a 3D mesh of the statue of David was obtained, represented using a billion polygons. Interactive visualization of such models, however, is not straightforward. Consider a current-day high-end GPU such as the *NVIDIA GTX 1080*, which offers 8 GB of memory. Assume that each vertex is stored using three 2-byte coordinates, and that each triangle is stored by giving three 4-byte indices into the list of vertices. Given the fact that, in general, the amount of triangles is approximately twice the amount of vertices, the mesh is

[1] `https://graphics.stanford.edu/projects/mich/`

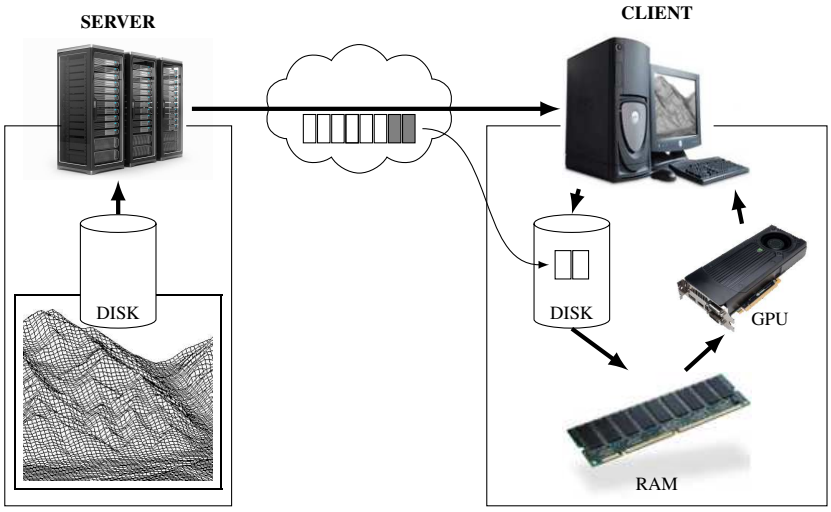



*Figure 1.7: Rendering system. In the most general case, a rendering system consists of the components depicted in the figure. 3D meshes are stored on a server. When data is requested by a client such as a PC or a smartphone, the data needs to be streamed to the local disk, after which it can be loaded into memory to be uploaded to graphics memory. More specific scenarios include cases where the data is read from a local DVD drive or USB disk, or cases where data is already copied or installed on the local drive in advance.*

stored using 30 bytes per vertex, i.e., 6 bytes for the vertices and $2 \times 12$ bytes for the triangle indices. Then this GPU can render models of maximally $250 \sim 300$ million vertices, not yet billions. Table 1.1 gives some additional figures, for the *NVIDIA GeForce 940M* found in laptops, and the *iPhone X* and *LG Nexus 5X* mobile phones. Note that these mobile phones share their memory between the CPU and GPU, such that the amount of memory available for graphics is limited by active applications.

| | **GPU memory** | **Largest mesh** | |
|---|---|---|---|
| NV GTX 1080 | 8 GB | $267 \times 10^6$ vertices | |
| NV GF 940M | 4 GB | $133 \times 10^6$ vertices | |
| iPhone X | 3 GB | $100 \times 10^6$ vertices | (shared with the CPU!) |
| LG Nexus 5X | 2 GB | $67 \times 10^6$ vertices | (shared with the CPU!) |

*Table 1.1: Mesh limitations w.r.t. storage capacity.*

For many applications, this amount of memory suffices even though these devices do not allow for rendering *billions* of vertices. However, real systems require more than just this GPU storage to be taken into consideration, as is also made clear by Figure 1.7. While a rendering system can operate smoothly when

all required data resides on the GPU itself, one needs to consider the latency and bandwidth requirements to offer a qualitative experience. When rendering at $60$ frames per second, approximately $16.7$ ms are available between rendering one frame and the next to do all required calculations for animations, physics, etc., to fetch all required data and to do the actual rendering. If models are not available in time, this results in a so-called *popping* effect by which geometry or texture data is not visible instantaneously and an end-user only sees it appearing on the display after a short instant. This severely reduces the quality of experience, and can be addressed by proper prefetching mechanisms which accurately predict what data will be needed in the near future and which fetches this data in advance if sufficient bandwidth is available. Alternatively, such popping effects can be addressed by progressive representations which allow for depicting lower-quality intermediate approximations while fetching the higher-quality original data.

Some numerical facts give more insight. The NVIDIA GTX 1080 offers a memory bandwidth of $320$ GB/s or $320$ MB/ms. Hence, if the data is readily available on the GPU, approximately $10\,000\,000$ vertices and their associated triangles can be rendered per millisecond. If the data still needs to be streamed to GPU memory over a 16-lane PCIe3.0, at $15.75$ GB/s or $15.75$ MB/ms only $500\,000$ vertices and their triangles can be streamed to the GPU per millisecond. If this data is not available in memory, it must be read into local memory first, either from local or remote storage. Table 1.2 indicates some common bandwidth figures. Again, the amount of vertices needs to be interpreted as the amount of vertices *and associated triangles* that can be rendered, streamed from local storage or streamed from remote storage. That is, a geometry rate of $n$ vertices/ms allows for streaming, on average, $n$ vertices and an estimated $2n$ triangles per millisecond.

| | **Bandwidth** | **Geometry per second** | |
|---|---|---|---|
| NV GTX 1080 | 320 GB/s | $10.6 \times 10^6$ vertices/ms | (rendering) |
| NV GF 940M | 14.4 GB/s | $480 \times 10^3$ vertices/ms | (rendering) |
| PCIe3.0 (x16) | 15.75 GB/s | $525 \times 10^3$ vertices/ms | (GPU upload) |
| SSD | 500 MB/s | $16.7 \times 10^3$ vertices/ms | (RAM upload) |
| WiFi | 100 Mbps | 417 vertices/ms | (LAN download) |
| Coax | 30 Mbps | 125 vertices/ms | (WAN download) |

*Table 1.2: Mesh transmission w.r.t. bandwidth.*

Observe that the limitations discussed above consider virtual environments where only a *single model* is to be viewed, and only the *geometry* of the approximated surface is relevant. When creating interactive virtual worlds, at least the visible parts of the *entire* scene need to be stored on the GPU and preferably even more. Depending on the scene complexity, tens or hundreds of objects need to be available simultaneously. And in addition to the geometry information

given by the three 2-byte coordinates, multiple additional attributes are required for a realistic environment. These can include diffuse and specular color values, information on surface reflectivity and transparency, normal vectors for lighting-related surface material interactions, etc. These additional attributes can be stored directly as vertex attributes or can be indirectly accessed in texture maps, which all have to be available in graphics memory next to the required geometry and connectivity information.

It is clear that compression and scalable representations are required to account for less efficient GPUs, slower local drives, less optimal local network fetching and even data retrieval over the world wide web. Indeed, in addition to the increasing size of the meshes, today one needs to consider lower-end devices such as tablets and smartphones with less processing power compared to commodity PCs, possibly connected at lower bandwidths which further increase download times. A scalable approach to handle meshes is well placed to address these challenges.

**Compression** Data needs to be compressed to allow for efficient storage and transmission. Compression techniques exploit correlations, similarities and predictability to store data in a vastly compacted fashion. Several general-purpose compression algorithms can be used for creating well-known *.zip* and *.rar* files. However, these exploit general statistical properties without knowledge on what the bytes actually represent. The compression of audio, image and video data is improved via transformations which allow for exploiting domain-specific knowledge, resulting for instance in *.mp3*, *.png* and *.mp4* files. This dissertation similarly explores knowledge on *3D* data to improve compression. The performance of a coder and decoder, i.e., a *codec*, is expressed as a *bit rate*: instead of comparing absolute file sizes which vary with model complexity, the performance is expressed by the required amount of *bits per vertex (bpv)*.

Compression results in a file with a vastly reduced storage and transmission footprint. However, such a file can no longer be used directly as it needs to be decompressed in order to reconstruct the vertices and triangles of the mesh. If reconstructing the mesh is only possible when decoding the entire file, i.e., a fixed amount of bits per vertex, such compression is called *single-rate* compression. Furthermore, decoded models still need to be loaded in GPU memory entirely, such that the memory required for interactive rendering remains unchanged.

**Levels of Detail** Single-rate compression as such does not suffice. Using state of the art single-rate compression methods, models have been represented at rates down to $1.6$ bpv. A billion-vertex model then needs $200$ MB of storage space. This still takes $400$ ms to fetch from an SSD and $16$ s to download from a local server. Furthermore, this does not yet account for the additional processing power and

time required for decoding the entire model before uploading it to the GPU. To reduce popping effects, the conventional approach is to use several *levels of detail (LODs)*. Each lower level of detail is typically a lower-resolution version of an original model. A low LOD can be visualized without delay, during which higher LODs are being loaded. This is called a *simulcast* solution: several LODs are encoded, stored and transmitted independently. This approach can be compared to a website storing both thumbnail images and full-size images, with the full-size images only depicted if a user needs to see them. Until today, employing several LODs of complex 3D models is the conventional approach in interactive rendering of virtual worlds. Artists and 3D modelers design several LODs of models *manually* to ensure that the resulting models are of an adequate quality level when considering specific memory constraints. This improves interactivity, but performs suboptimal as the similarities between several LODs are not exploited, neither for an optimal storage, nor for improved processing. After all, intuitively one can assume that the information needed to represent a lower LOD must in some sense be related to the information required to represent a higher LOD. These correlations are exploited to offer scalability. One needs to be able to scale the decoded data based on the requirements of the application or the limitations of the network and hardware using a single representation.

### 1.2.1 Resolution Scalability

To cope with high-resolution data, a first real requirement of any mesh coding system is *resolution scalability*, which improves upon LOD representations. An application often does not require all visible models at their highest resolution. Models far from a camera can be visualized using a reduced resolution without visual distortion. Additional detail information can then be loaded as the camera comes closer. Figure 1.8 displays three resolution levels.

In a resolution-scalable representation, only the *differences* with the previous resolution need to be obtained in order to improve the reconstructed model, contrary to LOD representations which have to decode an entirely new model per level. Furthermore, the amount of resolutions offered by a resolution-scalable system often surpasses the amount of levels of detail in an LOD system. This results in smoother and more fine-grained improvements. Finally, such finer-grained resolutions allow for more accurate memory and bandwidth management in environments where memory or bandwidth is limited. Given an entire scene of objects, the data can be streamed such that the distortion of the entire scene reduces optimally given the available bit rate. Eventually, compared to LOD systems, each object can be rendered more accurately at its most optimal resolution.

### 1.2.2 Quality Scalability

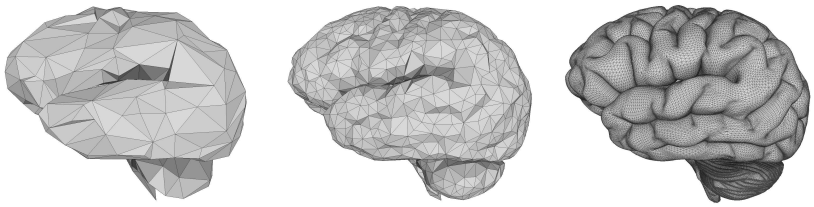
It is clear that both the resolution, i.e., the sampling density of the vertices, and the quality, i.e., the accuracy at which the vertices are given, affect the observed geometric distortion of a model (see Figure 1.6). At low resolutions, it makes less sense to reconstruct highly accurate vertices. This is illustrated in Figure 1.9. The transition from Figure 1.9d to Figure 1.9e shows the effect of adding a decimal digit of precision to the $x, y$ and $z$ components of low-resolution vertices, effectively increasing the quantization granularity tenfold. While there are obvious changes, the quality improvement is minimal. Adding more decimal digits of precision does not significantly alter the appearance of the brain model. In contrast, a tenfold increase of the quantization granularity has a drastic effect on the distinguishable features in the high-resolution mesh depicted on the top row, and additional decimal digits of precision further improve the visual quality.

*Quality scalability* hence is a desirable trait of mesh coding systems: the quality of the final vertices should be scalable, instead of blindly decoding all vertices at their highest accuracy without taking the resolution into account. Quality scalability ensures that the final accuracy of vertices can scale based on the accuracy required by the decoding application.

Resolution and quality scalability together allow for *RD optimization (RDO)*: a *rate-distortion* optimized storage and transmission order can be obtained, in which resolution and quality information is stored such that the distortion decreases optimally w.r.t. the rate. That is, each additional block of information either improves the reconstruction quality of existing vertices, or increases the resolution by adding new vertices, and the blocks are stored in such an order that the distortion is minimal at any arbitrary rate.

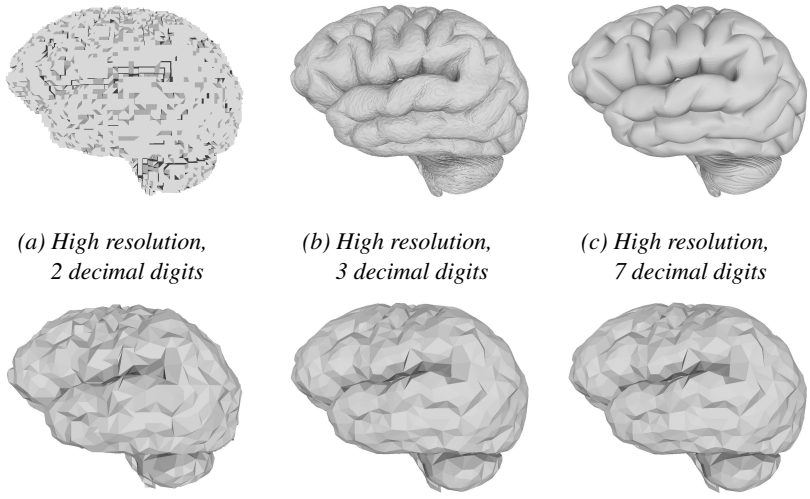
### 1.2.3 Region-of-Interest (ROI) Scalability

Finally, recall that a billion-vertex model cannot be stored entirely on the high-end NVIDIA GTX 1080. Resolution scalability allows for obtaining and visualizing an approximation which does fit on graphics memory. However, this does not allow for inspecting the finest possible details. Instead of waiting for better hardware to be developed, one can take into account that the amount of information that can be visualized is limited, considering limited display resolutions. A $1\,920 \times 1\,080$ display only shows approximately 2 million pixels. Hence, a billion-vertex mesh will never be entirely visible at its highest resolution. Either the entire model is visible and a lower-resolution approximation suffices without any observable deterioration, or the highest resolution of a specific part is required, leaving large parts of the model invisible outside of the screen. Figure 1.10 depicts two examples. In Figure 1.10a a particular part of the model is enhanced by the end-user, while Figures 1.10b and 1.10c show the automatic determination of front-



*(a)* 816 *vertices* *(b)* 3 952 *vertices* *(c)* 294 008 *vertices*

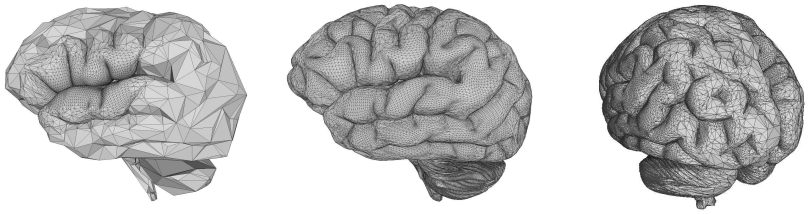
*Figure 1.8: Resolution scalability.*



*(a) High resolution, 2 decimal digits* *(b) High resolution, 3 decimal digits* *(c) High resolution, 7 decimal digits*

*(d) Low resolution, 2 decimal digits* *(e) Low resolution, 3 decimal digits* *(f) Low resolution, 7 decimal digits*

*Figure 1.9: Effect of quantization at different resolution levels.*



*(a) Box-selected ROI* *(b) Front-side ROI (front-view)* *(c) Front-side ROI (side-view)*

*Figure 1.10: Region-of-Interest scalability.*

facing regions. The side-view depicted in Figure 1.10c illustrates the reduced resolution of back-facing regions.

*Region-of-Interest (ROI)* support is a final requirement for modern mesh coding systems, allowing for adapting the resolution over a model based on the prioritization of an encoder or the requirements of a decoder. Consequently, an additional requirement to allow for ROI support, is *random accessibility* to avoid the transmission of unnecessary data.

## 1.3 Outline

This dissertation discusses a mesh representation and coding system which tackles the given requirements. A wavelet transform is described, by which a mesh is transformed to obtain several resolutions. Feature-preservation over all resolutions is an important property of the proposed transform. Consequently, even lower-resolution versions should preserve clearly recognizable geometric features. This preservation of features is achieved by allowing for irregularity in the mesh connectivity, i.e., vertices do not have to be connected in any predefined fashion. Such irregular connectivity information will allow for a more optimal trade-off between coding performance and geometric information per resolution, taking into account the fact that models need to be interactively *visualized* in addition to being stored and transmitted. The transformed model is encoded using a scalable coder, for which scalability entails the forms of scalability as discussed above, allowing for resolution and quality scalability, offering ROI support and permitting a rate-distortion optimized storage and transmission.

This thesis is structured as follows. Chapter 2 begins by describing the process of obtaining a digital mesh from a continuous, analogue surface. Then, some relevant properties of meshes are explained, after which methods for evaluating coding systems are discussed, including a novel rendering performance measure. Finally, the most significant works in the state of the art are covered.

Chapter 3 discusses the first part of this thesis, covering the global forms of scalability. First, a feature-preserving wavelet-based mesh transform is described, which transforms a high-resolution mesh into a lower-resolution mesh and a set of wavelet coefficients describing surface details along the mesh. Octree-based coding followed by *context-adaptive binary arithmetic coding (CABAC)* subsequently allows for an efficient storage by exploiting the statistical properties of the wavelet coefficients. Next, template meshes are introduced to decouple the representation from the encoding steps, which in turn allows for quality scalability and, consequently, for any arbitrary storage and transmission order of the data layers. This arbitrary storage and transmission order allows for *RD optimization (RDO)* and moreover allows for decoding the geometry information of multiple resolutions in parallel.

Next, Chapter 4 handles the second part of this thesis, which covers local ROI coding. First, ROI at the encoder side is tackled. Encoder-side ROI support entails prioritizing regions at the encoder side, resulting in faster quality improvements in these regions when decoding. This is achieved by employing boosted wavelet coefficients, leaving both the wavelet representation and wavelet encoding untouched. Decoder-side ROI support is subsequently discussed: the transmitted and decoded data needs to be adapted to the interactive needs of the *decoder*. An adaptive inverse wavelet transform is discussed, which allows for applying an inverse wavelet transform after separating the ROI from the background. To reduce the actually transmitted and decoded data, wavelet coefficients are partitioned into dynamic tiles, allowing for random accessibility. Finally, such tiled encoding allows for more fine-grained RDO as the data can now be reordered per tile instead of over the entire model. Tiled encoding furthermore allows for the decoding step per resolution to benefit from parallelization as well.

In the end, the overall conclusions of this work are given in Chapter 5.

## 1.4 Publications

The research described in this dissertation resulted in several publications, both in international journals and in international conference proceedings.

### 1.4.1 Publications in International Journals

1. J. El Sayeh Khalil, A. Munteanu, L. Denis, P. Lambert, and R. Van de Walle. *Scalable Feature-Preserving Irregular Mesh Coding*. Computer Graphics Forum, 36(6):275–290, September 2017.

2. J. El Sayeh Khalil, A. Munteanu, and P. Lambert. *Scalable Irregular Mesh Coding with Interactive ROI Support*. IEEE Transactions on Circuits and Systems for Video Technology, accepted with minor revisions.

### 1.4.2 Publications in International Conferences

1. J. El Sayeh Khalil, A. Munteanu, and P. Lambert. *3D Mesh Coding with Predefined Region-of-Interest*. In Proceedings of the 24th IEEE International Conference on Image Processing (ICIP), pages 1422–1426, Beijing, China, 17–20 September 2017.

2. J. El Sayeh Khalil, A. Munteanu, and P. Lambert. *Rate-Distortion Optimized Wavelet-based Irregular Mesh Coding*. In Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP), volume 1, pages 212–219, Porto, Portugal, 27 February – 1 March 2017.

3. J. El Sayeh Khalil, I. Saenen, P. Lambert, and R. Van de Walle. *Extreme Asset Simplification and the Preservation of Visual Appearance*. In Proceedings of the 9th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing (CGVCVIP), pages 155–162, Las Palmas de Gran Canaria, Spain, 22–24 July 2015.

4. I. Saenen, J. El Sayeh Khalil, P. Lambert, and R. Van de Walle. *Path Caching in Real-Time Strategy Games*. In Proceedings of the 9th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing (CGVCVIP), pages 147–154, Las Palmas de Gran Canaria, Spain, 22–24 July 2015.

5. J. El Sayeh Khalil, I. Saenen, G. Deglorie, P. Lambert, and R. Van de Walle. *Transforming Complex Animation Film Assets to 3D Game Assets Through the Use of a Voxel Model*. In Poster Proceedings of the 30th Spring Conference on Computer Graphics (SCCG), pages 24–27, Smolenice Castle, Slovakia, 28–30 May 2014.

6. J. El Sayeh Khalil, A. Demeulemeester, K. Samyn, P. Lambert, and R. Van de Walle. *Real-Time Semi-Procedural Crowd Animation*. In Proceedings of the 15th IEEE International Workshop on Multimedia Signal Processing (MMSP), pages 516–517, Pula (Sardinia), Italy, 30 September – 2 October 2013.

# 2
# Mesh Coding

Until today, triangle meshes have been the main representation of 3D models for *real-time rendering*. A mesh is often seen as the combination of both *geometry* information, i.e., the positions of sample points on a surface called vertices, and *connectivity* or *topology*, i.e., the edges (and consequently the triangles) between these vertices. Such a triangle mesh can be represented by a list of vertices and a list of triangles. Denote by $n_v$ the amount of vertices, and by $n_t$ the amount of triangles of a mesh $M$. In general, each vertex position is stored using three coordinates of $Q$ bits. In the remainder of this dissertation, storage is presented in a normalized way by presenting *bit rates*, i.e., the amount of bits required *per vertex*. This is also the convention in literature. Hence, the resulting geometry can be stored at a bit rate of $3Q$ bpv, a rate which is independent of the model complexity.

To allow for indexing all vertices, indices need to be at least $\lceil \log_2 n_v \rceil$ bit large, with $\lceil \cdot \rceil$ the ceiling function which rounds values up to the nearest integer value. Hence, each triangle is represented using $3\lceil \log_2 n_v \rceil$ bits: larger meshes require more bits for indexing all vertices. One can prove that $n_t \approx 2n_v$ for large meshes, hence the connectivity rate is $6\lceil \log_2 n_v \rceil$ bpv. Summarized, this results in a theoretical raw storage size $B$ of:

$$B_{\text{raw}} = 3Qn_v + 3\lceil \log_2(n_v) \rceil n_t \approx 3Qn_v + 6\lceil \log_2(n_v) \rceil n_v, \tag{2.1}$$

or, equivalently, a bit rate $R_{\text{raw}}$ of:

$$R_{\text{raw}} = \frac{B_{\text{raw}}}{n_v} \approx 3Q + 6\lceil \log_2(n_v) \rceil, \tag{2.2}$$

where the first term represents the geometry information, while the second term represents the connectivity information.

Table 2.1 presents the required bit rates and storage sizes using $Q = 32$, i.e., using the single-precision floating point format for vertex coordinates. The connectivity is assumed to be optimally stored as given by the second term of Equation 2.1; in practice, each index will often be aligned to a number of *bytes* instead of bits.

| vertices | | | rate | size | | |
|---|---|---|---|---|---|---|
| 513 | $\sim$ | 1 024 | $156 = 60_C + 96_G$ bpv | 10 KB | $\sim$ | 20 KB |
| 8 193 | $\sim$ | 16 384 | $180 = 84_C + 96_G$ bpv | 180 KB | $\sim$ | 360 KB |
| 65 537 | $\sim$ | 131 072 | $198 = 102_C + 96_G$ bpv | 1.6 MB | $\sim$ | 3.2 MB |
| 524 289 | $\sim$ | 1 048 576 | $216 = 120_C + 96_G$ bpv | 14 MB | $\sim$ | 28 MB |

*Table 2.1: Raw mesh storage rates and sizes using $Q = 32$ bits to quantize vertex coordinates. The rate is split up into the connectivity part "C" and the geometry part "G".*

**Practical Examples** Equation 2.2 gives the optimal rates for uncompressed representations and will be referred to in this dissertation when discussing the rate required for any uncompressed mesh. Nonetheless, in practice higher rates are observed. Instead of an exact amount of quantization bits and the optimal amount of bits per vertex index, binary representations align these sizes to a number of *bytes*. Furthermore, additional signaling is required to differentiate between several topological, geometrical and surface material attributes. Finally, human-readable formats are often used, which allow for easy modifications but further increase the bit rate.

Table 2.2 gives an overview of actual rates obtained by storing the connectivity and geometry information of several meshes in conventional 3D file formats using *Meshlab*. The *.ply* format reduces the additional signaling to a minimum by adhering to a strict order in which the information is stored, requiring only the *amount* of vertices and faces to be specified in the file header. Observe that the binary *.ply* format accurately depicts the most common uncompressed bit rate in practice: each vertex is stored using $4$ bytes, while each face is stored using $4$ bytes per vertex index and an additional byte to indicate the amount of vertices forming the face. As the meshes are triangle meshes, this results in a rate of

$$R_{\text{ply}} \approx 3 \times 32 + 2(3 \times 32 + 8) = 304. \tag{2.3}$$

If only triangular faces are allowed, the $8$ bits for indicating the face degree can be omitted, resulting in $288$ bpv. This is the optimal bit rate for uncompressed representations *in practice*, allowing for models with up to $4 \times 10^9$ vertices to be represented.

| model (vertices) | .obj size (rate) | .ply.asc size (rate) | .ply.bin size (rate) |
|---|---|---|---|
| epcot (770) | 45.6 KB (485 bpv) | 36.6 KB (390 bpv) | 28.7 KB (306 bpv) |
| bimba (8 857) | 584 KB (541 bpv) | 555 KB (514 bpv) | 328 KB (304 bpv) |
| rabbit (67 039) | 5.04 MB (631 bpv) | 4.19 MB (525 bpv) | 2.42 MB (304 bpv) |
| turbine (882 954) | 71.6 MB (681 bpv) | 55.5 MB (528 bpv) | 31.9 MB (304 bpv) |

*Table 2.2: Mesh storage in practice. Example mesh storage rates and sizes for the .obj file format, the human-readable .ply file format and the binary .ply format.*

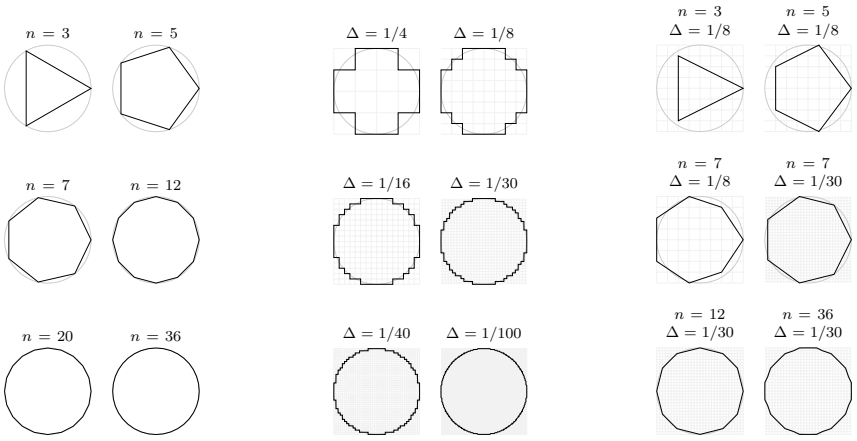
This chapter will first discuss issues related to the processes of surface sampling and quantization for digitizing continuous surfaces in Section 2.1. Then, after an overview of some relevant mesh properties in Section 2.2, a discussion on mesh distortion metrics and the comparison of mesh codecs is given in Section 2.3. Finally, a short overview of mesh compression is given. Single-rate mesh compression, where only a single bit rate is available for decoding an entire mesh, is discussed in Section 2.4.1; scalable mesh compression, which allows for decoding progressively more accurate reconstructions, is discussed in Section 2.4.2.

## 2.1 Sampling and Quantization

Before any digital processing can be performed, a surface $\mathcal{S}$ needs to be digitized to obtain a mesh $M$, as data needs to be stored on, and transmitted over, today's digital media. Contrary to the real world which is a *continuous* space (at least up to some quantum level), the digital world does not allow such freedom: we can only store a limited amount of bytes, and we can only transmit a limited amount of bits per second. An important question in signal processing then becomes: "what limited amount of information allows for reconstructing the most accurate representation of a continuous, real-world data set?"

Any analogue-to-digital conversion encompasses two important steps: first, the parametric domain needs to be discretized in a process called *sampling*. Instead of considering all points, only a limited amount of samples can be represented. These form the vertices of the mesh $M$. The remainder of the surface $\mathcal{S}$ is approximated by interpolating between the samples. Furthermore, these samples cannot be stored at an infinitesimal precision: a quantization step discretizes the range of the signal, allowing only a limited amount of sample positions where vertices can be located.

Hence, any digitization of a continuous signal introduces an error; denote this error as $\epsilon$. This error is caused by three factors [1]: acquisition noise $\epsilon_A$, sampling noise $\epsilon_S$ and quantization noise $\epsilon_Q$. Acquisition noise $\epsilon_A$ emerges because a surface $\mathcal{S}$ cannot be approximated more accurately than defined by the acquisition devices. For instance, one cannot model an object at micrometer accuracy if




*(a) Sampled* *(b) Quantized* *(c) Digitized*

*Figure 2.1: Circle approximation: sampling and quantization. Storing information to reconstruct a surface requires* sampling*, i.e., only considering a limited amount of points, and* quantization*, i.e., considering only a limited number of possible values for the samples. Sampling is depicted in (a): given the number of samples $n$, the approximated surface is obtained by linear interpolation between neighboring samples. Quantization is depicted in (b): an infinite amount of samples is considered, but the samples are rounded off to the nearest discretized value, i.e., the nearest crossing of two grid lines which lie a distance $\Delta$ apart. Finally, sampling and quantization are both required to allow for a finite representation of a surface. This is depicted in (c).*

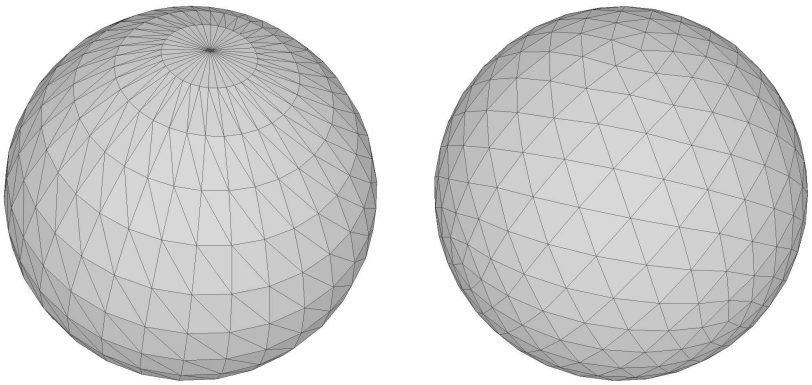
the measuring tools only provide millimeter-accurate results. Sampling noise $\epsilon_S$ originates from the *linear* interpolation between the samples in $M$ where the original surface $\mathcal{S}$ varies smoothly. Finally, quantization noise $\epsilon_Q$ is introduced because only a finite set of positions can be represented with a limited amount of bits.

### 2.1.1 Sampling

Sampling is the process of limiting the domain for which values are observed. Consider for instance the approximation of a circle around the origin $(0, 0)$ with radius $r$ in 2D, described by the parametric curve

$$\begin{aligned} \mathcal{C} : [0, 2\pi] \subset \mathbb{R} &\to \mathbb{R}^2; \\ \theta &\mapsto (x, y) = \mathcal{C}(\theta), \\ \text{with } &\begin{cases} x = r\cos(\theta) \\ y = r\sin(\theta) \end{cases} . \end{aligned} \tag{2.4}$$

A uniform sampling, as depicted in Figure 2.1a selects vertices spread out



*(a) UV sampling* *(b) Uniform sampling*

*Figure 2.2: Sampling a sphere.*

evenly over the circle. In this case, the amount of samples is directly related to the approximation quality: the sampling error $\epsilon_S$ is given by the area between the black and gray lines in Figure 2.1a. The required amount of sampling for a sufficient approximation depends on the scale at which the approximation is observed. Given $n = 20$ samples, the circle looks round until observed from a short distance where it again looks faceted. With $n = 36$ samples, which corresponds with a sample per $10°$, the resulting polygon looks nearly indistinguishable from the continuous circle. There is a threshold for the angles within the polygon, under which the human visual system will be unable to see corners; however, the above example shows that lower sampling densities often suffice, depending on the display size and the distance at which objects are observed.

Similar to how a curve (such as the circle in Figure 2.1) can be approximated using short and/or long straight strokes, a surface can be approximated using small and/or large polygons. Continuing on the sphere described by Equation 1.2, instead of considering *all* possible $\phi$ and $\theta$, one could rotate in steps of $10°$, resulting in only $36 \times 18 = 648$ samples for which we need a value. This is illustrated in Figure 2.2a. A linear interpolation between the samples allows for approximating the remaining values. The samples, called *vertices*, define the *geometry* of the model, while neighborhood information of the samples, which indicates how linear interpolation should be performed, defines the *connectivity* of the model. This neighborhood information is represented as *edges* between the vertices, and together they form polygons of a *polygon mesh* approximation of a surface.

As Figure 2.2a depicts, a perfectly regular sampling in the domain (in this case in perfect steps of $10°$ in both directions) does not necessarily distribute the

samples evenly along the surface. Observe that, near the poles, the samples are closer together. Moreover, at both poles themselves 36 samples have the exact same position. Figure 2.2b shows an alternative, more uniform sampling[1] where the sampling error will be spread more evenly over the entire surface.

### 2.1.2 Coordinate Quantization

Sampling alone does not suffice for a digital storage. The value of even a single sample can be any value of an infinite continuous set. Scalar quantization is the process of representing the infinite set of real numbers $\mathbb{R}$ by a finite set of symbols $V = \{v_0, v_1, v_2, ..., v_{n-1}\}$. After quantization, each of these symbols can now be represented using a binary representation. Using $b$ bits, one can represent $2^b$ possible values. For instance, using $4$ bits, one can map $v_0$ to $\texttt{0000}$, $v_1$ to $\texttt{0001}$, $v_2$ to $\texttt{0010}$, ..., and $v_{15}$ to $\texttt{1111}$. This is related to the *precision* of the samples: a fine quantization signifies that samples closer together can still be differentiated as a higher amount of bits is spent per sample; inversely, a coarser quantization represents nearby samples as being located at the exact same position.

Vector quantization is the process of representing the vectors in $\mathbb{R}^k$, i.e., $k$-tuples such as the 3-tuples $(x, y, z)$, by a finite set of symbols. In this dissertation however, vertex positions are quantized by employing scalar quantization on each of the three components independently.

Considering once more the circle defined by Equation 2.4. Both the $x$ and $y$ component of each sample can take *any* real value between $-r$ and $+r$. Quantization reduces these possible values, for instance given by the uniform quantization as depicted in Figure 2.1b where the difference between two subsequent values is given by $\Delta$. Despite sufficient samples, if the quantization step $\Delta$ is too big, clear blocking artifacts appear.

**Embedded Quantizers** In general, the set of real numbers $\mathbb{R}$ can be arbitrarily partitioned into $n$ cells, where each cell $C^i$ is associated with a symbol $q \in V$. Every value $X \in C^i$ is mapped to the same $q = Q(X)$ and after dequantization the symbol $q$ is mapped to a single value $\hat{X} = Q^{-1}(q)$. This is shown in Figure 2.3. Due to the arbitrary partitioning into cells $C^i$, several quantization granularities are not necessarily related.

For quality scalability, an important class of quantizers are *embedded quantizers*. As the name suggests, in this case the quantization cells of a finer quantizer are embedded within the cells of a coarser quantizer. Let $p \in \mathbb{N}$ denote the quantizer number when ordered from finer to coarser quantizers. If quantizer $Q_p$ has $n$ quantization cells $C_p^i$ and quantizer $Q_{p-1}$ has $k.n$ quantization cells $C_{p-1}^j$, then each $C_p^i$ embeds $k$ cells $C_{p-1}^j$. Consequently, this construction

[1] In fact, it is a subdivided *icosahedron*.

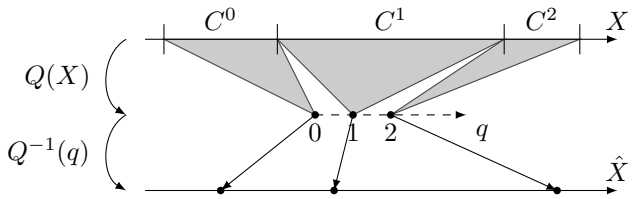



*Figure 2.3: Quantization $Q(X)$ and dequantization $Q^{-1}(q)$.*

allows for identifying the quantization cell for quantizer $Q_{p-1}$ by identifying the quantization cell for quantizer $Q_p$, followed by identifying the correct cell of the $k$ embedded quantization cells. This naturally leads to an embedded, truncatable data stream. Figure 2.4 illustrates this: each of the $n = 3$ cells $C_1^i$ embeds $k = 3$ cells $C_0^j$. Once $C_1^i$ is identified, the $k = 3$ possible superscripts for $C_0^j$ are found via $j = 3i, j = 3i + 1$ and $j = 3i + 2$.

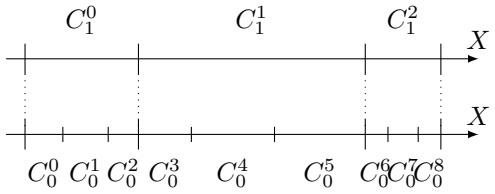



*Figure 2.4: Embedded quantizers.*

**Embedded Deadzone Quantizers** For binary representations, a popular choice is to have $k = 2$, i.e., each finer quantizer arbitrarily subdivides each coarser quantization cell into two finer cells, which can be identified by a single bit. An interesting category of embedded quantizers is given by *embedded deadzone quantizers*, where each coarser quantization cell is split into two *equal* finer quantization cells, and where a so-called *deadzone* around zero can be treated differently. In general, an embedded deadzone quantizer quantizes $X$ to the integer value

$$q_p = Q_p(X) = \begin{cases} \text{sign}(X) \cdot \lfloor \frac{|X|}{2^p\Delta} + \frac{\xi}{2^p} \rfloor & \text{if } \frac{|X|}{2^p\Delta} + \frac{\xi}{2^p} > 0 \\ 0 & \text{otherwise} \end{cases} , \tag{2.5}$$

where $\xi < 1$ determines the width of the deadzone and $2^p\Delta$ determines the quantization cell width. Only the deadzone of $Q_p$ deviates from this $2^p\Delta$ cell width: it has a width of $2(2^p\Delta - \xi\Delta)$, i.e. encompassing values $X \in (-2^p\Delta + \xi\Delta, 2^p\Delta - \xi\Delta)$. For $X > 0$, this is depicted in Figure 2.5 with $\xi = 0, \xi = 0.5$ and $\xi = -1$. An example quantization with $\xi = 0$ is given in

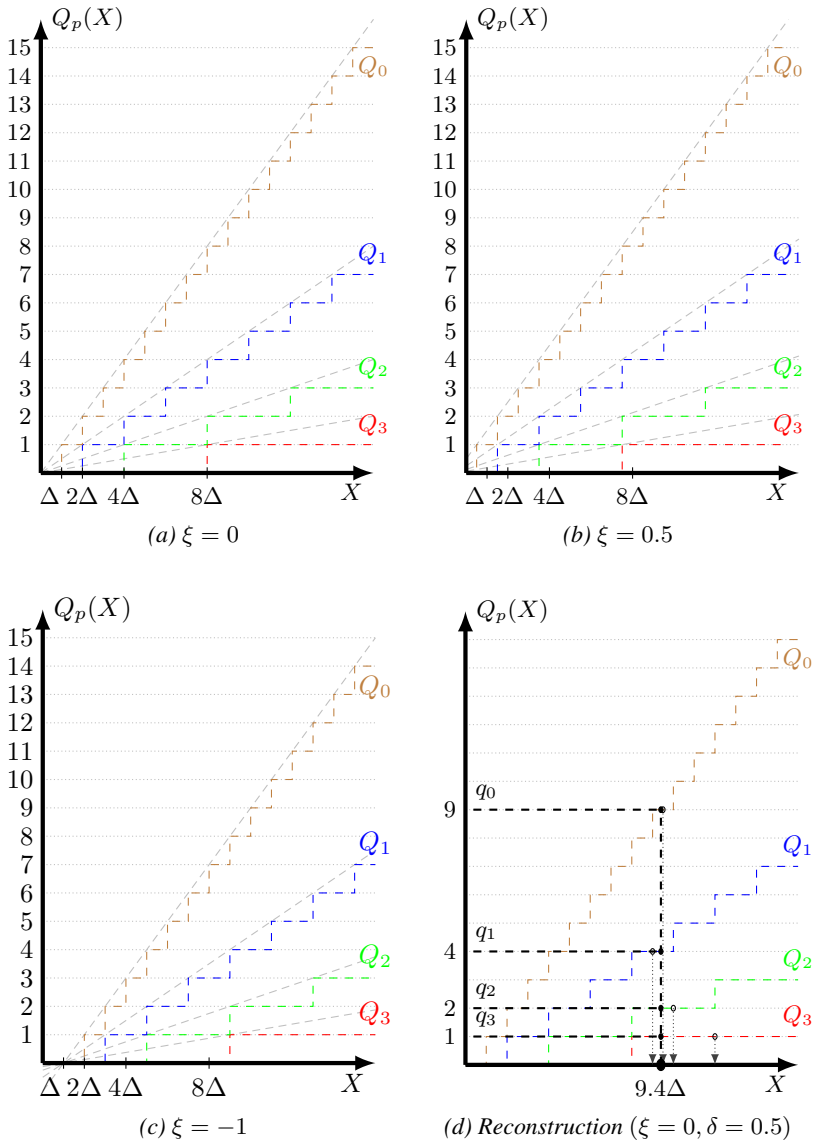



*Figure 2.5: Embedded deadzone quantization. Three examples are given for positive values, each with four embedded quantizers $\{Q_3, Q_2, Q_1, Q_0\}$. It is clear that the $\xi$ variable alters the deadzone around zero. Looking at $Q_0$, $\xi = 0$ results in a deadzone within $(-\Delta, \Delta)$, with $\xi = 0.5$ the deadzone shrinks to $(-\Delta/2, \Delta/2)$ and with $\xi = -1$ the deadzone grows to $(-2\Delta, 2\Delta)$. (d) illustrates the quantization and reconstruction of $X = 9.4\Delta$. The quantized values are indicated by $q_3, q_2, q_1$ and $q_0$ and the reconstructed values are indicated by the open circles.*

Figure 2.5d: the value $X = 9.4\Delta$ is quantized, resulting in

$$q_3 = Q_3(X) = 1, \tag{2.6}$$
$$q_2 = Q_2(X) = 2, \tag{2.7}$$
$$q_1 = Q_1(X) = 4, \tag{2.8}$$
$$q_0 = Q_0(X) = 9. \tag{2.9}$$

The figures show the embedding as well. $q_3$ can take two values, 0 or 1. Knowing $q_3$ leaves only two of the four possible values for $q_2$, i.e., 2 or 3 etc. This allows for a binary representation where each additional bit represents the value of a more fine-grained quantizer.

Conversely, the reconstruction is defined by

$$Q_p^{-1}(q_p) = \begin{cases} 0, & q_p = 0 \\ \text{sign}(q_p)\Big(|q_p| - \frac{\xi}{2^p} + \delta\Big)2^p\Delta, & q_p \neq 0 \end{cases}, \tag{2.10}$$

where $0 \leq \delta < 1$ determines which value within a quantization cell is used for reconstructing $\hat{X}$. With $\delta = 0.5$, values are reconstructed midway within the quantization cells. For the coarsest quantization $Q_3$, with $\xi = 0$ this signifies that each original value $X$ is approximated by either 0 or $12\Delta$. The finer quantization $Q_2$ allows for an approximation by $0, 6\Delta, 10\Delta$ or $14\Delta$. Similar reasoning allows for more accurate approximations by the even finer-grained quantizers $Q_1$ and $Q_0$.

This is again illustrated in the example of Figure 2.5d, for the given $X = 9.4\Delta$. The reconstructed values are

$$Q_3^{-1}(q_3) = 12\Delta, \tag{2.11}$$
$$Q_2^{-1}(q_2) = 10\Delta, \tag{2.12}$$
$$Q_1^{-1}(q_1) = 9\Delta, \tag{2.13}$$
$$Q_0^{-1}(q_0) = 9.5\Delta. \tag{2.14}$$

In this example, the distortion decreases for more fine-grained quantization. However, for $k = 2$ and $\delta = 0.5$ this does not always hold. Consider for instance $X = 9.9\Delta$. The same reconstruction values as above are obtained after quantization and reconstruction; yet, $Q_2^{-1}(q_2) = 10\Delta$ is now closer to the original value $X$ than $Q_0^{-1}(q_0) = 9.5\Delta$ is.

**Successive Approximation Quantization** In this dissertation, *successive approximation quantization (SAQ)* is used, which is a specific configuration of the *embedded deadzone quantizers*: it is a particular instance for which the deadzone width of quantizer $Q_p$ is twice as wide as the other cells of $Q_p$, that is, $\xi = 0$. This can be implemented via thresholding, by using thresholds of the form $\tau_{p-1} = \tau_p/2$

[2]. The binary value which is *appended* by each finer-grained quantizer $Q_p$ is given by:

$$\left\lfloor \frac{X}{\tau_p} \right\rfloor \mod 2. \tag{2.15}$$

In other words, a binary $0$ or $1$ can be assigned by determining whether the cell index is either odd or even. Additionally, the coarsest threshold $\tau_{p_{\max}}$ needs to be such that no $X$ surpasses $2\tau_{p_{\max}}$, otherwise so-called *overflow* issues arise as the quantizers do not suffice for representing all required values. The binary representations can be verified by Figures 2.5a and 2.5d, where $\tau_{p_{\max}} = \tau_3 = 8\Delta$.

$$q_3 = 1 =_b \quad \mathtt{1} \quad (\tau_3 = 8\Delta), \tag{2.16}$$
$$q_2 = 2 =_b \quad \mathtt{10} \quad (\tau_3 = 4\Delta), \tag{2.17}$$
$$q_1 = 4 =_b \quad \mathtt{100} \quad (\tau_3 = 2\Delta), \tag{2.18}$$
$$q_0 = 9 =_b \mathtt{1001} \quad (\tau_0 = \Delta). \tag{2.19}$$

For $X = 9.4\Delta$ in Figure 2.5d, $Q_3$ adds 1, $Q_2$ adds 0, $Q_1$ adds 0 and $Q_0$ adds 1, resulting subsequently in the conventional binary representations for the decimal numbers $q_3 = 1, q_2 = 2, q_1 = 4$ and $q_0 = 9$.

### 2.1.3 Numerical Example

Despite the quantization error $\epsilon_Q$ introduced by digitization, this error should not be considered as loss when $\epsilon_Q \approx \epsilon_A + \epsilon_S$; after all, values which are closer to the acquisition data are not guaranteed to be closer to the actual physical object.

For instance, consider a volume of $1\,m \times 1\,m \times 1\,m$ which holds a single object. If we employ a uniform 16 bit quantization in each dimension, we can represent each dimension in steps of $1/2^{16}\,m = 1/65\,536\,m = 15.3\,\mu m$. Hence, the surface of the object can be accurately represented up to details of several micrometers large. Similarly, a volume of $1\,km \times 1\,km \times 1\,km$ is represented at $15\,mm$ accuracy.

Consider another numerical example. Assume a terrain of a certain square area $A$ represented as a height map, i.e., represented by points $(x, y, h)$ where each point $(x, y)$ relates to a height $h$. If both horizontal dimensions are uniformly quantized using $b$ bits, there are $2^b \times 2^b = 2^{2b}$ possible values for $(x, y)$. Given the area $A$, the terrain can be accurately represented up to $A/2^{2b}$. Table 2.3 gives the required amount of bits to represent an area at $1\,m^2$ and $1\,cm^2$ accuracy.

| | Area | accuracy | | |
|---|---|---|---|---|
| | | $1\,\mathrm{m}^2$ | $1\,\mathrm{cm}^2$ | $1\,\mathrm{mm}^2$ |
| Ghent | $156.2\,km^2$ | 14 bit | 20 bit | 24 bit |
| East-Flanders | $13\,522\,km^2$ | 17 bit | 24 bit | 27 bit |
| Belgium | $30\,528\,km^2$ | 18 bit | 25 bit | 28 bit |
| Europe | $10\,180\,000\,km^2$ | 22 bit | 29 bit | 32 bit |
| Earth | $510\,100\,000\,km^2$ | 25 bit | 32 bit | 35 bit |

*Table 2.3: Quantization required for specific area accuracy.*

The area of Ghent is $156.2\,km^2$, so to ensure that each sample covers $1\,m^2$, $156\,200\,000$ samples are needed, which is possible using $b = 14$ bits. After all, 14 bits per dimension allow for $2^{28} \approx 268\,000\,000$ samples. Using similar reasoning, if East Flanders, Belgium, Europe and the Earth surface were square, these areas can be represented at $1\,m^2$ accuracy as illustrated in Table 2.3. Furthermore, changing the accuracy to $1\,cm^2$ adds $\log_2(100) = 6.6$ bits per dimension to this, increasing to $1\,mm^2$ adds $\log_2(1000) = 9.9$ bits. Hence, using $b = 32$ bits, which is the conventional storage size for integers, suffices to sample Europe at $1\,mm^2$ accuracy, assuming that it was perfectly square.

Consider finally the height values $h$. The highest mountain on earth, the Mount Everest, towers $8\,848\,m$ above sea level, while the lowest known point, Challenger Deep, is found $11\,034\,m$ below sea level. To represent this range of $19\,882\,m$ at $1\,m$ accuracy, $b = 15$ bits suffice, while $1\,mm$ accuracy requires $b = 25$ bits.

While these are very crude approximations, raising many questions and remarks, they justify the conventional view that sampling uniformly using $12 \sim 16$ bits, i.e., using $2^{12} = 4\,096 \sim 2^{16} = 65\,536$ samples per dimension suffices to accurately represent most models. An additional 16 bits per dimension, which allows for $64\,000$ times more accuracy per dimension, is often unnecessary. Table 2.4 repeats Table 2.1, now using $Q = 12$ bit quantization.

| vertices | | | rate | size | | |
|---|---|---|---|---|---|---|
| 513 | $\sim$ | 1 024 | $96 = 60_C + 36_G$ bpv | 6.16 KB | $\sim$ | 12.3 KB |
| 8 193 | $\sim$ | 16 384 | $120 = 84_C + 36_G$ bpv | 123 KB | $\sim$ | 246 KB |
| 65 537 | $\sim$ | 131 072 | $138 = 102_C + 36_G$ bpv | 1.13 MB | $\sim$ | 2.26 MB |
| 524 289 | $\sim$ | 1 048 576 | $156 = 120_C + 36_G$ bpv | 10.2 MB | $\sim$ | 20.4 MB |

*Table 2.4: Raw mesh storage rates and sizes, using $Q = 12$ bits to quantize vertex coordinates. The rate is split up into the connectivity part "C" and the geometry part "G".*

## 2.2 Properties of Meshes

Before going deeper into mesh compression, some relevant topics are discussed regarding mesh representations. First, neighborhood terminology is given in Section 2.2.1, followed by a discussion on mesh regularity in Section 2.2.2 and ending with some additional mesh properties in Section 2.2.3.

### 2.2.1 Neighborhood Information

In order to understand how vertices, edges and faces are related, some terminology on topological neighborhoods is given.

**Neighboring vertices**
Two vertices are neighbors if an edge connects them. In this dissertation, two indexed notations for vertices will be employed: a vertex can either be indicated using its absolute index as $v^i$, or via a relative, neighbor index as $v^{j,k}$, i.e., the $k^{\text{th}}$ neighbor of the $j^{\text{th}}$ vertex.

**Valence/vertex degree**
The amount of neighbors a vertex has; for a triangle mesh, this is 6 on average. The valence of vertex $v$ will be denoted as $\nu(v)$ in this dissertation.

**Neighboring faces**
Two faces are neighbors if they share an edge.

**Neighboring vertices of an edge**
The two vertices which define an edge.

**Neighboring faces of an edge**
The faces which have the edge in their border.

**Boundary edge**
An edge is a boundary edge if it only has a single neighboring face.

**Boundary vertex**
A vertex neighboring a boundary edge.

**Boundary face**
The face neighboring a boundary edge.

### 2.2.2 Mesh Regularity

The most important property within this dissertation is mesh regularity. On the one hand, regularity results in predictability which benefits the coding performance. However, as will be detailed next, on the other hand regularity limits the approximation quality of the representation.

**Geometry, Parameter and Connectivity Information** In many works, including this dissertation, the information to represent a model is split into the geometry of the model, which determines the positions of the vertices, and the connectivity of the model, determining how faces are formed between vertices.

However, as is illustrated by Khodakovsky et al. [1], the information for the positions of vertices can be further split into *geometry* and *parameter* information. This is based on the idea that any *tangential* displacement of vertices along the surface does not change the approximated surface geometry. Such parameter information is related to *where* samples are located on a surface, and as such altering this information affects the sampling noise $\epsilon_S$. This is illustrated in Figure 2.6 with a 2D example. Figure 2.6a depicts an original curve, and two approximations are given in Figures 2.6b and 2.6c, representing the same geometry information but differing in parameter information. Assuming a perfect acquisition and no quantization noise (i.e., $\epsilon_A = \epsilon_Q = 0$), it is clear that the reconstruction error due to sampling depends on the parameter information. For densely-sampled surfaces, the effect of parameter information on the reconstruction quality becomes negligible.

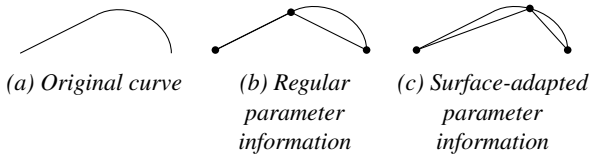
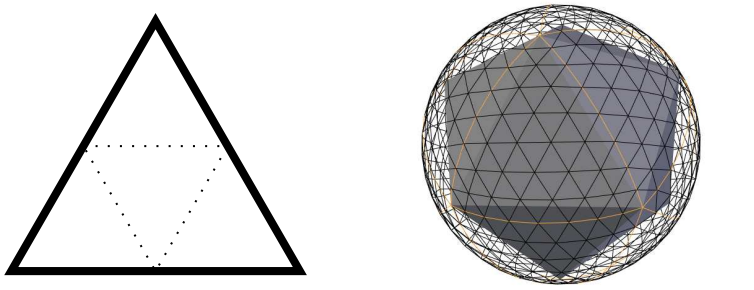



*Figure 2.6: Parameter information.*

**Explicit vs Implicit Information** A semi-regular mesh has a regular *connectivity* over large portions of its surface, with irregularity only found around a limited amount of so-called *extra-ordinary* vertices. In practice, such a semi-regular mesh is obtained by starting with a coarse approximation of a mesh, which is called the *base mesh*, and by then iteratively subdividing its faces, for instance using $1$-to-$4$ subdivision which subdivides each triangle into four new triangles as shown in Figure 2.7a. Figure 2.7b then shows the sphere of Figure 2.2b together with the underlying base mesh. The vertices of the original icosahedron can still be recognized as the extra-ordinary vertices with valence $5$ whereas the other, regular vertices have valence $6$. Once all information on the extra-ordinary vertices is known, i.e. once the base mesh is known, no additional connectivity information is required: any decoder can reconstruct the correct mesh connectivity if it is given the base mesh.

This immediately determines a first of three classes of semi-regular meshes. *Subdivision surfaces* only require a base mesh to be encoded. No additional

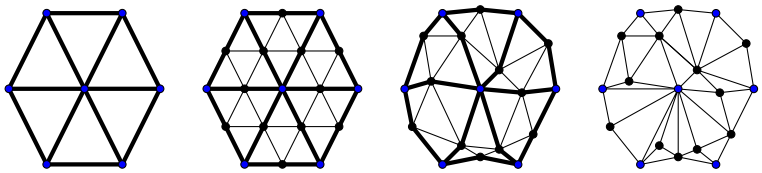


*(a)* 1*-to-*4 *subdivision* *(b) Icosahedron after three subdivisions*

*Figure 2.7: Semi-regular meshes.*

geometry or parameter information is given, and higher-resolution meshes are obtained through the implicit subdivision rules. Secondly, if vertices created by the subdivision procedure are only moved along the surface normal, only geometry information is added per subdivision step. This actually alters the geometry of the model but leaves parameter information unaltered. *Normal meshes* [3] are based on this idea. And finally, in general *semi-regular meshes*, the vertices obtained after subdivision can be perturbed tangentially over the surface of the sphere.

On the other hand, an *irregular mesh* is no longer implicitly constructed starting from a base mesh. In this case, connectivity information has to be explicitly provided in addition to the geometry and parameter information.



*(a) Base mesh* *(b) Subdivision surface* *(c) Semi-regular mesh* *(d) Irregular mesh*

*Figure 2.8: Mesh regularity. As is described in [1], a mesh can in fact be seen as composed of geometry, parameter and connectivity information. All meshes share the same geometry information, yet (b) is represented solely based on the subdivision of (a), the (c) has additional parameter information, and (d) also has connectivity information which makes it an irregular mesh.*

These ideas are illustrated in Figure 2.8. A low-resolution base mesh is shown in Figure 2.8a, and a resulting subdivision surface is depicted in Figure 2.8b. Parameter information is added to obtain the general semi-regular mesh shown in Figure 2.8c. Observe that the same (implicit) connectivity is seen as in Figure 2.8b; the vertices are merely displaced within the plane without altering the geometry.

Finally, Figure 2.8d depicts the same plane, but now the connectivity does not show any relationship with the connectivity at the lower resolution (Figure 2.8a). Table 2.5 summarizes this.

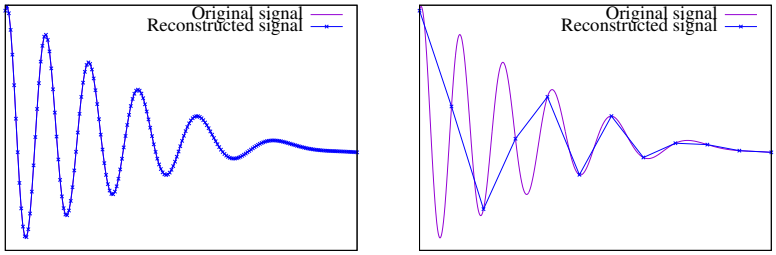
| type of mesh | Connectivity | Parameter | Geometry |
|---|---|---|---|
| subdivision surface meshes | implicit | implicit | implicit |
| normal meshes | implicit | implicit | *explicit* |
| semi-regular meshes | implicit | *explicit* | *explicit* |
| irregular meshes | *explicit* | *explicit* | *explicit* |

*Table 2.5: Implicit vs explicit information. In addition to a possible base mesh, this table depicts possible additional information that can be provided.*

**Mesh Coding** To process irregular meshes, any *semi-regular mesh codec* converts irregular meshes to semi-regular ones in a preprocessing step [1, 3, 4], referred to as *remeshing*. This remeshing step finds an appropriate base mesh and approximates the original surface via iterative subdivision. In applications where original sampling and its connectivity information are relevant, such a remeshing step is considered as a lossy step. After remeshing, normal meshes and other semi-regular coding schemes exploit the fact that connectivity is predetermined by the base mesh, while during remeshing they reduce the parameter information as much as possible, aiming for better coding performance by reducing irregularity and increasing predictability.

However, in this dissertation *irregular mesh coding* is considered, where arbitrary faces can be formed between the vertices as in Figure 2.8d. While it is challenging for an irregular mesh codec to achieve coding performances similar to those of semi-regular ones due to the added cost of storing connectivity information explicitly, its main advantage over semi-regular codecs is that irregular meshes allow for better approximations using fewer vertices and faces, and consequently using less memory for representing the intermediate approximations.

Indeed, it is a well-known fact that regular sampling, and by extension semi-regular meshes with minimal parametric information, require higher sampling densities compared to irregular sampling to preserve all details. A 2D example is shown in Figure 2.9, which shows two approximations of the purple curve which has high-frequency information on the left side and low-frequency information on the right side. The approximations illustrate high-density and low-density regular sampling. The high-density sampling in Figure 2.9a allows for accurately reconstructing all information in the purple curve. However, many samples are used on the right half of the curve which no longer necessarily aid in improving the reconstructed quality. Using fewer samples in the right half, such as depicted in Figure 2.9b suffices for representing these details. However, in this case the high-frequency information on the left side of the curve is lost. This effect is




*(a) High sampling density* *(b) Low sampling density*

*Figure 2.9: Sampling densities.*

also found in *semi-regular* mesh representations. Similar to the 2D example of Figure 2.9a, sufficient vertices need to be defined near high-frequency details to ensure sufficient preservation in these regions. Consequently, this results in denser vertex distributions in regions where this is not required. Irregular mesh representations tackle this, allowing for a better surface-adapted distribution of vertices and reducing the required amount of memory for rendering.

### 2.2.3 Additional Mesh Properties

A few additional mesh properties are discussed below. Depending on the algorithms used in different codecs, some properties do not allow meshes to be properly processed by specific codecs.

**Manifoldness** An important property of the represented surfaces is *2-manifoldness*: the neighborhood of each point on the surface is homeomorphic to a two-dimensional Euclidean space, that is, each neighborhood is homeomorphic to an open disk, or to a half-disk for points on the border. This is an important requirement for many mesh coding systems as it allows for unambiguously determining neighborhood information of vertices and faces. Figure 2.10 shows an example of a non-manifold edge and a non-manifold vertex.

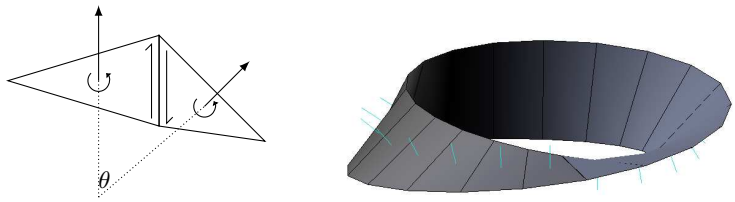
*(a) Non-manifold edge* *(b) Non-manifold vertex*

*Figure 2.10: Non-manifoldness.*

This dissertation also requires manifoldness; after all, any single, monolithic

model that corresponds to a *physical* object must be 2-manifold. However, due to the limited resolution and accuracy of any acquisition system, 2-manifoldness is not guaranteed. Similarly, 2-manifoldness is not necessarily guaranteed in models designed by 3D artists; depending on the applications in which these models are employed, such non-manifoldness is not an issue.

**Orientability** Furthermore, meshes are required to be orientable, which is defined as follows. Firstly, a face can have two orientations; informally, each triangle has a front and a back side, and either one can be on the outside of a surface. This determines the orientation of the surface normal and allows for defining dihedral angles between triangles. In practice, the orientation of a triangle is determined by the order in which its neighboring vertices are traversed when defining the triangle. Two neighboring faces are said to have a compatible orientation if the vertices of the shared edge are traversed in opposite directions within the neighboring triangles. A mesh is then said to be orientable if there exists an arrangement such that each pair of neighboring triangles is compatible. This is again necessary to allow for unambiguously determining neighborhood information. This is illustrated in Figure 2.11.




*(a) Compatible triangles* *(b) Möbius strip*

*Figure 2.11: Orientability. (a) shows two compatible triangles. The orientation determines the surface normal, and the order in which the vertices are defined, is indicated by the arrow around each normal vector. The traversal order of the shared vertices is explicitly indicated, showing the opposite direction and hence the compatibility of both triangles. A well-known example of a surface which is not orientable is the Möbius strip (b).*

**Closed vs Open Surfaces** An open surface as one or more loops of border edges and vertices. This is important as these vertices typically need to be treated differently, due to their neighborhood no longer consisting of a full ring of neighboring vertices. Several approaches are seen in literature to handle surface borders. A first approach is to explicitly handle these vertices differently, which provides the best results but complicates the algorithms. A second approach is to add *virtual* vertices to close such holes, and further handle these surfaces as closed surfaces. The decoder then needs to erase these virtual vertices again. This

leaves the algorithms unaltered but no longer handles borders appropriately. In the implementation for this dissertation, however, borders are not explicitly supported; if border vertices are present, these are forced to be preserved. The next step must be to handle border vertices explicitly, without adding virtual vertices.

**Genus of a Surface** The genus of a surface is related to the number of ways in which a surface can be cut along closed loops without resulting in multiple disconnected parts. Intuitively, this refers to the amount of handles seen in the surface. For instance, a sphere has genus 0, a donut has genus 1, a 3D figure eight has genus 2 and a toy typical fidget spinner has genus 3.

Important considerations are (a) whether a codec supports higher-genus models, and (b) whether this topological genus is preserved. For the codec proposed in the next chapters, the genus can be arbitrarily large, and will be preserved in the base mesh.

## 2.3 Comparing Codecs

Before starting a discussion on mesh compression and mesh approximations, the concepts of distortion measures need to be explored. This is done in Section 2.3.1. Afterwards, Sections 2.3.2 and 2.3.3 discuss how single-rate and multi-rate codecs are compared. The approaches used in the state of the art are detailed, as well as some measures introduced in this dissertation.

### 2.3.1 Mesh Distortions: Measuring Differences

Measuring distortions in 3D requires determining a distance metric. For point-to-point distances, e.g. between points $\mathbf{p}_1 = (x_1, y_1, z_1)$ and $\mathbf{p}_2 = (x_2, y_2, z_2)$, the classical Euclidean distance is often used:

$$d(\mathbf{p}_1, \mathbf{p}_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}. \tag{2.20}$$

The distortion of a reconstructed point is defined as the distance between the original position and the reconstructed position.

Measuring a surface-to-surface distance, however, is less straightforward. Ideally, for closed surfaces an objective metric would measure the volume of the space created between the two surfaces, but this is too computationally complex. Furthermore, such a metric cannot handle open surfaces.

In practice, the point-to-surface distance is used. Given a point $\mathbf{p}$ and a surface $\mathcal{S}$, the point-to-surface distance is defined as the shortest distance between $\mathbf{p}$ and any point $\mathbf{q}$ on the surface $\mathcal{S}$:

$$d(\mathbf{p}, \mathcal{S}) = \inf_{\mathbf{q} \in \mathcal{S}} d(\mathbf{p}, \mathbf{q}). \tag{2.21}$$

The distance from surface $\mathcal{S}_1$ to $\mathcal{S}_2$, $d(\mathcal{S}_1, \mathcal{S}_2)$, is now defined as the largest distance from any point $\mathbf{p}$ on $\mathcal{S}_1$ to $\mathcal{S}_2$:

$$d(\mathcal{S}_1, \mathcal{S}_2) = \sup_{\mathbf{p} \in \mathcal{S}_1} d(\mathbf{p}, \mathcal{S}_2). \tag{2.22}$$

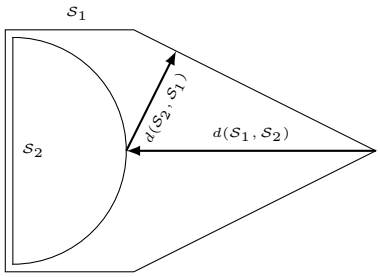



*Figure 2.12: Asymmetric distortion.*

Observe that $d(\mathcal{S}_1, \mathcal{S}_2)$ and $d(\mathcal{S}_2, \mathcal{S}_1)$ are not necessarily equal. This is illustrated in Figure 2.12. Based on this *unidirectional* surface-to-surface distance, the *Hausdorff distance* is defined as:

$$d_H(\mathcal{S}_1, \mathcal{S}_2) = \max\{d(\mathcal{S}_1, \mathcal{S}_2), d(\mathcal{S}_2, \mathcal{S}_1)\}. \tag{2.23}$$

The Hausdorff distance uses the largest distance from a point to a surface. Outliers hence have a large influence on the measured distortion. Alternatively, the point-to-surface distances can be integrated, resulting in, for instance, the *mean absolute deviation (MAD)* and *root mean squared (RMS)* deviation:

$$d_{\text{MAD}}(\mathcal{S}_1, \mathcal{S}_2) = \frac{1}{|\mathcal{S}_1|} \int_{\mathbf{p}} d(\mathbf{p}, \mathcal{S}_2) d\mathcal{S}_1, \tag{2.24}$$

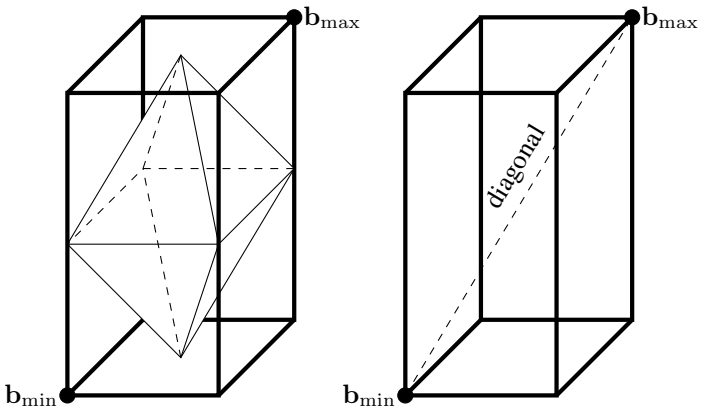
$$d_{\text{RMS}}(\mathcal{S}_1, \mathcal{S}_2) = \sqrt{\frac{1}{|\mathcal{S}_1|} \int_{\mathbf{p}} \big(d(\mathbf{p}, \mathcal{S}_2)\big)^2 d\mathcal{S}_1}. \tag{2.25}$$

Both are again unidirectional, as the distance $d(\mathcal{S}_1, \mathcal{S}_2)$ is not necessarily equal to $d(\mathcal{S}_2, \mathcal{S}_1)$. Similar to the Hausdorff distance, one can define distances using maximal values of the one-sided MAD and RMS deviation in the forward and in the backward direction. In literature, one often encounters the sum and average of the forward and backward distance, i.e.,

$$d_{\text{RMS}}^{\text{sum}} = d_{\text{RMS}}(\mathcal{S}_1, \mathcal{S}_2) + d_{\text{RMS}}(\mathcal{S}_2, \mathcal{S}_1), \tag{2.26}$$

$$d_{\text{RMS}}^{\text{avg}} = d_{\text{RMS}}^{\text{sum}}/2. \tag{2.27}$$

Finally, it is common practice to normalize these distances given the axis-aligned bounding box. Informally, this is the smallest box with its length, width




*(a) Definition* *(b) Diagonal*

*Figure 2.13: Bounding box. The bounding box is defined by the point $\mathbf{b}_{\min}$ and either the point $\mathbf{b}_{\max}$, or the bounding box diagonal $\mathbf{b}_{\max} - \mathbf{b}_{\min}$.*

and height parallel to the $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ axes which encompasses all points $\mathbf{p} \in \mathcal{S}$. This is illustrated in Figure 2.13.

Denote by $\mathbf{p}_x$, $\mathbf{p}_y$ and $\mathbf{p}_z$ the $x$, $y$ and $z$ component of $\mathbf{p}$ respectively. The bounding box can be defined by a *lower bound* $\mathbf{b}_{\min} = (x_{\min}, y_{\min}, z_{\min})$ and an *upper bound* $\mathbf{b}_{\max} = (x_{\max}, y_{\max}, z_{\max})$, such that:

$$x_{\min} = \inf_{\mathbf{p} \in \mathcal{S}} \mathbf{p}_x \text{ and } x_{\max} = \sup_{\mathbf{p} \in \mathcal{S}} \mathbf{p}_x, \tag{2.28}$$

$$y_{\min} = \inf_{\mathbf{p} \in \mathcal{S}} \mathbf{p}_y \text{ and } y_{\max} = \sup_{\mathbf{p} \in \mathcal{S}} \mathbf{p}_y, \tag{2.29}$$

$$z_{\min} = \inf_{\mathbf{p} \in \mathcal{S}} \mathbf{p}_z \text{ and } z_{\max} = \sup_{\mathbf{p} \in \mathcal{S}} \mathbf{p}_z. \tag{2.30}$$

The measured distances are then rescaled according to the bounding box diagonal $d(\mathbf{b}_{\min}, \mathbf{b}_{\max})$.

During the experimental evaluations throughout this dissertation, Hausdorff and RMS distortions are obtained using the *METRO* tool, which is the standard tool in literature for measuring mesh distortions. This tool numerically calculates these distances by sampling the first surface and finding the distances from each sample to the faces of the second surface. More details can be found in [5].

### 2.3.2 Comparing Single-Rate Codecs

Single-rate codecs, which only allow for a single mesh reconstruction, can be compared in two ways.

**Bit Rates** For *lossless* single-rate codecs, i.e. where the original mesh is *losslessly* reconstructed, from a *signal processing* perspective only the final bit rate needs to be evaluated as the reconstructed meshes are identical. The best encoding is the one which best exploits all correlations and hence best reduces the storage size.

**Distortion versus Bit Rate Trade-off** If one or more of the codecs are *lossy*, i.e. the original mesh is not perfectly reconstructed, trade-offs need to be considered. Better quality at a lower rate is obviously preferred over lower quality at a higher rate. However, when a higher rate offers better quality, the actual scenarios in which the codecs must be deployed, must be taken into account, e.g. whether distortion must be minimized or rate must not surpass a specific threshold.

### 2.3.3 Comparing Multi-Rate Codecs

For multi-rate codecs, which allow for reconstructing multiple approximations of an original mesh, the discussion of Section 2.3.2 still applies to the final bit rate, which is indicated by the *lossless rate* regardless of the final reconstruction being lossless or lossy. For comparing multi-rate codecs, the lossy rates need to be considered as well.

For comparing codecs at lossy rates, the main evaluation criterion in literature is the *rate-distortion* performance. Rate-distortion performance is discussed below, followed by a novel measure which allows for a more compact numerical comparison of multiple rate-distortion performance evaluations. In addition to rate-distortion performance, the scalable representation after decoding is equally important from a *rendering* point-of-view. This has been overlooked in the state of the art, but is valuable for any use-case where interactivity is required. This dissertation proposes a novel *triangle-distortion* measure which is related to the rendering performance. Finally, a measure for more compact numerical comparisons is similarly proposed.

**Rate-Distortion (RD) Curves** While a lossless rate comparison gives an objective number to compare coding systems, it entirely ignores any scalability functionalities. In this regard, the main measure for comparing scalable mesh coding systems has been *rate-distortion (RD) curves*. These plots show the distortions given the bit rates. For the distortion, either the Hausdorff distance or the RMS distance are generally considered in literature. In the literature, no strict convention is followed for the RMS distance whether the maximal value, the sum or the average of the forward and backward distances must be used. Furthermore, some results are reported with the distortions normalized w.r.t. the bounding box diagonal while others are reported as absolute numbers. For fair comparisons, in

this dissertation all values are normalized w.r.t. the bounding box of the original model. Figure 2.14 shows an example with three consecutive rate points. With increasing rate, the distortion needs to reduce. In practice, an encoder can indicate *non-decodable rate points* when additional rate increases the distortion. Such rate points are not considered for an RD comparison; instead, the curves are made convex by removing such non-convex rate points.

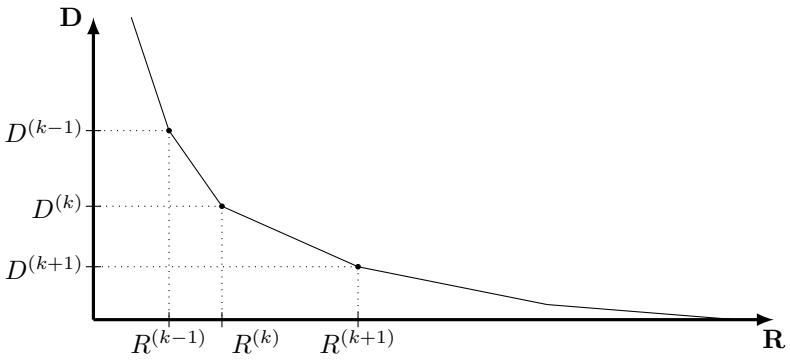



*Figure 2.14: Rate-distortion curve. The curve shows three rate points $k-1$, $k$ and $k+1$. Increasing the rate reduces the distortion.*

Evaluating results using such RD curves comes with several issues. First of all, as acknowledged in several multimedia domains, objective quality metrics such as the RMS distortion as described in Section 2.3.1 do not necessarily match subjective quality experience. This mismatch is even more notable for 3D meshes because (a) metrics can no longer be defined on regularly spaced samples as can for instance be done for images, and will, without proper weighing, result in even worse correlation with subjective experience; and (b) the qualitative impression of a 3D surface becomes even harder to express due to the interactive nature in which these models are used. For instance, if there are large distortions on a surface, but these distortions can only be seen from a very small space because of self-occlusions, such distortions should be preferred over smaller distortions which are spread over an entire model. Additional lighting in a virtual world can either enhance or hide geometric aberrations. And finally, once additional vertex attributes such as colors, specularity, transparency, etc., have to be taken into account, traditional objective quality metrics are entirely insufficient to measure mesh quality in an interactive environment. Due to this mismatch between objective quality metrics and subjective visual quality, visual results accompany an RD comparison.

**Average Rate Difference** The second issue of evaluating using RD curves is that such a curve compares several codecs or parameters for a *single* model. From a

practical point-of-view, it becomes cumbersome to compare codecs over a larger test set. To account for this, a new metric is proposed which is similar to the *Bjøntegaard delta rate (BD-rate)* [6]. The BD-rate uses four rate points to match two *peak signal-to-noise ratio (PSNR)* curves and integrate the rate values over a range of quality values. In this dissertation, an *average rate difference* $\Delta^{\mathbf{r}}_{\mathbf{avg}}$ is proposed: linearly interpolate the rate points and then take the average of the differences in required rates to obtain specific, densely sampled distortion values.

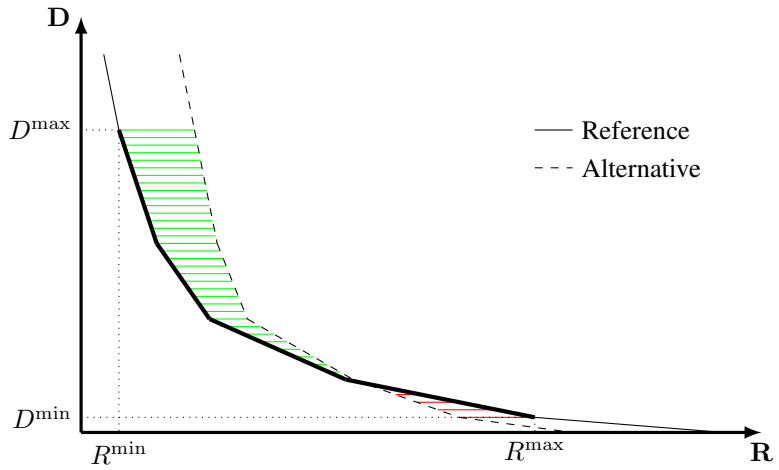



*Figure 2.15: Average rate difference. The green area indicates where the reference outperforms the alternative: to obtain a specific quality a lower rate suffices, or with the same rate a better quality can be obtained. Conversely, the red area shows where the reference codec is outperformed.*

This is shown in Figure 2.15. For a desired range of rates given by $[R^{\min}, R^{\max}]$, the accompanying distortions for a reference codec can be determined. Now, for each distortion $d \in [D^{\min}, D^{\max}]$, the rates required by the reference codec and an alternative codec can be determined. The rate differences can now be averaged to give a single number which compares the two codecs within the initial range $[R^{\min}, R^{\max}]$. The figure illustrates that a codec with a worse lossless coding rate can still be valuable if it offers better quality at lossy rates and only performs suboptimal at rates where distortions become nearly unobservable.

**Triangle-Distortion (TD) Curves** Observe that conventional rate-based evaluations only consider the coding step, which gives valuable information from a storage and transmission perspective, but does not take into account memory usage for visualization. While a semi-regular mesh codec is superior because no connectivity information is required, such coding systems result in approximations

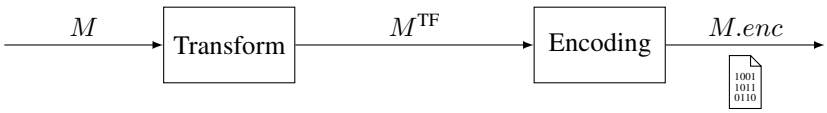



*Figure 2.16: Basic architecture of any mesh encoding system. A mesh $M$ is transformed into some format $M^{TF}$ with reduced entropy. The actual compression is obtained through the encoding block, which makes use of residual or subband encoding and entropy coding to obtain a compressed representation $M_{enc}$ of the input mesh.*

which have far more triangles compared to approximations by irregular mesh codecs with similar distortions. Consequently, despite requiring fewer bytes, a semi-regular mesh codec can require more memory compared to an irregular mesh codec for rendering at a specific quality level.

In this regard, a novel *triangle-distortion (TD)* measure is proposed in addition to the classical RD quality measure: instead of evaluating the distortion in function of the required *rate*, the distortion is measured given a specific percentage of reconstructed *triangles*. Hence, this considers how the quality evolves, not from a coding perspective but from a rendering perspective, and is related to the mesh *representation* instead of its encoding. A better curve means that the same quality can be obtained using less memory, or a higher quality can be obtained using the same amount of memory.

**Average Triangle Difference** Similar to the BD-rate measure, two TD curves can be compared by their *average triangle percentage difference* $\Delta^{\mathbf{t}}_{\mathbf{avg}}$, obtained by linearly interpolating the rate points and averaging the triangle percentage differences.

## 2.4 Mesh Compression

This final section on mesh coding covers some mesh compression techniques. Mesh compression has been investigated for over two decades. Thorough surveys can be found in, for instance, [7], [8], [9] and more recently [10] and [11]. Nonetheless, a short overview is given, covering the basic ideas of mesh compression. Single-rate coding is discussed first in Section 2.4.1. Subsequently, multi-rate or scalable coding techniques are covered in Section 2.4.2, differentiating between fine-grained (Section 2.4.2.1) and coarse-grained (Section 2.4.2.2) scalability.

Figure 2.16 depicts the architecture of a conventional mesh encoding system. The process starts from an initial mesh $M$. First, a *transform* will change the way the data is represented such that predictability emerges. This is indicated by $M^{\text{TF}}$, the *T*rans*F*ormed mesh. This data is now sent through an encoding block which

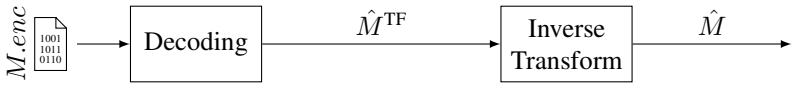



*Figure 2.17: General single-rate decoding system.*
*In a single-rate decoding, a model can only be decoded at one rate, i.e., the intended rate. This results in a single $\hat{M}^{TF}$ and consequently a single reconstructed mesh $\hat{M}$. This reconstruction can be lossless (where $M = \hat{M}$), or lossy ($M \neq \hat{M}$). Observe the symmetry with Fig. 2.16.*

exploits the reduced entropy in $M^{\text{TF}}$, and results in a binary file $M_{enc}$. In many cases, this encoding step is where compression is obtained.

*Entropy* is a measure of 'chaos'; reducing entropy signifies increasing structure and consequently predictability. Such predictability can be exploited, for instance by storing patterns that occur more frequently using fewer bits than what is used for storing less-frequently occurring patterns.

### 2.4.1 Single-Rate Mesh Compression

Single-rate mesh compression entails that the coding scheme shown in Figure 2.16 results in an encoding $M_{enc}$ which can only be decoded at a single rate; this single-rate decoding is visualized in Figure 2.17. Table 2.6 gives single-rate compression results when compressing the models of Table 2.2 to the *general-purpose* .zip archive file format using default settings.

| **model (vertices)** | **.obj.zip** | **.ply.asc.zip** | **.ply.bin.zip** |
|---:|:---:|:---:|:---:|
| epcot (770) | 9.88 KB (105 bpv) | 9.13 KB (97.2 bpv) | 3.05 KB (96.4 bpv) |
| bimba (8 857) | 182 KB (169 bpv) | 175 KB (163 bpv) | 174 KB (161 bpv) |
| rabbit (67 039) | 1.55 MB (195 bpv) | 1.41 MB (177 bpv) | 1.25 MB (158 bpv) |
| turbine (882 954) | 14.3 MB (137 bpv) | 13.0 MB (124 bpv) | 11.0 MB (105 bpv) |

*Table 2.6: General-purpose compression on meshes. Examples of Table 2.2, compressed to .zip archive files.*

Research into single-rate *mesh* compression started over two decades ago. A first improvement on the storage using lists of vertices and triangles was given by Deering [12]. He suggested that the connectivity of a mesh can be partitioned in triangle strips and triangle fans. This allows for implicitly defining a triangle per added vertex. After storing an initial triangle, a triangle strip creates a new triangle by combining the new vertex with the last two created vertices. A triangle fan creates new triangles by combining the new vertex with the last created vertex and the very first created vertex. Ideally, the connectivity information converges to a third of the original size as only one vertex index is needed per triangle, instead of

three. The geometry is stored by looking at the difference between the last vertex and the new one. These values are usually very small, and variable-length coding will result in smaller values for the geometry information.

Taubin and Rossignac [13] proposed an approach called *topological surgery (TS)*. The connectivity is represented using a vertex spanning tree within the graph of edges and vertices. Such a tree connects all vertices without visiting vertices multiple times, i.e., without creating loops. Neighboring vertices within this tree are likely to be close together geometrically, so representing the difference between two vertices results in smaller values. Touma and Gotsman [14] make clever use of the valences of the vertices. For the connectivity information they traverse all vertices, always keeping track of the encoded portion and the unvisited portion. Using three commands and the valences of the vertices, both of which are very predictable and interesting for entropy coding, the decoder can reconstruct the same connectivity. Two modes for predicting the positions of the vertices are used; either a prediction similar to Deering's approach, using the last encountered vertex as the prediction, or a prediction using a *parallelogram rule*. This rule assumes that a new vertex will form a parallelogram with the neighboring triangle. An even more advanced approach considers the expected angle for the edge within this parallelogram. The reported averages are $1.4$ bpv for the connectivity and $9.0$ bpv for the geometry, when encoding at $8$ bit precision. These results are reported to be resp. $66\%$ and $31\%$ better w.r.t. the results of [13].

Rossignac described the *Edgebreaker* algorithm [15]. This is a traversal approach which does not consider the vertex valences, but uses the relation between a newly visited vertex and the boundary of the already encoded vertices. Depending on whether it is a new vertex, or it is on the boundary, either to the left or right of the current edge, the only remaining vertex, or any other vertex, one of five symbols is encoded per triangle. The geometry encoding approach is not detailed, but a prediction approach is assumed.

An improvement upon the work of Touma and Gotsman, which is nearly optimal in the regular case, was proposed by Alliez and Desbrun [16]. Their work mainly improves the results on coarse irregular meshes. Alliez and Desbrun suggest to replace the deterministic conquest by an adaptive conquest which makes the three commands and their parameters even more predictable. The geometry coding has not been altered.

### 2.4.2 Scalable Mesh Compression

Single-rate mesh compression is great for storage and distribution, but does not *scale* well for interactive applications. Regardless of the implementation, models growing over certain sizes will require several seconds or minutes to decode and visualize. A simulcast solution, of which a general scheme is depicted in Figure

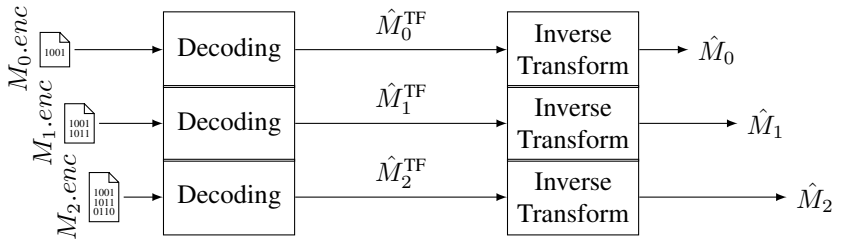



*Figure 2.18: General simulcast decoding system.*
*In a simulcast system, a limited form of scalability is offered by having multiple versions of $M$, e.g., $M_1$, $M_2$ and $M_3$, sending these each through a single-rate encoding system and storing them either in physically separate files or in a single byte stream $M_{enc}$. Based on the requirements of the system, a decoding system can now reconstruct either $\hat{M}_0$, $\hat{M}_1$ or $\hat{M}_2$. Progressivity can be offered by decoding them in order, decoding $\hat{M}_0$ first, then $\hat{M}_1$ and finally $\hat{M}_2$. Note that these decoding steps are completely independent of each other; consequently, on the one hand they can be performed in parallel, but on the other hand the decoding of $\hat{M}_1$ does not take advantage of the work done in decoding $\hat{M}_0$. Each of the three pipelines can be seen as a system as depicted in Fig. 2.17.*

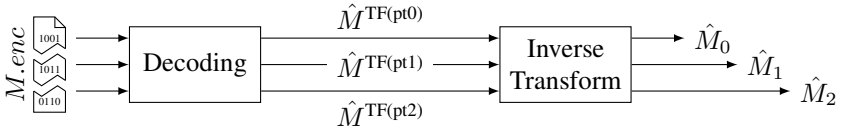



*Figure 2.19: General multi-rate decoding system.*
*In multi-rate decoding, a model can be decoded at several rates. Given the parts of $M_{enc}$ which are streamed, this results in transformed data $\hat{M}_i^{TF}$. Using this data, several possible reconstructions of mesh $M$ can be obtained; three reconstructions, $\hat{M}_0$, $\hat{M}_1$ and $\hat{M}_2$, are found. These versions can differ in amounts of vertices (resolution scalability) or accuracy of the reconstructed vertices (quality scalability), where these differences can either be found in the entire model or even just in selected parts (ROI decoding).*

2.18, addresses this issue: several reduced versions are created and independently encoded, allowing for (1) a coarse visualization while a higher detailed model is loading, and (2) only visualizing the appropriate detail if an application allows for a reduced version. Such a reduced version could mean either reducing the vertex quality by using a coarser quantization, or reducing the resolution by downsampling the model which results in fewer samples and consequently fewer vertices and triangles.

While such a simulcast solution of several LODs addresses the interactivity issue, and exploits *intra-LOD* correlations using a single-rate coder, it is clearly a suboptimal solution. Due to the independent handling of the LODs, the storage requirements and total decoding time increases due to the storage and possible decoding of several lower LODs. There are clear similarities between several LODs $\hat{M}_i$; after all, they are approximations of the same high-detailed model $M$. Hence, capturing such *inter-LOD* correlations reduces the storage cost and decoding time, by considering the difference between subsequent LODs.

Depending on the granularity of subsequent LODs, the literature can be partitioned into fine-grained continuous LOD systems and coarse-grained discrete LOD systems, as discussed next.

#### 2.4.2.1 Continuous LOD Systems

The first proposed approach to consider LODs and their relations was given by Hoppe [17, 18], and was appropriately termed the *progressive mesh (PM) representation*. This approach was further generalized to any number of dimensions in [19]. In this system, the transformations occurred by iteratively collapsing a single edge. This collapsing operation considers a single edge and merges the two neighboring vertices, while removing the neighboring triangles. This edge collapse operation is repeated until some base mesh $M_0$ is obtained. The transformed representation $M^{\text{TF}}$ comprises this base mesh $M_0$ and a series of vertex split operations which invert the edge collapse operations. Each edge split operation takes one vertex, splits it into two vertices and creates an edge and two triangles between them. The PM representation is progressive in the sense that each additional part of $M^{\text{TF}}$ allows for reconstructing $M$ at an increasingly higher accuracy. For a model with thousands or millions of vertices, this results in thousands or millions of LODs. The differences between these become nearly unnoticeable at higher resolutions, such that despite the amount of LODs still being discrete, such *vertex-by-vertex* systems can be considered as continuous LOD (cLOD) systems.

In his original work, Hoppe does not describe any encoding system, i.e., $M^{\text{TF}}$ is directly written to $M_{enc}$ (one can consider the encoding block as a simple serialization operation). Hence the representation does not offer compression; the data is merely given in a different way. As the name suggest, the advantage w.r.t.

conventional mesh representations is the fact that partially reading $M_{enc}$ gives a progressively improving approximation of $M$. An encoding step can be added as described by Pajarola and Rossignac [20]. By grouping the vertex split operations in batches and jointly encoding them, they obtain files at half the original storage size.

In light of real-time rendering, aimed at higher resolutions, this work soon shifted towards view-dependent representations. Such representations are related to ROIs, which are typically not an issue for cLOD systems. [21] describes a view-dependent variant for progressive meshes, which was later implemented for terrain data in [22].

A similar transform was proposed in the IPR-codec of Valette et al. [23]. Their refinement operator, called an *edge split*, can be translated to the vertex splits of the PM representation. After obtaining a base mesh $M_0$ using an approach similar to [24], their operation processes the $i^{\text{th}}$ largest edge at every step, adding a single vertex and optimally assigning the quantization for the surrounding vertices. Such an approach considers both resolution and quality to obtain better RD performance. By ignoring the original connectivity information, all intermediate resolutions can be stored more efficiently; the cost of obtaining the actual connectivity is only required when the final vertex has been added, for actually lossless decoding. The index $i$ of which edge to split when the edges are ordered from largest to smallest, is represented as a unary code, i.e., a series of $i - 1$ zeros followed by a one. These bits are called midpoint codes. Arithmetic coding is used both for these midpoint codes and for the prediction errors of the vertex coordinates, which are predicted at the centers of the edges.

A different approach was taken by Alliez and Desbrun [25] who propose a valence-driven conquest of vertices, resolution per resolution, inspired by the work of Touma and Gotsman [14]. In this algorithm, a mesh is traversed in a *decimation conquest*, forming patches based on some valence considerations. The retriangulation per patch is done in a deterministic way to ensure that the decoder can find the patches without additional information. Following the decimation conquest, a *cleaning conquest* is performed, where only valence-3 vertices are handled. Finally, the resulting data is arithmetically encoded.

#### 2.4.2.2 Discrete LOD Systems

Inspired by the PM representation of Hoppe [17] and the TS single-rate codec of Taubin and Rossignac [13], Taubin et al. [26] developed their *progressive forest split (PFS)* compression algorithm. The PFS scheme is more coarse-grained compared to the fine-grained PM scheme. In view of the continuous LOD terminology given above, such coarser-grained LOD systems are termed discrete LOD (dLOD) systems.

An important paradigm in many multimedia domains is the wavelet transform

to obtain multiple resolutions and encode them in an efficient way. A wavelet transform iteratively transforms a higher-resolution signal into a lower-resolution signal and a wavelet subband. However, whereas 3D surfaces are inherently irregularly sampled, wavelet coding conventionally considers regularly sampled data. The regular connectivity of semi-regular meshes has been used in literature for building wavelet transforms. Such meshes are often obtained via the interpolating Butterfly [27] scheme or the approximating Loop [28] subdivision scheme, where each subdivision step results in a resolution increment. After a 1-to-4 subdivision, the Butterfly subdivision perturbs the newly added vertices based on neighborhood information. For a vertex on a given edge, this neighborhood information encompasses the two triangles neighboring this edge, and their four additional neighboring triangles. Figure 2.20a depicts this and illustrates the origin of the name '*butterfly*' subdivision. Denote by $j$ the amount of performed subdivision steps. Given a tension parameter $w$, each new vertex $v_{j+1}^o$ is located at

$$v_{j+1}^o = \frac{1}{2}(v_j^A + v_j^B) + 2w(v_j^C + v_j^D) - w(v_j^E + v_j^F + v_j^G + v_j^H). \tag{2.31}$$

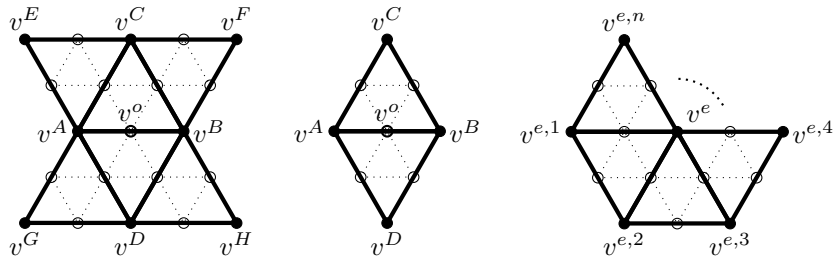
The Loop subdivision scheme on the other hand transforms both new and existing vertices given the masks shown in Figure 2.20b and 2.20c respectively. Each new vertex $v_{j+1}^o$ is positioned at

$$v_{j+1}^o = \frac{3}{8}v_j^A + \frac{3}{8}v_j^B + \frac{1}{8}v_j^C + \frac{1}{8}v_j^D, \tag{2.32}$$

while an exist vertex $v_j^e$ with degree $\nu(v_j^e) = n$ is repositioned, again given a weighing factor $w$, to

$$v_{j+1}^e = (1 - nw)v_j^e + w\sum_{i=0}^{n} v_j^{e,i}. \tag{2.33}$$

Wavelet-based mesh coding was initially proposed for such semi-regular meshes: wavelets are defined on the 2D parametric surface defined by the base mesh, and each higher-resolution wavelet subband is then related to the lower-resolution subband given the subdivision scheme. The stencils depicted in Figure 2.20 and the weights given in Equations 2.31, 2.32 and 2.33 determine the scaling coefficients, describing how a mesh is upscaled from a lower resolution to a higher resolution. Wavelet coefficients then refine the vertices of the mesh, for representing detail information which was not present in the lower-resolution representation. Pioneering work was proposed by Khodakovsky et al. [1], describing their *progressive geometry compression (PGC)* algorithm which processes meshes by using semi-regular wavelet transforms and zerotree coding [2]. Such a tree exploits the fact that the descendants of a wavelet coefficient which is non-significant given a specific threshold $\tau$ are often non-significant as




*(a) Butterfly* *(b) Loop (new vertices)* *(c) Loop (existing vertices)*

*Figure 2.20: Butterfly and Loop neighborhoods.*

well for the same threshold $\tau$, and can be encoded together with a single *zero*. This approach and other approaches such as those proposed by Khodakovsky and Guskov [29] or by Avilés et al. [30] exploit *interband* correlations, i.e., correlations between wavelet coefficients in subsequent wavelet subbands: at specific regions, the properties are expected to be similar across resolutions and are encoded together. The main issue with such approaches is that they do not allow for resolution scalability.

While *intraband* correlations, i.e., the correlations between wavelet coefficients *within* wavelet subbands, have been investigated for image compression (see, for instance, [31–33]), few intraband mesh codecs have been proposed. Payan and Antonini [34] describe a semi-regular mesh codec which employs the Loop [28] Discrete Wavelet Transform and encodes the quantized wavelet coefficients using independent *embedded block coding with optimized truncation (EBCOT)* [33] of the embedded bit streams. The statistical dependencies within and across wavelet subbands have been analyzed by Satti et al. in [35] and [36] for semi-regular meshes and normal meshes respectively. These works concluded that intraband dependencies are stronger than interband dependencies, and that composite codecs which exploit both intraband and interband statistical dependencies perform best.

For directly processing irregular meshes, few codecs have been proposed. Early work was done by Bonneau [37], which compresses the *data* contained in an irregular mesh, for instance the color data over an irregular mesh representing the earth. As such, this work considers the changing mesh resolutions as "given" and maps the color data to this domain. Actual compression results for the mesh itself are not taken into consideration. Wavemesh, by Valette and Prost [38], is a state-of-the-art wavelet-based irregular mesh coding system. In Wavemesh, the classical 1-to-4 subdivision is generalized to any subdivision of triangles which adds vertices to one, two or all three of the edges of a triangle. As with any irregular mesh

coding system, irregularity comes at a cost: contrary to semi-regular codecs where the mesh connectivity can be implicitly reconstructed at a decoding side, irregular mesh codecs require additional information to properly reconstruct the connectivity. The Wavemesh connectivity encoding has been reused by Valette et al. in [39] which proposes zerotree encoding [2]. Lee et al. make use of the Wavemesh representation but propose novel connectivity and geometry coding approaches [40].

Roy et al. [41] have reformulated the PM representation, which was already encoded in a batched form by Pajarola and Rossignac in [20], as a multiresolution analysis problem. They additionally take into account multiple attributes per vertex, but do not consider them together to optimize the coding performance.

Finally, Maglo et al. [42] similarly use edge collapses to generate several LODs by decimating an original mesh, grouping edge collapse operations which are mutually independent. At the finer LODs, clustering and their independent compression enables random access, allowing for refining specific clusters more or less than their neighboring clusters.

### 2.4.3 Decoding Granularity

The difference between single-rate (Section 2.4.1), cLOD (Section 2.4.2.1) and dLOD decoding (Section 2.4.2.2) is illustrated in Figure 2.21. This figure depicts the distortion w.r.t. the amount of reconstruction. In the end, all (lossless) schemes reconstruct the same model. A single-rate codec *only* constructs this model, so the error when limiting the reconstruction becomes infinite as there is *no* approximation available. A dLOD system has a *discrete amount of reconstructions* which can be obtained. For a system which relies on $1$-to-$4$ subdivision of vertices, a model with $n$ vertices will have $log_4(n)$ resolutions; for instance, a $1\,000\,000$-vertex model will be represented using $10$ resolution levels. Finally, a cLOD system practically has a *continuous amount of reconstructions*. For instance, the same $1\,000\,000$-vertex model would result in almost $1\,000\,000$ reconstruction levels.

## 2.5 Conclusions

This chapter covered the basic principles of mesh compression. A discussion on sampling and quantization detailed how a continuous surface is represented digitally by using a polygon mesh. From this, one can learn that $12$ to $16$ bit quantization per component often largely suffices for representing the mesh geometry. Successive approximation quantization (SAQ) was covered, which will allow for quality scalability due to the embedded nature of the quantizers. Next, some definitions and properties of meshes are discussed, focusing mainly on mesh

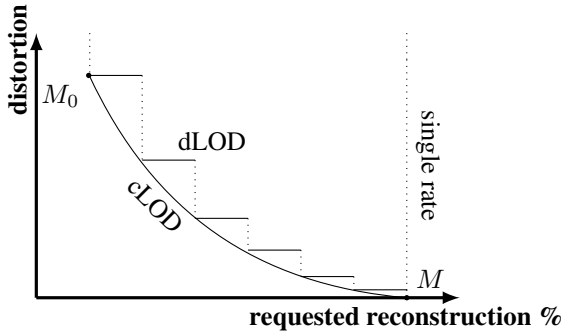



*Figure 2.21: Reconstruction granularity of single-rate and LOD decoding. The full lines show obtainable rate points. For single-rate decoding (on the right), only the full reconstruction can be requested, requesting a lower reconstruction gives an "infinite distortion" as the model is simply undefined. The bottom curved line and the top step line respectively show cLOD and dLOD reconstructions. While cLOD is in fact still discrete, plotting all (possibly millions of) reconstructed samples will give a nearly continuous curve. In both cases, the minimal reconstruction is $M_0$, i.e., the* base mesh*.*

regularity. This reveals that, in general, semi-regular meshes can be compressed more efficiently as no connectivity information needs to be stored. In contrast, irregular meshes require more information to be encoded, but allow for better approximations using fewer vertices due to better allocation of vertex densities. This trade-off is further investigated in Chapter 3.

Subsequently, mesh distortion measures have been discussed. Both the Hausdorff and RMS measure have been used in literature for rate-distortion (RD) comparisons; however, this disregards memory requirements for real-time rendering which relates to the amount of vertices and triangles used to represent a model. This chapter proposed a triangle-distortion (TD) comparison to account for this. Additionally, an average rate difference and an average triangle difference have been suggested for more succinct comparisons of codecs over larger test sets.

Finally, an overview of mesh compression was given. This dissertation advances the state of the art in wavelet-based discrete LOD (dLOD) systems for irregular meshes. The main reference for such systems is Wavemesh. Furthermore, IPR will be used for comparison with a state-of-the-art continuous LOD (cLOD) system.

The novel evaluation measures have been introduced in the *Computer Graphics Forum* publication:

- J. El Sayeh Khalil, A. Munteanu, L. Denis, P. Lambert, and R. Van de Walle. *Scalable Feature-Preserving Irregular Mesh Coding*. Computer Graphics Forum, 36(6):275–290, September 2017.

# References

[1] Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. *Progressive Geometry Compression*. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 271–278, New Orleans (Louisiana), United States, 23–28 July 2000.

[2] Jerome M. Shapiro. *Embedded Image Coding using Zerotrees of Wavelet Coefficients*. IEEE Transactions on Signal Processing, 41(12):3445–3462, December 1993.

[3] Igor Guskov, Kiril Vidimče, Wim Sweldens, and Peter Schröder. *Normal Meshes*. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 92–102, New Orleans (Louisiana), United States, 23–28 July 2000.

[4] Frédéric Payan, Céline Roudet, and Basile Sauvage. *Semi-Regular Triangle Remeshing: A Comprehensive Study*. Computer Graphics Forum, 34(1):86–102, 2015.

[5] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. *Metro: Measuring Error on Simplified Surfaces*. Computer Graphics Forum, 17(2):167–174, June 1998.

[6] Gisle Bjøntegaard. *Calculation of Average PSNR Differences between RD-Curves*. Technical Report VCEG-M33, ITU-T SG16/Q6, Austin (Texas), United States, 2001.

[7] Craig Gotsman, Stefan Gumhold, and Leif Kobbelt. *Simplification and Compression of 3D Meshes*, pages 319–361. Tutorials on Multiresolution in Geometric Modelling: Summer School Lecture Notes (Mathematics and Visualization). Springer Berlin Heidelberg, 2002.

[8] Pierre Alliez and Craig Gotsman. *Recent Advances in Compression of 3D Meshes*, pages 3–26. Advances in Multiresolution for Geometric Modelling (Mathematics and Visualization). Springer Berlin Heidelberg, 2005.

[9] Jingliang Peng, Chang-Su Kim, and C.-C. Jay Kuo. *Technologies for 3D Mesh Compression: A Survey*. Journal of Visual Communication and Image Representation, 16(6):688–733, December 2005.

[10] Huaiyu Li, Weilang Meng, and Xiaopeng Zhang. *A Survey on Recent Approaches of Mesh Compressions*. In Proceedings of the 4th International Conference on Virtual Reality and Visualization (ICVRV), pages 50–57, Shenyang Aerospace University, China, 30–31 August 2014.

[11] Adrien Maglo, Guillaume Lavoué, Florent Dupont, and Céline Hudelot. *3D Mesh Compression: Survey, Comparisons, and Emerging Trends*. ACM Computing Surveys, 47(3):44:1–44:41, February 2015.

[12] Michael Deering. *Geometry Compression*. In Proceedings of the 22nd Annual Conference on Computer Graphics (SIGGRAPH), pages 13–20, Los Angeles (California), United States, 6–11 August 1995.

[13] Gabriel Taubin and Jarek R. Rossignac. *Geometric Compression Through Topological Surgery*. ACM Transactions on Graphics, 17(2):84–115, April 1998.

[14] Costa Touma and Craig Gotsman. *Triangle Mesh Compression*. In Proceedings of the Graphics Interface Conference (GI), pages 26–34, Vancouver (British Columbia), Canada, 18–20 June 1998.

[15] Jarek R. Rossignac. *Edgebreaker: Connectivity Compression for Triangle Meshes*. IEEE Transactions on Visualization and Computer Graphics, 5(1):47–61, January/March 1999.

[16] Pierre Alliez and Mathieu Desbrun. *Valence-Driven Connectivity Encoding for 3D Meshes*. Computer Graphics Forum, 20(3):480–489, 2001.

[17] Hugues Hoppe. *Progressive Meshes*. In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 99–108, New Orleans (Louisiana), United States, 4–9 August 1996.

[18] Hugues Hoppe. *Efficient Implementation of Progressive Meshes*. Computers & Graphics, 22(1):27–36, February 1998.

[19] Jovan Popović and Hugues Hoppe. *Progressive Simplicial Complexes*. In Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 217–224, Los Angeles (California), United States, 3–8 August 1997.

[20] Renato Pajarola and Jarek R. Rossignac. *Compressed Progressive Meshes*. IEEE Transactions on Visualization and Computer Graphics, 6(1):79–93, January 2000.

[21] Hugues Hoppe. *View-Dependent Refinement of Progressive Meshes*. In Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 189–198, Los Angeles (California), United States, 3–8 August 1997.

[22] Hugues Hoppe. *Smooth View-dependent Level-of-detail Control and its Application to Terrain Rendering*. In Proceedings of the IEEE Conference on Visualization (VIS), pages 35–42, Research Triangle Park (North Carolina), United States, 18–23 October 1998.

[23] Sébastien Valette, Raphaëlle Chaine, and Rémy Prost. *Progressive Lossless Mesh Compression via Incremental Parametric Refinement*. Computer Graphics Forum, 28(5):1301–1310, 2009.

[24] Michael Garland and Paul S. Heckbert. *Surface Simplification Using Quadric Error Metrics*. In Proceedings of the 24th Annual Conference on Computer Graphics (SIGGRAPH), pages 209–216, Los Angeles (California), United States, 3–8 August 1997.

[25] Pierre Alliez and Mathieu Desbrun. *Progressive Compression for Lossless Transmission of Triangle Meshes*. In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 195–202, Los Angeles (California), United States, 12–17 August 2001.

[26] Gabriel Taubin, André Guéziec, William Horn, and Francis Lazarus. *Progressive Forest Split Compression*. In Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 123–132, Orlando (Florida), United States, 19–24 July 1998.

[27] Nira Dyn, David Levine, and John A. Gregory. *A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control*. ACM Transactions on Graphics, 9(2):160–169, April 1990.

[28] Charles Teorell Loop. *Smooth Subdivision Surfaces Based on Triangles*. Master's thesis, Department of Mathematics, The University of Utah, August 1987.

[29] Andrei Khodakovsky and Igor Guskov. *Compression of Normal Meshes*, pages 189–206. Geometric Modeling for Scientific Visualization (Mathematics and Visualization). Springer Berlin Heidelberg, 2004.

[30] Marcos Avilés, Francisco Morán, and Narciso García. *Progressive Lower Trees of Wavelet Coefficients: Efficient Spatial and SNR Scalable Coding of 3D Models*. In Proceedings of the 6th Pacific-Rim Conference on Advances in Multimedia Information Processing (PMC) – Volume Part I, pages 61–72, Jeju Island, Korea, 11–13 November 2005.

[31] Adrian Munteanu, Jan Cornelis, Geert Van der Auwera, and Paul Cristea. *Wavelet-based Lossless Compression Scheme with Progressive Transmission Capability*. International Journal of Imaging Systems and Technology, 10(1):76–85, 1999.

[32] Adrian Munteanu, Jan Cornelis, Geert Van Der Auwera, and Paul Cristea. *Wavelet Image Compression – The Quadtree Coding Approach*. IEEE Transactions on Information Technology in Biomedicine, 3(3):176–185, September 1999.

[33] David Taubman. *High Performance Scalable Image Compression with EBCOT*. IEEE Transactions on Image Processing, 9(7):1158–1170, July 2000.

[34] Frédéric Payan and Marc Antonini. *Multiresolution 3D Mesh Compression*. In Proceedings of the 9th IEEE International Conference on Image Processing (ICIP), volume 2, pages 245–248, Rochester (New York), United States, 22–25 September 2002.

[35] Shahid Mahmood Satti, Leon Denis, Adrian Munteanu, Jan Cornelis, and Peter Schelkens. *Estimation of Interband and Intraband Statistical Dependencies in Wavelet-based Decomposition of Meshes*. In Proceedings of SPIE – Wavelet Applications in Industrial Processing VI, volume 7248, pages 1–10, San Jose (California), United States, 18–22 January 2009.

[36] Leon Denis, Shahid Mahmood Satti, Adrian Munteanu, Jan Cornelis, and Peter Schelkens. *Scalable Intraband and Composite Wavelet-Based Coding of Semiregular Meshes*. IEEE Transactions on Multimedia, 12(8):773–789, December 2010.

[37] Georges-Pierre Bonneau. *Multiresolution Analysis on Irregular Surface Meshes*. IEEE Transactions on Visualization and Computer Graphics, 4(4):365–378, October 1998.

[38] Sébastien Valette and Rémy Prost. *Wavelet-Based Progressive Compression Scheme for Triangle Meshes: Wavemesh*. IEEE Transactions on Visualization and Computer Graphics, 10(2):123–129, March/April 2004.

[39] Sébastien Valette, Alexandre Gouaillard, and Rémy Prost. *Compression of 3D Triangular Meshes with Progressive Precision*. Computers & Graphics, 28(1):35–42, 2004.

[40] Dae-Youn Lee, Sanghoon Sull, and Chang-Su Kim. *Progressive 3D Mesh Compression using MOG-based Bayesian Entropy Coding and Gradual Prediction*. The Visual Computer, 30(10):1077–1091, October 2014.

[41] Michaël Roy, Sebti Foufou, and Frédéric Truchetet. *Multiresolution Analysis for Irregular Meshes with Appearance Attributes*, pages 80–86. Computer Vision and Graphics: International Confonference, ICCVG 2004, Warsaw, Poland, September 2004, Proceedings (Computer Imaging and Vision). Springer Netherlands, 2006.

[42] Adrien Maglo, Ian Grimstead, and Céline Hudelot. *POMAR: Compression of Progressive Oriented Meshes Accessible Randomly*. Computers & Graphics, 37(6):743–752, 2013.

# 3

# Resolution-Scalable and Quality-Scalable Coding

The first major part of this dissertation tackles the issue of *resolution scalability*. As introduced, computer graphics are omnipresent, with applications ranging far outside of the entertainment sector. Main concerns are the increasing size of the data, the growing diversity in processing power and capabilities of rendering devices, and the larger variety of available bandwidths. Compression as such does not suffice for interactive use, due to the processing required to decode an *entire* mesh. A representation which is allowed to scale based on the actual application requirements, device limitations and network properties proves valuable for any mesh coding system.

Similar to other signal processing domains, there is a need for a multiresolution representation that takes advantage of the inherent similarities between *levels of detail (LODs)*. For many years, wavelets have been used for representing data in a multiresolution fashion, and their usage has already been extended to 3D meshes. A wavelet-based solution uses a set of high-pass and low-pass filters to obtain a low-resolution base mesh where all high-frequency information has been removed, and a set of wavelet subbands containing this increasingly higher frequency information. However, even today scalable representations have not yet seen a breakthrough into the commercial world. Using these scalable systems at low bit rates results either in a high number of polygons or a low quality; hence, *manually* designing high-quality LODs with a low number of polygons is still preferred.

**Contributions** This first part of the dissertation proposes a lossless wavelet-based multiresolution representation and coding system for irregular meshes. The focus lies on improving the feature preservation using a limited number of triangles, resulting in a good *rate-distortion (RD)* trade-off while also improving the rendering performance. This work improves over the state of the art in several aspects.

- A signal-adaptive downsampling *and* retriangulation procedure targets feature-preservation by design, without impeding the filtering process near geometric features.
- Adaptive retriangulation is not purely topology-based but takes into account the geometric properties.
- The downsampling procedure decreases the resolution at most by half for each decomposition level, resulting in a higher granularity in terms of levels of detail.
- A novel octree-based encoding of the connectivity information decouples decoding from any mesh traversal order.
- Spatial correlations of wavelet coefficients are exploited by employing an octree-based encoding of geometry information as a novel way to process a connectivity-driven transform by a geometry-based encoding.
- A comparison of the distortion w.r.t. the number of triangles shows the performance from a rendering point-of-view, rather than a storage or transmission perspective.

By exploiting fine-grain quality scalability, which allows for scaling the quality of reconstructed data by decoding per wavelet subband bit plane, data can be transmitted such that the distortion in the reconstructed mesh decreases optimally. The codec is then extended to offer quality scalability, further improving over the state of the art in two ways.

- The *coding performance* at low bit rates is improved by the proposed algorithm by performing RD optimization (RDO).
- *Functionally*, an additional form of scalability is offered without negatively impacting the lossless coding rate.

The remainder of this chapter is structured as follows. First a more detailed discussion on related work is given in Section 3.1. The proposed coding scheme is presented in Section 3.2, with a discussion on the transform in Section 3.2.1 and on the coding part in Section 3.2.2. Requirements for quality scalability are then

discussed in Section 3.3, and RDO using both resolution and quality scalability is discussed in Section 3.4. Next, the experimental evaluation is given in Section 3.5, and finally Section 3.6 concludes this chapter.

## 3.1 Related Work

Whereas single-rate coding has not seen many improvements since the state-of-the-art coder of Touma and Gotsman [1], multi-rate coding has been tackled using several very different approaches. Multi-rate or scalable coding is possible on a fine-grained *vertex-by-vertex* level as introduced by Hoppe [2], resulting in a nearly continuous sequence of resolutions; or on a more coarse-grained *multi-vertex* level as pioneered by Lounsbery et al. [3], resulting in a discrete sequence of resolutions.

Pioneering work in *progressive mesh representations* was done by Hoppe [2], defining a mesh in terms of a base mesh and a sequence of vertex splits. In this work, Hoppe describes how a mesh can be simplified vertex per vertex, minimizing an energy function at every step. This results in a continuous LOD (cLOD) chain as introduced in Section 2.4.2.1: this progressive mesh representation generates a nearly continuous spectrum of LODs where each new level is obtained by splitting one vertex of the previous level. It provides the optimal mesh given a fixed triangle budget, but does not provide an efficient encoding solution. Pajarola and Rossignac [4] proposed refining this progressive mesh representation by grouping the individual increments in batches, each batch splitting half of the decoded vertices. This allows for better compression results but reduces the granularity.

Alliez and Desbrun [5] describe a valence-driven progressive compression approach which creates patches and retriangulates these in a deterministic fashion, layer by layer. Within each layer, a deterministic traversal order removes vertices, aiming at optimizing the valences of the vertices globally. Similarly, Maglo et al. [6] describe a progressive compression scheme for polygon meshes existing of polygons in general, i.e., not restricted to triangular faces. The state of the art in cLOD codecs is given by the IPR-codec of Valette et al. [7]. In this work the authors present a compression scheme which predicts the edge splits and the required precision for each vertex. The prediction of which edge to split at every step ensures a visually pleasing result by keeping the triangle areas similar to each other. However, keeping these areas similar results in undersampling near high-frequency regions, which is why the preservation of geometric features cannot be guaranteed. Finally, Peng et al. also considered the importance of feature preservation in their *Feature Oriented Progressive Lossless Mesh Coder (FOLProM)* [8]. In their work, the authors achieved this goal by introducing feature-based prioritization of vertex split operations.

The idea of constructing scalable representations and compression systems is well-known in signal processing. In this context, wavelets play a major role,

being used to generate multiresolution representations of an input signal and to build scalable codecs based on them. Wavelet-based scalable codecs include well-known examples for images [9–11], for video [12–14], and were introduced for surfaces by Lounsbery et al. [3]. Essentially, Lounsbery et al. established the link between subdivision schemes and multiresolution analysis for meshes. Subdivision schemes result in *semi-regular* meshes; however, models are most efficiently represented using *irregular* meshes, allowing for adaptive sampling of a surface as described in Section 2.2.

To allow for adaptive sampling of a surface, the lossy remeshing step which constructs semi-regular meshes from irregular meshes needs to be avoided, and instead the original irregular mesh must be processed. Additionally, some applications do not allow for a lossy remeshing step. This reveals a trade-off: the superior representation efficiency offered by an adaptive vertex density can compensate for the inferior compression performance due to explicit connectivity information.

Few wavelet transforms for *irregular meshes* have been proposed in the literature. The issue was tackled by Bonneau [15] who described a generalization of Haar-wavelets for piecewise constant functions defined on irregular triangular meshes. Valette et al. [16, 17] describe a wavelet transform as an extension of the subdivision-based multiresolution analysis introduced by Lounsbery et al. in [3]. The main difference with the latter is that the downsampling procedure is not solely restricted to inverse subdivision, where four triangles of the higher resolution mesh are merged to form a single triangle in the lower resolution approximation. Rather, the simplification algorithm of Valette et al. can merge two, three or four triangles depending on the connectivity of the mesh, thus eliminating the need for semi-regular connectivity. As the downsampling procedure is no longer trivial, each simplification step must be stored to allow for reconstructing the wavelet-transformed mesh. Later, Valette and Prost proposed *Wavemesh* [18], a codec employing this wavelet transform for irregular meshes; this codec represents the state of the art in wavelet-based irregular mesh coding.

A comprehensive overview and classification of all scalable coding systems falls out of the scope of this dissertation; the interested reader is referred to one of the many in-depth overviews such as [19], [20], [21], [22] and more recently [23]. Whereas the proposed coder is *connectivity-based* just as the related work described above, i.e., the connectivity is improved after which the geometry is reconstructed, in this thesis *geometry-based* octree-based data structures are employed similar to how these are employed by the semi-regular mesh codec of Denis et al. [24]. Octree-based data structures have also been successfully used for *geometry-based* coding in the past, as shown by the state-of-the-art coder by Peng and Kuo [25]. Being geometry-based, such approaches can easily be extended to point clouds [26].

State-of-the-art coders mainly focus on optimizing their RD performance [27]. While optimizations in $L^2$ sense might yield optimized performance in the mean square sense, this does not necessarily imply that visual quality is optimal. In this respect, the importance of geometric features in a model needs to be emphasized. Human perception is focused on high-frequency changes [28], and as such, preservation of geometric features will better serve the perceived visual quality. Codecs have rarely considered this by design, often offering feature preservation by prohibiting reductions near features. Furthermore, one should question the focus of minimizing the *rate* as such. From a storage and transmission perspective the rate is of key importance, but for interactivity, efficient rendering requires minimizing the distortion for a given *triangle limit*, which is directly related to a given memory limit.

## 3.2 Designing a Core Resolution-Scalable Feature-Preserving Irregular Mesh Codec

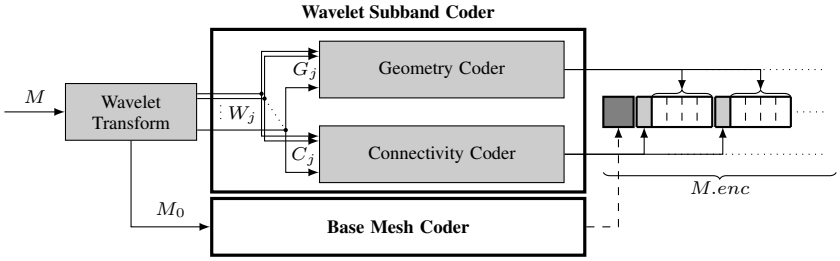



*Figure 3.1: Basic architecture of a wavelet-based mesh encoding system. A mesh $M$ is transformed into a base mesh $M_0$, and a set of wavelet subbands $W_j$ required to reconstruct intermediate meshes $M_j$. Each subband consists of both connectivity information $C_j$ for reconstructing the topology of the* irregular *mesh, and geometry information $G_j$ for reconstructing accurate geometry.*

The conceptual overview, as presented in Figure 3.1, depicts the major components of the proposed encoding system.

An input mesh is subjected to the proposed wavelet transform, resulting in a base mesh and a sequence of wavelet subbands. The base mesh can be encoded using any single-rate coder, such as the state-of-the-art coder of Touma and Gotsman [1]. The wavelet subbands require a connectivity coder and a geometry coder to encode the connectivity increments and the geometry refinements respectively. These data streams are multiplexed, resulting in a bit stream which is resolution scalable and, within each resolution, quality scalable. A key feature of this coding system will be implicit geometric feature preservation while allowing

for several forms of scalability. As such, the resulting coding system is named *SFWInCS*, the *Scalable Feature-preserving Wavelet-based Irregular mesh Coding System*.

Figure 3.2 shows several intermediate resolutions when applying the proposed codec to the 5-million vertex *Thai statue* model. The visual results show the growing importance of the lossy domain with increasing mesh sizes: while a complete lossless decoding requires 27 bpv, Figure 3.2c shows that very good approximations are possible at low bit rates. The proposed lossless codec aims to improve upon the state of the art at these lossy bit rates. Furthermore note that Figure 3.2c is obtained using only $13.5\%$ of the triangles. Contrary to many state-of-the-art wavelet-based coders where a low bit rate generates a large amount of triangles due to the regularity to which they owe their efficient compression, the proposed codec generates fewer triangles as the intermediate meshes have irregular topologies, this in turn resulting in a lower memory footprint for real-time rendering without impeding the visual quality (as discussed in Section 2.2).

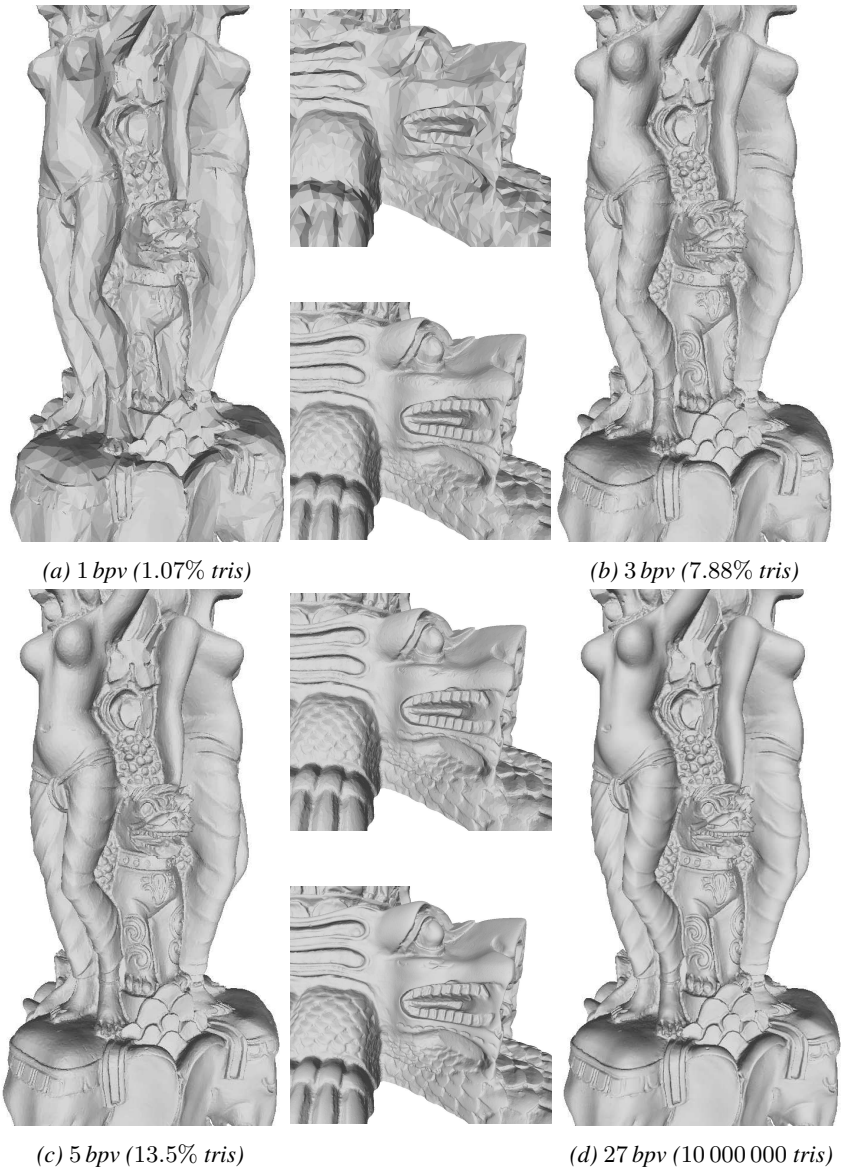
The next sections describe the main components of SFWInCS in more detail. Section 3.2.1 describes the wavelet transform employing a novel adaptive downsampling and retriangulation step to preserve geometric features, together with a two-mode feature-aware prediction step, while Section 3.2.2 discusses the encoding: Section 3.2.2.1 handles the novel encoding of the connectivity changes using an octree data structure, and Section 3.2.2.2 describes the proposed wavelet coding system for irregular meshes.

## 3.2.1 Wavelet Transform

The conventional wavelet transform used in mesh coding is given in Figure 3.3, depicting the transform of a mesh at resolution $j$ to a mesh at resolution $j-1$ and a corresponding wavelet subband. This scheme is generic and stems from the classical synthesis of wavelet transforms based on lifting [29, 30], and is discussed for irregular meshes in Section 3.2.1.1. The novel contributions are discussed in the subsequent sections.

### 3.2.1.1 Conventional Wavelet Transform for Irregular Meshes

In general, the lifting scheme proposes three steps in transforming input data: a high-pass and low-pass filtering step, a prediction step, and an update step. In the filtering step the vertices of $M_j$ are partitioned in odd and even vertices $M_j^o$ and $M_j^e$, respectively. After downsampling to preserve only the even vertices $M_j^e$, retriangulating the connectivity and updating the vertex positions, the vertices $v_j^{e_i} \in M_j^e$ form the vertices $\gamma(v_j^{e_i}) = v_{j-1}^i \in M_{j-1}$, where $\gamma$ is the bijective operator that maps $M_j^e$ to $M_{j-1}$. To allow for reconstructing $M_j$ by upsampling the lower-resolution mesh and refining the odd vertices, connectivity



*(a)* 1 *bpv (*1.07% *tris)* *(b)* 3 *bpv (*7.88% *tris)*

*(c)* 5 *bpv (*13.5% *tris)* *(d)* 27 *bpv (*10 000 000 *tris)*

*Figure 3.2: Rendered model:* Thai statue. *Several resolution levels of this 5-million-vertex model, encoded with* 16 *bit quantization. The middle column shows close-ups of the pedestal of the statue, at the same bit rates.*

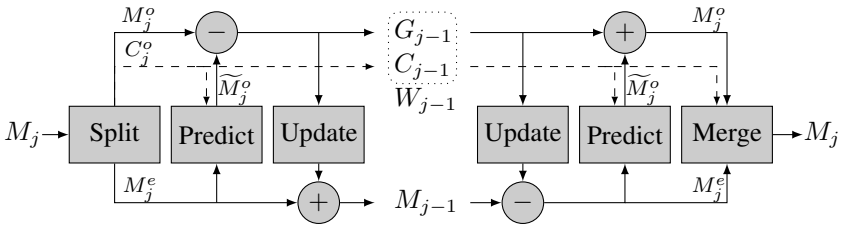



*Figure 3.3: Overview of the wavelet transform. During the analysis step, a mesh $M_j$ is decomposed into a lower-resolution mesh $M_{j-1}$ and wavelet coefficients $W_{j-1}$. To allow for irregular mesh coding, additional connectivity information has to be provided (denoted by $C_{j-1}$) in addition to the geometry information represented as conventional wavelet coefficients (denoted by $G_{j-1}$). The sets of even and odd vertices after partitioning are respectively indicated by $M_j^e$ and $M_j^o$, with additional connectivity information represented by $C_j^o$; the set of* predicted *odd vertices is indicated by $\widetilde{M}_j^o$.*

and geometry information is required in the form of the wavelet subband $W_{j-1}$. Define the wavelet coefficients corresponding to the odd vertices $v_j^{o_i} \in M_j^o$ as $\omega(v_j^{o_i}) = w_{j-1}^i \in W_{j-1}$, where $\omega$ is the bijective operator that maps $v_j \in M_j^o$ to $w_{j-1} \in W_{j-1}$.

For semi-regular meshes, connectivity information is implicit, and needs not be transmitted. To accommodate for irregular meshes, the dashed lines in Figure 3.3 have been added to indicate that connectivity information can no longer be assumed to be implicitly known for this type of meshes. This connectivity information is represented by $C_j^o = C_{j-1}$, and the need for explicitly representing this information is common to all irregular mesh codecs.

Hence, for irregular meshes each wavelet subband encompasses the *geometry information* represented by the *conventional* wavelet coefficients $G_{j-1}$ obtained via lifting, together with explicit *connectivity information* $C_{j-1}$ to ensure the correct connectivity. The connectivity information $C_{j-1}$ and geometry information $G_{j-1}$ are denoted together as the wavelet subband $W_{j-1}$.

The reconstruction of the high-resolution mesh $M_j$, given the low-resolution mesh $M_{j-1}$ and wavelet subband $W_{j-1}$, is denoted as:

$$M_j = WT^{-1}(M_{j-1}, W_{j-1}), \tag{3.1}$$

where $WT^{-1}$ denotes the inverse wavelet transform.

Denote by $w_c \in C_{j-1}$ and $w_g \in G_{j-1}$ the connectivity and geometry information of wavelet coefficient $w_{j-1}^i$ respectively. To simplify notations, the index $i$ and subband $j-1$ are not explicitly given for these two variables. In the inverse wavelet transform, each $v_j^{o_i} = \omega^{-1}(w_{j-1}^i)$ is reconstructed as follows:

$$v_j^{o_i} = \Big( \sum_{k \in |M_{j-1}|} w_c^k v_{j-1}^k \Big) + w_g = \widetilde{v}_j^{o_i} + w_g, \tag{3.2}$$

where $|\cdot|$ is the cardinality of a set. In this equation, the first term represents the '*Predict*' step in the wavelet transform in Figure 3.3, weighing all vertices $v_{j-1}^k$ of the lower-resolution mesh by weights given by the connectivity information in $w_{j-1}^i$. This results in the predicted vertex $\widetilde{v}_j^{o_i} \in \widetilde{M}_j^o$. The second term corrects this prediction. Define now the support $S(w_{j-1}^i)$ for a wavelet coefficient $w_{j-1}^i$ as the set of vertices with a non-zero weight in Equation 3.2:

$$S(w_{j-1}^i) = \{v_{j-1}^k \in M_{j-1} \mid w_c^k \neq 0\}. \tag{3.3}$$

That is, $S(w_{j-1}^i)$ corresponds to the set of vertices $v_{j-1}^k \in M_{j-1}$ required for reconstructing $v_j^{o_i} = \omega^{-1}(w_{j-1}^i)$.

The next sections discuss these steps in more detail. In Section 3.2.1.2, the partitioning of vertices into even and odd vertices is discussed, followed by the retriangulation after downsampling. Section 3.2.1.3 then tackles the prediction and update steps, and the reconstruction is discussed in Section 3.2.1.4.

#### 3.2.1.2 Signal-Adaptive Downsampling and Retriangulation

The first step of the wavelet transform is splitting the mesh information into lower-resolution surface information and higher-frequency detail information. This is done by first splitting the vertices in even and odd vertices. After downsampling, which preserves the even vertices and eliminates the odd ones, the resulting polygonal patches are retriangulated to form a lower-resolution triangle mesh. In the subsequent step, discussed in Section 3.2.1.3, the lower-resolution information is used to predict the detail information.

**Vertex Partitioning** First, the vertices of $M_j$ are partitioned into the two subsets $M_j^e$ and $M_j^o$ of even and odd vertices respectively. Vertices of the input mesh are labeled in the following manner.

Choose an arbitrary start vertex $v_j^S$ with a valence greater than 3 (for reasons that will become clear in Section 3.2.2.1 on connectivity coding), and mark it as odd. Next, iterate over the neighbors of $v_j^S$, marking them as even vertices. Push all unlabeled neighbors of these newly marked even vertices on a queue which keeps track of vertices that still need to be processed. Figure 3.4b illustrates an odd vertex (depicted in green as it is the start vertex $v_j^S$) and its neighbors, classified as even vertices. When all neighbors of $v_j^S$ have been marked, and their unlabeled neighbors are added to a queue, as shown in Figure 3.4c, the labeling procedure is repeated for the first element in the queue. The algorithm halts when the queue is empty and no more vertices need to be processed. These steps are formalized in Algorithm 3.1.

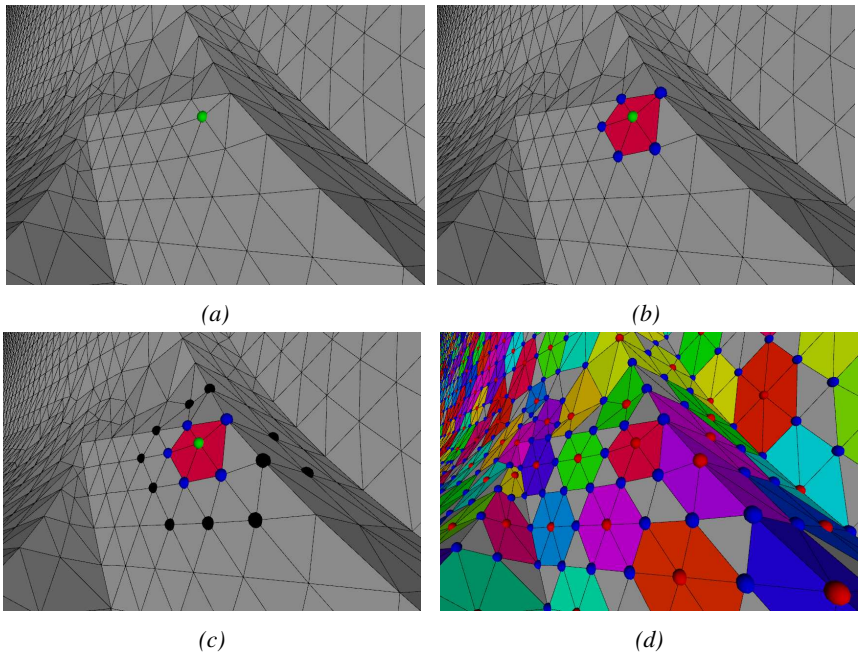



*Figure 3.4: Selecting odd (red) and even (blue) vertices. (a) shows an arbitrary start vertex $v_j^S$, depicted in green, which is considered as the first odd vertex. In (b), its neighbors $v_j^{S,i}$ are marked as even vertices, and (c) shows, in black, the vertices that are added to a queue. The first black vertex is considered as the next start vertex $v_j^S$ and the process is repeated. The resulting patches are shown in (d).*

Figure 3.4d shows the resulting patches. The colored triangles are merged to form patches, but due to the irregular nature of the connectivity, not all triangles are necessarily associated with a patch. The latter, drawn in gray, are referred to as *even polygons* as they are strictly made out of even vertices.

The largest possible reduction occurs if a mesh can be completely partitioned in quadrilateral patches. This can reduce the amount of triangles by half at most, resulting in a higher granularity compared to other wavelet-based coders which often aim at $4$-to-$1$ reduction. Observe furthermore that improved results can be obtained by steering the vertex partitioning instead of employing the current approach using a random start vertex and a first-in-first-out queue. A better vertex partitioning can improve the downsampling efficiency by aiming for a better patch coverage over the surface, the TD performance by improved overall triangulation, or the RD performance by providing better meshes for encoding.

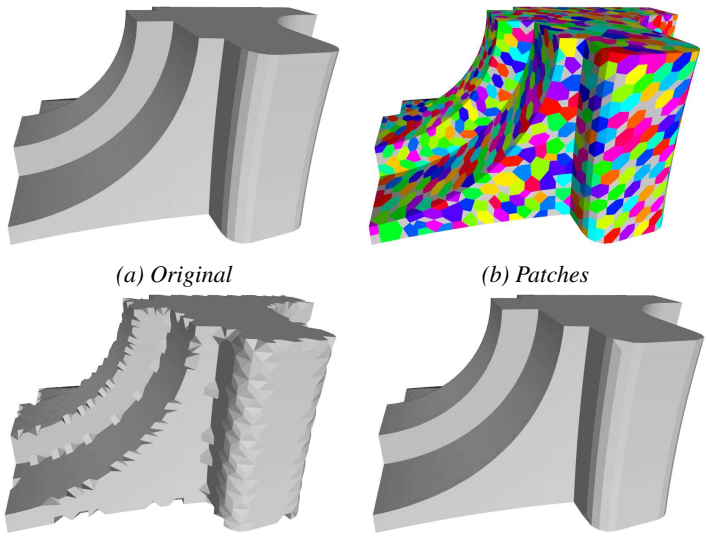
Because the connectivity information is straightforward and implicit in most wavelet schemes, e.g., compression of images or semi-regular meshes, the triangulation of $M_j^e$ is never emphasized in any of the papers describing lifting.

However, for irregular meshes this triangulation requires proper attention, and requires explicit information to be stored to allow for reconstructing the high-resolution mesh. This connectivity information $C_j^o$ describes the patches in the retriangulated low-resolution mesh, as discussed next.

$\mathcal{O} \leftarrow \varnothing$ ▷ odd vertices
$\mathcal{E} \leftarrow \varnothing$ ▷ even vertices
$\mathcal{U} \leftarrow \{v : v \in M_j\}$ ▷ unlabeled vertices
$\mathcal{Q} \leftarrow \varnothing$ ▷ vertex queue
$v^S \leftarrow v_{\text{random}} \in M_j$
**while** $\mathcal{U} \neq \varnothing$ **do**
    $\mathcal{U} \leftarrow \mathcal{U} \setminus \{v^S\}$
    $\mathcal{O} \leftarrow \mathcal{O} \cup \{v^S\}$
    **for all** $i \in [1, \nu(v^S)]$ **do**
        $v^N \leftarrow v^{S,i}$
        $\mathcal{U} \leftarrow \mathcal{U} \setminus \{v^N\}$
        $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{v^N\}$
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{v^N\}$
    **end for**
    **for all** $i \in [1, \nu(v^S)]$ **do**
        $v^N \leftarrow v^{S,i}$
        **for all** $k \in [1, \nu(v^N)]$ **do**
            **if** $v^{N,k} \in \mathcal{U}$ **then**
                $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{v^{N,i}\}$
            **end if**
        **end for**
    **end for**
    $v^S \leftarrow \mathcal{Q}[0]$
    $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{v^S\}$
**end while**

*Algorithm 3.1: Vertex partitioning*

**Adaptive Retriangulation** After downsampling, the resulting mesh given by the vertices $M_j^e$ has to be retriangulated to obtain a triangular mesh. As the patch borders and the even polygons already make up for a large portion of the connectivity of the lower resolution approximation, the triangulation problem is reduced to triangulating each patch individually. The patches are similar to those generated by Cohen-Or et al. in [31]. Contrary to [31], however, the retriangulation is adapted to the underlying geometry, and allows for *any* retriangulation. Figure 3.5 illustrates how different triangulations of the patches can dramatically affect the visual quality of the lower resolution approximation. The original mesh is depicted in Figure 3.5a. Figures 3.5c and 3.5d show the result when removing the odd vertices and applying two different retriangulations on the patches shown in



*(a) Original* *(b) Patches*

*(c) Feature-disrupting retriangulation* *(d) Feature-preserving retriangulation*

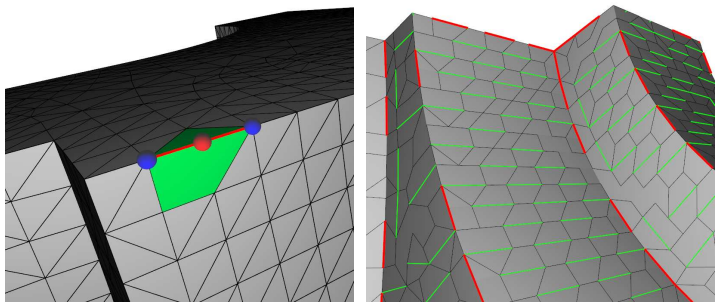
*Figure 3.5: Importance of feature-preserving patch retriangulation.*

Figure 3.5b. Though both meshes have the same resolution, it is clear that Figure 3.5d resembles 3.5a much more closely, demonstrating the importance of a proper, feature-preserving triangulation of the patches.

Ultimately, the goal is to preserve the geometric features of the original mesh. The term *geometric feature* is defined as a portion of the mesh that determines in great extent the visual appearance of that mesh. More concretely, features are sharp edges that introduce large discontinuities in the surface normals, greatly influencing the final renderings of the mesh. Hence, the key to preserving geometric features and thus, maintaining object fidelity, is to assure that those normal discontinuities are kept. The following discusses how to achieve the latter, by introducing a simple, yet very efficient criterion.

Consider the odd vertex lying on a geometric feature, as depicted in Figure 3.6a. From the figure, the key observation is that the normal discontinuity inside the green patch can be captured after removing the odd vertex by ensuring the existence of the red colored edge, constructed by connecting the two even vertices indicated in the figure. The main idea of the retriangulation algorithm is to identify for *every* patch a feature *candidate*; this is done in the following manner.

Define the edge connecting vertices $v^a$ and $v^b$ as $E(v^a, v^b)$. Define



*(a)* *(b)*

*Figure 3.6: Feature candidates. (a) depicts an odd vertex along a geometric feature. After retriangulation, the red edge is the* feature candidate *for the green patch, splitting the patch in two sub-patches. In (b), the feature candidate of each patch is depicted. The colors are merely illustrative, indicating whether the dihedral angle along each feature candidate surpasses* $45^\circ$ *(red) or not (green).*

furthermore the distance from $v$ to $E(v^a, v^b)$ by

$$d\Big(v, E(v^a, v^b)\Big) = \frac{|(v - v^a) \times (v - v^b)|}{|v^b - v^a|}. \tag{3.4}$$

This equation makes use of the fact that the magnitude of a cross product of two vectors is the area of the parallelogram spanned by these two vectors; this is illustrated in Figure 3.7.

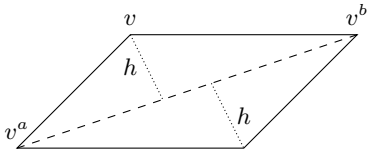



*Figure 3.7: Vertex-to-edge distance. To find h, i.e., the distance from v to* $E(v^a, v^b)$*, one can compute the area of the parallelogram spanned by* $E(v, v^a)$ *and* $E(v, v^b)$ *and divide this area by the base of the two formed triangles,* $|E(v^a, v^b)|$*.*

Each patch consists of a center odd vertex $v^o$ and a neighboring ring of even vertices $v^{o,k}$, with $k \in [1, \nu(v^o)]$, where $\nu(v)$ denotes the valence of vertex $v$. The feature candidate $FC(v^o)$ corresponding to the given patch around $v^o$ is the edge which minimizes the Euclidean distance between $E(v^{o,k_a}, v^{o,k_b})$ and $v^o$ . Mathematically, this is expressed as: $FC(v^o) = E(v^{o,k'_a}, v^{o,k'_b})$, with

$$(k'_a, k'_b) = \underset{(k_a, k_b)}{\arg\min} \quad d\Big(v^o, E(v^{o,k_a}, v^{o,k_b})\Big). \tag{3.5}$$

Figure 3.6b shows the result obtained when using this equation in the triangulation procedure. This approach does not require introducing an explicit threshold to actually differentiate between feature candidates lying along actual geometric features and candidates lying within a planar surface; for illustrative purposes the edges along which a large dihedral angle (i.e., larger than $45°$) was found are displayed in red, while the other candidates are displayed in green. The figure clearly indicates that the feature candidates for patches that actually embed geometric discontinuities, are capable of maintaining large normal discontinuities typifying the overall shape of the mesh without any threshold definition or feature detection preprocessing step.

*(a) No Delaunay triangulation* *(b) Valid Delaunay triangulation*

*Figure 3.8: Delaunay triangulation condition. This condition states that the circumcircles of all triangles must have empty interiors. In (a) it is not met as the red vertex is located within the dashed circle, i.e, within the circumcircle of the hatched triangle. In (b) a valid Delaunay triangulation is shown: no vertex lies inside the circumcircle of the right triangle as illustrated, while similarly no vertex lies in the circumcircle of the left triangle.*

Once the feature candidates are computed, the patches are retriangulated: as demonstrated by Figure 3.6a, a feature candidate divides its corresponding patch into two smaller sub-patches. Hence, triangulation of $M_j^e$ now consists of triangulating these sub-patches. Although this step only slightly impacts the overall visual quality of the mesh, it is wise to target a good triangulation in order to facilitate the extraction of the lower resolution approximations for further wavelet decompositions. In the implementation, a Delaunay triangulation [32] is targeted for the sub-patches. For vertices in 3D, this triangulation is obtained by projecting the vertices on their best fitting plane, performing Delaunay triangulation in this plane, and mapping the projected vertices back to their respective original vertices. A Delaunay triangulation optimizes the resulting triangles by avoiding long thin triangles in favor of near-equilateral triangles. Figure 3.8 depicts the Delaunay triangulation condition which must be met for all triangles. As both sub-patches are assumed to be nearly planar, few issues may arise when using this projection and triangulation. Care only has to be taken to avoid adding triangles when the projected vertices form a concave polygon. In this case, only the triangles created *within* the borders of the original patch will be used in the low-resolution patch.

In addition, whichever triangulation is used, triangle flips within the sub-patches are not allowed after retriangulation. Assuming that a sub-patch is relatively smooth, a triangle flip is defined as the appearance of a triangle whose

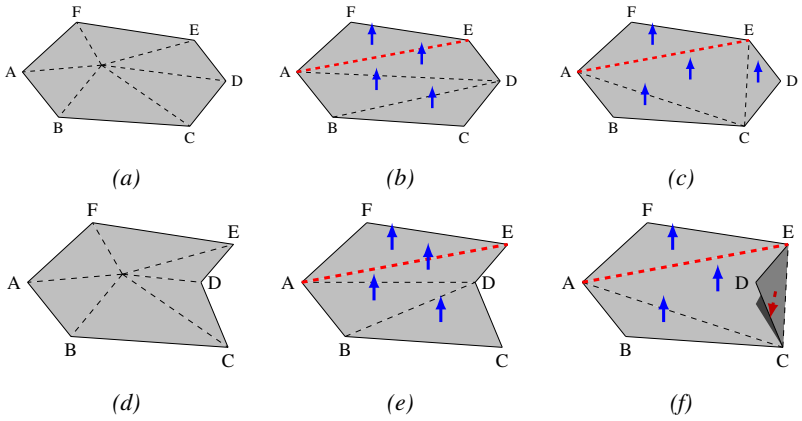



*Figure 3.9: Effect of geometry on triangulation. The top and bottom rows are topologically equivalent. (a) can be retriangulated as in (b) or (c), (d) as in (e) or (f). The red dashed line indicates the selected feature candidate. Although no real geometric features are present for these nearly-coplanar vertices, a wrong triangulation can introduce a sharp crease.*

normal is rotated more than $90°$ compared to the other normals within the sub-patch. Figure 3.9 illustrates this. Both rows share the same connectivity information, but depending on the actual geometry, retriangulations such as observed in Figure 3.9f are undesirable: this triangulation introduces a new geometric feature $CE$ between triangles $\Delta ACE$ and $\Delta ECD$, changing the underlying geometry. If the vertices are nearly coplanar, this will also result in overlapping triangles and visual artifacts when considering the triangulation of the neighboring patches. Hence, a triangulation as shown in Figure 3.9e should be preferred.

**Multiple-Feature Patches** Though the previously detailed feature candidates are very capable of maintaining the overall shape of the input mesh in most cases, there are some occasions where they prove to be insufficient. Consider Figures 3.10a and 3.10c, which depict two odd vertices lying on geometric corners of the surface. The results when removing those particular odd vertices and applying the triangulation procedure are given in Figure 3.10b and 3.10d respectively. As is obvious from the figures, the geometry of the original mesh is significantly altered in both cases. This stems from the fact that any retriangulation of patches can only accommodate for at most one normal discontinuity. In both Figures 3.10a and 3.10c the normals of the polygons incident to the odd vertex show three normal discrepancies. As is illustrated by Figures 3.10b and 3.10d, respectively none or at most one of the discrepancies can be captured by a feature candidate in the

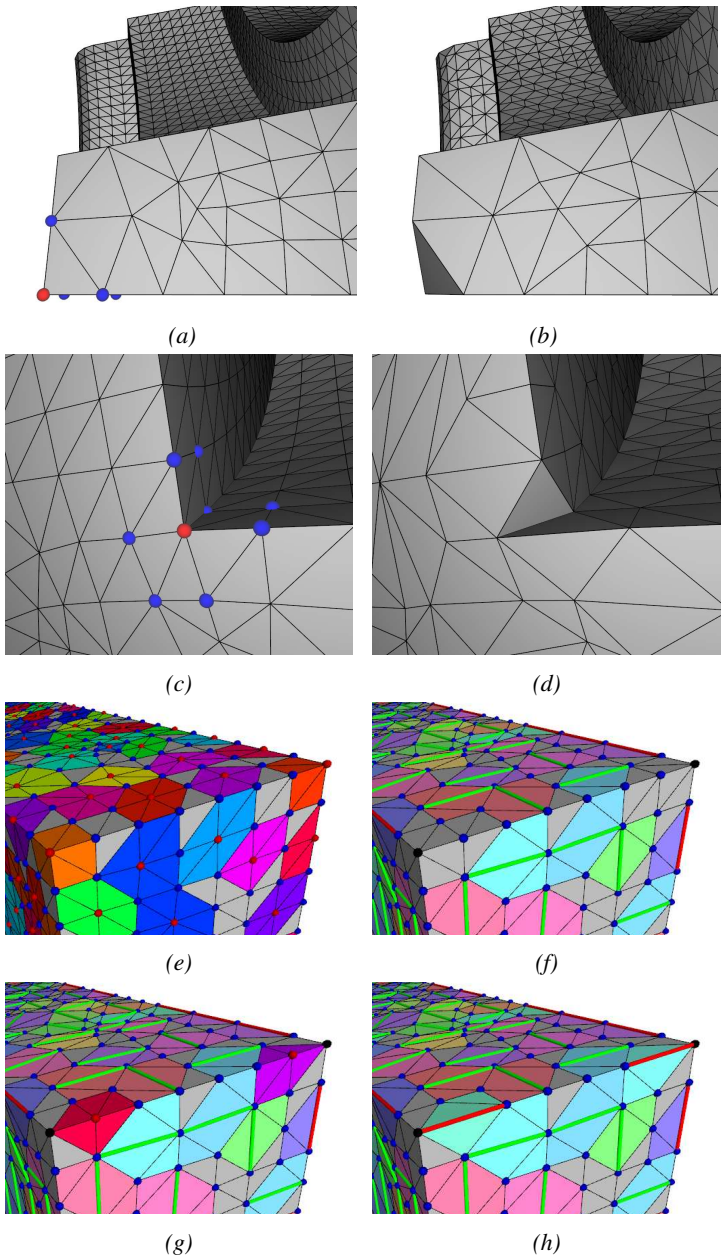



*Figure 3.10: Multiple-feature patches. Removing the odd vertex shown in (a) and (c) alters the visual appearance of the mesh due to aliasing, as is illustrated by (b) and (d), respectively. (e) – (h) show how new patches are found around multiple-feature vertices.*

retriangulated patch. Currently, large normal discontinuities are detected based on a dihedral angle threshold $\tau_{\text{MF}}$: in the original patch, only two edges leaving the odd vertex can lie along a large normal discontinuity; if more dihedral angles surpassing $\tau_{\text{MF}}$ are found, these patches are referred to as multiple-feature patches, and the odd vertex will be preserved in $M_j^e$. Depending on the defined patches in the neighborhood of the odd vertex, alternative patches can possibly be found at the same resolution which now define this vertex as an even vertex around a neighboring vertex that is allowed to change its label to 'odd'. This is shown in Figures 3.10e – 3.10h; the new patch retriangulations now appropriately preserve features. As before, the feature candidate colors are again purely illustrative to distinguish "sharper" feature candidates; these colors have no significance for the wavelet transform.

The effect of the threshold $\tau_{\text{MF}}$ has not been investigated rigorously, yet the value of $\tau_{\text{MF}} = 45^\circ$ has resulted in adequate feature-preservation. The effects of choosing this value too high or too low can be understood intuitively. Setting threshold $\tau_{\text{MF}}$ too large will result in artifacts as shown in Figures 3.10b and 3.10d, i.e., multiple-feature patches go undetected. Setting threshold $\tau_{\text{MF}}$ too low will limit the simplification performance as too many odd vertices are preserved. This approach can become sensitive to noise, resulting in an overestimation of the amount of multiple-feature patches and consequently a reduced downsampling efficiency. Noise-resilience can be improved by denoising and more advanced feature detection techniques. Lei He and Scott Schaefer [33] have shown impressive results. However, investigating such techniques is left as topic of further investigation; currently the models being considered are manually crafted, such as high-complexity CAD models, video game assets or animation film assets. In these cases, *implicitly preserving* a single feature within a patch, and *explicitly filtering* multiple-feature patches using dihedral angles and threshold $\tau_{\text{MF}}$, suffices.

#### 3.2.1.3 Two-Mode Feature-Aware Prediction without Update

Following the *downsampling and retriangulation* (Section 3.2.1.2), indicated in Figure 3.3 by the *Split* block, the proposed *prediction* and *update* steps of the lifting-based multiresolution analysis are discussed next. The high-frequency data is predicted using the local patch information: the lower-resolution mesh $M_j^e$ and the information for upscaling the mesh connectivity as provided by $C_j^o$ together allow for predicting odd vertices indicated by $\widetilde{M}_j^o$ in Figure 3.1, determining a prediction error per patch. Good prediction leads to a wavelet representation with low entropy, which in turn is of particular interest for compression. A predictor with two operating modes is proposed. If the largest dihedral angle in the low-resolution patch surpasses the feature prediction threshold $\tau_{\text{FP}}$, the new vertex is predicted along the appropriate edge. If not, then no obvious geometric features are detected and the new vertex is predicted as the average of the positions of the

patch border vertices. For both operating modes, more advanced predictors will improve the performance even further.

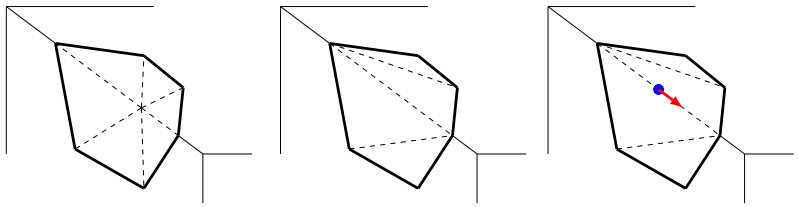
The coding results have not been evaluated extensively w.r.t. this threshold value $\tau_{\text{FP}}$, but short experiments have shown that $\tau_{\text{FP}}= 45^\circ$ is an acceptable threshold for obtaining good compression results. Yet, this value could be optimized for each model individually; even more, an optimal $\tau_{\text{FP}}$ could be determined *per resolution level*. The effect of this threshold is obvious. By choosing $\tau_{\text{FP}}$ too high, some geometric features are not detected and the prediction as a patch average will result in larger prediction error values. Choosing $\tau_{\text{FP}}$ too low results in detecting random features when no actual geometric feature is embedded, again resulting in larger prediction error values. Note that by determining whether an edge is a geometric feature based solely on the low-resolution mesh, both the encoder and decoder can do this same prediction without communicating feature candidates.

Additionally, although theoretically and practically feasible, the choice was made not to include the update step in the multiresolution analysis because of the following reasons. In the original lifting-based designs, updating was introduced to avoid aliasing. However, by preserving odd vertices in multiple-feature patches aliasing artifacts are already reduced. Another objective of the update step is to smooth the geometry in $M_{j-1}$, making the lower resolution approximation more visually appealing. However, in the case of feature-rich meshes, smoothing $M_{j-1}$ would also lead to a blurring of sharp corners such as the ones depicted in Figures 3.10a and 3.10c, hence, severely altering the geometry of the original mesh. Altering sharp discontinuities is totally unacceptable; the objective is to generate high quality low-resolution meshes. Also note that the wavelet transforms used in the state of the art in semi-regular mesh coding - see [24] - do not employ the update step.

#### 3.2.1.4 Reconstruction

Figure 3.11 illustrates the wavelet transform for a single patch which embeds a geometric feature. Figure 3.11b shows the preservation of a feature due to the adaptive retriangulation, and Figure 3.11c shows the prediction of the odd vertex (blue) and the accompanying wavelet coefficient (red). Due to the presence of the geometric feature in the low-resolution mesh, the new vertex will be predicted along this feature, i.e., along the middle internal edge.

Reconstruction merely involves inserting a new vertex for each patch and recovering the original triangulation. As detailed above in Section 3.2.1.3, predicting the position of the new vertex is based on the presence of geometric features in the low-resolution mesh, predicting the vertex either at the midpoint of a geometric feature or as the average of the positions of the patch border vertices, depending on the threshold $\tau_{\text{FP}}$. In both cases, the new vertex $\widetilde{v} \in \widetilde{M^o}$ is refined by



*(a) High-resolution patch* *(b) Downsampling & retriangulation* *(c) Prediction & wavelet coefficient*

*Figure 3.11: Conceptual example of the wavelet transform.*

displacing it, given the associated prediction error stored in the form of a wavelet coefficient as shown in Equation 3.2. The connectivity of the high-resolution mesh is rebuilt by discarding the triangulation inside the patches of the low-resolution mesh, and connecting the newly inserted vertices with their corresponding patch border vertices.

### 3.2.2 Wavelet Subband Coding

As depicted in Figure 3.1, the wavelet transform results in a base mesh $M_0$ and a set of wavelet subbands $W_j$. The next sections describe the encoding of the wavelet subbands. First the encoding of the connectivity information $C_j$ of these subbands, which allows for properly upsampling irregular meshes, is described. This is followed by a description of the encoding of the geometry information $G_j$, required for accurately refining the odd vertices per resolution.

#### 3.2.2.1 Connectivity Coder

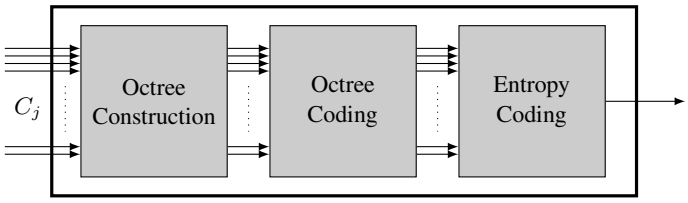



*Figure 3.12: Overview of the connectivity coder.*

As detailed in Section 3.2.1, connectivity information comes in the form of patches that have been constructed. Recall that, after downsampling a mesh $M_{j+1}$, it has to be retriangulated to obtain a *triangular* mesh $M_j$. This retriangulation step is required because the downsampling step results in a polygon mesh consisting not only of triangles, but also of quadrilaterals, pentagons, hexagons, etc. Each

of these polygons corresponds with a single patch. Hence, the patches can be recovered by identifying which edges where available in the downsampled polygon mesh, and which edges were created when retriangulating these polygons. These edges are identified using a single bit per edge. Contrary to similar patch-based coders, e.g., the work of Cohen-Or et al. [31], this approach does not pose any restrictions on the retriangulation process, as reconstruction will always be possible.

There is one caveat: when a patch in the high-resolution mesh consists of three triangles, i.e., the odd vertex had valence three, it is merged into one triangle, making it indistinguishable from even triangles that were simply preserved when downsampling. To avoid the overhead of signaling these triangular patches separately, such patches are not allowed to be created in the first place by disallowing valence-three-vertices to be marked as odd vertices.

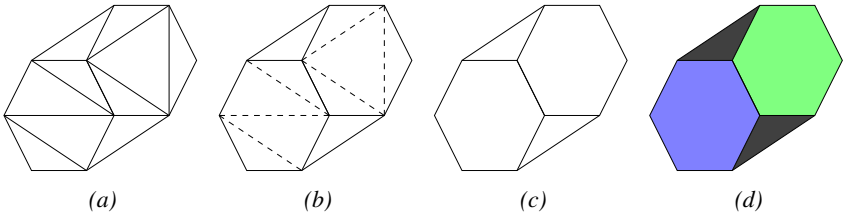



*Figure 3.13: Patch recovery with higher-order polygons allowed.*

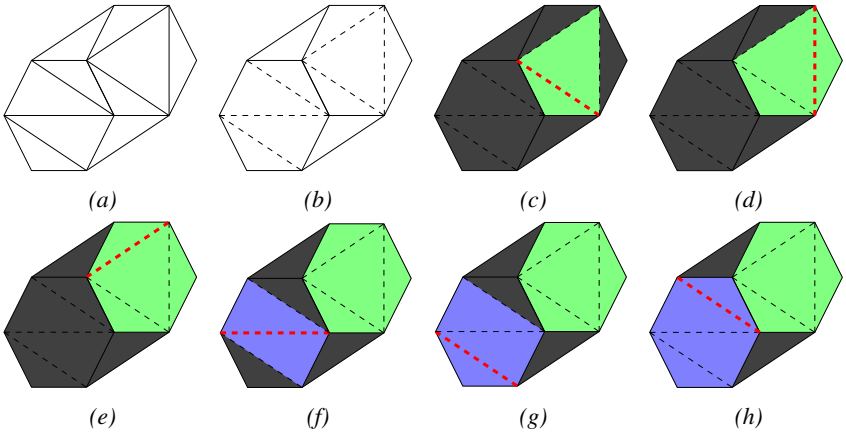



*Figure 3.14: Patch recovery with a triangle mesh.*

To recover the patches when given the edges that were obtained through retriangulation, several approaches are possible. If a data structure is employed which allows for quadrilaterals, pentagons, hexagons, etc. in the polygon mesh

representation, then these edges simply need to be dissolved, merging triangles into higher-order polygons. A traversal of the mesh then collects all higher-order polygons, corresponding with the patches that need to be upsampled. This is illustrated in Figure 3.13.

If the data structure only allows for triangles, then patches can be recovered by realizing that the two neighboring triangles of each given edge are part of the same patch. Figure 3.14 depicts this.

**Connectivity Octree Construction** To encode the connectivity information, represented as a single bit per low-resolution edge as discussed above, the low-resolution mesh $M_j$ needs to be traversed to visit each edge. In this case, one has to take care to ensure that both the encoder and decoder apply the same traversal order, ensuring that edges are visited in the exact same order such that the decoder assigns the binary values to the appropriate edges. Alternatively, to avoid defining and imposing any traversal order, in this dissertation the connectivity information is encoded in an octree data structure, which has been successfully used for *geometry* coding (e.g., [25]). Such an octree structure makes use of the underlying low-resolution geometry to assign a spatial location to each edge, and the binary values are encoded by iteratively subdividing the space spanned by these locations into eight smaller subcells. While this approach does not significantly affect the connectivity coding performance, it allows for implementation optimizations as subcells can be processed in parallel. The octree construction is detailed next.

The binary values associated with the edges are embedded in spatial cells as follows. When denoting the components $\mathbf{p}_x$, $\mathbf{p}_y$ and $\mathbf{p}_z$ of a point $\mathbf{p}$ by $p_0$, $p_1$ and $p_2$, then a spatial cell in $\mathbb{R}^3$ can be compactly defined as

$$C(\mathbf{k},\mathbf{u}) = \{\mathbf{p} \in \mathbb{R}^3 | p_i \in [k_i, k_i + u_i[, \text{for } 0 \leq i \leq 2\}, \mathbf{k}, \mathbf{u} \in \mathbb{R}^3. \tag{3.6}$$

Such a spatial cell is illustrated in Figure 3.15a, encompassing all points $\mathbf{p}$ which are located within the drawn box. Furthermore, a partitioning rule is defined to iteratively segment spatial cells $C(\mathbf{k},\mathbf{u})$ in eight subcells $C(\mathbf{k}^*,\mathbf{u}/2)$ with $\mathbf{k}^* = (k_0 + \gamma_0 u_0, k_1 + \gamma_1 u_1, k_2 + \gamma_2 u_2)$ and each $\gamma_i \in \{0, 1/2\}$. This is illustrated in Figure 3.15b: a cell $C(\mathbf{k},\mathbf{u})$ is refined into eight smaller cells, where the particular $\mathbf{k}^*$ in Figure 3.15b is obtained with $\gamma_0 = 0, \gamma_1 = 1/2$ and $\gamma_2 = 0$.

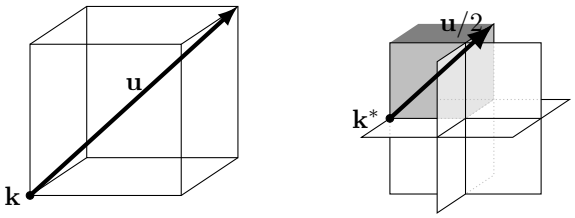
This octree construction can now be used for encoding the connectivity information. First, samples are taken at the midpoints of the edges: that is, let $e_j^i$ be the edge defined by $v_j^{i_a}$ and $v_j^{i_b} \in M_j$, then the embedding of the sample $\beta(e_j^i)$ for the edge $e_j^i$ is defined such that:

$$\beta(e_j^i) \in C(\mathbf{k},\mathbf{u}), \text{ if } \left[\frac{1}{2}v_j^{i_a} + \frac{1}{2}v_j^{i_b}\right] \in C(\mathbf{k},\mathbf{u}). \tag{3.7}$$

Now define $C_{root} = C(\mathbf{k}_r,\mathbf{u}_r)$ such that all the samples are embedded within this cell, i.e., so that $\beta(e_j^i) \in C_{root}, \forall e_j^i \in M_j$. This can be done by considering

the bounding box of the mesh, as described in Section 2.3.1 (Figure 2.13 on page 38), and considering the lower bound $\mathbf{b}_{\min}$ as $\mathbf{k}_r$, and the bounding box diagonal $(\mathbf{b}_{\max} - \mathbf{b}_{\min})$ as $\mathbf{u}_r$.

The cell refinement is guided by the significance of the samples. A sample is considered *significant* if the corresponding edge was added during retriangulation and hence will be dropped when reconstructing the high-resolution mesh. Cells are only refined when significant samples are present, and iterative refinement halts when the number of samples within a particular cell drops below a user-specified threshold $\lambda_C > 0$. The samples within such a *leaf cell* are ordered based on the x value of the edge midpoint, subsequently on its y value and finally on its z value, resulting in a deterministic ordering.



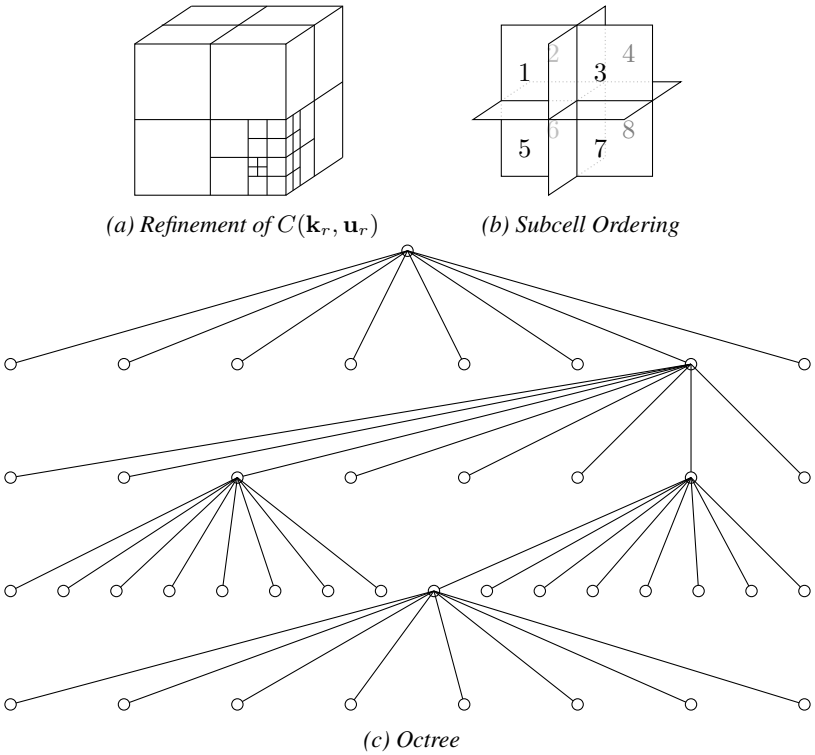

*(a) Spatial cell $C(\mathbf{k}, \mathbf{u})$* *(b) Cell refinement*

*Figure 3.15: Spatial cell refinement.*

**Example** An example of this is shown in Figure 3.16. When the connectivity tree construction is finished, the cell configuration can look, for instance, as depicted in Figure 3.16a. With the numbering convention as depicted in 3.16b, this octree can be visualized as in Figure 3.16c. There is at least one significant edge in $C_{root}$ such that it is split into eight new cells. As can be seen, both in Figure 3.16a and 3.16c, only the 7$^{\text{th}}$ cell is further refined; the other cells either contain *no* significant edges, or these are leaf cells, where the number of edges does not surpass $\lambda_C$. These steps are repeated: of this 7$^{\text{th}}$ cell, the 3$^{\text{rd}}$ and 7$^{\text{th}}$ are further refined and finally, of the latter refinement, the first cell is once more subdivided.

An illustration of an actual mesh with its samples embedded in an octree is shown in Figure 3.17.

**Octree and Entropy Coding** The octree, as constructed above, can now be traversed and its embedded samples encoded by finding significant samples. Starting from $C_{root}$, refinements will result in symbols $s_c$ and $n_c$ for significant and non-significant cells respectively, while the encoding of leaf cells will result in symbols $s_e$ and $n_e$ for significant and non-significant edges.



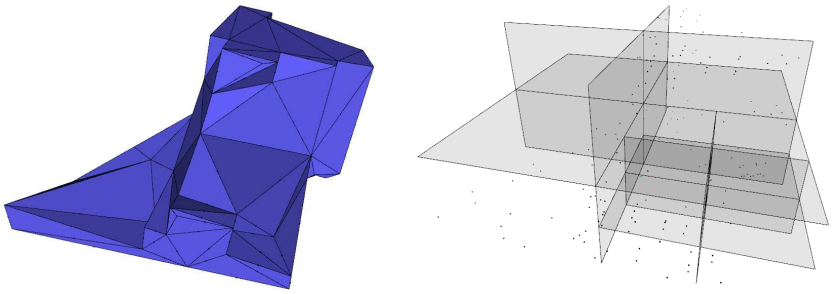

*(a) Refinement of $C(\mathbf{k}_r, \mathbf{u}_r)$* *(b) Subcell Ordering*

*(c) Octree*

*Figure 3.16: Example octree.*

At every point during the coding procedure a symbol of only one specific symbol pair is expected, either $s_c$ or $n_c$, or either $s_e$ or $n_e$. Consequently, no additional signaling is required to differentiate between these two binary signals. To encode this, *context-adaptive binary arithmetic coding (CABAC)* is used, as in several video coding standards [34]. In the connectivity codec, a separate context is used for each of the two binary signals, i.e., one context for the bits indicating cell significance, and a second context for the bits signaling edges being kept or dropped.

**Example (cont.)** The symbol stream for encoding the samples embedded in the octree of Figure 3.16c starts by $s_c$. For each of the first six subcells: either the number of samples in the cell surpasses $\lambda_C$ and a single symbol $n_c$ encodes all samples, or it is a leaf cell and each sample is encoded by a single symbol $s_e$ or $n_e$. Then the $7^{\text{th}}$ subcell is processed by encoding $s_c$ after which the steps above are repeated for the new subcells. In the end, the $8^{\text{th}}$ cell is encoded, again either by a single $n_c$ or by a symbol $s_e$ or $n_e$ per sample.



*(a) Low-resolution mesh* *(b) Sampling in an octree*

*Figure 3.17: Embedding connectivity in an octree. The samples, taken at the edge middles shown in (a) are used for constructing the octree depicted in (b). In this example there is only a single patch, located in the lower right corner (cell 7 as indicated in Figure 3.16b).*

The following gives a *possible* symbol stream, where the '␣' character is merely added for clarity, indicating the refinement level. Only four samples in leaf cells are encoded and are shown in boldface, two samples after three refinement steps and two samples after four refinement steps:

$s_c$,
␣ $n_c, n_c, n_c, n_c, n_c, n_c, s_c$,
␣␣ $n_c, n_c, s_c$
␣␣␣ $n_c, n_c, n_c, n_c, n_c, n_c, n_c, \mathbf{s_e}, \mathbf{n_e}$,
␣␣ $n_c, n_c, n_c, s_c$
␣␣␣ $s_c$,
␣␣␣␣ $\mathbf{n_e}, \mathbf{s_e}, n_c, n_c, n_c, n_c, n_c, n_c, n_c$,
␣␣␣ $n_c, n_c, n_c, n_c, n_c, n_c, n_c$,
␣␣ $n_c$,
␣ $n_c$.

As the decoder starts the procedure with the same low-resolution mesh, it obtains the same sample locations and consequently makes the same decision to either decode all samples in a cell (then, the binary digit is interpreted as either $s_e$ or $n_e$) or decode whether the cell is significant or non-significant (then, the digit is interpreted as either $s_c$ or $n_c$).

#### 3.2.2.2 Geometry Coder

When predicting the positions of the odd vertices (as detailed in Section 3.2.1.3), the deviations from the actual odd vertex positions are encoded as wavelet coefficients, of which the distribution in each subband is typically zero-mean Laplacian.

The geometry coder is based on the state-of-the-art SIM-codec for semi-

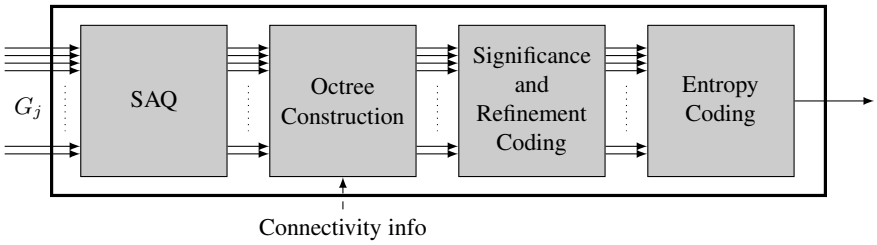



*Figure 3.18: Overview of the geometry coder.*

regular meshes [24]. It avoids interband coding techniques as these prohibit resolution scalability by traversing all subbands when encoding each bit plane. The SIM-codec is an intraband codec which also uses an octree structure, to exploit statistical dependencies between wavelet coefficients in each subband. It is the extension into 3D of quadtree-based coding techniques used in the past in the context of image coding [35].

**Quantization** In order to provide quality scalability per resolution, SAQ is used as discussed in Section 2.1.2, i.e., the significance of the wavelet coefficients are determined with respect to thresholds of the form $\tau_p = 2^p, p > 0$. The resulting quantization indices are encoded using a bit plane coder, as described further in the paragraph on *intraband bit plane coding*.

**Geometry Octree Construction** As the mesh vertices are quantized, the wavelet coefficients can be represented using a limited number of bits, implicitly determining their significance with respect to a series of monotonically decreasing thresholds of the form $2^p$. The proposed scheme stores the wavelet components in a hierarchical octree data structure, analogous to [24] and similar to the storage of connectivity bits as described in Section 3.2.2.1. Consider again $C(\mathbf{k}, \mathbf{u})$ defined in Equation 3.6. This spatial cell in $\mathbb{R}^3$ now embeds wavelet coefficients. Denoting $\omega(v_{j+1}^{o_i})$ as the wavelet coefficient corresponding to the odd vertex $v_{j+1}^{o_i}$, define the embedding such that

$$\omega(v_{j+1}^{o_i}) \in C(\mathbf{k}, \mathbf{u}), \text{ if } \widetilde{v}_{j+1}^{o_i} \in C(\mathbf{k}, \mathbf{u}). \tag{3.8}$$

Recall that $\widetilde{v}$ in this equation is the *predicted* position of $v$, which is determined at both the encoder and decoder side without additional geometry information whereas the *actual* position of $v$ is unknown for a decoder, prior to decoding the wavelet subband. As neighboring wavelet coefficients are encoded by the same octree cells, this approach is able to exploit spatial correlations and to

improve the compression performance by more efficient encoding of neighboring (non-)significance.

Contrary to the connectivity information where binary signals are trivially determined as being significant, an operator $\sigma^p(.)$ now determines the significance of a wavelet coefficient with respect to the quantization threshold $2^p$, mapping $\omega(v_j)$ to the binary value

$$\omega^p(v_j) = \sigma^p(\omega(v_j)) = \begin{cases} 0 & \text{if } |\omega(v_j)| < 2^p \\ 1 & \text{if } |\omega(v_j)| \geq 2^p \end{cases} . \tag{3.9}$$

This allows again for defining a partitioning rule segmenting $C(\mathbf{k}, \mathbf{u})$ in eight adjacent subcells $C(\mathbf{k}^*, \mathbf{u}/2)$ as described in Section 3.2.2.1. A cell is partitioned if at least one significant wavelet coefficient is present. This partitioning rule allows for constructing a hierarchical octree. The construction starts by creating the root $C_{root} = C(\mathbf{k}_r, \mathbf{u}_r)$ so that $\omega(v_j) \in C_{root}, \forall v_j \in M_j$. Next, $C_{root}$ is segmented using the partitioning rule and recursively applying it to all newly created subcells $C(\mathbf{k}^*, \mathbf{u}/2)$ until the number of coefficients in each cell is smaller than a user-specified threshold $\lambda_G > 0$. The wavelet coefficients within the leaf cells are ordered analogous to the ordering of connectivity samples as described in Section 3.2.2.1, i.e., based on the x-, y-, and z-values of the predicted positions $\widetilde{v}$.

**Intraband Bit Plane Coding** The wavelet coefficients, now stored in an octree data structure, are encoded in exactly the same way as described by Denis et al. [24] for *semi-regular* 3D mesh coding, i.e., using a significance pass, non-significance pass and refinement pass. This is similar to the *set partitioning in hierarchical trees (SPIHT)* coding approach proposed by Said and Pearlman [9] for image coding. Yet, whereas SPIHT exploits inter-resolution correlations for its set partitioning, the proposed coding approach performs set partitioning, i.e., octree cell refinement, only based on intra-resolution information.

The significance pass encodes whether a cell is significant or non-significant using symbols $s_c$ and $n_c$. When the number of coefficients drops under the user-defined threshold $\lambda_G$, this pass encodes whether the wavelet coefficients are significant or non-significant using symbols $s_w$ and $n_w$. The significant wavelet coefficients are stored in a refinement list, while the non-significant wavelet coefficients are stored in a non-significance list. Note that, as soon as a wavelet coefficient is found to be significant, its sign is encoded as being positive, $p_s$, or negative, $n_s$.

The non-significance pass traverses the non-significance list and encodes, with $s_w$ and $n_w$, whether or not each coefficient has become significant. If so, the coefficient is moved from the non-significance list to the refinement list, and its sign is encoded.

Finally, the refinement pass refines the significant wavelet coefficients in the refinement list using symbols $s_r$ and $n_r$.

Similar to the coding as described in Section 3.2.2.1, the symbols as obtained by the intraband coder are again encoded using CABAC. At every step in the coding process a symbol of only one symbol pair can be emitted, that is, either $s_c$ or $n_c$, either $s_w$ or $n_w$, either $p_s$ or $n_s$, or either $s_r$ or $n_r$. This allows the binary encoding of these symbols using four separate contexts, which can again be distinguished without additional signaling.

## 3.3 Introducing Template Meshes to Unlock Quality Scalable Irregular Mesh Coding

The wavelet transform and encoding in SFWInCS as discussed above in Section 3.2 allows for resolution scalability and only for a limited form of quality scalability: at each resolution, the decoder can determine the amount of quality bits for the newly added vertices; however, a new resolution can only be decoded after fully decoding the previous resolution. This is a disadvantage of using the lower-resolution mesh for embedding the connectivity information and wavelet coefficients in octrees. Dependencies between data blocks are shown in Figure 3.19a, clearly showing that this approach allows only one meaningful order in which data can be transmitted, depicted in Figure 3.19b.

Instead, quality-scalable encoding of irregular meshes has been explored where the wavelet subbands are encoded at possibly different quality levels. Moreover, the subband bit planes are encoded in an RD-optimal manner, ensuring minimal distortion in the reconstructed mesh for any target bit rate.

To allow for this, a so-called template mesh is introduced in order to decouple the transform step from the encoding step, and to allow for quality scalability. The necessity of such a template mesh originates from the octree encoding and the embedding of samples therein based on geometrical information; it distinguishes between the geometry of the transform step and the geometry used for embedding. This approach using template meshes was proposed in [24] for *semi-regular* meshes: proper decoding requires that the samples which are embedded within the octrees can be embedded unambiguously, making use only of data which is available at the decoder side.

Hence, for each subband $W_{j-1}$ a template mesh $M_{j-1}^T$ is maintained, which represents at least all connectivity information in the original mesh. This is the minimal information which is required for decoding without drift, independent of the reconstructed quality of $M$. As both $M_{j-1}$ and $M_{j-1}^T$ share the same connectivity information, there exists a bijective mapping $\mu$ linking the vertices

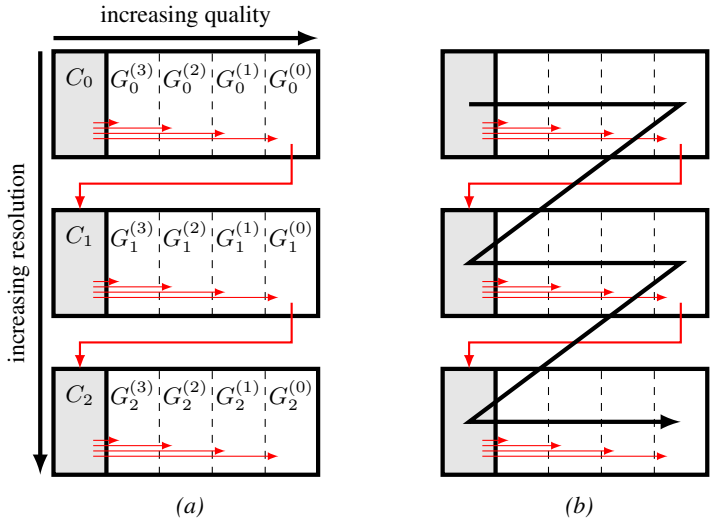



*Figure 3.19: Data dependencies for resolution-scalable coding. (a) shows dependencies in resolution-scalable coding, and (b) the only possible coding order. For resolution $j$, $C_j$ represents the connectivity data and $G_j^{(i)}$ the geometry data at bit plane $i$.*

of both, that is:

$$\forall v \in M_{j-1} : \mu(v) \in M_{j-1}^T, \tag{3.10}$$

$$\forall v^T \in M_{j-1}^T : \mu^{-1}(v^T) \in M_{j-1}. \tag{3.11}$$

The embedding of connectivity samples in an octree explicitly makes use of the mapping. Hence, Equations 3.7 and 3.8 become:

$$\beta(e_j^i) \in C(\mathbf{k}, \mathbf{u}), \text{ if } \left[\frac{1}{2}\mu(v_j^{i_a}) + \frac{1}{2}\mu(v_j^{i_b})\right] \in C(\mathbf{k}, \mathbf{u}), \tag{3.12}$$

$$\omega(v_{j+1}^{o_i}) \in C(\mathbf{k}, \mathbf{u}), \text{ if } \mu(\widetilde{v}_{j+1}^{o_i}) \in C(\mathbf{k}, \mathbf{u}). \tag{3.13}$$

That is, the embedding is determined by the geometry of the template mesh instead of the real mesh, via the mapping $\mu$.

Observe that the embedding, and consequently the encoding itself, only requires the information available at the current resolution. Hence, the connectivity information is required only up to the highest resolution where wavelet coefficients are being refined; the quality does not need to be equal over all subbands. This is an interesting difference with the work of Valette et al. [36], which proposes quality scalability for the irregular wavelet transform of [17] and requires all wavelet coefficients for its zerotree coding, thus requiring all connectivity information to be encoded before any geometry information can be processed.

### 3.3.1 Template Meshes for Resolution-Scalable Coding

When reformulating the approach as described in Section 3.2.2 following this *template* paradigm, the template mesh for reconstructing $M_j$ is an exact copy of the low-resolution mesh: $M_{j-1} = M_{j-1}^T$, and $\mu$ is the identity operator, leaving its argument unaltered. Denote by $\widetilde{M}_{j+1}$ the upsampled version of $M_j$ before refining the odd vertices. Equivalently, this would be the reconstruction of $M_{j+1}$ assuming that $w_g = 0 \, \forall w \in W_j$. Algorithm 3.2 reformulates the approach suggested in Section 3.2.2 using template meshes.

1: $M_0^T \leftarrow M_0$
2: **for all** $j \in [0, R-1]$ **do**
3: $\quad C_j = \text{DECODECONNECTIVITY}(M_j^T)$
4: $\quad \widetilde{M}_{j+1} = \text{UPSAMPLE}(M_j, C_j)$
5: $\quad \widetilde{M}_{j+1}^T = \text{UPSAMPLE}(M_j^T, C_j)$
6: $\quad G_j = \text{DECODEGEOMETRY}(\widetilde{M}_{j+1}^T)$
7: $\quad M_{j+1} = \text{REFINE}(\widetilde{M}_{j+1}, G_j)$
8: $\quad M_{j+1}^T = \text{REFINE}(\widetilde{M}_{j+1}^T, G_j)$ ▷ Note: $M_{j+1}^T \equiv M_{j+1}$
9: **end for**

*Algorithm 3.2: Resolution-scalable coding using template meshes*

By ensuring that the geometry of this template mesh $M_j^T$ is close to the geometry of the real mesh $M_j$, this approach remains able to exploit spatial correlations. The template mesh suggested above, which is a direct implementation of the resolution-scalable encoding as described in Section 3.2, matches the real mesh perfectly by construction (as $M^T = M$). This ensures optimal preservation of spatial correlations, but disallows full quality scalability. Only *within* a resolution, a decoder can decide on the reconstruction quality of the wavelet coefficients.

To allow for quality scalability, this template mesh construction is altered as described next.

### 3.3.2 Template Meshes for Quality-Scalable Coding

To allow for quality scalability, the following *geometry-agnostic* approach is proposed. The base mesh is used as a first template mesh again. However, instead of reconstructing each resolution using its previous resolution, effectively doing an inverse transform on the template mesh which is identical to the inverse transform performed on the real mesh, a modified inverse transform is performed which only uses connectivity information. This results in a template mesh which increases in resolution by *only predicting* the added vertices without requiring the decoded

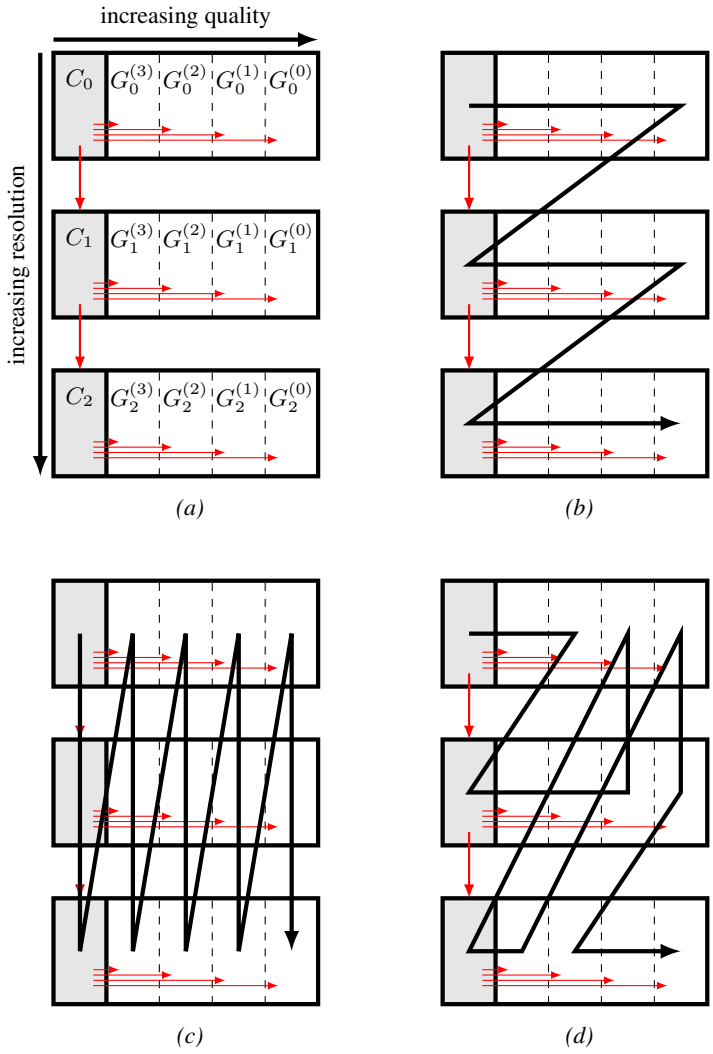



*Figure 3.20: Data dependencies for quality-scalable coding. (a) shows dependencies when quality scalability is allowed, (b), (c) and (d) show possible coding orders. For resolution $j$, $C_j$ represents the connectivity data and $G_j^{(i)}$ the geometry data at bit plane $i$.*

wavelet coefficients to refine these vertices. The result is that the connectivity and geometry of a resolution can be decoded as soon as the connectivity of the previous resolution is known. The adapted algorithm is given in Algorithm 3.3, and is very similar to Algorithm 3.2, only line 8 has been changed as this line introduced the dependencies which made quality scalability without drift impossible.

1: $M_0^T \leftarrow M_0$
2: **for all** $j \in [0, R-1]$ **do**
3: $\quad C_j = \text{DECODECONNECTIVITY}(M_j^T)$
4: $\quad \widetilde{M}_{j+1} = \text{UPSAMPLE}(M_j, C_j)$
5: $\quad \widetilde{M}_{j+1}^T = \text{UPSAMPLE}(M_j^T, C_j)$
6: $\quad G_j = \text{DECODEGEOMETRY}(\widetilde{M}_{j+1}^T)$
7: $\quad M_{j+1} = \text{REFINE}(\widetilde{M}_{j+1}, G_j)$
8: $\quad M_{j+1}^T = \widetilde{M}_{j+1}^T$ ▷ Note: $M_{j+1}^T \not\equiv M_{j+1}$
9: **end for**

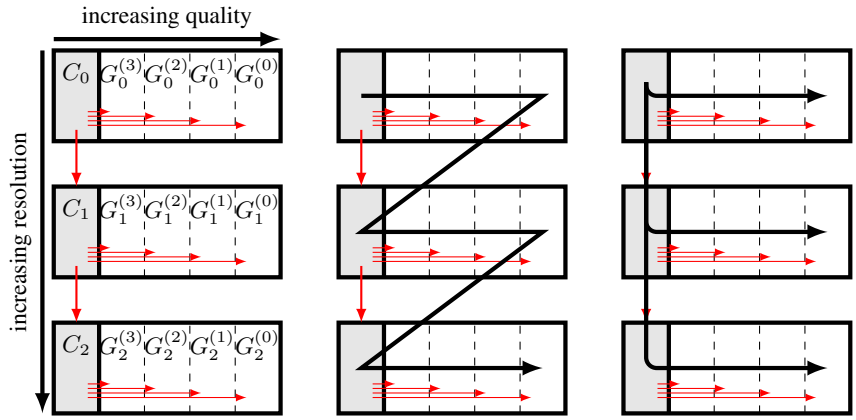
*Algorithm 3.3: Quality-scalable coding using template meshes*

The resulting dependencies are depicted in Figure 3.20a: the encoder is able to store the data blocks in an unrestricted order as long as (a) the order of connectivity information blocks is maintained, (b) geometry information within each resolution is stored in the correct order, and (c) connectivity information for a specific resolution is encoded before geometry information. This allows for storing and transmitting the blocks in any order: resolution per resolution as before (Figure 3.20b), bit-plane-by-bit-plane or purely quality-scalable (Figure 3.20c), or in any arbitrary order that meets the dependencies as indicated above (e.g., Figure 3.20d).

### 3.3.3 Geometry-Agnostic Template Meshes and Parallelization

The geometry-agnostic template meshes suggested in the previous section allow for any arbitrary storage and transmission order. This allows for an encoder to determine an optimal order in which to encode all data.

An added advantage of a geometry-agnostic template mesh is the parallelization opportunities it offers at the decoder side. Assuming that a resolution-scalable coding order is employed such as depicted in Figure 3.20b, and furthermore assuming that sufficient signaling is provided such that a decoding application is able to extract the individual resolutions from the data stream, then each wavelet subband can be decoded in parallel, as soon as the appropriate connectivity information is decoded. This has been illustrated in Figure 3.21c.




*(a)* *(b) Storage order* *(c) Decoding order*

*Figure 3.21: Data dependencies and parallel decoding. Whereas the connectivity data $C_j$ needs to be read in the correct order, the geometry data no longer needs to be processed per resolution: $G_{j_1}$ and $G_{j_2}$ can be processed in parallel.*

## 3.4 Global RD Optimization

RDO requires encoding data blocks such that the distortion is minimal at all bit rates. Such distortion optimization depends on the distortion measure used and as such does not yet have an unambiguous solution. The aim is to show that such optimizations are indeed *possible* by proposing an RDO algorithm, proving that an optimized subband bit plane storage and transmission order is enabled by the codec architecture.

### 3.4.1 Computing Distortions per Bit Plane

The optimization algorithm is constructed by considering the distortions introduced by the wavelet transform. With $N_j = |W_j|$ the number of wavelet coefficients for resolution $j$, the remaining distortion $D_j^{(p)}$ related to this $j^{th}$ resolution decoded up to bit plane $p$ is given by:

$$D_j^{(p)} = \sum_{i=1}^{N_j} \alpha_j^i d_j^{i^{(p)}}, \tag{3.14}$$

with $d_j^{i^{(p)}}$ the distortion on the odd vertex $v_j^{o_i}$ when the most significant bits of wavelet coefficient $i$ of resolution $j$ are decoded up to the $p^{th}$ bit plane, i.e.,

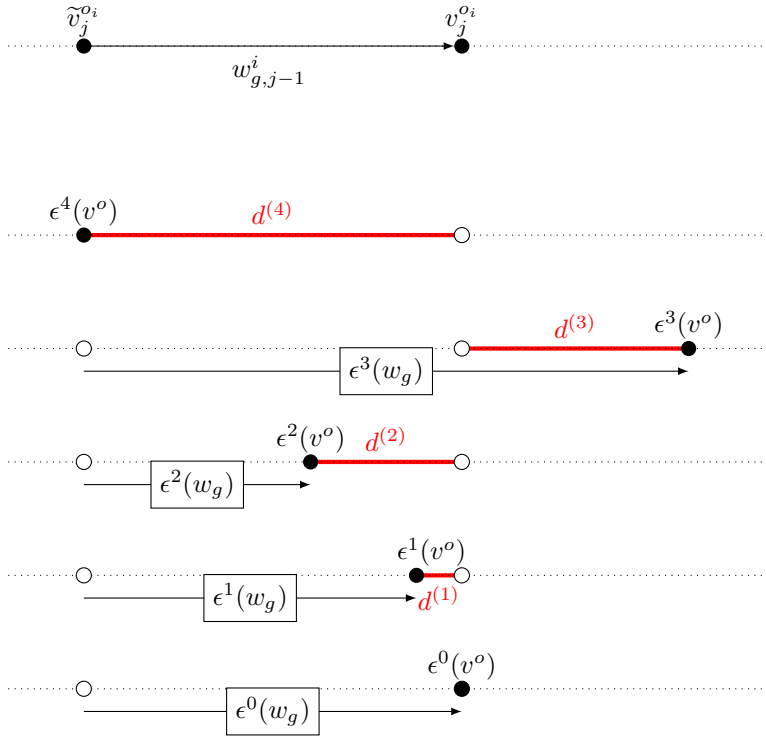



*Figure 3.22: Example reconstruction errors of a 4 bit wavelet coefficient. Except for the first line, the resolution subscript $j$ and the wavelet coefficient index superscript $i$ are omitted to simplify notations. $\epsilon^p(w_g)$ represent the reconstructed wavelet coefficients and $\epsilon^p(v^o)$ represent the odd vertex reconstruction when decoding up to bit plane $p$.*

$$\begin{aligned} d_j^{i^{(p)}} &= |v_j^{o_i} - \epsilon^p(v_j^{o_i})| \\ &= |v_j^{o_i} - \left(\widetilde{v}_j^{o_i} + \epsilon^p(w_{g,j-1}^i)\right)|. \end{aligned} \tag{3.15}$$

In Equation 3.15, $\widetilde{v}_j^{o_i}$ is again the predicted position of odd vertex $v_j^{o_i}$, and $\epsilon^p(v_j^{o_i})$ denotes its reconstructed position when decoding the most significant bit planes of the accompanying wavelet coefficient $w_{j-1}^i = \omega(v_j^{o_i})$ up to bit plane $p$; that is, when considering the partly reconstructed wavelet coefficient $\epsilon^p(w_{g,j-1}^i)$. These notations are illustrated in Figure 3.22 where the example reconstruction of a 4 bit wavelet coefficient (and the corresponding reconstructed odd vertex) is given. To simplify notations, the superscript $j$ and subscript $i$ are dropped as the following paragraphs always handle a specific odd sample $i$ of a specific resolution $j$.

Denoting the total number of bit planes by $Q$, then $\epsilon^Q(w_g) = 0$, i.e., when all $Q$ bit planes are still to be encoded, the wavelet coefficient is zero and the

1: **for all** $v \in M$ **do**
2: $\beta(v) \leftarrow 1/n_v$
3: **end for**
4: **while** downsampling **do**
5: **for all** $v_j^{o_i} \in M_j^o$ **do**
6: **for all** $v \in neighborhood(v_j^{o_i})$ **do**
7: $\beta(v) \leftarrow \beta(v) + \beta(v_j^{o_i})/\nu(v_j^{o_i})$
8: **end for**
9: **end for**
10: **end while**

*Algorithm 3.4: Assigning weights to vertices*

prediction is not refined: $d^{(Q)} = |v^o - \widetilde{v}^o|$. The distortion becomes exactly 0 as soon as the least-significant bit plane is decoded: $\epsilon^0(w_g) = w_g = v^o - \widetilde{v}^o$ and $d^{(0)} = |v^o - (\widetilde{v}^o + \epsilon^0(w_g)| = |v^o - (\widetilde{v}^o + v^o - \widetilde{v}^o)| = 0$.

As discussed in Section 3.2.2.2, the wavelet coefficients are encoded using SAQ (see Section 2.1.2). Let $q_p = Q_p(w_g)$; the wavelet coefficient up to bit plane $p$ can be *dequantized* as $\epsilon^p(w_g) = Q_p^{-1}(q_p)$ using Equation 2.10, where $0 \leq \delta < 1$ determines the placement of $\epsilon^p(w_g)$ within the quantization cell. In the current implementation, it is chosen to be $\delta = 0.5$.

To find weights $\alpha_j^i = \alpha\big(\omega(v_j^{o_i})\big)$ for the wavelet coefficients, first assign weights $\beta(v_j^i)$ to *all* vertices. These weights indicate an estimation of the effect on the full-resolution mesh of repositioning vertices. The reconstructed position of each odd vertex $v_j^{o_i}$ is influenced by the positions of its neighbors. Hence, the weights of these neighbor vertices of $v_j^{o_i}$ each increase by $\beta(v_j^{o_i})/\nu(v_j^{o_i})$ with $\nu(v_j^{o_i})$ the degree of $v_j^{o_i}$. The weight of a wavelet coefficient is given by its accompanying odd vertex, that is, $\alpha\big(\omega(v_j^{o_i})\big) = \beta(v_j^{o_i})$.

At the highest resolution, each of the $n_v$ vertices (and consequently each wavelet coefficient) has a weight of $1/n_v$. If the downsampling terminates at a tetrahedral base mesh counting four vertices, each vertex (and consequently each wavelet coefficient on average) has a weight of $1/4$. This indicates that a single wavelet coefficient at a lower resolution is equally valuable as a multitude of wavelet coefficients at a higher resolution. Note that the use of such weights is similar to the scaling of wavelet coefficients prior to encoding – see e.g. [9], rendering biorthogonal transforms approximately unitary. The complete algorithm for assigning weights is presented in Algorithm 3.4.

### 3.4.2 Global RDO Implementation

At each resolution, an RD curve such as shown in Figure 3.23 can be found. Optimization comes down to considering the RD curves for every resolution,

and coding at every step the information which introduces the largest distortion decrease at the lowest rate. With $P_j$ the last encoded bit plane of resolution $j$, encode the next bit plane of resolution $j'$ with

$$j' = \underset{j:P_j \neq 0}{\arg\max} \frac{D_j^{(P_j)} - D_j^{(P_j-1)}}{R_j^{(P_j-1)} - R_j^{(P_j)}}. \tag{3.16}$$

Up to this point, connectivity blocks have not been considered. To start decoding a specific resolution, the connectivity information of all previous resolutions has to be decoded. This decoding comes at a rate but does not introduce a distortion decrease in vertex-based mean-squared-error sense. Furthermore, as the most-significant bit planes are mostly zero, decoding these highest bit planes also requires rate often without decreasing distortion either. Hence, Equation 3.16 is adapted to consider multiple bit planes together to find the most optimal slope. The definition of $P_j$ is generalized to encompass *any* data block required to encode resolution $j$. Consequently, $P_j$ will start at $Q+1$ as it counts for all $Q$ wavelet subband bit planes and an additional data block for the connectivity information. With $L$ the first unencoded resolution, encode $k'$ bit planes of resolution $j'$ using:

$$(j', k') = \underset{j \in [0,L], k \in [1,P_j]}{\arg\max} \frac{D_j^{(P_j)} - D_j^{(P_j-k)}}{R_j^{(P_j-k)} - R_j^{(P_j)}}. \tag{3.17}$$

For this, the following conventions are made:

$$R_j^{(Q+1)} = 0, \tag{3.18}$$

$$R_j^{\text{conn}} = R_j^{(Q)} - R_j^{(Q+1)} = R_j^{(Q)} \neq 0, \tag{3.19}$$

$$R_j^{\text{geom}} = R_j^{(0)} - R_j^{(Q)}, \tag{3.20}$$

$$D_j^{(Q+1)} = D_j^{(Q)}. \tag{3.21}$$

This states that encoding the connectivity information, i.e., the first data block of a resolution, introduces a rate of $R_j^{\text{conn}}$ while not decreasing the distortion; this corresponds to adding vertices to the decoded mesh without refining their locations. The distortions $D_j^{(p)}, \forall p \in [0, Q]$ are defined in Equation 3.14.

## 3.5 Evaluation

To evaluate SFWInCS as presented in Sections 3.2, 3.3 and 3.4, assume a raw storage of the base mesh as given in Equation 2.1. In practice, the wavelet transform stops at a resolution which is small enough to benefit from scalability, while remaining qualitative enough to be useful as a *base* resolution. To evaluate, however, the transform is applied until no more downsampling can be performed.

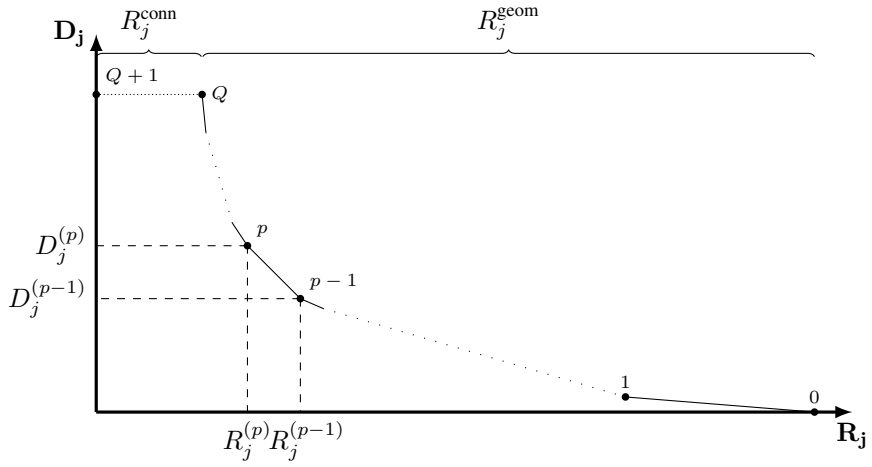



*Figure 3.23: Quality-scalable rate-distortion curve. With decreasing bit plane $p$ the rate increases and the distortion decreases. For each subband $j$ such an RD curve is obtained.*

The portion of the encoded data related to the base mesh is then negligible, and the produced coding numbers can be entirely ascribed to the wavelet subband codec itself. Resolution scalability is first evaluated as such in Section 3.5.1, followed by an evaluation of the cost and benefit of allowing for quality scalability, in Section 3.5.2.

### 3.5.1 Resolution-Scalable Coding

This first evaluation section compares the codec presented in Section 3.2 with the state of the art in wavelet-based coding for irregular meshes, which is Wavemesh [18]. The experiments for Wavemesh were carried out using the software made publicly available by its authors, using the standard settings with the *wavelet geometrical criterion (WGC)* enabled as this avoids downscaling if the resulting geometric distortion would be too large. For the proposed codec, the thresholds were set to $\tau_{\text{MF}} = \tau_{\text{FP}} = 45^\circ$ for every tested model, and the employed leaf size thresholds were $\lambda_C = \lambda_G = 32$. Furthermore, the results are compared with IPR [7], which is not wavelet-based but does outperform Wavemesh in general. As there only was access to a limited set of experimental results of IPR and not the software itself, there is only a limited number of models for which both Wavemesh and IPR can be compared.

These codecs are compared in three ways. First the RD performance of the codecs is compared in Section 3.5.1.1. This distortion is measured against the model after quantization, i.e., the quantization error is not taken into account, and

lossless decoding actually results in zero distortion. For the distortion, both the Hausdorff distance as well as the RMS distance are shown. The resulting RD curves have been made convex by removing non-convex rate points. Next, the distortion is discussed in function of the used number of triangles, in Section 3.5.1.2. This novel measure was introduced as the TD performance in Section 2.3.3 and is proportional to the rendering performance as it indicates runtime memory usage; while it does not relate to storage or transmission efficiency, it is equally important for practical visualization applications to have either a better quality with the same number of triangles, or the same quality using fewer triangles. Finally, a visual comparison is given for the organic model *horse* and the synthetic model *fandisk* in Section 3.5.1.3.

This comparison is followed by a discussion on the computational complexity of the implemented codec in Section 3.5.1.4.

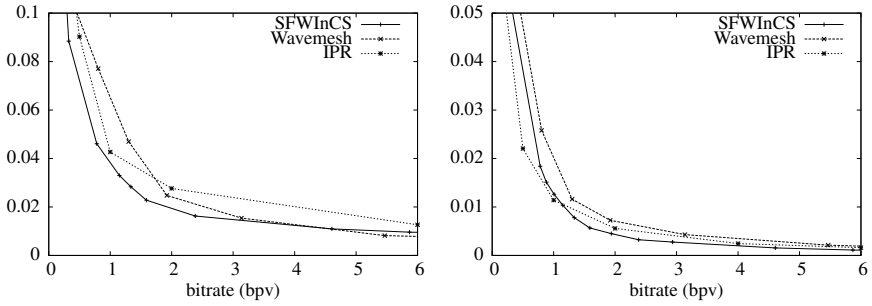
#### 3.5.1.1 Rate-Distortion Performance

Figure 3.24 depicts the RD curves for three test models, *fandisk*, *horse* and *rabbit* (both for Hausdorff and RMS distortions). In case of *fandisk* (Figures 3.24a and 3.24b), where clear visual features are present, the improvements of the proposed codec over Wavemesh are obvious. Experiments have shown that it takes a rate higher than $10$ bpv for Wavemesh to show lower distortions than the proposed solution, at which point the improvements of Wavemesh over the proposed solution are only marginal. The proposed codec also outperforms the state-of-the-art IPR coder for such a feature-rich model.

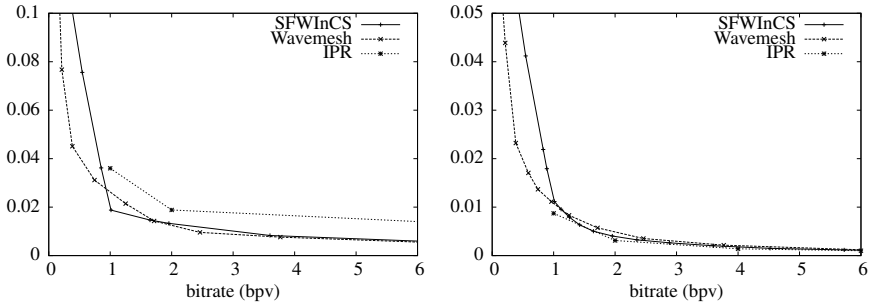
Figures 3.24c and 3.24d show the RD curves for *horse*, and Figures 3.24e and 3.24f show the distortion curves for *rabbit*. The *horse* model still has a few distinct geometric features, *rabbit* does not. Nonetheless the proposed codec performs comparable or better compared to both Wavemesh and IPR. Note that different metrics do not always lead to the same conclusions. Below, the visual comparison of two of these models is discussed, showing improvements of the proposed codec over the state of the art even more significantly than can be deduced from such graphs.

More numerical results are given in Table 3.1, showing the distortions of intermediate meshes at specific rate points. A final column adds average rate differences which were introduced in Section 2.3.3, where the sampled distortion values are taken in an interval such that the rate-values fall in the interval $[1, 6]$, i.e., comparing the average rate difference in the low-to-mid rate range. This number $\Delta^{\mathbf{r}}_{\mathbf{avg}}$ indicates the amount of bits per vertex other codecs need more (if the difference is positive) or less (if the difference is negative) compared to SFWInCS.

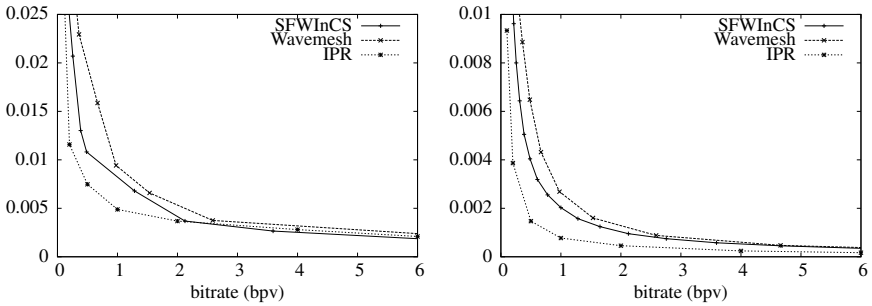
In conclusion, the figures and tables show the improved RD performance of the proposed solution over the state of the art when geometric features are present. Also in the absence of these features the proposed solution stays competitive




*(a) Fandisk (Hausdorff)* *(b) Fandisk (RMS)*




*(c) Horse (Hausdorff)* *(d) Horse (RMS)*




*(e) Rabbit (Hausdorff)* *(f) Rabbit (RMS)*

*Figure 3.24: RD performance: resolution-scalable coding w.r.t. the state of the art. The distortion is expressed either as the Hausdorff distance (left column) or the forward+backwards RMS distance (right column), both computed relative to the bounding box diagonal.*

| Model (#verts) | Coder | Bitrates (in bpv) 1.0 | 2.0 | 4.0 | 6.0 | 8.0 | 10.0 | |
|---|---|---|---|---|---|---|---|---|
| | | **Hausdorff Distortion** | | | | | | $\Delta^{\mathbf{r}}_{\mathbf{avg}}$ |
| fandisk (6 475) | SFWInCS | 43 | 27 | 19 | 9.6 | 9.0 | 8.9 | |
| | Wavemesh | 77 | 25 | 15 | 8.2 | 8.2 | 6.2 | +0.43bpv |
| | IPR | 43 | 28 | 24 | 13 | 10 | 10 | +1.5bpv |
| horse (19 851) | SFWInCS | 46 | 13 | 8.3 | 7.1 | 4.7 | 3.2 | |
| | Wavemesh | 27 | 14 | 7.6 | 7.6 | 5.4 | 5.4 | −0.13bpv |
| | IPR | 36 | 19 | 19 | 18 | 17 | 16 | +4.4bpv |
| fertility (241 607) | SFWInCS | 5.9 | 4.5 | 2.6 | 1.4 | 1.3 | 1.3 | |
| | Wavemesh | 6.9 | 3.2 | 2.4 | 1.7 | 1.2 | 1.2 | −0.53bpv |
| | IPR | 2.6 | 1.7 | 1.1 | 0.71 | 0.65 | 0.59 | −1.6bpv |
| igea (134 345) | SFWInCS | 6.4 | 6.2 | 3.3 | 2.2 | 2.2 | 2.2 | |
| | Wavemesh | 8.0 | 7.5 | 3.7 | 2.7 | 2.7 | 2.3 | +0.30bpv |
| | IPR | 4.2 | 3.1 | 1.8 | 1.8 | 1.7 | 1.6 | −0.97bpv |
| rabbit (67 039) | SFWInCS | 9.5 | 7.1 | 2.7 | 2.6 | 1.8 | 1.8 | |
| | Wavemesh | 9.4 | 6.6 | 3.7 | 3.4 | 3.4 | 1.3 | +0.59bpv |
| | IPR | 4.9 | 3.7 | 2.8 | 2.1 | 2.1 | 2.1 | −0.30bpv |
| vaselion (38 728) | SFWInCS | 99 | 28 | 12 | 12 | 6.9 | 6.9 | |
| | Wavemesh | 36 | 26 | 15 | 8.4 | 7.8 | 7.8 | −0.47bpv |
| bimba (8 857) | SFWInCS | 100 | 32 | 17 | 13 | 11 | 8.9 | |
| | Wavemesh | 65 | 52 | 18 | 12 | 12 | 8.8 | +0.18bpv |
| golfball (122 882) | SFWInCS | 3.6 | 2.9 | 1.8 | 1.7 | 0.82 | 0.82 | |
| | Wavemesh | 1.8 | 1.3 | 0.54 | 0.29 | 0.29 | 0.29 | −1.9bpv |
| | | **RMS Distortion** | | | | | | |
| fandisk (6 475) | SFWInCS | 15 | 4.5 | 2.3 | 1.1 | 0.76 | 0.56 | |
| | Wavemesh | 26 | 7.3 | 4.3 | 2.1 | 2.1 | 0.57 | +0.90bpv |
| | IPR | 11 | 5.6 | 2.4 | 1.6 | 0.98 | 0.92 | +0.45bpv |
| horse (19 851) | SFWInCS | 16 | 4.0 | 2.0 | 1.2 | 0.87 | 0.65 | |
| | Wavemesh | 11 | 5.7 | 2.1 | 2.1 | 1.1 | 1.1 | +0.18bpv |
| | IPR | 8.7 | 3.1 | 1.4 | 1.0 | 0.75 | 0.60 | −0.16bpv |
| fertility (241 607) | SFWInCS | 0.89 | 0.56 | 0.29 | 0.18 | 0.14 | 0.11 | |
| | Wavemesh | 1.5 | 0.60 | 0.36 | 0.21 | 0.11 | 0.11 | −0.015bpv |
| | IPR | 0.33 | 0.20 | 0.12 | 0.073 | 0.056 | 0.046 | −1.3bpv |
| igea (134 345) | SFWInCS | 1.6 | 1.1 | 0.53 | 0.31 | 0.24 | 0.19 | |
| | Wavemesh | 2.0 | 1.3 | 0.77 | 0.43 | 0.43 | 0.20 | +0.21bpv |
| | IPR | 0.69 | 0.39 | 0.21 | 0.15 | 0.096 | 0.051 | −1.4bpv |
| rabbit (67 039) | SFWInCS | 2.6 | 1.2 | 0.58 | 0.45 | 0.26 | 0.26 | |
| | Wavemesh | 2.7 | 1.6 | 0.88 | 0.47 | 0.47 | 0.20 | +0.36bpv |
| | IPR | 0.77 | 0.46 | 0.24 | 0.17 | 0.12 | 0.084 | −1.3bpv |
| vaselion (38 728) | SFWInCS | 35 | 8.0 | 3.4 | 2.3 | 1.5 | 1.5 | |
| | Wavemesh | 11 | 7.4 | 3.8 | 2.4 | 1.4 | 1.4 | −0.50bpv |
| bimba (8 857) | SFWInCS | 68 | 16 | 5.6 | 3.6 | 2.9 | 2.4 | |
| | Wavemesh | 32 | 17 | 6.8 | 4.0 | 4.0 | 2.3 | −0.27bpv |
| golfball (122 882) | SFWInCS | 1.7 | 0.94 | 0.45 | 0.35 | 0.20 | 0.20 | |
| | Wavemesh | 1.5 | 0.71 | 0.29 | 0.15 | 0.15 | 0.15 | −0.72bpv |

*Table 3.1: Hausdorff and RMS distances (both in* $1\,000$*s) w.r.t. the bounding box at specific bit rates. The rightmost column shows the average rate differences* $\Delta^{\mathbf{r}}_{\boldsymbol{avg}}$ *relative to SFWInCS, within the bit rate range* $[1, 6]$ *bpv. Negative figures correspond to rate savings compared to SFWInCS; positive figures, vice versa, correspond to rate savings of SFWInCS compared to the state of the art.*

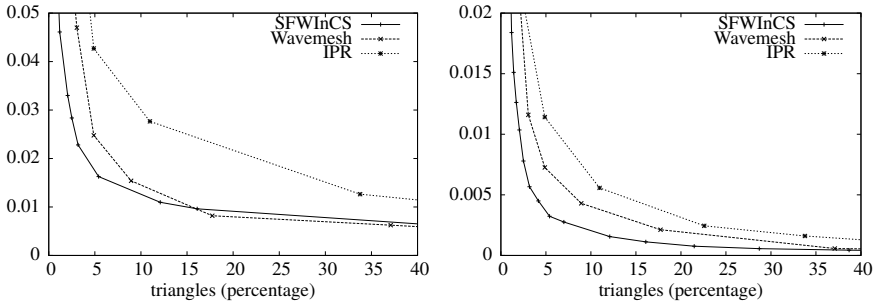
against the state of the art. The results focus on the low-to-midrange bit rates as many applications require high-quality, feature-rich models specifically at these bit rates. Only in lossless encoding, the proposed codec still performs suboptimal; on average over a small set of $15$ test models quantized at $12$ bit precision, Wavemesh requires approximately $17.02$ bpv compared to $25.81$ bpv for the proposed solution. Of these $25.81$ bpv, the connectivity information stays close to $8$ bpv, independent of the model being used. Wavemesh mainly shows superior lossless results for this connectivity information, which is reduced to a minimal amount for the regular meshes in the test set. Observe, however, that the irregular topology in the downscaled models of the proposed codec, which increases the connectivity information cost due to the added cost *per triangle*, does allow for a higher quality using *fewer triangles overall*, which is beneficial for practical uses as is discussed in the next paragraph.

#### 3.5.1.2 Rendering Performance

Compression rates are important from a storage and transmission point-of-view. From a rendering point of view, compression matters as data has to be streamed to memory. The rendering performance, however, is proportional to the complexity of the decoded meshes, as meshes are stored as decoded vertices and triangles on the graphics hardware. Thus, the amount of vertices and triangles is directly related to the memory usage during real-time visualization. This is why considerable importance is also attached to the relation between the number of triangles and the distortion, evaluating the scalable representation without considering the encoding itself, as discussed in Section 2.3.3. Figure 3.25 shows the rendering performance by plotting the distortion in function of the percentage of triangles used (both for Hausdorff and RMS distortions). Table 3.2 gives numerical overviews of a larger test set, giving the obtainable quality when limiting the number of triangles used. The final column again adds a BD-rate-like measure, i.e., the *average triangle percentage difference* $\Delta^{\mathbf{t}}_{\mathbf{avg}}$ introduced in Section 2.3.3. This is done in an interval such that the triangle percentages fall within $[5\%, 40\%]$, i.e., again comparing the differences in the low-to-mid rate range. Similarly, this number $\Delta^{\mathbf{t}}_{\mathbf{avg}}$ indicates the triangle percentage other codecs need more or less compared to *the proposed* codec.

When geometric features are present, as in Figures 3.25a and 3.25b for *fandisk* and Figures 3.25c and 3.25d for *horse*, the proposed solution clearly improves upon the state of the art, by better preserving these features when downsampling. When no sharp geometric features are present, as is the case for *rabbit* (Figures 3.25e and 3.25f), the improvements are still visible, albeit not as pronounced.

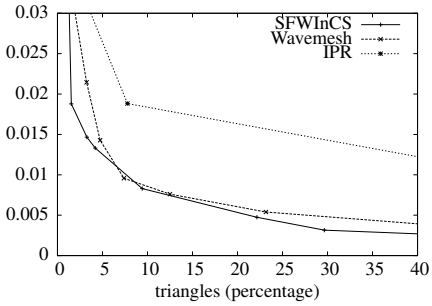
Additionally, the figures display the finer resolution granularity compared to Wavemesh. Over a small set of models, one observes a significant increase in the number of resolution levels offered: SFWInCS offers on average $40$ resolution



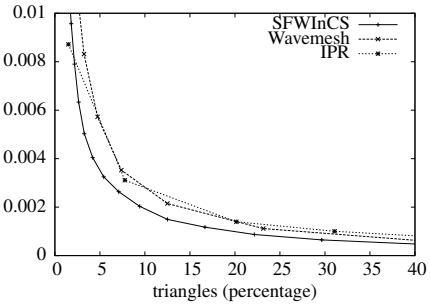

*(a) Fandisk (Hausdorff)*



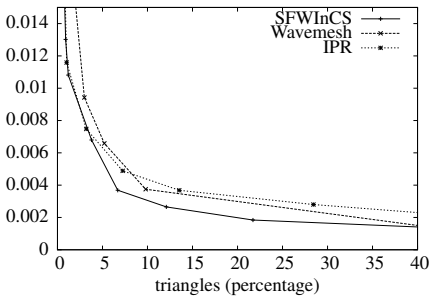

*(b) Fandisk (RMS)*



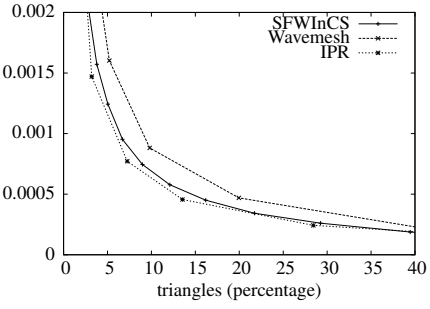

*(c) Horse (Hausdorff)*




*(d) Horse (RMS)*




*(e) Rabbit (Hausdorff)*




*(f) Rabbit (RMS)*

*Figure 3.25: TD performance: resolution-scalable coding w.r.t. the state of the art. The distortion is expressed either as the Hausdorff distance (top row) or the forward+backwards RMS distance (bottom row), both computed relative to the bounding box diagonal.*

| Model (#verts) | Coder | Percentage of triangles 5 | 10 | 20 | 30 | 50 | $\Delta^{\mathbf{t}}_{\mathbf{avg}}$ |
|---|---|---|---|---|---|---|---|
| | | **Distortion** | | | | | |
| fandisk (6 475) | SFWInCS | 27 | 19 | 9.6 | 8.9 | 7.8 | |
| | Wavemesh | 25 | 15 | 8.2 | 8.2 | 6.2 | $-0.14\%$ |
| | IPR | 43 | 43 | 28 | 24 | 10 | $+27\%$ |
| horse (19 851) | SFWInCS | 13 | 8.3 | 7.1 | 3.2 | 2.9 | |
| | Wavemesh | 14 | 9.6 | 7.6 | 5.4 | 3.4 | $+4.1\%$ |
| | IPR | 36 | 19 | 19 | 19 | 17 | $+48\%$ |
| fertility (241 607) | SFWInCS | 4.5 | 3.2 | 1.9 | 1.3 | 1.3 | |
| | Wavemesh | 5.1 | 3.2 | 2.4 | 1.7 | 1.2 | $+0.68\%$ |
| | IPR | 4.4 | 2.6 | 1.7 | 1.7 | 1.1 | $-0.80\%$ |
| igea (134 345) | SFWInCS | 6.2 | 3.3 | 3.2 | 2.2 | 2.2 | |
| | Wavemesh | 7.5 | 3.7 | 2.7 | 2.7 | 2.3 | $+4.8\%$ |
| | IPR | 8.0 | 4.2 | 3.1 | 3.1 | 1.8 | $+3.0\%$ |
| rabbit (67 039) | SFWInCS | 6.8 | 3.7 | 2.6 | 1.8 | 1.6 | |
| | Wavemesh | 9.4 | 3.7 | 3.4 | 3.4 | 1.3 | $+6.6\%$ |
| | IPR | 7.5 | 4.9 | 3.7 | 2.8 | 2.1 | $+12\%$ |
| vaselion (38 728) | SFWInCS | 21 | 12 | 8.8 | 6.9 | 4.5 | |
| | Wavemesh | 26 | 15 | 8.4 | 7.8 | 7.8 | $+1.8\%$ |
| bimba (8 857) | SFWInCS | 22 | 17 | 11 | 8.9 | 8.7 | |
| | Wavemesh | 52 | 18 | 12 | 8.8 | 6.5 | $+1.4\%$ |
| golfball (122 882) | SFWInCS | 3.5 | 1.9 | 1.7 | 0.82 | 0.63 | |
| | Wavemesh | 1.8 | 1.3 | 0.54 | 0.29 | 0.29 | $-11\%$ |
| | | **Distortion** | | | | | |
| fandisk (6 475) | SFWInCS | 4.5 | 2.3 | 1.1 | 0.56 | 0.44 | |
| | Wavemesh | 7.3 | 4.3 | 2.1 | 2.1 | 0.57 | $+10\%$ |
| | IPR | 11 | 11 | 5.6 | 2.4 | 0.98 | $+21\%$ |
| horse (19 851) | SFWInCS | 4.0 | 2.0 | 1.2 | 0.65 | 0.48 | |
| | Wavemesh | 5.7 | 3.5 | 2.1 | 1.1 | 0.45 | $+5.1\%$ |
| | IPR | 8.7 | 3.1 | 3.1 | 1.4 | 0.75 | $+7.4\%$ |
| fertility (241 607) | SFWInCS | 0.72 | 0.36 | 0.23 | 0.14 | 0.11 | |
| | Wavemesh | 1.0 | 0.60 | 0.36 | 0.21 | 0.11 | $+2.7\%$ |
| | IPR | 0.70 | 0.33 | 0.20 | 0.20 | 0.12 | $-0.48\%$ |
| igea (134 345) | SFWInCS | 1.3 | 0.68 | 0.42 | 0.24 | 0.19 | |
| | Wavemesh | 1.3 | 0.77 | 0.43 | 0.43 | 0.20 | $+3.6\%$ |
| | IPR | 1.1 | 0.69 | 0.39 | 0.39 | 0.21 | $-0.70\%$ |
| rabbit (67 039) | SFWInCS | 1.6 | 0.75 | 0.45 | 0.26 | 0.19 | |
| | Wavemesh | 2.7 | 0.88 | 0.47 | 0.47 | 0.20 | $+4.4\%$ |
| | IPR | 1.5 | 0.77 | 0.46 | 0.24 | 0.17 | $-1.0\%$ |
| vaselion (38 728) | SFWInCS | 6.4 | 3.4 | 1.8 | 1.5 | 0.83 | |
| | Wavemesh | 7.4 | 3.8 | 2.4 | 1.4 | 1.4 | $+1.6\%$ |
| bimba (8 857) | SFWInCS | 11 | 5.6 | 2.9 | 2.4 | 1.5 | |
| | Wavemesh | 17 | 6.8 | 4.0 | 2.3 | 1.2 | $+1.5\%$ |
| golfball (122 882) | SFWInCS | 1.2 | 0.57 | 0.35 | 0.20 | 0.15 | |
| | Wavemesh | 1.5 | 0.71 | 0.29 | 0.15 | 0.15 | $-3.3\%$ |

*Table 3.2: Hausdorff and RMS distances (both in* $1\,000s$*) w.r.t. the bounding box at specific triangle budgets. The rightmost column shows the average triangle percentage differences* $\Delta^{\mathbf{t}}_{avg}$ *relative to SFWInCS, within the triangle percentage range* $[5\%, 40\%]$*. Negative figures correspond to savings compared to SFWInCS; positive figures then correspond to savings of SFWInCS compared to the state of the art.*

levels whereas Wavemesh offers on average 23 resolution levels. Moreover, the penultimate resolution counts, on average, $73\%$ of the triangles for SFWInCS compared to $39\%$ for Wavemesh, which Wavemesh obtains due to exploiting $4-$to$-1$ subdivision regularity which results in a penultimate resolution with only $25\%$ of the triangles in the perfectly regular case. Offering more fine-grained control allows for more optimal memory management when using these models in a real-time rendering environment, which is an advantage that cLOD systems such as IPR have over wavelet-based systems such as Wavemesh.

#### 3.5.1.3 Visual Comparisons

To conclude the comparison with the state of the art, some visual examples belonging to the distortion curves given in Figures 3.24 and 3.25 are shown. Pay special attention to the visual features, such as the ears and the hooves of *horse*, and the sharp outlines of *fandisk*. Also pay attention to the number of triangles used. The topology at lower resolutions of the proposed solution stays irregular, which increases the coding cost per vertex and per triangle. However, fewer triangles are required to obtain similar visual quality, and consequently similar bit rates still deliver similar and even superior quality, while reducing the load on graphics memory.

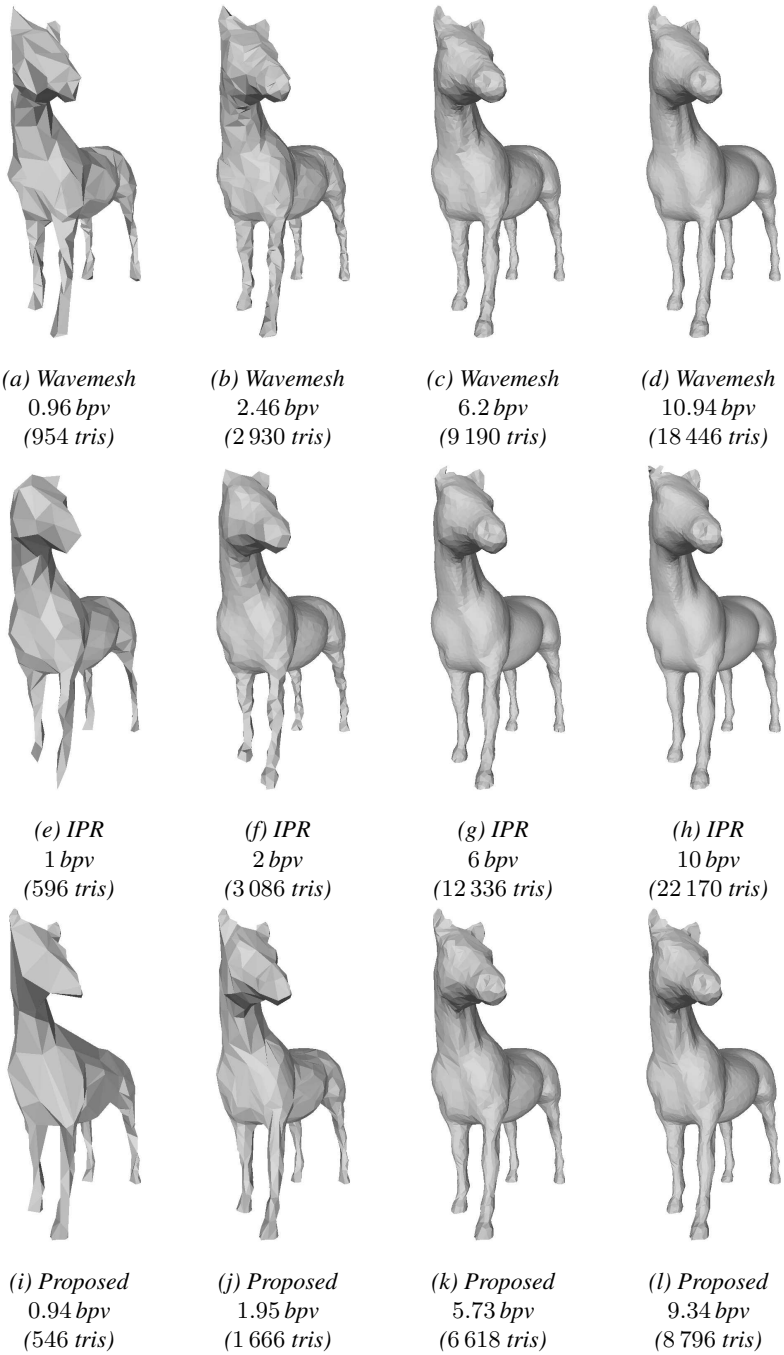
The first set of examples, in Figure 3.26, shows *horse* at up to $10$ bpv. The results show that Wavemesh achieves similar quality at similar bit rates, but needs more triangles to render this. IPR generates even more triangles but is not able to accurately reconstruct all visual features even at $10$ bpv.

The second set of examples, in Figure 3.27, shows *fandisk* at bit rates up to $10$ bpv. At $0.5$ bpv, SFWInCS has preserved slightly more visual features than Wavemesh. At $2$ bpv the sharp features are already fully reconstructed, while Wavemesh (even with the geometry criterion enabled) still has an obvious visual artifact. The intermediate models produced by SFWInCS clearly are better suited for rendering applications, as features are better preserved.

Observe the discrepancy between objective measures and (subjective) visual assessment, which is also observable in other domains such as image processing. Whereas distortion metrics show nearly equal quality, the visual quality is still remarkably higher with SFWInCS due to the better avoidance of aliasing effects.

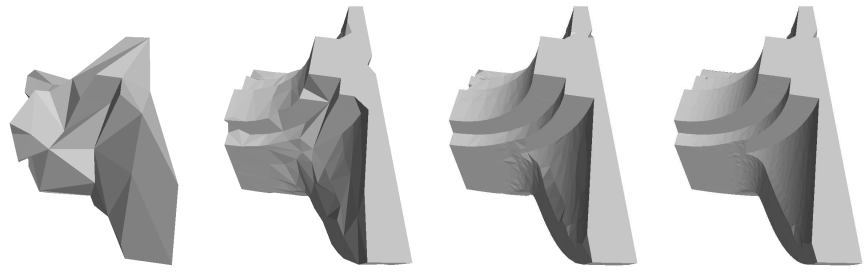
#### 3.5.1.4 Complexity

The experiments were single-threadedly executed on a $2.40$*GHz Intel Core 2 Quad CPU with* $8$ *GB of RAM* for models up to $5 \times 10^6$ vertices. Figure 3.28 shows encoding speeds in the order of $3\,000 \sim 8\,000$ vertices/s and decoding speeds in the order of $20\,000 \sim 50\,000$ vertices/s. Variations are caused by the sparsity of the test set, comprised of $30$ models with their number of vertices distributed



*(a) Wavemesh* 0.96 *bpv* *(954 tris)*
*(b) Wavemesh* 2.46 *bpv* *(2 930 tris)*
*(c) Wavemesh* 6.2 *bpv* *(9 190 tris)*
*(d) Wavemesh* 10.94 *bpv* *(18 446 tris)*
*(e) IPR* 1 *bpv* *(596 tris)*
*(f) IPR* 2 *bpv* *(3 086 tris)*
*(g) IPR* 6 *bpv* *(12 336 tris)*
*(h) IPR* 10 *bpv* *(22 170 tris)*
*(i) Proposed* 0.94 *bpv* *(546 tris)*
*(j) Proposed* 1.95 *bpv* *(1 666 tris)*
*(k) Proposed* 5.73 *bpv* *(6 618 tris)*
*(l) Proposed* 9.34 *bpv* *(8 796 tris)*

*Figure 3.26: Visual comparison* horse.



*(a) Wavemesh* 0.55 *bpv (*128 *tris)*



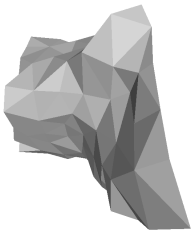
*(b) Wavemesh* 1.92 *bpv (*632 *tris)*



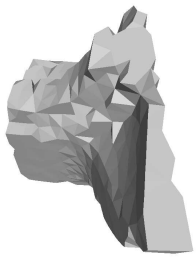
*(c) Wavemesh* 5.47 *bpv (*2 296 *tris)*



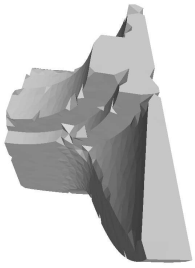
*(d) Wavemesh* 9.61 *bpv (*4 802 *tris)*



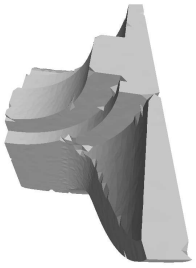
*(e) IPR* 0.5 *bpv (*284 *tris)*



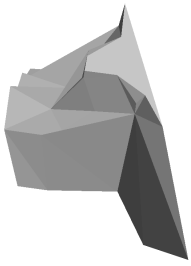
*(f) IPR* 2 *bpv (*1 420 *tris)*



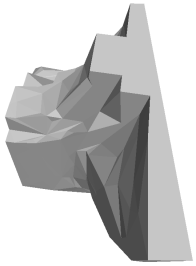
*(g) IPR* 6 *bpv (*4 372 *tris)*



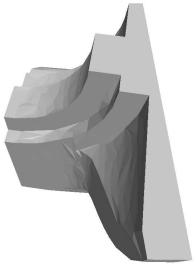
*(h) IPR* 10 *bpv (*7 604 *tris)*



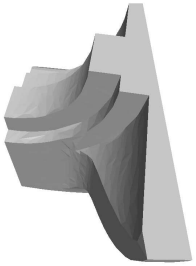
*(i) Proposed* 0.45 *bpv (*74 *tris)*



*(j) Proposed* 1.94 *bpv (*540 *tris)*

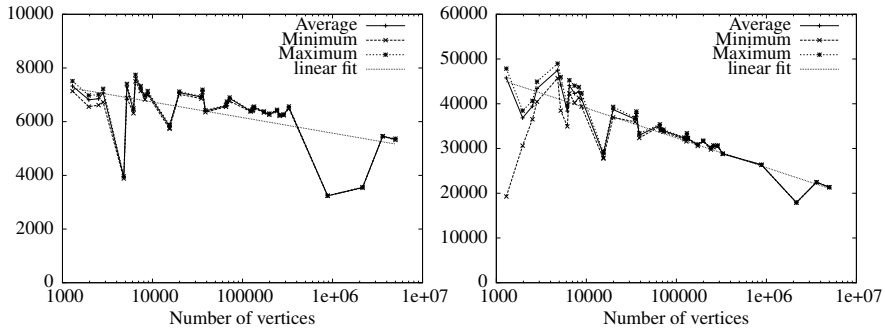


*(k) Proposed* 5.87 *bpv (*2 084 *tris)*



*(l) Proposed* 9.66 *bpv (*3 716 *tris)*

*Figure 3.27: Visual comparison* fandisk.

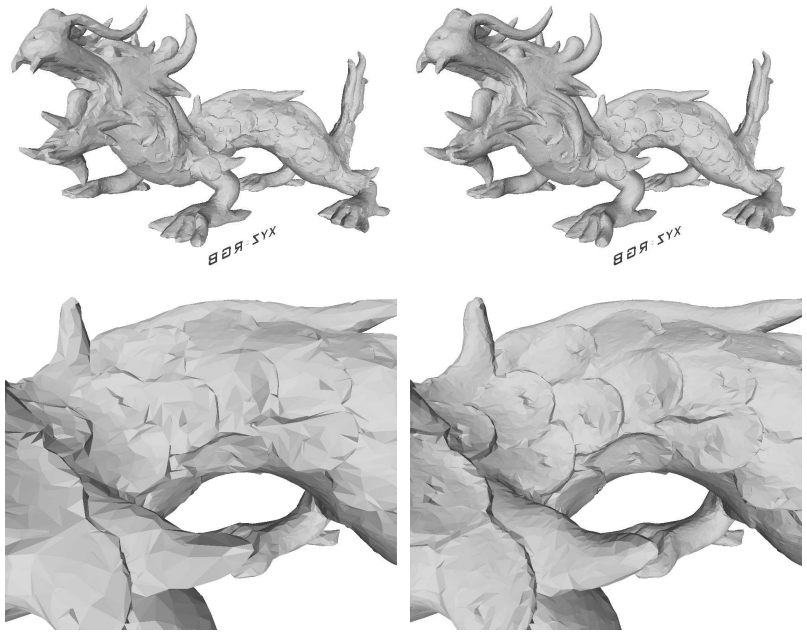



*(a) Encoding speed* *(b) Decoding speed*

*Figure 3.28: Encoding and decoding speed. The figures show the amount of vertices encoded (a) and decoded (b) per second. An approximation of the encoding speed is given by* $f(x) = 9\,053.85 - 251.98 \log x$*, while the decoding speed is approximated by* $f(x) = 65\,628.10 - 2\,888.33 \log x$*.*

over a fairly large span. The encoding and decoding times depend on the number of vertices as well as on other mesh properties. Nonetheless, except for a few outliers, a slight linear decrease in encoding speed is still visible when the number of vertices increases exponentially as is shown in the figure. This implies that both the encoding and decoding *speeds* decrease $O(\log(n))$, such that the complexity of both the encoder and decoder in terms of encoding and decoding *times* is $O(n \log(n))$, with $n$ the number of vertices. In future research, using a larger benchmark of low- to high-complexity models should confirm this observation.

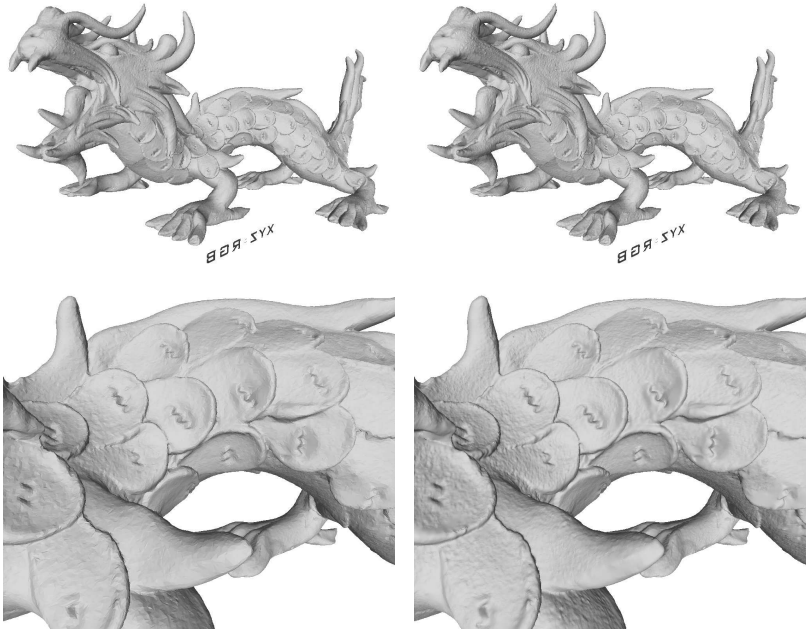
**High-Resolution Models** Despite the absence of a discussion on high-resolution models in state-of-the-art papers, where high-resolution models should nowadays be considered to be made up of at least over $500\,000$ vertices, a few of these results are depicted here without any comparable data available. Figure 3.2 already showed such a model, i.e., the *Thai statue* which counts $4\,999\,999$ vertices. Two additional examples are found in Figures 3.29 and 3.30. The figures point out an important consequence of the current evolution towards higher resolutions: whereas lossless compression is still desired for storage and transmission, the lossy rates are becoming much more important for practical rendering applications. This can be explained by the fact that the fine detail often does not add geometric information and can be dropped without visual distortion. In practice, the small curvature distortion will even be visually masked by shader techniques which interpolate the surface normal over the triangle surfaces. Nevertheless, the figures show that even without such shader techniques, very good visual results are obtained already in the low-to-midrange bit rates.



*(a)* 0.5 *bpv (*0.610% *tris)*



*(b)* 1 *bpv (*2.08% *tris)*

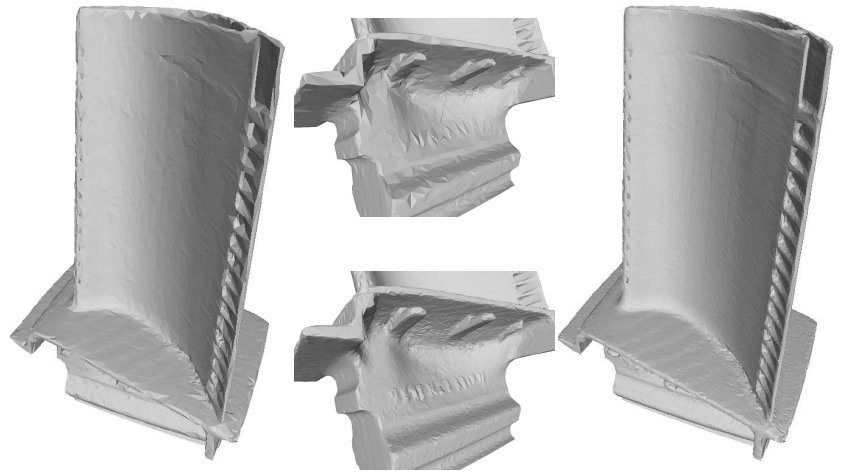


*(c)* 5.75 *bpv (*17.5% *tris)*

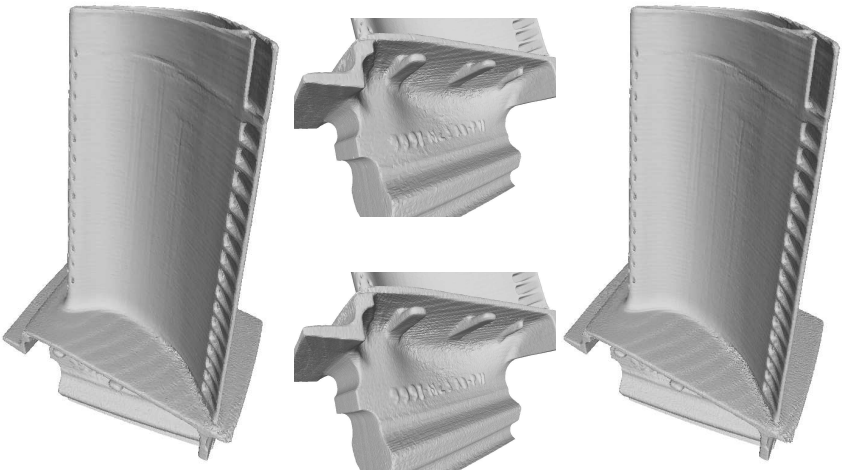


*(d)* 24.7 *bpv (*7 219 045 *tris)*

*Figure 3.29: Rendered model:* Asian dragon*. Several resolution levels of the 3.5-million-vertex* Asian dragon*, encoded with* 16 *bit quantization.*



*(a)* 2.9 *bpv (*2.33% *tris)*



*(b)* 4.8 *bpv (*7.38% *tris)*



*(c)* 12.4 *bpv (*29.6% *tris)*



*(d)* 32.6 *bpv (*1 765 388 *tris)*

*Figure 3.30: Rendered model:* turbine blade*. Several resolution levels of the 900-thousand-vertex* turbine blade*, encoded with* 16 *bit quantization. The middle column shows a close-up of the text on the bottom half of the model at the same bit rates.*

| Model (#verts) | Rate increase |
|---|---|
| teapot (1 292) | $+0.068$bpv (+0.21%) |
| beethoven (2 521) | $+0.187$bpv (+0.55%) |
| triceratops (2 832) | $-0.017$bpv (-0.052%) |
| elk (5 194) | $+0.015$bpv (+0.049%) |
| fandisk (6 475) | $-0.015$bpv (-0.057%) |
| maxplanck (7 399) | $-0.013$bpv (-0.044%) |
| venushead (8 268) | $+0.022$bpv (+0.074%) |
| bimba (8 857) | $+0.094$bpv (+0.31%) |
| horse (19 851) | $+0.022$bpv (+0.087%) |
| screwdriver (65 538) | $+0.182$bpv (+0.89%) |
| rabbit (67 039) | $+0.073$bpv (+0.32%) |
| dino (129 026) | $+0.074$bpv (+0.37%) |
| **Average** | **+0.23%** |

*Table 3.3: Additional cost when using a geometry-agnostic template. All models have been encoded using* $12$ *bit precision.*

### 3.5.2 Quality-Scalable Coding

To evaluate quality-scalable coding, Section 3.5.2.1 handles the effect of using the geometry-agnostic template meshes as constructed in Section 3.3.2. Subsequently, the unlocked quality scalability allows for RDO which is evaluated in Section 3.5.2.2. Finally, the resolution-scalable mode of SFWInCS, as described in Section 3.2, will be denoted as SFWInCS$^R$ and a comparison of SFWInCS with SFWInCS$^R$ and with the state of the art is given in Section 3.5.2.3.

The distortion values are reported as RMS distortions, and the resulting RD curves have again been made convex.

#### 3.5.2.1 Accuracy of the Template Mesh

Using a template mesh which does not take into account geometry information, i.e., a template which develops using only connectivity information as described in Section 3.3.2, had an effect on the implementation: possibly overlapping vertices in the template mesh have to be handled properly, by repositioning these vertices equally by both the encoder and decoder *without relying on the encounter order*. Note that these geometric modifications do not alter the geometry of the decoded meshes after inverse wavelet transformation; they can only alter where samples are encoded in the octrees.

Both the use of this alternative template mesh as well as the occasional repositioning of overlapping sample positions can have an influence on the exploitability of the spatial correlations within the data. However, the effects of using the new geometry-agnostic template proved to be negligible.

This can also be seen in the final bit rates. Table 3.3 lists the changes in lossless bit rates for several models. Observe that the bit rate increases on average by

$0.23\%$, while in some cases the bit rate even goes down. These results show that lowering the accuracy of the template mesh does not hinder the exploitation of spatial correlations, on the contrary it is clear that topological locality information is preserved. Vertices that are located closely together will have mapped vertices in the template mesh which are also located closely together due to topological proximity, albeit possibly at another global position (and consequently, falling within a different octree cell) due to geometry information not being taken into account.

#### 3.5.2.2 Rate-Distortion Optimization

RDO orders the data such that the quality gains come at the lowest rates. Figure 3.31 gives the results for the three models for which RD curves were given in Section 3.5.1. For *fandisk* and *horse* the improvements compared with a resolution-scalable transmission order are clear at low bit rates, obtaining similar qualities at a lower cost. At higher bit rates the improvements are minimal. This indicates that the resolution-scalable transmission at high bit rates is in general already nearly optimal in rate-distortion sense.

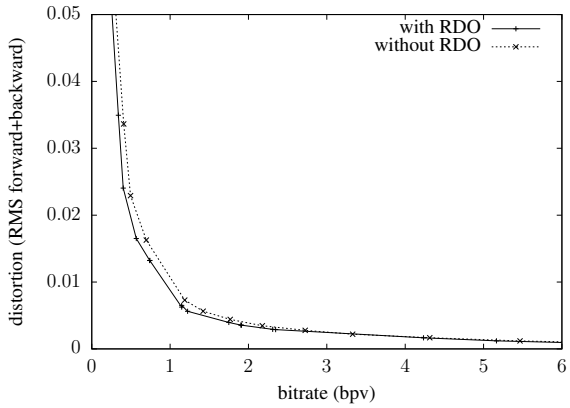
Visual results of the proposed codec are given in Figure 3.32 for *fertility*. As it was also observed in Section 3.5.1, the information in the lower resolution wavelet subbands contributes the most to the shape of such densely sampled models, while higher resolutions only serve to increase the quality when rendering from very nearby: Figure 3.32c already resembles very closely the original mesh depicted in Figure 3.32d.

In theory, an RD optimization should *always* be on par or better; in practice this is the case if the optimization algorithm calculates distortions in the spatial domain. The results show that an optimization based on the proposed weighted wavelet coefficients while ignoring topological information also proves to give superior results. The coding overhead is minimal: the storage increases only a fraction of a bit per vertex, due to the need to identify the resolution of each subsequent data block.

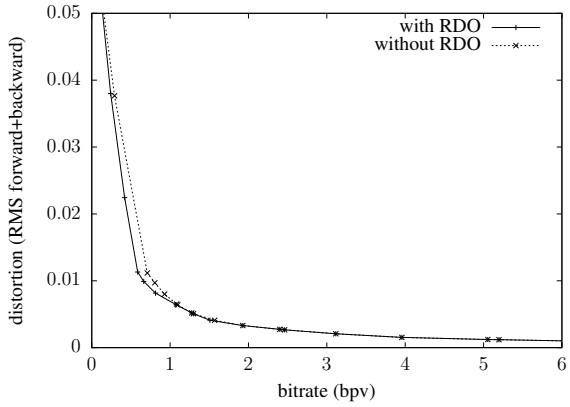
#### 3.5.2.3 Comparison with the State of the Art

For comparison, mainly the overall improvement over SFWInCS$^R$ (for which the results are given in Section 3.5.1) is investigated. As in Section 3.5.1, Wavemesh [18] and IPR [7] are also compared with. SFWInCS$^R$, presented in Section 3.2, was implemented using rate points at every *resolution*; this is more coarse-grain than the quality-scalable codec which gives a rate point after every bit plane.

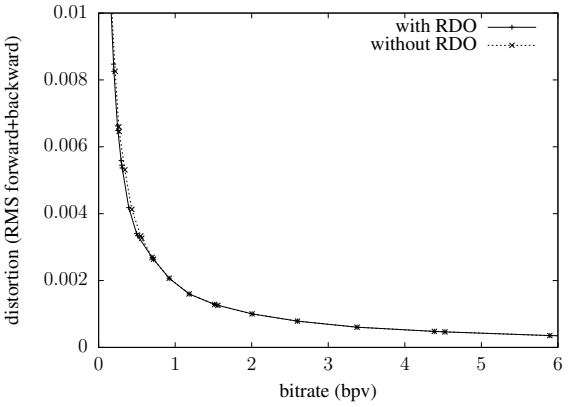
Figure 3.33 shows a comparison with the state of the art. It shows even more competitive results brought by SFWInCS compared with the resolution-scalable mode, SFWInCS$^R$. In the case of the feature-rich model *fandisk*, the results after




*(a) Fandisk*

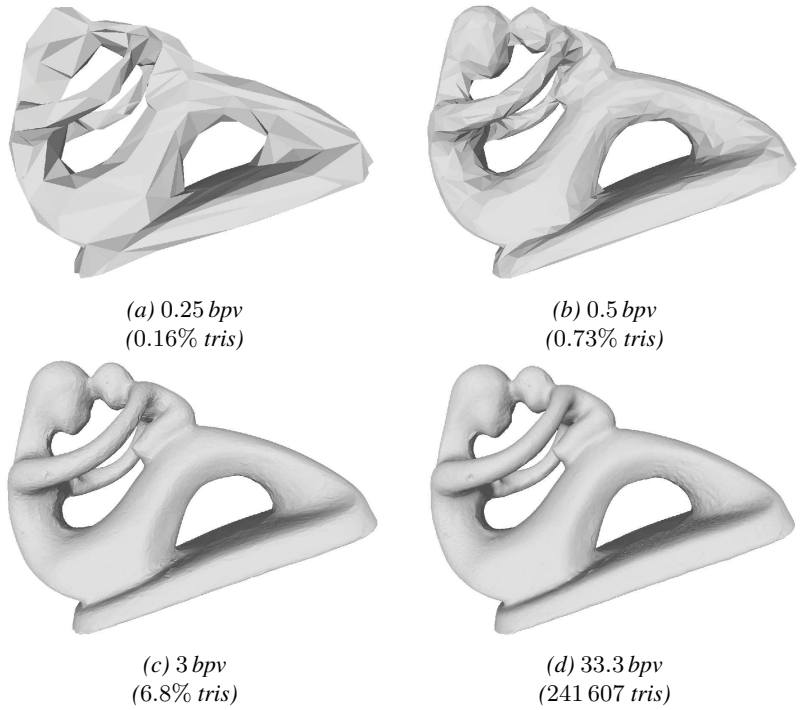



*(b) Horse*




*(c) Rabbit*

*Figure 3.31: RD performance: effect of RD optimization.*



*(a)* 0.25 *bpv* (0.16% *tris)*

*(b)* 0.5 *bpv* (0.73% *tris)*

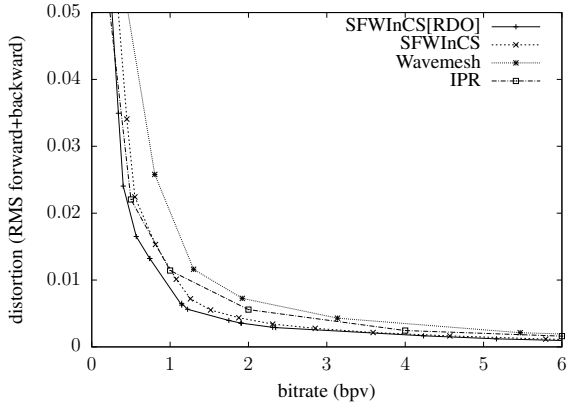
*(c)* 3 *bpv* (6.8% *tris)*

*(d)* 33.3 *bpv* (241 607 *tris)*

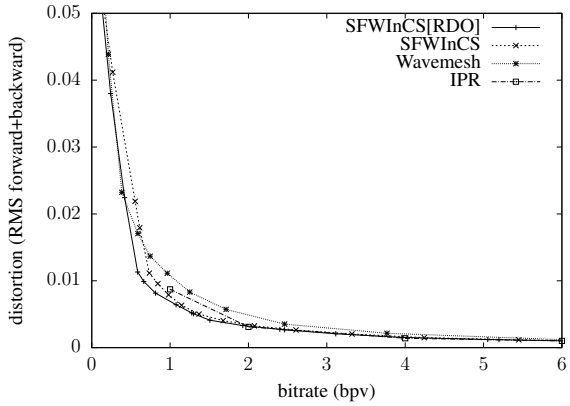
*Figure 3.32: Rendered model:* fertility. *(a), (b) and (c) show the model at increasing rates. Observe that* 3 *bpv is still in the low bit rate range considering a lossless rate of* 33.3 *bpv for this* 18 *bit quantized* fertility *model shown in (d).*

RDO improve over both Wavemesh and IPR. In the case of *horse*, the results are entirely on-par with or better than previous work, even at the lowest rates. Finally, for the feature-poor model *rabbit*, results remain nearly unchanged compared with resolution-scalable coding; IPR remains the better solution.

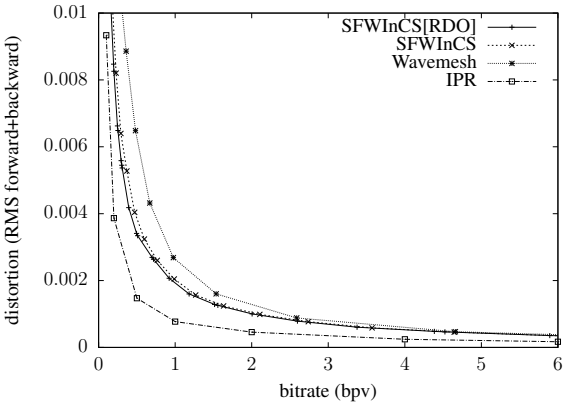
Results over a small set of models are summarized in Table 3.4. This table makes use of the *average rate difference* $\Delta^{\mathbf{r}}_{\mathbf{avg}}$ which as introduced in Section 2.3.3, and which was used in Section 3.5.1. Additionally, in a similar way the maximal and minimal rate differences are reported as $\Delta^{\mathbf{r}}_{\mathbf{max}}$ and $\Delta^{\mathbf{r}}_{\mathbf{min}}$. This gives a maximal, minimal and average value of the rate differences, with a positive difference indicating that a state-of-the-art codec requires more bits for the same quality, and a negative difference indicating that the state-of-the-art codec outperforms the proposed codec. In this case, the limited rate range over which measurements are done is taken up to 3 bpv for the proposed codec. Furthermore, RDO and quality-scalable decoding improve all results of resolution-scalable coding. At such low bit rates, relatively high gains of up to 1 bpv are obtained. Observe that the minimal rate difference is zero as both the proposed codec and resolution-




*(a) Fandisk*




*(b) Horse*




*(c) Rabbit*

*Figure 3.33: RD performance: RD-optimized coding w.r.t. the state of the art.*

| Model | Coder | $\Delta^{\mathrm{r}}_{\mathrm{avg}}$ | $\Delta^{\mathrm{r}}_{\mathrm{max}}$ | $\Delta^{\mathrm{r}}_{\mathrm{min}}$ |
|---|---|---|---|---|
| teapot (1 292) | SFWInCS | $+0.22$ | $+1.00$ | $0.0$ |
| beethoven (2 521) | SFWInCS | $+0.32$ | $+0.81$ | $0.0$ |
| triceratops (2 832) | SFWInCS | $+0.07$ | $+0.59$ | $0.0$ |
| | Wavemesh | $-0.72$ | $+0.43$ | $-1.30$ |
| elk (5 194) | SFWInCS | $+0.16$ | $+0.60$ | $0.0$ |
| | Wavemesh | $+0.10$ | $+1.20$ | $-0.28$ |
| fandisk (6 475) | SFWInCS | $+0.08$ | $+0.73$ | $0.0$ |
| | Wavemesh | $+0.25$ | $+0.93$ | $+0.04$ |
| | IPR | $+0.43$ | $+2.90$ | $-0.03$ |
| maxplanck (7 399) | SFWInCS | $+0.09$ | $+0.39$ | $0.0$ |
| | Wavemesh | $-0.25$ | $+0.04$ | $-0.37$ |
| venushead (8 268) | SFWInCS | $+0.06$ | $+0.73$ | $-0.04$ |
| bimba (8 857) | SFWInCS | $+0.10$ | $+0.39$ | $0.0$ |
| | Wavemesh | $+0.28$ | $+1.1$ | $+0.14$ |
| horse (19 851) | SFWInCS | $+0.10$ | $+0.39$ | $0.0$ |
| | Wavemesh | $+0.15$ | $+0.84$ | $-0.17$ |
| | IPR | $+2.40$ | $+7.60$ | $+0.55$ |
| screwdriver (65 538) | SFWInCS | $+0.01$ | $+0.78$ | $0.0$ |
| | Wavemesh | $-0.04$ | $+0.01$ | $-2.1$ |
| rabbit (67 039) | SFWInCS | $+0.011$ | $+0.29$ | $0.0$ |
| | Wavemesh | $+0.06$ | $+1.3$ | $0.0$ |
| | IPR | $-0.15$ | $+0.40$ | $-0.80$ |
| dino (129 026) | SFWInCS | $+0.02$ | $+0.94$ | $0.0$ |
| | Wavemesh | $-0.08$ | $0.0$ | $-2.50$ |

*Table 3.4: Rate savings w.r.t. the state of the art. To obtain the same quality at rates up to 3 bpv, the numbers indicate in bpv ($\Delta^{\mathrm{r}}_{avg}$) the average rate savings, ($\Delta^{\mathrm{r}}_{max}$) the largest rate savings, and ($\Delta^{\mathrm{r}}_{min}$) the smallest rate savings. A positive value means more rate is required than the proposed codec, a negative value means the proposed codec performs worse.*

scalable coding start at the same base mesh, resulting in the same distortion at the same rate. This confirms that the RD-optimized codec never performs worse than resolution-scalable coding, as the minimal differences are never negative. The comparison with Wavemesh and IPR shows that in most cases the proposed codec is also more favorable at these low bit rates, except for *dino* where it is on par at best.

## 3.6 Conclusions

In this first part of the dissertation a new wavelet-based representation of meshes has been proposed, which relies on an adaptive subsampling and retriangulation process to preserve geometric features of an original mesh. Visual results and an analysis of the rendering performance showed its effectiveness compared to the state of the art. Additional work concerning the transform could focus on

improving the used predictors to take curvature into account. Furthermore, a two-mode update step similar to the proposed two-mode predictor can be investigated in order to further improve the quality at lower resolutions. Also, one notes that thresholding dihedral angles is sensitive to noise; a deeper investigation could handle the processing of noisy models as, e.g., obtained by scanning equipment.

Furthermore, a novel coding system was proposed, using octree coding principles both to avoid defining and imposing any mesh traversal order for connectivity encoding, and to exploit spatial correlations between wavelet coefficients for geometry encoding. The proposed representation and coding system improves the compression performance over the state of the art for feature-rich models at low-to-midrange bit rates. Additional work can be aimed at a better connectivity coding which has to exploit the statistical dependencies within (intra-) and in between (inter-) various resolution levels. For geometry coding, describing wavelet coefficients in tangential components, in effect splitting the geometric information from the parametric information, should further improve upon the rate-distortion performance.

Thirdly, requirements were shown to achieve quality scalability and an optimized rate-distortion performance. Gains up to 1 bpv are obtained at low bit rates, while at higher bit rates the original codec design is nearly optimal in rate-distortion sense. Furthermore, the results and working implementation serve as a proof-of-concept that an unrestricted storage and transmission of subband bit planes can be provided using the proposed framework.

Overall, these improvements upon the state of the art are vital in the context of automatic LOD generation and the use of models in a streaming environment. The approach is a step forward in constructing multiresolution representations with high-quality feature-preserving levels-of-detail.

The work presented in this chapter has led to the following publications:

- J. El Sayeh Khalil, A. Munteanu, L. Denis, P. Lambert, and R. Van de Walle. *Scalable Feature-Preserving Irregular Mesh Coding*. Computer Graphics Forum, 36(6):275–290, September 2017.

- J. El Sayeh Khalil, A. Munteanu, and P. Lambert. *Rate-Distortion Optimized Wavelet-based Irregular Mesh Coding*. In Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP), volume 1, pages 212–219, Porto, Portugal, 27 February – 1 March 2017.

# References

[1] Costa Touma and Craig Gotsman. *Triangle Mesh Compression*. In Proceedings of the Graphics Interface Conference (GI), pages 26–34, Vancouver (British Columbia), Canada, 18–20 June 1998.

[2] Hugues Hoppe. *Progressive Meshes*. In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 99–108, New Orleans (Louisiana), United States, 4–9 August 1996.

[3] Michael Lounsbery, Tony D. DeRose, and Joe D. Warren. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. ACM Transactions on Graphics, 16(1):34–73, January 1997.

[4] Renato Pajarola and Jarek R. Rossignac. *Compressed Progressive Meshes*. IEEE Transactions on Visualization and Computer Graphics, 6(1):79–93, January 2000.

[5] Pierre Alliez and Mathieu Desbrun. *Progressive Compression for Lossless Transmission of Triangle Meshes*. In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 195–202, Los Angeles (California), United States, 12–17 August 2001.

[6] Adrien Maglo, Clément Courbet, Pierre Alliez, and Céline Hudelot. *Progressive Compression of Manifold Polygon Meshes*. Computers & Graphics, 36(5):349–359, 2012.

[7] Sébastien Valette, Raphaëlle Chaine, and Rémy Prost. *Progressive Lossless Mesh Compression via Incremental Parametric Refinement*. Computer Graphics Forum, 28(5):1301–1310, 2009.

[8] Jingliang Peng, Yan Huang, C.-C. Jay Kuo, Ilya Eckstein, and Meenakshisundaram Gopi. *Feature Oriented Progressive Lossless Mesh Coding*. Computer Graphics Forum, 29(7):2029–2038, 2010.

[9] Amir Said and William A. Pearlman. *A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees*. IEEE Transactions on Circuits and Systems for Video Technology, 6(3):243–250, June 1996.

[10] Adrian Munteanu, Jan Cornelis, Geert Van der Auwera, and Paul Cristea. *Wavelet-based Lossless Compression Scheme with Progressive Transmission Capability*. International Journal of Imaging Systems and Technology, 10(1):76–85, 1999.

[11] David S. Taubman and Michael W. Marcellin. *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers, Norwell (Massachusetts), United States, 2001.

[12] David S. Taubman and Avideh Zakhor. *Multirate 3-D Subband Coding of Video*. IEEE Transactions on Image Processing, 3(5):572–588, September 1994.

[13] Jens-Rainer Ohm. *Three-dimensional Subband Coding with Motion Compensation*. IEEE Transactions on Image Processing, 3(5):559–571, September 1994.

[14] Yiannis Andreopoulos, Adrian Munteanu, Joeri Barbarien, Mihaela van der Schaar, Jan Cornelis, and Peter Schelkens. *In-band Motion Compensated Temporal Filtering*. Signal Processing: Image Communication, 19(7):653–673, August 2004. Special Issue on Subband/Wavelet Interframe Video Coding.

[15] Georges-Pierre Bonneau. *Multiresolution Analysis on Irregular Surface Meshes*. IEEE Transactions on Visualization and Computer Graphics, 4(4):365–378, October 1998.

[16] Sébastien Valette, Yun-Sang Kim, Ho-Youl Jung, Isabelle Magnin, and Rémy Prost. *A Multiresolution Wavelet Scheme for Irregularly Subdivided 3D Triangular Mesh*. In Proceedings of the 6th IEEE International Conference on Image Processing (ICIP), volume 2, pages 171–174, Kobe, Japan, 24–28 October 1999.

[17] Sébastien Valette and Rémy Prost. *Wavelet-Based Multiresolution Analysis of Irregular Surface Meshes*. IEEE Transactions on Visualization and Computer Graphics, 10(2):113–122, March/April 2004.

[18] Sébastien Valette and Rémy Prost. *Wavelet-Based Progressive Compression Scheme for Triangle Meshes: Wavemesh*. IEEE Transactions on Visualization and Computer Graphics, 10(2):123–129, March/April 2004.

[19] Paolo Cignoni, Claudio Montani, and Roberto Scopigno. *A Comparison of Mesh Simplification Algorithms*. Computers & Graphics, 22(1):37–54, February 1998.

[20] Pierre Alliez and Craig Gotsman. *Recent Advances in Compression of 3D Meshes*, pages 3–26. Advances in Multiresolution for Geometric Modelling (Mathematics and Visualization). Springer Berlin Heidelberg, 2005.

[21] Jingliang Peng, Chang-Su Kim, and C.-C. Jay Kuo. *Technologies for 3D Mesh Compression: A Survey*. Journal of Visual Communication and Image Representation, 16(6):688–733, December 2005.

[22] Faxin Yu, Hao Luo, Zheming Lu, and Pinghui Wang. *3D Mesh Compression*, pages 91–160. Three-Dimensional Model Analysis and Processing (Advanced Topics in Science and Technology in China). Springer Berlin Heidelberg, 2010.

[23] Adrien Maglo, Guillaume Lavoué, Florent Dupont, and Céline Hudelot. *3D Mesh Compression: Survey, Comparisons, and Emerging Trends*. ACM Computing Surveys, 47(3):44:1–44:41, February 2015.

[24] Leon Denis, Shahid Mahmood Satti, Adrian Munteanu, Jan Cornelis, and Peter Schelkens. *Scalable Intraband and Composite Wavelet-Based Coding of Semiregular Meshes*. IEEE Transactions on Multimedia, 12(8):773–789, December 2010.

[25] Jingliang Peng and C.-C. Jay Kuo. *Geometry-guided Progressive Lossless 3D Mesh Coding with Octree (OT) Decomposition*. ACM Transactions on Graphics, 24(3):609–616, July 2005.

[26] Yan Huang, Jingliang Peng, C.-C. Jay Kuo, and Meenakshisundaram Gopi. *A Generic Scheme for Progressive Point Cloud Coding*. IEEE Transactions on Visualization and Computer Graphics, 14(2):440–453, 2008.

[27] Ho Lee, Guillaume Lavou, and Florent Dupont. *Rate-distortion Optimization for Progressive Compression of 3D Mesh with Color Attributes*. The Visual Computer, 28(2):137–153, February 2012.

[28] Stéphane G. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 2nd edition, 1999.

[29] Ingrid Daubechies and Wim Sweldens. *Factoring Wavelet Transform into Lifting Steps*. The Journal of Fourier Analysis and Applications, 4(3):247–269, May 1998.

[30] Wim Sweldens. *The Lifting Scheme: A Construction of Second Generation Wavelets*. SIAM Journal on Mathematical Analysis, 29(2):511–546, March 1998.

[31] Daniel Cohen-Or, David Levin, and Offir Remez. *Progressive Compression of Arbitrary Triangular Meshes*. In Proceedings of the IEEE Conference on Visualization (VIS), pages 67–72, San Francisco (California), United States, 24–29 October 1999.

[32] Boris Delaunay. *Sur la sphère vide. A la mémoire de Georges Voronoï.* Bulletin de l'Académie des Sciences de l'URSS, (6):793–800, 1934.

[33] Lei He and Scott Schaefer. *Mesh Denoising via L0 Minimization*. ACM Transactions on Graphics, 32(4):64:1–64:8, July 2013.

[34] Detlev Marpe, Heiko Schwarz, and Thomas Wiegand. *Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard*. IEEE Transactions on Circuits and Systems for Video Technology, 13(7):620–636, July 2003.

[35] Adrian Munteanu, Jan Cornelis, Geert Van Der Auwera, and Paul Cristea. *Wavelet Image Compression – The Quadtree Coding Approach*. IEEE Transactions on Information Technology in Biomedicine, 3(3):176–185, September 1999.

[36] Sébastien Valette, Alexandre Gouaillard, and Rémy Prost. *Compression of 3D Triangular Meshes with Progressive Precision*. Computers & Graphics, 28(1):35–42, 2004.

# 4

# Region-of-Interest Coding

The second part of this dissertation handles the issue of *Region-of-Interest (ROI)* coding. In addition to resolution and quality scalability, which was tackled in Chapter 3 and which allows for scaling the resolution and quality of an entire model, a valuable functionality of any coding system is to allow for prioritizing specific spatial regions. Such functionality allows, for instance, for prioritizing the front side of a model or the face of a human virtual character. From the point-of-view of an encoder, such functionality can enforce specific regions to be transmitted and decoded prior to other regions. This can ensure that the most valuable regions are visualized faster in case of reduced bandwidths, or this can allow for limiting the decoding of the least valuable regions, only to be performed if these regions are actually required. In such scenarios, the prioritized regions are fixed by the encoding application; in other words, these regions are fixed within the data stream. From the point-of-view of a decoder, such ROI functionality can allow the decoding to adapt to specific use-cases, which in turn allows for minimizing the memory usage or bandwidth requirements. Such functionality becomes increasingly important with larger models which are to be processed. In these scenarios, it no longer suffices to have an ROI-adapted representation of a model: the functionality of *random accessibility* comes into play. Random access entails that a decoding system is able to pick from a data stream the parts that are required for decoding the desired ROIs without needing to read any other parts of this data stream.

**Contributions** In this second part of the dissertation, the required improvements have been investigated to enhance a wavelet-based irregular mesh coding system to allow for ROI coding both on the encoder and decoder side. These improvements have been applied to SFWInCS which was described in Chapter 3, resulting in a codec which has all fundamental functionalities to allow for the processing of increasingly larger meshes in a scalable way.

The work described in this chapter improves over the state of the art in several ways. From an encoder side, few prior work has been done considering prioritizing spatial regions for wavelet-based encoding.

- *Encoder-side ROI support for irregular meshes.* This functionality has not been proposed in literature so far, yet poses valuable opportunities as it allows for prioritizing regions in more memory-efficient irregular meshes.

- *ROI-steered upsampling.* In addition to supporting ROIs, the use of irregular meshes allows for upscaling *only* the ROI while representing the remaining regions at a lower resolution. This is valuable as inaccurately reconstructed high-resolution vertices still consume graphics memory space when rendering, even if little processing power is spent in generating them. An ROI-steered upsampling procedure avoids generating these altogether.

Such encoder-side ROI capabilities allow for optimizing data transmission in order to reduce bit rate, transmission time and memory use while improving visual quality in the region(s) of interest as determined valuable *by an encoder*. From a decoding perspective, the investigated techniques improve over the state of the art in the following ways:

- *An adaptive inverse wavelet transform for irregular meshes.* The wavelet coefficients are filtered *and* the appropriate connectivity information is provided before performing the inverse wavelet transform. This offers ROI support and allows for reduced graphics memory requirements compared to wavelet-based ROI decoding of *semi-regular* meshes.

- *Dynamic tiling per resolution in the wavelet domain.* By tiling in the wavelet domain, random access is provided while avoiding tiling artifacts in the spatial domain. The tiling adapts to the sampling densities within each resolution, allowing for an optimal trade-off between coding performance and random access granularity *per resolution*.

- *Tile-based RD optimization (RDO).* The tiled coding approach allows for optimizing the *rate-distortion (RD)* performance. As the tiles are independently encoded, this allows for reordering the tiles within a resolution level, and even across resolutions, while still preserving the

benefits offered by quality scalability. An RD-optimized bit plane transmission order is then obtained such that those bit planes of the tiles which give the largest quality gain at the lowest rate are coded first.

The remainder of the chapter discusses the following topics. Section 4.1 will discuss work related both to encoder-side and decoder-side ROI coding. In Section 4.2, ROIs for meshes are introduced in general. Section 4.3 then handles work done concerning encoder-side ROI coding. In this respect, Section 4.3.1 details the propagation of ROIs to ensure proper reconstruction, Section 4.3.2 tackles wavelet coefficient boosting which allows for encoding without altering the coding system, Section 4.3.3 then discusses the approach to enable the transmission of a predefined ROI, and Section 4.3.4 discusses how to further improve the results by providing an ROI-steered upsampling. Next, Section 4.4 handles the work done to allow for decoder-side ROIs. Here, Section 4.4.1 discusses the adaptive inverse wavelet transform and Section 4.4.2 continues by defining tiles in the mesh data. Subsequently, Section 4.5 details how these tiles can be exploited to improve the RD performance. The evaluation is presented in Sections 4.6.1 and 4.6.2, followed by conclusions in Section 4.7.

## 4.1 Related Work

Recall that a conventional mesh coding system initially transforms a mesh $M$ into $M^{\text{TF}}$, after which an encoding step results in a compressed representation $M_{enc}$ of the input mesh (see Figure 2.16, page 42). Scalable systems then allow for progressively refining a mesh. In Chapter 3, such refinement operations are performed over the entire mesh surface. The current chapter investigates refinement operations which are localized in specific ROIs, defined either at the encoder side or at the decoder side. This corresponds to encoder-side and decoder-side ROI coding functionalities respectively.

The encoding of an *encoder-side ROI* is symmetric: the decoded data is equivalent to the encoded data. Encoder-side ROIs are useful for prioritizing regions within $M$ for coping with reduced bandwidths; e.g., prioritizing facial details of a virtual human character over details in the remainder of his body. This codec functionality is further discussed in Section 4.1.1.

Conversely, a *decoder-side ROI* adapts the resolution over the mesh surface to the needs of the decoder, requiring an encoding procedure which can no longer be symmetric. A decoder must be able to request specific parts of a model at specific resolutions, affecting either $M_{enc}$, $M^{\text{TF}}$ or both. This type of functionality is further addressed in Section 4.1.2.

### 4.1.1 Encoder-Side Regions of Interest

The premise for encoder-side ROIs is easy to formulate: store data such that the prioritized regions are transmitted and decoded before the unprioritized background regions. *Vertex-by-vertex based, cLOD systems* such as [1] or [2] can trivially prioritize regions: vertices are prioritized individually, directly altering the refinement and encoding order. *Multi-resolution systems* such as [3–6] and SFWInCS presented in Chapter 3 call for specific designs, as individual vertices are no longer accessible after the transform which results in $M^{\text{TF}}$. Few solutions have been proposed for encoder-side ROIs for multi-resolution systems. A solution which was inspired by the JPEG 2000 maxshift and general scaling operations [7] was proposed for semi-regular codecs by Zheng et al. [8]. Encoder-side wavelet-based ROI for *irregular* meshes has not been proposed in literature so far.

### 4.1.2 Decoder-Side Regions of Interest

Contrary to encoder-side ROIs which are based on predefined prioritizations of specific spatial regions in the mesh, a decoder-side ROI is more involved as a single bit stream needs to provide for a dynamic, arbitrary ROI selection in an efficient way.

A first 3D graphics domain where arbitrary ROI decoding was required was terrain visualization. Gobbetti et al. [9] proposed adaptive meshes for terrain data. The employed grid-based structure of [9] allowed for easy random access support as each two adjacent triangles form a root for further subdivision and can be processed individually. Similar tiled approaches have been suggested e.g. in [10, 11], and have proven to be successful for $2.5D$ surfaces, i.e. surfaces where each $(x, y)$ couple corresponds to, at most, a single point $(x, y, z)$ on the surface. However, such approaches are too restrictive for the general case of 3D surfaces.

In the following, a short overview of the literature is given by focusing on *transform-based* ROI coding in Section 4.1.2.1 and *tile-based* ROI coding in Section 4.1.2.2.

#### 4.1.2.1 Transform-Based Region-of-Interest Coding

For cLOD representations, decoder-side ROIs (e.g., a view-dependent selection) can be obtained by either carefully applying or skipping refinements. Such approaches are often termed as *selective refinement* methods. Hoppe [12] proposed such an ROI decoding approach for his progressive meshes method [1], which was implemented on graphics hardware by Hu et al. [13]. For *multi-resolution* representations, where each refinement step adds multiple vertices, the refinements need to be restricted based on the required ROIs. Gioia et al. [14] have proposed ROI support for wavelet representations of *semi-regular meshes*, whereby the

resolution can be adapted over the geometric surfaces by suppressing wavelet coefficients. This corresponds to what is proven to be an adaptive inverse wavelet transform for *semi-regular meshes*. Roy et al. [15] have proposed a multi-resolution analysis for meshes with surface attributes, using the edge collapse operations of [1] and its selective refinement [12]. Similar to Gioia et al., the detail coefficients are also selectively filtered [15].

These solutions are termed *transform-based* ROI coding methods as the transform is enhanced to offer ROI scalability. This allows for adapting the resolution in a fine-grained manner over the surface. However, these solutions suffer from the fact that only the refinement steps are ROI-aware without providing for random access within the data stream; that is, the ROIs are only defined on the decoded transformed mesh $\hat{M}^{\text{TF}}$ (see Figure 2.17 on page 43), requiring full decoding of $M_{enc}$.

#### 4.1.2.2 Tile-Based Region-of-Interest Coding

The idea of tiling and encoding a mesh was proposed by [16] to allow for out-of-core processing, i.e., to handle data which no longer fits into main memory. However, this approach did not offer random access due to the dependencies between the tiles during encoding. In most works, a global tiling is decided upon, after which each tile is individually encoded. These approaches are termed as *tile-based*, whereby ROI scalability is offered by the tiling process itself.

Initial tile-based approaches used a *single-rate codec operating on a global tiling* of a mesh. Choe et al. [17] proposed a tiling which resulted in a wire-net mesh representing the tile borders, and encoding each tile independently using a single-rate encoder. In [18], the authors improved upon [17], allowing for more control over random accessibility, better compression, and support for large meshes which require out-of-core processing. Yoon and Lindstrom [19] also group triangles in tiles, after which each of these tiles is compressed using their streaming mesh compression approach [20]. In this method, each tile is also encoded at a single rate. Such approaches make binary decisions on tile granularity: either the tile is (partly) visible and is entirely decoded, or it is not visible and the decoding is skipped. The main downside is that this does not allow for scaling the resolution or quality of the visible regions, with issues similar to those of single-rate coding without tiling.

By employing a *progressive encoding per global tile*, each tile can be decoded at a different level of detail, for instance depending on the distance to a virtual camera, or depending on some specified region of interest. Liu and Zhang describe a wavelet-based mesh compression scheme with random access [21], as one of the first works which combine random access with scalable coding, as opposed to utilizing a single-rate compression per tile. Making use of [4], their work operates on semi-regular meshes to obtain progressive encoding per

tile, while random access is offered by exploiting the employed zerotree-based encoding and considering the base triangles as tiles. A binary decision is still made based on back-face culling: if an oriented triangle of the base mesh has its front side facing the virtual camera, the corresponding trees of coefficients are transmitted. This avoids tiling artifacts but results in a high sampling density, which is common for semi-regular mesh representations. Roudet et al. [22] similarly use [4], and define global tiles by projecting high-resolution data onto the base triangles. In [23], Cheng et al. improve the progressive coder of [24] to use multi-granular quantization, and use this coder to encode tiles obtained after global tiling. No details are given concerning approaches to avoid artifacts at the tile borders. Maglo et al. [25] continued on this approach, using the progressive codec of [26] combined with post-processing of the tile borders to ensure a valid topology. This is done by moving the border vertices of the higher-resolution tile towards the border vertices of the lower-resolution tile, and triangulating any remaining holes. This process results in visual blocking artifacts. This method was further continued in [27], allowing for smooth transitions between adjacent tiles without post-processing, with the restriction that neighboring regions can differ by at most one resolution level. Du et al. [28] propose a coding method based on the codec of Gandoin and Devillers [29], which refines vertex locations iteratively via increasingly finer-grained quantizers. A two-level tree is proposed where the root represents the coarsely-quantized base mesh to be decoded completely, while the subtrees, i.e., the tiles, can be individually decoded each to its desired level. Again, care has to be taken for border triangles lying across subtrees. Finer granularity and random access for vertex-based selective-refinement coding was proposed by Kim et al. in [30] by partitioning the original vertex hierarchy into sub-blocks acting as tiles for random accessibility.

The downside of these *tile-based* solutions is that they cannot adapt the resolution in a fine-granular way as offered by *transform-based* approaches; the granularity is limited by the tile granularity. Consequently, care has to be taken near tile borders to avoid artifacts, which can be challenging if the resolutions of neighboring tiles differ by multiple levels. Tile-based approaches are compared to transform-based approaches in Table 4.1.

| | **Transform-Based ROI** | **Tile-Based ROI** |
|---|---|---|
| **Transform** | single altered inverse transform | unaltered instance per tile |
| **Encoding** | unaltered encoding of *all data* | unaltered encoding of *data per tile* |
| ROI granularity | resolution varies over the surface | single resolution per tile |
| Fixing tiling artifacts | not necessary | careful resolution selection |
| Random access | not possible | tiles independently encoded |

*Table 4.1: Summary of decoder-side ROI approaches. Transform-based approaches define ROIs on the decoded, transformed data and require all data to be decoded. Tile-based approaches partition a mesh, and transform and encode each partition individually.*

**Dynamic Tiling** A solution where the tiles used for random access change dynamically is given by Courbet and Hudelot [31]. The authors propose a recursive mesh splitting approach to obtain hierarchical random access for polygonal meshes. The resulting representation allows to randomly access arbitrarily small portions of a mesh, but does not support resolution scalability: the portions of a mesh within the ROI have to be iteratively subdivided until the original triangle mesh is obtained for the ROI. The subsequently smaller tiles of [31] allow for more efficient processing.

## 4.2 Wavelet-Based Coding and Regions of Interest

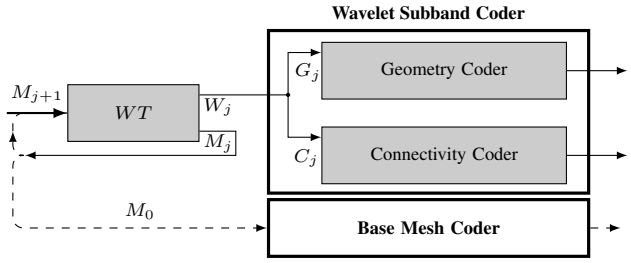



*Figure 4.1: Single pass through a wavelet-based mesh encoding system. This figure depicts a single pass through the mesh encoding system as was depicted in Figure 3.1. A mesh $M_{j+1}$ is transformed into a lower-resolution mesh $M_j$ and a wavelet subband $W_j$, consisting of connectivity information $C_j$ and geometry information $G_j$. This transformation is repeated until a base mesh $M_0$ is obtained.*

In Chapter 3, a resolution and quality scalable mesh coding system named SFWInCS, was presented. Figure 3.1 (page 61) depicts the basic architecture for such a wavelet-based mesh coding system, of which a single pass is once more illustrated in Figure 4.1. The wavelet transform iteratively downsamples an original mesh $M$, generating the multi-resolution representation ($M$=)$M_R$, $M_{R-1}$, ..., $M_0$, i.e., there are $R$+1 resolutions. Downsampling $M_{j+1}$ results in a lower-resolution mesh $M_j$, a wavelet subband $W_j$, consisting of geometry information $G_j$ and, for an *irregular* mesh codec, connectivity information $C_j$. For a semi-regular mesh codec, no connectivity information is required as the connectivity of a *semi-regular* mesh is determined by a subdivision scheme.

The base mesh $M_0$ is encoded using any arbitrary single-rate coder, such as the state-of-the-art codec of Touma and Gotsman [32]. The wavelet subbands and connectivity information are encoded using the *Geometry Encoder* and *Connectivity Encoder* respectively. As indicated in Figure 3.1, encoding the connectivity information results in a single data block per resolution. The wavelet coefficients however are typically quantized using *successive approximation*

*quantization (SAQ)* [33] and are encoded in bit-plane-by-bit-plane fashion. This is a conventional approach, e.g. employed by Denis et al. [6], which allows for quality scalability and RDO (as was shown in Sections 3.3 and 3.4).

In the context of ROI coding, the concepts of *Region-of-Interest (ROI)* and *background (BG)* are used. The ROI is the prioritized region, i.e., the regions prioritized by an encoder or requested by a decoder. The BG, conversely, is formed by the unprioritized regions or the regions that are not requested. To investigate the ROI coding functionality, the codec presented in Chapter 3, which yields state-of-the-art compression performance for irregular meshes, is enhanced. Given meshes $M_j$, an (encoder-side or decoder-side) application can define subsets of vertices as *spatial* regions of interest $ROI_j^S \subset M_j$. Employing the support of a wavelet coefficient $S(w)$ as defined in Equation 3.3, each $ROI_j^S$ is translated to a region of interest in the wavelet domain $ROI_j^W$ as follows:

$$ROI_j^W = \{w \in W_j \mid S(w) \cap ROI_j^S \neq \varnothing\}, \tag{4.1}$$

i.e., $ROI_j^W$ is the subset of those wavelet coefficients in $W_j$ whose support overlaps with the given $ROI_j^S$. Additionally, define the support of the set $ROI_j^W$ as:

$$S(ROI_j^W) = \bigcup_{w \in ROI_j^W} S(w). \tag{4.2}$$

To accommodate for the varying resolution across a surface, the notations $M_{\alpha \leq j}$,$M_{\beta \leq j}$,... are introduced to identify several *partial* reconstructions of $M_j$, i.e., the resolution varies over the surface, reaching at most resolution $j$. Given such a partially reconstructed mesh $M_{\alpha \leq j}$ and $ROI_j^W$, Equation 3.1 is generalized as follows:

$$\begin{aligned} M_{\alpha \leq j+1} &= WT^{-1}(M_{\alpha \leq j}, ROI_j^W), \\ &\text{with } S(ROI_j^W) \subset M_{\alpha \leq j}. \end{aligned} \tag{4.3}$$

The requirement in Equation 4.3 states that if the support for each wavelet coefficient $w \in ROI_j^W$ is present in the *partially* reconstructed $M_{\alpha \leq j}$, then the inverse wavelet transform results in an upsampled mesh $M_{\alpha \leq j+1}$. Otherwise, if there exists at least one wavelet coefficient with a support which is not entirely present in $M_{\alpha \leq j}$, then the inverse transform is topologically ill-defined as the prediction term in Equation 3.2 cannot be evaluated properly.

## 4.3 Encoder-Side Region-of-Interest Coding

The encoding of an encoder-side ROI, or a *predefined* ROI, is based on the ROI coding methods defined in the JPEG 2000 image coding standard [34]. The standard defines (1) a general scaling based method which allows for scaling

rectangular or elliptical regions at arbitrary scaling values, and (2) the maximum shift (maxshift) method which allows for arbitrarily-shaped regions to be encoded. The former allows for choosing for each region a relative importance w.r.t. the BG but requires transmitting the ROI masks. The latter only allows for one shift, i.e., the ROI gets full precedence over the BG, but the ROI masks are arbitrarily shaped and are not encoded. This dissertation applies the maxshift method to 3D mesh coding.

The next sections discuss the propagation of higher-resolution ROIs to lower resolutions in Section 4.3.1 and wavelet coefficient boosting in Section 4.3.2. Section 4.3.3 then details an ROI-aware transmission order, and Section 4.3.4 finally tackles deterioration in the BG by proposing an ROI-steered upsampling step.

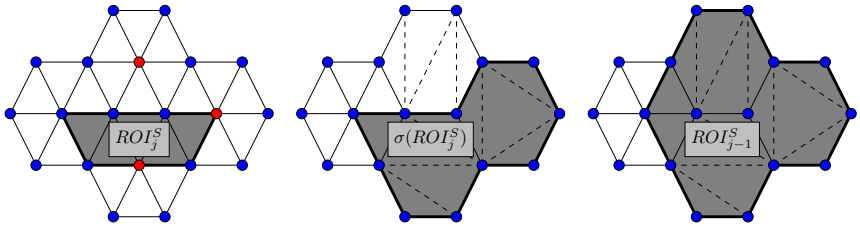
### 4.3.1 Propagating an ROI Mask

An encoder can decide, at any resolution, which regions are of interest. This arbitrary choice is restricted by the fact that $ROI_j^S$ must be reconstructable after upsampling the ROIs of all lower resolutions $k < j$. To ensure this, $ROI_j^S$ at resolution $j$ is propagated to the region $\sigma(ROI_j^S)$ at resolution $j-1$, additionally requiring that $\sigma(ROI_j^S) \subseteq ROI_{j-1}^S$, whereby $\sigma$ is the region propagation operator. This was described in [7] for image coding; analogue reasoning can be applied for 3D meshes. The region $\sigma(ROI_j^S)$ is obtained as follows:

$$\begin{aligned} &\sigma(ROI_j^S) = \\ &\{v \in M_{j-1} \mid \gamma^{-1}(v) \in ROI_j^S\} \quad \cup \\ &\{v \in M_{j-1} \mid \exists v_j^o \in (ROI_j^S \cap M_j^o), v \in S(\omega(v_j^o))\}. \end{aligned} \tag{4.4}$$

That is, $\sigma(ROI_j^S)$ consists of each vertex $v \in M_{j-1}$ which can either be mapped to an (even) vertex in $ROI_j^S$, or an odd vertex $v_j^o \in ROI_j^S$ can be found whose wavelet coefficient support $S(\omega(v_j^o))$ contains $v$. Hence, all even vertices in $ROI_j^S$ are preserved in $\sigma(ROI_j^S)$ while odd vertices require the support of the corresponding wavelet coefficients to be present as well. Requiring that $\sigma(ROI_{k+1}^S) \subseteq ROI_k^S \ \forall k < j$ ensures that any reconstruction $M_{\alpha \leq j}$ can be obtained with $ROI_j^S$ properly reconstructed. The relationships between the high-resolution $ROI_j^S$ and the low-resolution $\sigma(ROI_j^S)$ and $ROI_{j-1}^S$ are illustrated in the example of Figure 4.2.

Following this process, a base mesh $M_0$ is obtained, together with a set of wavelet subbands $W_j$ for which their $ROI_j^W$ are known. These subbands $W_j$ are now encoded *with* information on their $ROI_j^W$ without making changes to the wavelet subband coder, as discussed next.




*(a) ROI in $M_j$* *(b) Propagation to $M_{j-1}$* *(c) Final ROI in $M_{j-1}$*

*Figure 4.2: ROI propagation. (a) shows mesh $M_j$ with a selected ROI. After downsampling, the odd vertices (drawn in red) are no longer available in $M_{j-1}$ and the ROI is expanded in (b) to properly represent $ROI_j^S$ at resolution $j-1$. This ROI, denoted as $\sigma(ROI_j^S)$, is the propagation of $ROI_j^S$ to the lower resolution $j-1$. $\sigma(ROI_j^S)$ must be a subset of the final ROI defined in $M_{j-1}$, denoted by $ROI_{j-1}^S$ and depicted in (c).*

### 4.3.2 Boosted Wavelet Coefficients

As wavelet coefficients are quantized, the same ideas underlying the maxshift coding method can be applied, allowing for storage and transmission with ROI support without altering the employed encoder and decoder. This is done by using boosted wavelet coefficients: the quantized wavelet coefficients $w \in ROI_j^W$ are premultiplied by a scaling value $s_j$. Given $s_j > |w|, \forall w \in W_j \setminus ROI_j^W$, the set of boosted wavelet coefficients $W_j^B$ is constructed as:

$$\begin{cases} \forall w \in ROI_j^W : s_j w \in W_j^B \\ \forall w \in W_j \setminus ROI_j^W : w \in W_j^B \end{cases}. \tag{4.5}$$

That is, the wavelet coefficients within the ROI are upscaled by a scaling factor surpassing the largest wavelet coefficient magnitude found within the BG, as illustrated in Figure 4.3.

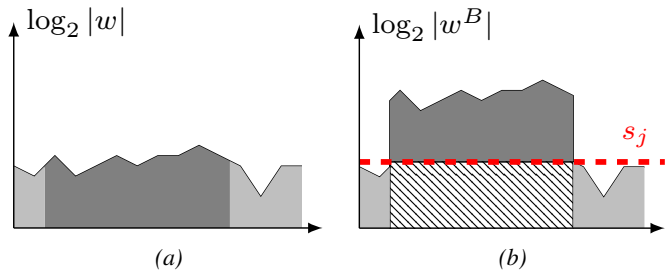



*Figure 4.3: Boosted wavelet coefficients. An example magnitude profile of wavelet coefficients $w \in W_j$ is given in (a), with the coefficients within the ROI shown in darker gray. (b) shows the boosted wavelet coefficients $w^B \in W_j^B$.*

The encoder processes $W_j^B$ instead of $W_j$, and is unaware of any ROI functionality. As the wavelet coefficients pertaining to the ROI occupy bit planes with a higher significance after scaling, these wavelet coefficients will be encoded prior to the coefficients in the BG. At the decoder side, after decoding $W_j^B$ the original wavelet coefficients can be obtained as:

$$\begin{cases} \forall w \in W_j^B \wedge w \geq s_j : w/s_j \in W_j \\ \forall w \in W_j^B \wedge w < s_j : w \in W_j \end{cases}. \tag{4.6}$$

The decoder classifies wavelet coefficients with a magnitude larger than $s_j$ as being part of the ROI and will scale them back down by this factor $s_j$, after which the classical inverse wavelet transform is performed.

### 4.3.3 ROI-Aware Transmission Order

*Per wavelet subband*, the ROI is stored or transmitted before the BG. One can also ensure that the ROIs of *all* resolutions are stored or transmitted before any BG information. To this end, the bit plane coding order must be altered such that the coding layers associated with the BG information in every resolution are stored at the end of the bit stream, after storing the ROIs.

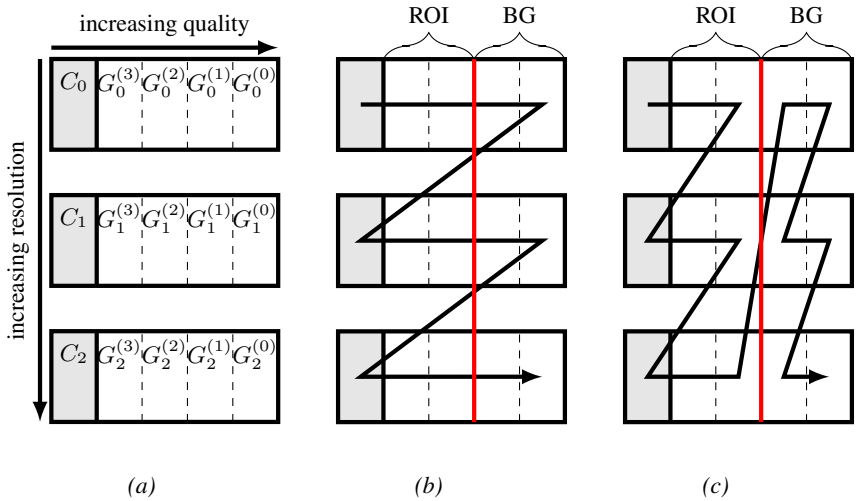



*Figure 4.4: Predefined ROI transmission orders. (a) shows the data layers after coding; for resolution $j$, $C_j$ represents connectivity information and $G_j^{(i)}$ geometry information for bit plane $i$. (b) shows the default resolution-scalable encoding order of Chapter 3. (c) depicts the ROI-aware transmission order.*

Figure 4.4a shows a simplified example using 2 bit quantization and scaling the obtained wavelet coefficients with $s = 2^2$. Note that this scaling value

(and consequently the number of bit planes used for the BG information) varies, in general, across subbands depending on the largest magnitude coefficient in the corresponding subband. At each resolution level $j$, the system produces a connectivity layer $C_j$ and corresponding geometry layers $G_j^{(k)}$. The maxshift method described above implicitly encodes the ROI before the BG, *within each subband*. This is depicted in Figure 4.4b, where it is shown that the ROI bit planes are coded before the BG bit planes. This prioritizes the ROI information only within each resolution level, but not across resolutions: an ROI at a given resolution is only streamed after streaming both the ROI and BG information of the previous resolution. Analogue to how the rate allocator in JPEG 2000 reorders the bit stream parts across coding blocks and wavelet subbands to have the ROI streamed prior to the BG information, bit stream parts need to be reordered across resolutions.

Section 3.3.2 showed that an arbitrary transmission order of wavelet coefficient bit planes is possible without a negative impact on the lossless coding rate. Hence, the transmission can be made ROI-aware by streaming per resolution level the bit planes pertaining to the ROI, subsequently followed by streaming the BG information of every resolution. This transmission order is shown in Figure 4.4c.

### 4.3.4 ROI-Steered Upsampling

Consider the ROI-aware transmission order described above and let $Q$ denote the amount of quantization bits, $s_j$ the scaling value for subband $j$, and $k_j = \log_2 s_j$; the decoder reads base mesh $M_0$ and then receives $C_0$, $G_0^{(Q+k_0-1)}$, $\ldots$, $G_0^{(k_0+1)}$, $G_0^{(k_0)}$ to reconstruct $M_1$ for which the ROI will be accurately refined. Subsequently, $C_1$, $G_1^{(Q+k_1-1)}$, $\ldots$, $G_1^{(k_1+1)}$, $G_1^{(k_1)}$ are received to reconstruct $M_2$, and so on. The ROI is again accurately refined; in the BG, however, geometric errors accumulate. Figure 4.5 shows the back of *fandisk* after decoding the ROIs for all resolutions, i.e., up to $G_{R-1}^{(k_{R-1})}$, without decoding BG information.

This quality degradation appears because the connectivity information is not ROI-aware, that is, the set $\widetilde{M}_j^o$ of reconstructed odd vertices, i.e.

$$\widetilde{M}_j^o = \{\omega^{-1}(w) \mid w \in W_{j-1}\}, \tag{4.7}$$

creates *all* vertices when upsampling while only the vertices in the ROI are accurately positioned. The small distortions due to prediction errors accumulate with each resolution, resulting in increasingly worse artifacts in the BG.

A trivial solution involves low-pass filtering the BG vertices to ensure a smooth BG surface after decoding. However, irregular meshes allow for a more straightforward and beneficial approach, which is to limit the upsampling to only create vertices in the decoded ROI, that is:

$$\widetilde{M}_j^o = \{\omega^{-1}(w) \mid w \in ROI_{j-1}^W \subset W_{j-1}\}. \tag{4.8}$$

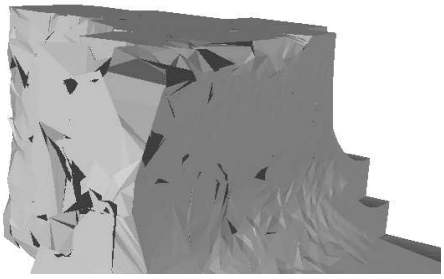


*Figure 4.5: Geometric degradation in the BG*

$ROI^W_{j-1}$ can be simply detected as the set of all *non-zero* wavelet coefficients when only decoding layers associated with the ROI; however, one may miss possible zero-magnitude coefficients in $ROI^W_{j-1}$. These zero-magnitude coefficients which are effectively within the ROI are then nonetheless determined as BG and the corresponding patches are not retriangulated, which may lead to topological issues at higher resolutions: subsequent ROI-aware reconstruction steps possibly depend on undetected ROI vertices.

To avoid this problem, in addition to pre-multiplication and post-division, one can add a pre-increment and post-decrement to Equations 4.5 and 4.6 respectively, to ensure accurate encoding and detection of $ROI^W_{j-1}$. Smoothness in the BG is now ensured by the reduced resolution instead of by smoothed samples. Adapting the resolution over the surface ensures a minimal amount of vertices and triangles, which reduces memory load when rendering.

## 4.4 Decoder-Side Region-of-Interest Coding

Whereas encoder-side ROIs discussed in Section 4.3 allow for an encoder to statically prioritize specific regions at specific resolutions, decoder-side ROIs are dynamically determined and ask for an improved codec design to ensure efficient data retrieval and mesh representations.

The next two sections describe decoder-side ROI coding. In Section 4.4.1 a novel adaptive inverse wavelet transform is described which allows for ROI decoding of irregular meshes. Such an adaptive inverse transform offers fine-grained resolution control over the entire mesh surface, ensuring an optimal mesh *representation*. To allow for random access into the encoded data, ensuring efficient data *retrieval*, dynamic tiling is described in Section 4.4.2.

### 4.4.1 Adaptive Inverse Wavelet Transform

The ROI-steered upsampling discussed in Section 4.3.4 is suitable for *encoder-side* ROI approaches for which ROIs are predefined at the encoding side: per resolution $j$, a spatial-domain $ROI_j^S$ will be accurately reconstructed in $M_{\alpha\leq j}$. In contrast, *decoder-side* ROI approaches have to accommodate *interactive* ROIs, which are arbitrarily defined while resolutions are being reconstructed. That is, at each resolution $j$, an $ROI_j^S$ is specified by the user at the decoder side. During upsampling, these arbitrary ROIs cannot take into account (unknown) higher-resolution ROIs, as done in the encoder-side ROI approach presented in Section 4.3.

The proper reconstruction of $M_{\alpha\leq j+1}$ given $ROI_j^S$ possibly requires modifying lower-resolution meshes $M_{\alpha\leq k}, k \leq j$ to obtain additional samples to satisfy the condition in Equation 4.3. These lower-resolution meshes $M_{\alpha\leq k}$ need to be reconstructed with a larger ROI denoted by $ROI_{k|j}^S$, which is the *expansion* of $ROI_k^S$ to yield a proper reconstruction at the higher-resolution $j$.

Consider for instance that $M_{\alpha\leq j}$ is accurately reconstructed given all lower-resolution regions of interest $ROI_k^S, k < j$. To reconstruct $M_{\alpha\leq j+1}$ given $ROI_j^S$, $M_{\alpha\leq j}$ needs to be modified to $M_{\beta\leq j}$, obtaining the additional samples to ensure $S(ROI_j^W) \subset M_{\beta\leq j}$. First, the given $ROI_j^S$ needs to be expanded to $ROI_{j|j}^S$ in order to encompass the support $S(ROI_j^W)$ defined in Equation 4.2, i.e., to include the supports of the wavelet coefficients $w \in W_j$ which overlap with $ROI_j^S$:

$$ROI_{j|j}^S = ROI_j^S \cup S(ROI_j^W). \tag{4.9}$$

To reconstruct $M_{\beta\leq j}$, the lower-resolution mesh $M_{\alpha\leq j-1}$ needs to be upsampled considering the interactively-specified ROI at resolution $j-1$, $ROI_{j-1}^S$, expanded to $ROI_{j-1|j}^S$, in order to ensure its accurate reconstruction at resolution $j$. One can write:

$$ROI_{j-1|j}^S = ROI_{j-1|j-1}^S \cup \sigma(ROI_{j|j}^S). \tag{4.10}$$

The new ROI encompasses both $ROI_{j-1|j-1}^S$, i.e., the expansion of $ROI_{j-1}$ which yields a proper reconstruction at resolution $j-1$, and the propagation of $ROI_{j|j}^S$ to resolution $j-1$, as defined in Equation 4.4.

Recursively, to reconstruct $M_{\beta\leq j}$ given this further expanded $ROI_{j-1|j}^S$, $M_{\alpha\leq j-1}$ needs to be modified to $M_{\beta\leq j-1}$, reconstructed from the subsequent lower-resolution mesh $M_{\alpha\leq j-2}$ by considering $ROI_{j-2|j}^S$, with:

$$ROI_{j-2|j}^S = ROI_{j-2|j-1}^S \cup \sigma(ROI_{j-1|j}^S). \tag{4.11}$$

Equations 4.10 and 4.11 can be generalized, defining the expansions for $k < j$ recursively as:

$$ROI_{k|j}^S = ROI_{k|j-1}^S \cup \sigma(ROI_{k+1|j}^S). \tag{4.12}$$

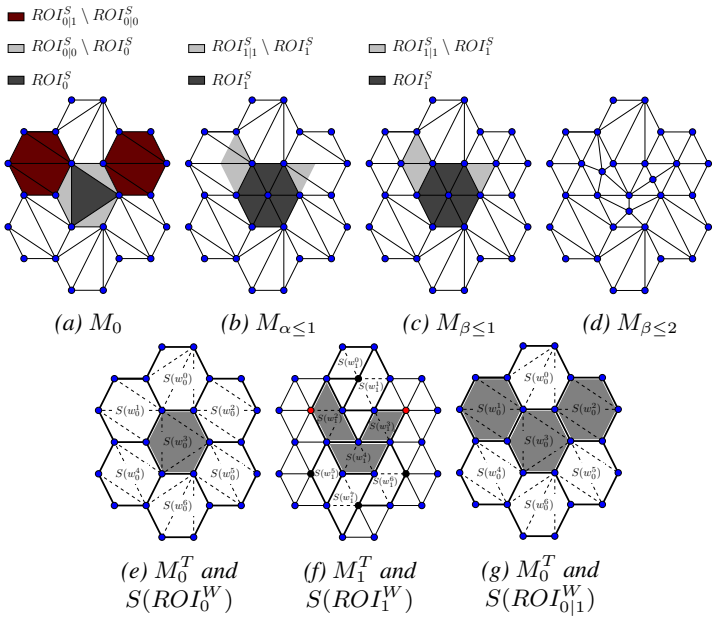



*Figure 4.6: Conceptual example of the adaptive inverse wavelet transform. The figure shows the decoded mesh, with determined ROIs indicated in dark gray and expanded ROIs in light gray. The inverse transform operations subsequently transform $M_0$ (a) to $M_{\alpha\leq 1}$ (b) at resolution 1, a second version $M_{\beta\leq 1}$ (c) and finally $M_{\beta\leq 2}$ (d). The requested $ROI^S$ are shaded in dark gray, and the corresponding $S(ROI^W)$ are depicted on the bottom line.*

The expansion $ROI^S_{k|j}$ of $ROI^S_k$ given the *newly determined* $ROI^S_j$ is the union of its expansion $ROI^S_{k|j-1}$ complying for the *already known* $ROI^S_{j-1}$, and the propagation $\sigma(ROI^S_{k+1|j})$ of the higher-resolution expansion $ROI^S_{k+1|j}$.

Given the spatial-domain $ROI^S_j$ with wavelet-domain $ROI^W_j$ obtained via Equation 4.1, and $S(ROI^W_j) \not\subset M_{\alpha\leq j}$, the adaptive inverse wavelet transform can now be recursively defined to obtain $M_{\beta\leq j}$:

$$M_{\beta\leq 1} = WT^{-1}(M_0, ROI^W_{0|j}), \tag{4.13}$$

$$M_{\beta\leq k} = WT^{-1}(M_{\beta\leq k-1}, ROI^W_{k-1|j}). \tag{4.14}$$

With this, $WT^{-1}(M_{\beta\leq j}, ROI^W_j)$ is properly defined.

The adaptive inverse wavelet transform is illustrated via the example of Figure 4.6. The figure shows a mesh $M$ on the top line, and its template mesh $M^T$ on the bottom line. The ROIs in Figure 4.6 are shaded as indicated in the figure. The performed steps are described in Listing 4.1.

1: **function** UPSAMPLE($ROI_0^S$)
2: **given** $M_0$ and $ROI_0^S$ ▷ (Fig. 4.6a)
3: $ROI_0^W = \{w_0^3\}$
4: $S(ROI_0^W) = S(w_0^3) \subset M_0$ ▷ (Fig. 4.6e)
5: **return** $M_{\alpha\leq 1} = WT^{-1}(M_0, ROI_0^W)$ ▷ (Fig. 4.6b)
6: **end function**

7: **function** UPSAMPLE($ROI_1^S$)
8: **given** $ROI_1^S$ ▷ (Fig. 4.6b)
9: $ROI_1^W = \{w_1^2, w_1^3, w_1^4\}$
10: $S(ROI_1^W) = S(w_1^2) \cup S(w_1^3) \cup S(w_1^4)$ ▷ (Fig. 4.6f)
11: $\implies S(ROI_1^W) \underline{\underline{\not\subset M_{\alpha\leq 1}}}$ ▷ (Fig. 4.6b)
12: **function** UPSAMPLE($ROI_{0|1}^S$)
13: $ROI_{0|1}^S = ROI_{0|0}^S \cup \sigma(ROI_{1|1}^S)$ ▷ (Fig. 4.6a)
14: $ROI_{0|1}^W = \{w_0^1, w_0^2, w_0^3\}$
15: $S(ROI_{0|1}^W) = S(w_0^1) \cup S(w_0^2) \cup S(w_0^3)$ ▷ (Fig. 4.6g)
16: **return** $M_{\beta\leq 1} = WT^{-1}(M_0, ROI_{0|1}^W)$ ▷ (Fig. 4.6c)
17: **end function**
18: **return** $M_{\beta\leq 2} = WT^{-1}(M_{\beta\leq 1}, ROI_1^W)$ ▷ (Fig. 4.6d)
19: **end function**

*Example 4.1: Adaptive inverse wavelet transform*

Observe that the mapping $\mu$ is no longer surjective if $M_{\alpha\leq j} \neq M_j$:

$$v^T \in M_j^T \nRightarrow \exists v \in M_{\alpha\leq j} : \mu(v) = v^T. \tag{4.15}$$

Although not *all* $v^T \in M_j^T$ have a corresponding $v \in M_{\alpha\leq j}$, the recursive expansion and propagation of ROIs (Equation 4.12) does ensure such a correspondence for the *required* vertices.

The main advantage of such an adaptive inverse wavelet transform is that a rendering system is allowed to select its desired ROI, reducing the processing power spent for the inverse wavelet transform, and reducing the data transmission to, and the memory usage on, the graphics hardware. The main disadvantage is that this approach still requires decoding all data in $M_{enc}$. Additionally, one notes that the size of the transformed data $M^{\text{TF}}$, i.e., before any arithmetic or entropy coding, is often proportional to the size of $M$ itself. Hence, while graphics memory requirements are optimized, the memory required to obtain this ROI-adapted model is not.

### 4.4.2 Dynamic Tile-Based Coding

To reduce memory and bandwidth requirements when reconstructing $M_{\alpha\leq j+1}$ given $ROI_j^S$, only the wavelet coefficients $w_j \in ROI_j^W$ and $w_k \in ROI_{k|j}^W$

for $k < j$ need to be decoded. This reveals the conventional trade-off in randomly-accessible coding: at one end of the spectrum, all samples are individually decodable which means there is *no entropy coding* but perfect random accessibility; at the other end of the spectrum, all samples are encoded simultaneously resulting in optimal coding performance but *no random accessibility*. To solve this trade-off, the proposed coding paradigm makes use of dynamic tile-based coding, detailed in this section.

The connectivity and geometry coder of Chapter 3 were employed, which can be summarized as follows. For each wavelet coefficient $w \in W_j$, the connectivity information $w_c \in C_j$ is represented by assigning a binary value $\beta(e_j^k)$ to each edge $e_j^k$, indicating whether or not the edge is preserved when upsampling. This is shown in Figure 4.6e for $M_0^T$ using full lines for edges that are preserved, and dashed lines across which triangles are merged. Merging triangles across such edges results in non-triangular faces which are immediately recognized as *patches*. An odd vertex $v_{j+1}^{o_i}$ is added per patch, and the patch is retriangulated (depicted in Figure 4.6f). The geometry information $w_g \in G_j$ allows for an accurate reconstruction of the vertex positions.

The connectivity samples, i.e. a sample per edge for $C_j$, and the geometry samples, i.e. a single sample per wavelet coefficient for $G_j$, are encoded using a connectivity coder and a geometry coder respectively. Both make use of octree coding, by embedding the samples through the template mesh. Let $v_j^{k_a}$ and $v_j^{k_b}$ denote the two vertices which define $e_j^k$. Given a spatial cell $C(\mathbf{k}, \mathbf{u})$ as defined in Equation 3.6, the connectivity samples with binary values $\beta(e_j^k)$ are embedded at

$$\check{\mathbf{e}}_j^k = \Big[\frac{1}{2}\mu(v_j^{i_a}) + \frac{1}{2}\mu(v_j^{i_b})\Big], \tag{4.16}$$

such that

$$\beta(e_j^i) \in C(\mathbf{k}, \mathbf{u}) \text{ if } \check{\mathbf{e}}_j^k \in C(\mathbf{k}, \mathbf{u}), \tag{4.17}$$

and the geometry samples for the wavelet coefficients $\omega(v_{j+1}^{o_i})$ are embedded at

$$\check{\mathbf{v}}_{j+1}^{o_i} = \mu(\widetilde{v}_{j+1}^{o_i}), \tag{4.18}$$

such that

$$\omega(v_{j+1}^{o_i}) \in C(\mathbf{k}, \mathbf{u}), \text{ if } \check{\mathbf{v}}_{j+1}^{o_i} \in C(\mathbf{k}, \mathbf{u}). \tag{4.19}$$

In Chapter 3 the template mesh was used for embedding both the connectivity and geometry samples in $\mathbb{R}^3$ as described above. In Section 4.4.1, the same template mesh is additionally used to map the wavelet coefficient supports $S(w)$ from $M_j^T$ where they are ensured to be accurately represented, to $M_{\alpha \leq j}$ where these supports are not necessarily fully reconstructed. The next sections discuss approaches to avoid decoding an *entire* subband before performing an adaptive inverse transform. Section 4.4.2.1 discusses tiling of the geometry samples $\check{\mathbf{v}}_{j+1}^{o_i}$; Section 4.4.2.2 goes on by discussing how the connectivity samples $\check{\mathbf{e}}_j^k$ can be tiled.

#### 4.4.2.1 Tiled Geometry Information

For an ROI-based reconstruction of $M_{\alpha \leq j+1}$, the decoding of the geometric samples can be limited to $w \in ROI^W_{k|j}$ for each $k \leq j$. A partial, ROI-based decoding of these geometric samples does not hinder subsequent decoding steps as each $M^T_j$, which embeds the samples, is reconstructed using only connectivity information.

At each resolution, $C_j$ is fully decoded. To allow for random access into the geometry data $G_j$, the geometry samples can be partitioned into tiles based on any criterion that can be mirrored by a decoder; for instance, based on the topology of $M^T_{j+1}$ or based on the sampling locations $\tilde{\mathbf{v}}^{o_i}_{j+1}$ within $M^T_{j+1}$.

Each odd vertex corresponds to a single geometry sample, and can consequently be mapped to a single tile after partitioning. Denote the mapping of odd vertices $v^o_j$ to tile $T^{\mathbf{x}}_j$ as $T(v^o_j) = T^{\mathbf{x}}_j$. The set of tiles required for $ROI^W_j$ is:

$$\mathcal{T}^{\text{req}}_j = \bigcup_{w \in ROI^W_j} T\big(\omega^{-1}(w)\big). \tag{4.20}$$

Except for signaling the tiles within a bit stream, the only lossless rate penalty is caused by the trade-off between fine-granular random access (requiring smaller tiles) and high coding performance (requiring larger tiles). As the portion of geometry information vastly surpasses the connectivity information (see, for instance, [2, 5, 35]), large speed-ups and rate savings for ROI decoding can be obtained. However, using a template mesh $M^T$ having the same number of vertices as the original mesh has memory limitations: for instance, decoding only a fraction of a multi-million-vertex model $M$ still requires a multi-million vertex $M^T$ in memory. One can solve this problem by tiling also the connectivity information, as proposed next.

#### 4.4.2.2 Tiled Connectivity Information

Tiling geometry samples is possible in a straightforward manner because the entire template mesh $M^T_j$ at each resolution $j$ is available for embedding the samples before (partial) decoding, *and* each wavelet coefficient corresponds to a single encoded sample. Connectivity information cannot be handled similarly as these two assumptions are no longer true.

Firstly, partial decoding of the connectivity information results in only partly upsampling $M^T_j$ to $M^T_{\alpha \leq j+1}$, breaking the symmetry with the encoder side. Denote the tiles obtained after tiling using any criterion as $\widetilde{T}^{\mathbf{x}}_j$. Contrary to the tiling of geometry samples in Section 4.4.2.1 where $M^T_j$ can be used *entirely* by both the encoder and decoder to create identical tiles, the tiling of connectivity samples which are only partly decoded can no longer depend on all template mesh information.

To tackle this, one approach is to tile the samples only once, as is done in literature and which results in *fixed tiles*. While this tiling operation is often performed on the base mesh, it can in general be performed at any resolution $j$. As the samples are not tiled at resolutions $k < j$, $M_j^T$ will be entirely reconstructed by the decoder and the tiling operation can still use all template mesh information. Alternatively, a *dynamic tiling* approach is suggested in this dissertation. By allowing the amount of tiles to change per resolution level, the tiling can be adapted to the global average sampling density. Additionally, by allowing for non-uniform tiling, the tiling can be adapted to local sampling densities. In this approach, only the first tiling operation can use *all* template mesh information. Subsequent tiling operations can only consider information *local* to each tile in order to preserve the symmetry with the encoder. Consequently, either additional signaling in the data stream allows for uniform tiling, or decoder-side tiling decisions result in non-uniform tiling as only local information can be considered.

Secondly, whereas the geometry is decoded given a *single* sample $\omega(v_j^{o_i})$ per wavelet coefficient, properly upsampling the connectivity requires multiple samples $\beta(e_j^k)$ to define the wavelet coefficient supports. As these supports are only known to a decoder *after* decoding connectivity samples, a tiling of the connectivity samples *before* decoding them necessarily needs to duplicate vertices of neighboring tiles to avoid patches being encoded only partially within a tile.

Two approaches are discussed for extending the obtained tiles $\widetilde{T}_j^{\mathbf{x}}$ to account for patches across tile borders:

- extend tile samples with the minimal amount of samples such that each patch is fully represented,
- or extend tile samples such that all subsequent resolutions are decodable given the current tile.

The following two sections discuss these two approaches for extending the tiles $\widetilde{T}_j^{\mathbf{x}}$ to form the tiles $T_j^{\mathbf{x}}$ which are encoded.

Due to only partially reconstructing $M_{\alpha \leq j}^T$, the mapping of vertices to tiles $T(v_j) = T_j^{\mathbf{x}}$ is no longer straightforward. The vertices at resolutions where (fixed or dynamic) tiling is performed, are trivially mapped. However, for vertices $v^o$ reconstructed at *higher* resolutions, $S(\omega(v^o))$ possibly lies across a tile border and is consequently duplicated across multiple tiles. In the implementation, higher-resolution even vertices $v_j^e$ and odd vertices $v_j^o$ are mapped as follows:

$$T(v_j^e) = T\big(\gamma(v_j^e)\big), \tag{4.21}$$

$$T(v_j^o) = \bigcup_{v \in S(\omega(v_j^o))} T(v). \tag{4.22}$$

That is, reconstructed vertices inherit the (possibly different) tiles of the vertices in their support. The set of required tiles $\mathcal{T}_j^{\text{req}}$ per resolution $j$ can then be determined

via:

$$\mathcal{T}_j^{\text{req}} = \bigcup_{v \in ROI_j^S} T(v), \tag{4.23}$$

where a single tile of each $T(v)$ suffices.

Contrary to Equation 4.20 the required tiles can no longer be determined directly in the wavelet-domain $ROI_j^W$. As connectivity information is only partially decoded, not all $w_c \in C_j$, needed for determining $S(w)$ using Equation 3.3, are known. Hence, $ROI_j^W$ cannot be determined using Equation 4.1. The extended tiles corresponding to the vertices in the *spatial-domain* $ROI_j^S$ are decoded. Consequently, the extension of the tiles must ensure that sufficient samples are being decoded such that $ROI_j^W$ can accurately be obtained for reconstructing $M_{\alpha \leq j+1}$.

**Minimal Connectivity Information per Tile** After partitioning, the vertices in patches which lie across tile borders are scattered over multiple tiles. Hence, decoding only a specific tile $\widetilde{T}_j^{\mathbf{x}}$ can result in inaccurate connectivity near the tile borders, which in turn results in drift when decoding geometry samples for tile $\widetilde{T}_j^{\mathbf{x}}$. The most straightforward approach encodes the tiles $T_j^{\mathbf{x}}$ which extend $\widetilde{T}_j^{\mathbf{x}}$ as follows:

$$\begin{aligned} &\widetilde{T}_j^{\mathbf{x}} \subset T_j^{\mathbf{x}} \text{ and} \\ &\forall w_j \in W_j : S(w_j) \cap \widetilde{T}_j^{\mathbf{x}} \neq \varnothing \implies S(w_j) \subset T_j^{\mathbf{x}}. \end{aligned} \tag{4.24}$$

That is, if one vertex of the wavelet coefficient support $S(w_j)$ was partitioned into $\widetilde{T}_j^{\mathbf{x}}$, then the entire support needs to be encoded in $T_j^{\mathbf{x}}$.

Observe that this tiling cannot be mirrored by a decoder: $\widetilde{T}_j^{\mathbf{x}}$ can be determined in the same way as done at the encoder side, but without knowledge of $W_j$, $T_j^{\mathbf{x}}$ cannot be found. The proposed approach encodes additional vertex rings per tile $\widetilde{T}_j^{\mathbf{x}}$ until $T_j^{\mathbf{x}}$ is entirely taken into account; the number of additional vertex rings is a parameter which needs to be encoded in the bit stream. Each encoded sample which is not found in $T_j^{\mathbf{x}}$ is encoded as a null-value to ensure no patches are determined outside of $T_j^{\mathbf{x}}$. The actual values for these samples are irrelevant for decoding $\widetilde{T}_j^{\mathbf{x}}$; if a decoder needs these values, the appropriate tile will be decoded.

The construction of the tiles $T_j^{\mathbf{x}}$ considering wavelet coefficient supports, as given in Equation 4.24, ensures that decoding the tiles given in Equation 4.23 provides the necessary wavelet coefficients of $W_j$ for accurately determining $ROI_j^W$ defined in Equation 4.1, which are, in turn, required for reconstructing $M_{\alpha \leq j+1}$. However, as the encoder only takes into account $W_j$ without considering higher resolutions, there is no guarantee that a subsequent decoding step of a higher resolution will be possible without drift.

*Figure 4.7: Tiling with minimal duplication. In (a), an adaptive inverse wavelet transform requires the data of subband* 0*, limited to tile* $\widetilde{T}^{x,y}$*. The black wave line indicates data outside of this tile required to properly reconstruct the template mesh* $M^T_{\alpha\leq 1}$ *which perfectly reconstructs* $M^T_1$ within tile $\widetilde{T}^{x,y}$*. As* $M^T_0$ *is at resolution* 0 *over all tiles, all required data is available. In (b), data of subband* 1 *is required to reconstruct* $M^T_{\alpha\leq 2}$*, again limited to tile* $\widetilde{T}^{x,y}$*. Additional data outside of this tile (indicated by the red wave line) is no longer guaranteed to be available, as* $M^T_{\alpha\leq 1}$ *is not necessarily at resolution* 1 *in the tiles neighboring* $\widetilde{T}^{x,y}$*. To ensure that all required data is available, the neighboring tiles need to be at resolution* 1 *before reconstructing* $M^T_{\alpha\leq 2}$ *(c). In general, if data of subband* $j$ *is required in a tile* $\widetilde{T}^{x,y}$*, tiles minimally need to be decoded as depicted in (d).*

Figure 4.7 illustrates this effect. Figure 4.7a shows all tiles of $M_0^T$. The wave line indicates the samples added to $\widetilde{T}_0^{x,y}$ to obtain $T_0^{x,y}$. For the next upsampling step, such additional samples are not necessarily available in Figure 4.7b as the neighboring tiles $T_0^{x\pm1,y}$, $T_0^{x,y\pm1}$ and $T_0^{x\pm1,y\pm1}$ are not decoded, nor were these additional samples taken into account when constructing $T_0^{x,y}$ using Equation 4.24. Consequently, these neighboring tiles need to be decoded before $T_1^{x,y}$ can be decoded (Figure 4.7c). In general, if an $ROI_j^S$ requires $\widetilde{T}_j^{x,y}$, the minimal resolutions are shown in Figure 4.7d: neighboring tiles can differ by, at most, one resolution level.

This tiling approach has the disadvantage that much of the decoding effort is spent to support neighboring tiles instead of the highest resolution *within* the tile. Furthermore, while dynamic tiling is possible, ensuring the appropriate resolutions per tile becomes even more involved if tiles are non-uniformly distributed. To solve these issues, an alternative tiling methodology is presented next.

**Sufficient Connectivity Information for Independent Tiles** Alternatively, given tiles $\widetilde{T}_j^{\mathbf{x}}$, a tiling $T_j^{\mathbf{x}}$ is proposed which takes into account *all* higher resolutions. In a first step, traverse all higher resolutions to find wavelet coefficients affected by vertices in $\widetilde{T}_j$, defining temporary tiles $\widehat{T}_k^{\mathbf{x}}$ for $k \geq j$. Let $\widehat{T}_j^{\mathbf{x}} = \widetilde{T}_j^{\mathbf{x}}$; $\widehat{T}_k^{\mathbf{x}}$ is recursively determined as follows:

$$\begin{aligned} &\forall v_k^e \in M_k^e : \gamma(v_k^e) \in \widehat{T}_{k-1}^{\mathbf{x}} \implies v_k^e \in \widehat{T}_k^{\mathbf{x}}, \\ &\forall v_k^o \in M_k^o : S(\omega(v_k^o)) \cap \widehat{T}_{k-1}^{\mathbf{x}} \neq \varnothing \implies v_k^o \in \widehat{T}_k^{\mathbf{x}}, \end{aligned} \tag{4.25}$$

for each $k \in [j+1, R]$, with $R$ the highest resolution. $\widehat{T}_R^{\mathbf{x}}$ encompasses all highest-resolution vertices which are, directly or indirectly, affected by the vertices in $\widetilde{T}_j^{\mathbf{x}}$. To ensure that these highest-resolution vertices are accurately reconstructed, traverse back to resolution $j$ to ensure that all wavelet coefficient supports are taken into account. Define temporary tiles $\breve{T}_k^{\mathbf{x}}$ for the traversal back to resolution $j$. Let $\breve{T}_R^{\mathbf{x}} = \widehat{T}_R^{\mathbf{x}}$; we have:

$$\begin{aligned} &\forall v_{k+1}^e \in (\breve{T}_{k+1}^{\mathbf{x}} \cap M_{k+1}^e) : \gamma(v_{k+1}^e) \in \breve{T}_k^{\mathbf{x}}, \\ &\forall v_{k+1}^o \in (\breve{T}_{k+1}^{\mathbf{x}} \cap M_{k+1}^o) : S(\omega(v_{k+1}^o)) \in \breve{T}_k^{\mathbf{x}}. \end{aligned} \tag{4.26}$$

Using Equation 4.26 eventually gives $\breve{T}_j^{\mathbf{x}}$. The minimally required tile data which needs to be encoded is $T_j^{\mathbf{x}} = \breve{T}_j^{\mathbf{x}}$. This ensures that, given an initial $\widetilde{T}_j$, sufficient additional samples are provided to decode this resolution $j$ as was described for the previous approach, *and* to decode future resolutions $j + k$ as long as $ROI_{j+k}^S$ is composed of vertices which are (directly or indirectly) affected by vertices in the original $\widetilde{T}_j^{\mathbf{x}}$. Additionally, as tiles can now be treated independently, tiling can be easily adapted to the decoded sample densities per resolution.

Similar to the approach for tiling with minimal connectivity information above, the decoder has no knowledge about $W_k$ with $k \geq j$ and cannot reconstruct $T_j^{\mathbf{x}}$. Consider again the vertex rings around $\widetilde{T}_j^{\mathbf{x}}$ which need to be added until $T_j^{\mathbf{x}}$ is entirely taken into account. This number of vertex rings is communicated to ensure that the same tiles are used at the encoder and decoder sides. Finally, observe that the same $T_j^{\mathbf{x}}$ is obtained as in Equation 4.24 if only looking one resolution higher, i.e., considering $R = j + 1$ and thus only apply Equations 4.25 and 4.26 once.

This approach lends itself perfectly to dynamic tiling, producing tiles of various sizes, adapted on the local density of the tessellation. In the implementation, the partitioning is based on the recursive octree decomposition of the bounding box of template mesh vertices. Let $\tau^{\text{TS}}$ be the given tile-split threshold which controls the amount of vertices within any given tile. The dynamic tiling algorithm initially considers all vertices to be contained within a single tile $\widetilde{T}_0^0 = T_0^0$. The tile partitioning approach considers the axis-aligned bounding box of all vertices within a tile, i.e., the box containing the points $(x, y, z)$ with $x \in [x_{\text{min}}, x_{\text{max}}], y \in [y_{\text{min}}, y_{\text{max}}], z \in [z_{\text{min}}, z_{\text{max}}]$ and

$$x_{\text{min}} = \min_{v \in T_j^0} v_0 \text{ and } x_{\text{max}} = \max_{v \in T_j^0} v_0, \tag{4.27}$$

$$y_{\text{min}} = \min_{v \in T_j^0} v_1 \text{ and } y_{\text{max}} = \max_{v \in T_j^0} v_1, \tag{4.28}$$

$$z_{\text{min}} = \min_{v \in T_j^0} v_2 \text{ and } z_{\text{max}} = \max_{v \in T_j^0} v_2. \tag{4.29}$$

In these equations $v_i$ indicates the $i^{\text{th}}$ component of the position of $v$.

The vertices are partitioned by considering eight octants around the bounding box center $\mathbf{v}_C$. The index of each of the octants is given by a triple $\mathbf{x} = (x_0, x_1, x_2)$ with $x_i \in \{0, 1\}$, where $x_i = 0$ indicates that the $i^{\text{th}}$ component of each vertex position in the octant is smaller than $\mathbf{v}_{C,i}$, i.e., the $i^{\text{th}}$ component of $\mathbf{v}_C$. Conversely, $x_i = 1$ indicates that the vertices have an embedded position with the $i^{\text{th}}$ component larger than $\mathbf{v}_{C,i}$. This condition can be represented compactly:

$$\forall v \in \widetilde{T}_j^{x_0,x_1,x_2} : (-1)^{x_i}(\mathbf{v}_{C,i} - v_i) > 0, i \in [0, 2]. \tag{4.30}$$

The dynamic tile decoding algorithm is described in Example 4.1. $\mathcal{T}$ represents the set of decodable tiles. At each resolution, $ROI_j^S$ determines the required tiles $\mathcal{T}_j^{\text{req}}$ using Equation 4.23. Each required tile that still needs to be decoded, denoted by $T_j$ in Listing 4.1, is removed from the decodable tiles (line 4), is decoded and upsampled (line 5), and either the upsampled tile $T_{j+1}$ is added to the decodable tiles as such (line 7) or $T_{j+1}$ is first split and each of the subtiles are added to the decodable tiles (line 13).

$\mathcal{T} \leftarrow \{T_0^0\}$
**for all** $j \in [0, R-1]$ **do**
  **for all** $T_j \in \mathcal{T} : T_j \in \mathcal{T}_j^{\text{req}}$ **do**
    $\mathcal{T} \leftarrow \mathcal{T} \setminus \{T_j\}$
    $T_{j+1} = \text{DECODEANDUPSAMPLE}(T_j)$
    **if** $|T_j| < \tau^{\text{TS}}$ **then** ▷ keep $T_{j+1}$
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_{j+1}\}$
    **else** ▷ split $T_{j+1}$
      $\mathbf{v}_C = (\frac{x_{\min}+x_{\max}}{2}, \frac{y_{\min}+y_{\max}}{2}, \frac{z_{\min}+z_{\max}}{2})$
      **for all** $\mathbf{x} \in \{0,1\}^3$ **do**
        $\widetilde{T}_{j+1}^{\mathbf{x}} = \{v \in T_{j+1} : (-1)^{x_i}(\mathbf{v}_{C,i} - v_i) > 0\}$
        $T_{j+1}^{\mathbf{x}}$: obtained using Equations 4.25 and 4.26
        $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_{j+1}^{\mathbf{x}}\}$
      **end for**
    **end if**
  **end for**
**end for**

*Algorithm 4.1: Dynamic tile decoding*

An example illustrating an encoding and decoding of the proposed dynamic tiling approach is shown in Figure 4.8, where tiles are split binary; tile indices $\mathbf{x}$ are now scalar indices which stay constant between the resolution at which the tile is created and the resolution where the tile is further split. A base mesh is encoded using a single tile $T^0$. Tile $T_0^0$ encodes the first subband $W_0$, i.e., $G_0$ and $C_0$. Tiles $T_1^0$ and $T_2^0$ encode the next two subbands. However, the amount of samples after upsampling $T_2^0$ surpasses $\tau^{\text{TS}}$ so instead of encoding a single tile $T_3^0$, the tile is split and $T_3^1$ and $T_3^2$ are encoded instead (see Figure 4.8). Information is duplicated, such that each subtree is now processed independently. Because tiles are processed independently they need not be split at the same resolution, allowing for the tile splitting to adapt to the sampling densities within the model.

The decoding process follows the same steps. The given $ROI_j^S$ per resolution $j$ determines the tiles $\mathcal{T}_j^{\text{req}}$ which need to be decoded, while no dependencies occur between neighboring tiles. For instance, for a given ROI, tile $T^8$ can be decoded up to resolution 7 while the neighboring tile $T^2$ can remain at resolution 3, without any blocking artifacts showing up due to the construction of the tiles. At each step of the decoding process, the state is represented by a *tile front* which is formed as illustrated by the dashed line in Figure 4.8, and corresponds with $\mathcal{T}$ in Listing 4.1. In this example, $\mathcal{T} = \{T_5^3, T_6^7, T_7^8, T_3^2\}$ represents the four tiles and their respective resolutions for decoding a specific mesh $M_{\alpha \leq 7}$. $T_3^2$ spans approximately half of the bounding box, $T_5^3$ spans a quarter and $T_6^7$ and $T_7^8$ span $1/8$, showing that the decoded tiles appropriately adapt to the sampling densities required for the requested ROIs.

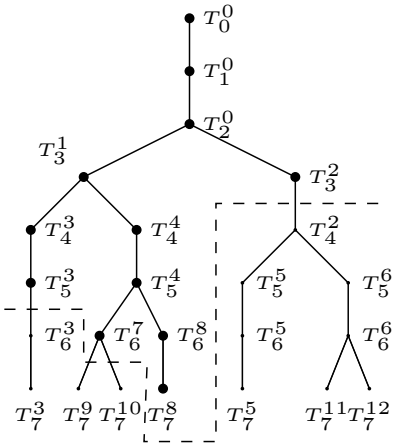



*Figure 4.8: Dynamic tiling: tile hierarchy. This figure visualizes the tiles hierarchically. The dashed line represents a possible front of tiles which are decoded at a particular time.*

Contrary to the approach for tiling with minimal connectivity information, tiles are only decoded in order to supply wavelet coefficients for the mesh reconstruction within the tile and not as support for decoding neighboring tiles located in the ROI. That is, given an $ROI_j^S$, only the tiles given by Equation 4.23 need to be decoded, as sufficient samples are provided for decoding up to resolution $j$ without drift. Compared to regular tiling when using minimal connectivity information, this reduces the amount of tiles being decoded for a given resolution level and improves random accessibility. This is illustrated in Figure 4.9 which compares the proposed dynamic tiling method (Figure 4.9b) with the regular tiling when using minimal connectivity information (Figure 4.9a). One notices that providing any given resolution level requires less tiles when following dynamic tiling compared to the regular tiling when using minimal connectivity information.

Although the proposed method with independent tiles (Figure 4.9b) improves random accessibility compared to the method with minimally-sized tiles (Figure 4.9a), it also introduces an additional rate penalty. On the one hand, additional samples are taken into account further outside of the tile borders; this is illustrated by the overshoots of the dashed lines over the tile marks (indicated by the vertical lines) in Figure 4.9b. If tiles are not decoded until the highest resolution, some of these additional samples are irrelevant. On the other hand, information will be duplicated over several neighboring tiles; this is indicated by the gray areas in Figure 4.9b which illustrate data near the tile borders that has been decoded twice.

Nonetheless, while the lossless coding rate increases due to duplicate information being encoded, the required rate when decoding specific ROIs is vastly

lowered due to the tile resolutions scaling down as fast as the inverse wavelet transform does. An example with 8 tiles is shown in Figure 4.10. For each tile, the template mesh which is depicted is used to embed samples in its own octree. It is clear from the figure that the connectivity information can be duplicated over several tiles, as shown for instance by the connectivity information for the left nostril of the dragon. Each of these tiles can be processed individually, hence each tile can also be split further depending on the sampling density.

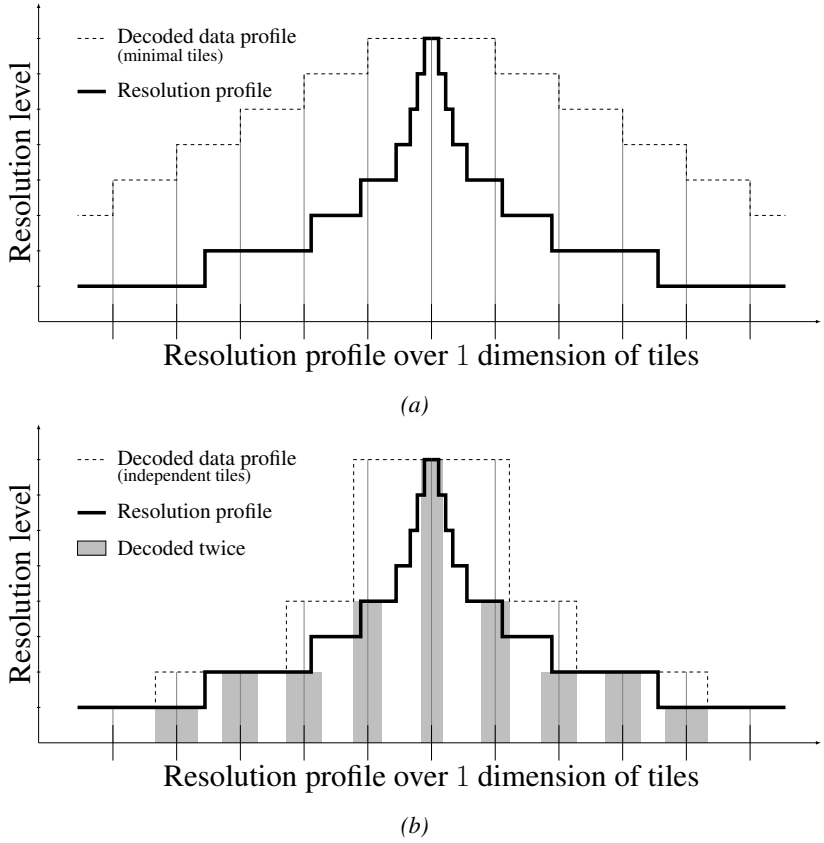



*Figure 4.9: Tile granularity. These two plots qualitatively compare tiling with minimal connectivity information and tiling with sufficient information for independent tiles respectively. Ideally, the dashed line which represents the* decoded data *(at tile granularity) perfectly follows the full line which represents the amount of* data used *for the wavelet transform (at triangle granularity). The gray areas indicate information which is duplicated in order to provide random accessibility while avoiding tile dependencies.*

*Figure 4.10: Independent tiles for model* dragon*. The two front top tiles clearly show that overlapping information is present within the tiles: the left nostril of the dragon is encoded in both tiles, as the reconstruction of higher resolution vertices near the tile border depends on all these vertices around this nostril.*

## 4.5 Local RD Optimization

With this tiling available, RD optimization of the coding system can now aim at optimally allocating rate *across different tiles*. RD optimization for *untiled* wavelet-based irregular mesh encoding was investigated in Section 3.4. This approach is now generalized for dynamic tiling, as detailed next.

Without tiling, at each moment during encoding, either a new resolution is decoded up to a specific number of bit planes, or the quality of an existing resolution is improved by encoding an additional number of bit planes. This is formalized as follows. Consider all wavelet coefficients being represented using $Q$ bit planes, and the connectivity information using an additional data layer. The data layers are labeled decreasingly as $Q$ for the connectivity information, and $Q-1, \ldots, 0$ for the geometry information from most significant to least significant bit planes. This allows for constructing RD curves per resolution, where each curve has $Q+2$ rate points, one for each encoded data layer (including the case when no information is sent for the given resolution). Each rate point is denoted by $(R_j^{(p)}, D_j^{(p)})$, with $R_j^{(p)}$ the bit rate required to obtain the rate point with label $p$ of resolution $j$, and $D_j^{(p)}$ the accompanying distortion.

Let $P_j$ be the last encoded layer of resolution $j$, and let $L$ be the first unencoded resolution (hence, $P_L = Q+1$, the rate point *before* encoding any connectivity information); in general, when performing RDO, we want to encode $k'$ layers of

resolution $j'$ by maximizing the distortion-rate slopes given by:

$$(j', k') = \underset{j \in [0,L], k \in [1,P_j]}{\arg\max} \frac{D_j^{(P_j)} - D_j^{(P_j - k)}}{R_j^{(P_j - k)} - R_j^{(P_j)}}. \tag{4.31}$$

Per resolution $j$, the amount of data layers $k \in [1, P_j]$ which yields the largest slope is determined. These data layers are encoded for the resolution for which this largest slope is maximal.

With the availability of tiles, RDO can be further improved as the tiles can be present at different resolutions and quality levels. Given that each resolution $j$ counts $T_j$ tiles, the RD curves can now be found per resolution *and* per tile. Each such curve has again $Q + 2$ rate points, denoted by $(R_{(j,t)}^{(p)}, D_{(j,t)}^{(p)})$. $P_{j,t}$ now denotes the last encoded layer *of tile* $t$ in resolution $j$, and $R_{(j,t)}^{(p)}$ and $D_{(j,t)}^{(p)}$ respectively give the bit rate and distortion after encoding layer $p$ *of tile* $t$ at resolution $j$. We now want to encode $k'$ layers of tile $t'$ of the $j'^{\text{th}}$ resolution by determining:

$$(j', t', k') = \underset{\substack{j \in [0,L] \\ t \in [0, T_j - 1] \\ k \in [1, P_{j,t}]}}{\arg\max} \frac{D_{(j,t)}^{(P_{j,t})} - D_{(j,t)}^{(P_{j,t} - k)}}{R_{(j,t)}^{(P_{j,t} - k)} - R_{(j,t)}^{(P_{j,t})}}. \tag{4.32}$$

In these equations the following conventions are made, similar to the conventions in Section 3.4:

$$R_{(j,t)}^{(Q+1)} = 0, \tag{4.33}$$

$$R_{(j,t)}^{\text{conn}} = R_{(j,t)}^{(Q)} - R_{(j,t)}^{(Q+1)} = R_{(j,t)}^{(Q)}, \tag{4.34}$$

$$R_{(j,t)}^{\text{geom}} = R_{(j,t)}^{(0)} - R_{(j,t)}^{(Q)}, \tag{4.35}$$

$$D_{(j,t)}^{(Q+1)} = D_{(j)}^{(Q)}. \tag{4.36}$$

These conventions signify that the rate for each tile at each resolution starts at $0$ as given by Equation 4.33; the rate for each tile $t$ of each resolution $j$ can be attributed to connectivity information by the first data layer (Equation 4.34) and geometry information in the remaining layers (Equation 4.35). Distortion decrease by decoding the connectivity information (Equation 4.36) is not considered.

## 4.6 Evaluation

### 4.6.1 Encoder-Side Evaluation

ROI-aware encoding is compared with the ROI-agnostic encoding of Chapter 3 using a set of 25 conventional models ranging from $1\,000$ up to $350\,000$ vertices.

The wavelet coefficients were quantized using 12 bits, and a scaling value was determined per wavelet subband as illustrated in Figure 4.3. The ROI was defined based on the surface orientation: regions where the surface normal has a component in a specific direction are part of the ROI, i.e., the *front-facing regions*; the other regions are part of the BG, i.e., the *back-facing regions*.

| Model | rate | p | s | g | vert% |
|---|---|---|---|---|---|
| teapot (1 292) | 32.7 | 4.05 | 6.71 | 2.66 | 74% |
| drill bit (1 961) | 33.3 | 3.29 | 7.01 | 3.72 | 77% |
| beethoven (2 521) | 34.1 | 3.13 | 5.06 | 1.93 | 83% |
| triceratops (2 832) | 32.4 | 2.73 | 4.15 | 1.42 | 87% |
| elk (5 194) | 30.5 | 3.03 | 5.97 | 2.94 | 76% |
| parthenon (5 936) | 27.2 | 2.73 | 5.09 | 2.36 | 74% |
| atomium (6 150) | 26.8 | 2.42 | 4.24 | 1.82 | 80% |
| fandisk (6 475) | 26.2 | 2.71 | 6.43 | 3.72 | 63% |
| maxplanck (7 399) | 29.6 | 2.87 | 7.89 | 5.02 | 66% |
| venushead (8 268) | 29.6 | 3.03 | 6.86 | 3.84 | 70% |
| bimba (8 857) | 30.1 | 2.67 | 5.84 | 3.16 | 75% |
| horse (19 851) | 25.1 | 2.59 | 6.23 | 3.64 | 65% |
| bunny (34 834) | 24.2 | 2.26 | 5.61 | 3.35 | 67% |
| vaselion (38 728) | 27.4 | 2.27 | 5.26 | 2.99 | 75% |
| screwdriver (65 538) | 20.6 | 2.15 | 5.44 | 3.29 | 56% |
| rabbit (67 039) | 22.6 | 1.60 | 6.07 | 4.47 | 60% |
| golfball (122 882) | 21.8 | 1.24 | 6.26 | 5.02 | 56% |
| dino (129 026) | 19.9 | 1.90 | 4.01 | 2.12 | 66% |
| headus (131 074) | 19.8 | 1.44 | 4.53 | 3.09 | 58% |
| armadillo (172 974) | 20.8 | 1.73 | 5.08 | 3.35 | 62% |
| igea (198 658) | 19.2 | 1.50 | 5.18 | 3.67 | 52% |
| fertility (241 607) | 21.3 | 1.90 | 6.12 | 4.23 | 57% |
| feline (258 046) | 18.7 | 1.85 | 4.27 | 2.43 | 54% |
| heptoroid (286 678) | 20.1 | 2.22 | 5.30 | 3.08 | 52% |
| skeleton hand (327 323) | 19.8 | 2.09 | 5.44 | 3.35 | 50% |
| **Average** | **25.4** | **2.376** | **5.602** | **3.226** | **66%** |

*Table 4.2: Rate penalty and savings. The columns give, respectively, the models with their amount of vertices, the original rate, the rate penalty $p$ in lossless coding in bpv, the saved rate $s$ in bpv when viewing from the front, the final gain $g$ in bpv, and the percentage of vertices used for this viewing angle.*

#### 4.6.1.1 Penalty in Lossless Coding

The shifting method introduces a penalty for lossless coding due to the additional bit planes which need to be encoded.

The second column in Table 4.2 gives the lossless rate for resolution-scalable coding in bpv. On average, a coding rate of $25.4$ bpv, a rate which decreases with increasing number of vertices, is observed. The rate penalty introduced by allowing ROI-aware decoding is given in the third column. An average rate penalty of $2.38$ bpv is observed.

#### 4.6.1.2 Rate Decrease for ROI Decoding

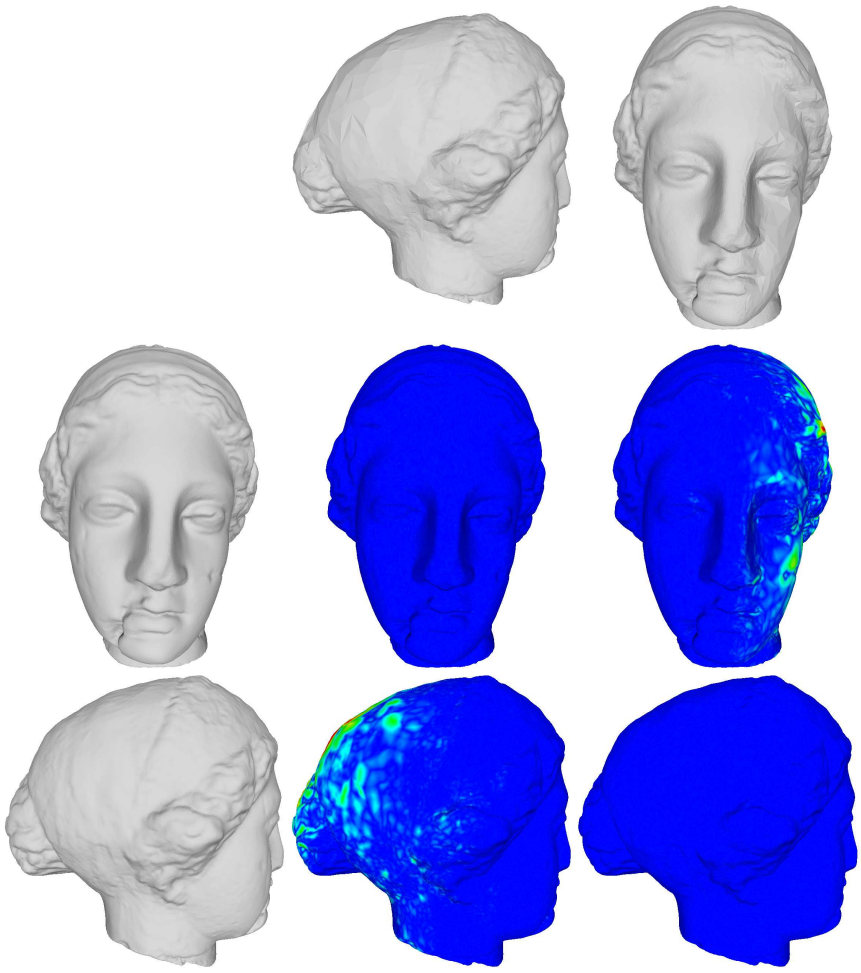
When viewing the models from the front, the described front-back ROI suffices to attain *visually* lossless results when only decoding the ROI. Table 4.2 shows an average rate saving of $5.60$ bpv in this case. Taking into account the rate penalty, gains up to $5$ bpv are observed; on average $3.23$ bpv was saved. Furthermore, the ROI-aware inverse wavelet transform reduced the amount of vertices for such visually lossless front-view rendering by $34\%$. Observe also that the ROI-aware inverse transform performs better on higher-resolution data: with approximately half of the vertices in the ROI, the decoded data saves up to $50\%$ with increasing model size.

A decreased quality can only be observed when viewing from an angle. Figure 4.11 shows the *igea* model, with Figure 4.11b showing the visually lossless reconstruction of the front-facing regions, while Figure 4.11c shows the visually lossless reconstruction of side-facing regions. The top line depicts visual results, the second line depicts the distortions seen from the front, and the final line depicts distortions visible from the side. In Figure 4.12 the wireframe renderings of both predefined ROI examples are depicted.

#### 4.6.1.3 ROI Coding with Visual Loss

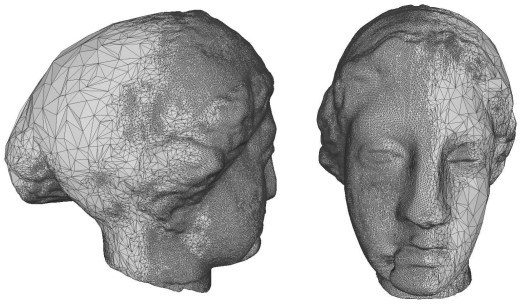
Finally, the restriction of requiring all regions facing a specific direction is softened. Denote the angle formed between a triangle normal and the desired direction as $\theta$, then the ROI determined before was given by $\theta \leq 90°$. After all, these are the regions of a surface which are physically visible from a given direction, and in an efficient real-time rendering engine only these triangles are rendered. Consider now ROIs determined by $\theta \leq \theta_{\text{lim}} \leq 90$. Table 4.3 shows the increased rate gains and the reduced amount of vertices used, for $\theta_{\text{lim}} \in \{90°, 70°, 45°\}$. While the rate penalty is not significantly influenced by altering the amount of wavelet coefficients which are encoded in the ROI, the rate savings, and consequently the final gains, are significant. Similarly, the amount of required vertices is significantly reduced. Consequently, a careful selection of the ROIs while ensuring adequate quality can significantly reduce the final bit rates when only decoding the ROI. Next, this subjective visual quality is investigated.

Figure 4.13 clearly demonstrates the adaptive inverse wavelet transform, where the front-facing regions have a high resolution, while back-facing regions are smoothed out by providing fewer vertices. Figure 4.14 then demonstrates that lower resolutions result in smoother surfaces as seen on the top row, where details in the diadem are largely lost in Figure 4.14c. The errors are clearly increasing as shown in the middle row. However, geometric errors in the visible areas are nearly unobservable despite a lossy ROI selection. While no *objective* conclusions can be made, one can visually claim, specifically for the *igea* model and the selected front



*(a) Original* *(b) front ROI* *(c) side ROI*

*Figure 4.11: Lossless predefined ROIs for model* igea*: visual results. Blue colors represent accurate geometry reconstructions while red colors represent the largest distortions. In (a) the original* igea *model is shown from the front and from the side. (b) shows the example of a front-side ROI, while (c) shows an ROI on the side. No visual distortions can be observed when viewing the front ROI from the front or when viewing the side ROI from the side.*



*(a) front ROI* *(b) side ROI*

*Figure 4.12: Lossless predefined ROIs for model* igea*: wireframe results. Observe that the sampling density varies while preserving a valid topology.*

| **igea (19.2 bpv)** | **p** | **s** | **g** | **vert%** |
|---|---|---|---|---|
| $\theta_{\text{lim}} = 90°$ | 2.08 | 5.30 | 3.22 | 52 |
| $\theta_{\text{lim}} = 70°$ | 2.07 | 6.91 | 4.84 | 39 |
| $\theta_{\text{lim}} = 45°$ | 1.90 | 8.07 | 6.17 | 29 |

*Table 4.3: Lossy rate penalty and savings for* igea*. The columns give, respectively, the maximal angle with the viewing direction, the rate penalty $p$ in lossless coding in bpv, the saved rate $s$ in bpv when viewing from the front, the final gain $g$ in bpv, and the percentage of vertices used for this viewing angle.*

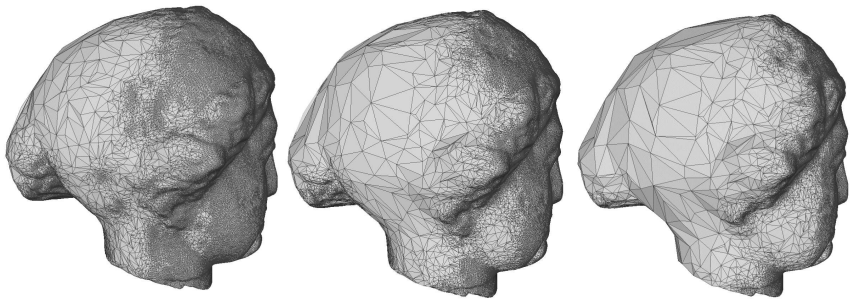
side, that the ROI with $\theta_{\text{lim}} = 45°$ does not reduce the quality when looking from the front, while significant additional coding gains and vertex savings are obtained.

### 4.6.2 Decoder-Side Evaluation

For the evaluation of the proposed methods, experiments have been performed with models up to the *Asian Dragon* model of $3\,609\,600$ vertices. In the literature, no proper evaluation criteria have been proposed for comparing different approaches w.r.t. the quality of ROI decoding or the accuracy offered by random accessibility. Comparative studies are usually limited to comparing the *lossless rates*, and *visual results* without rate indications. These visual results are obtained for instance using a click-and-drag approach in a virtual environment.

This section provides both lossless rates and visual results, but these are also complemented by experimental results for two ROI decoding scenarios: *front-view ROI* and *point-based ROI*.

The *front-view ROI* selects all "front-facing" triangles. Let $\mathbf{n}$ be the surface normal of a triangle, and $\mathbf{v}$ the direction to the camera. As depicted in Figure 4.15a, a triangle is front-facing if $\mathbf{n}$ and $\mathbf{v}$ form an angle smaller than $90°$, signifying that the front of the triangle is visible when looking from the direction of the camera.



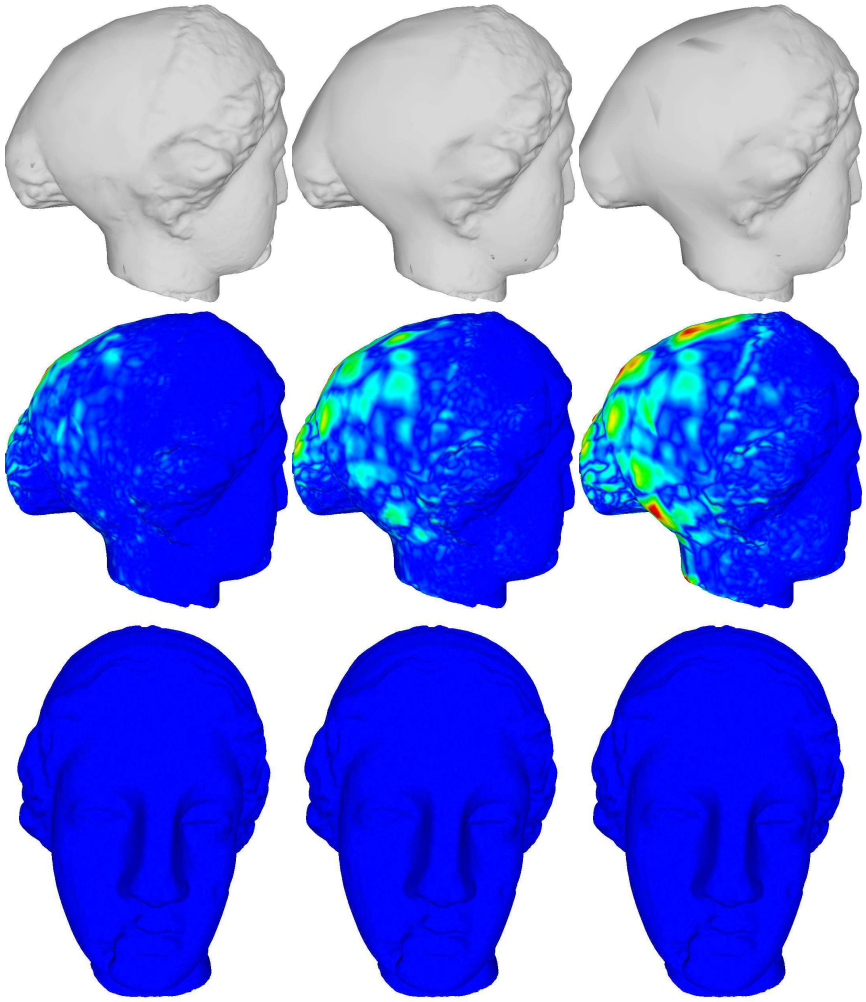
*(a)* $\theta_{lim} = 90^\circ$ *(b)* $\theta_{lim} = 70^\circ$ *(c)* $\theta_{lim} = 45^\circ$

*Figure 4.13: Lossy predefined ROIs for model* igea*: wireframe results. Observe that the sampling density varies while preserving a valid topology.*

For the visible triangles, the proposed ROI coding methodology should guarantee *visually* lossless results. Figure 4.16 shows examples for the *heptoroid*, *fertility* and *golfball* models. Depending on the model, front-facing triangles can either be found mainly in the front half of the model (e.g., *golfball*) or over the entire surface (e.g., *heptoroid*). To avoid selecting ROIs over the entire surface, experimental results are added where the front-view results are limited to those ROIs in the front half, indicated by "*front-view (half)*". This approach is depicted in Figure 4.15b, showing the bounding box of a model and shading, in gray, the volume where ROIs are allowed when viewing from the right. The result for the fertility model is shown in Figure 4.16d.

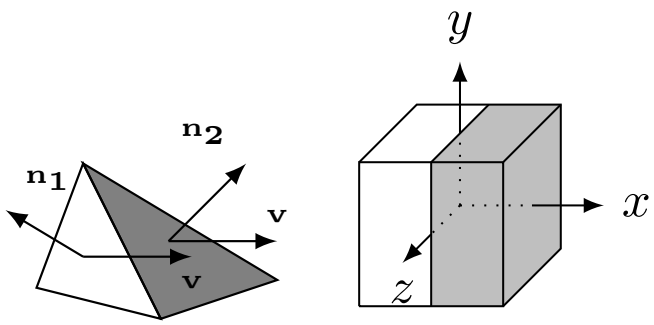
The *point-based ROI* selects a random vertex of the base mesh; at each resolution $j$ those triangles surrounding this selected vertex are selected as part of $ROI_j^S$, while each higher resolution either keeps the same vertex or one of the newly created vertices to continue this process. Visual results with this method are shown in Figure 4.17.

Such experiments are sensitive to the orientation, geometry and topology of the models, but still give insights on how the proposed ROI decoding methodology performs. The proposed evaluation scenarios are considered as *soft extrema*. On the one hand, the *front-view* results give an estimation on the maximally useful ROIs, considering the camera position but disregarding occlusion and lower resolution requirements due to the distance to the camera. On the other hand, *point-based* decoding can be considered as an estimation on the minimally useful ROIs: only a single vertex is considered as ROI and the resolution over the reconstructed mesh surface is minimal in order to provide for a valid mesh topology.



*(a)* $\theta_{lim} = 90^\circ$ *(b)* $\theta_{lim} = 70^\circ$ *(c)* $\theta_{lim} = 45^\circ$

*Figure 4.14: Lossy predefined ROIs for model* igea*: visual results. Observe that models still appear identical when rendered from the front (bottom row).*




*(a) front-view* *(b) front-view (half)*

*Figure 4.15: Front-view ROI selection. (a) shows the selection of front-facing triangles. With* $\mathbf{v}$ *pointing towards the camera viewpoint and* $\mathbf{n}$ *the surface normal, the vertices of a triangle are part of the ROI if* $\mathbf{n} \cdot \mathbf{v} > 0$*. In (b), an alternative ROI is depicted, where only half of the model is eligible for the ROI (i.e., the triangles in the gray volume).*

#### 4.6.2.1 Adaptive Inverse Wavelet Transform

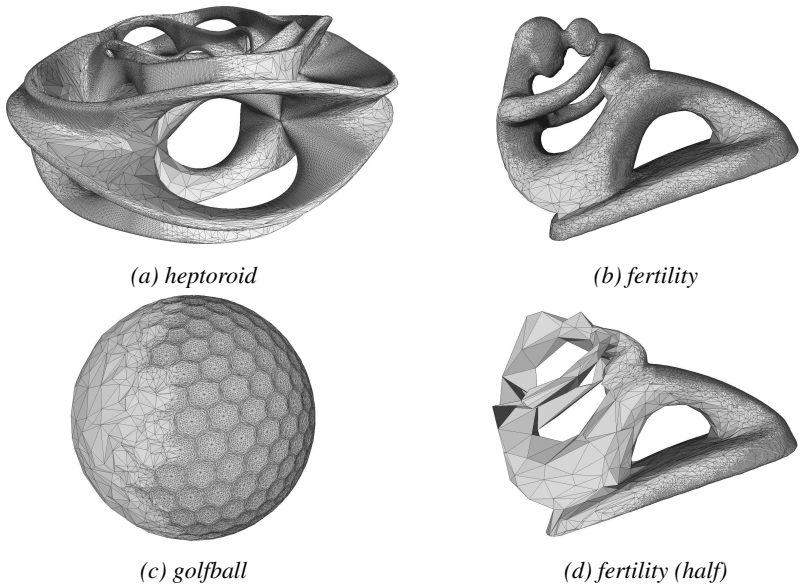
This section evaluates the inverse wavelet transform proposed in Section 4.4.1, which is independent of any tiling decisions. The amount of decoded vertices is investigated for the three ROI decoding scenarios, i.e. *front-view*, *front-view (half)* and *point-based*, indicating the ratio of ROI vertices w.r.t. the total amount of vertices as:

$$\rho = \frac{v_{ROI}}{v_{total}}. \tag{4.37}$$

The results are shown in Figure 4.18. The lines connecting the samples have no significance but were added for clarity.

For front-view decoding, $\rho$ converges to $50\%$, which was expected assuming that approximately half of the triangles are facing any camera in general. The overhead caused by the ROI propagation (see Equation 4.12 and Figure 4.6c) is only significant for smaller models, where $\rho$ goes up to $80\%$. For point-based decoding, $\rho$ converges to $0\%$ as the ROI, i.e., a single vertex, reduces relative to the full-resolution sizes.

Such accurate representations are possible because the wavelet transform does not depend on any tiling decisions. For tile-based solutions, the smallest $\rho$ depends on the tiling granularity. These results are valuable from a rendering perspective: a reduction in the amount of vertices directly relates to the reduction in memory usage for real-time rendering. However, as mentioned at the end of Section 4.4.1, without tiling, this still requires lossless decoding and substantial amounts of memory that linearly scale with the mesh sizes.



*(a) heptoroid* *(b) fertility*

*(c) golfball* *(d) fertility (half)*

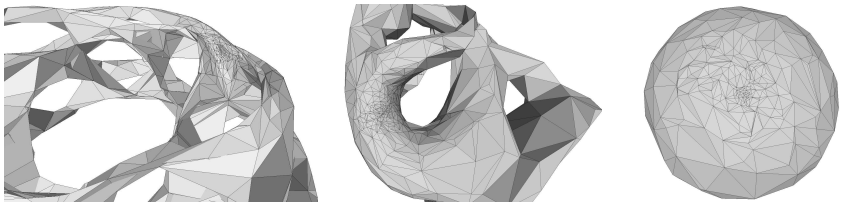
*Figure 4.16: Front-view ROI selection: wireframe results. For each of the "Front-view ROI" (a, b, c) scenarios, all regions facing a specific direction are considered ROI. In the example of* fertility*, the front is considered as pointing towards the right: the naive approach (b) selects regions at the back, i.e., the left side in the figure, while the "Front-view (half)" approach (d) only allows ROIs to be selected on the front side, i.e., the right half in the figure. For more topologically complex models, such as* heptoroid *(a), such an approach does not suffice and more advanced occlusion checking and better adapted resolutions per region are required. Less topologically complex models such as* golfball *will not have significantly different results when limiting the ROIs to the front.*

#### 4.6.2.2 Dynamic Tile-Based Coding

The next set of experiments investigates the dynamic tiling discussed in Section 4.4.2, introduced to reduce the coding rate required by the ROI-aware inverse transform. Consider again the *front-view*, *front-view (half)* and *point-based* ROI coding scenarios. Additionally, the bit rates for lossless decoding are presented, both with and without tiling, i.e., *ROI-aware* and *ROI-agnostic*. These figures reveal the actual rate penalty introduced by tiling.

Given a fraction $p$, the effect of splitting tiles is evaluated by setting $\tau^{\text{TS}} = p.n_v$, with $n_v$ the amount of vertices in a model. This results in similar tiling for all models, independent of the model sizes. Results are given for models up to $350\,000$ vertices at $12$ bit quantization, and models up to over $3.6$ million vertices at $21$ bit quantization.

High values of $p$ result in only splitting the tiles once. When splitting at $p = 40\%$ (see plots in Figures 4.19a and 4.19b), the single base tile (see Figure



*(a) heptoroid* *(b) fertility* *(c) golfball*

*Figure 4.17: Point-based ROI selection: wireframe results. With a single point considered as the ROI, the figures depict how the resolution reduces when the distance to this point increases.*

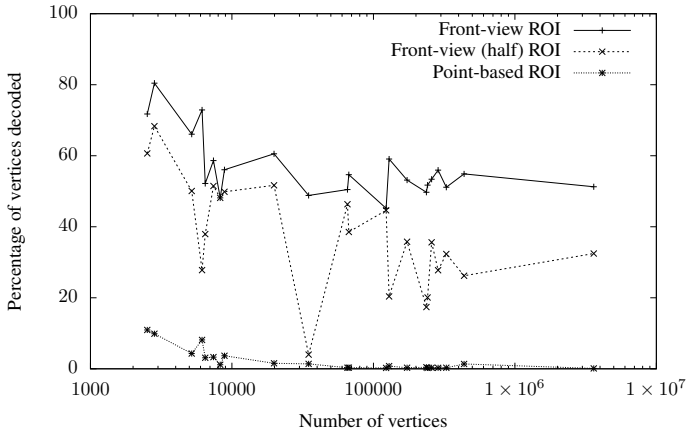



*Figure 4.18: Percentage of vertices of ROI-decoded models. This plot shows the percentage $\rho$ of decoded triangles when (i) considering front-facing regions (Front-view ROI), (ii) front half regions (Front-view (half) ROI), and (iii) only considering the region surrounding a random point (Point-based ROI).*

4.8) will only be split at a high resolution. In general, none of the eight new tiles will surpass this $\tau^{\mathrm{TS}}$ again, resulting in a highest resolution with eight tiles. Observe that the lossless penalty is minimal, which reduces with increasing model sizes. Rate gains when decoding a single vertex are also observed. Finally, when increasing the amount of quantization bits, the obtainable rate gains also increase, showing that the proposed approach is valuable for high-accuracy models. The gains are limited due to the large amount of data that is still encoded in a single tile.

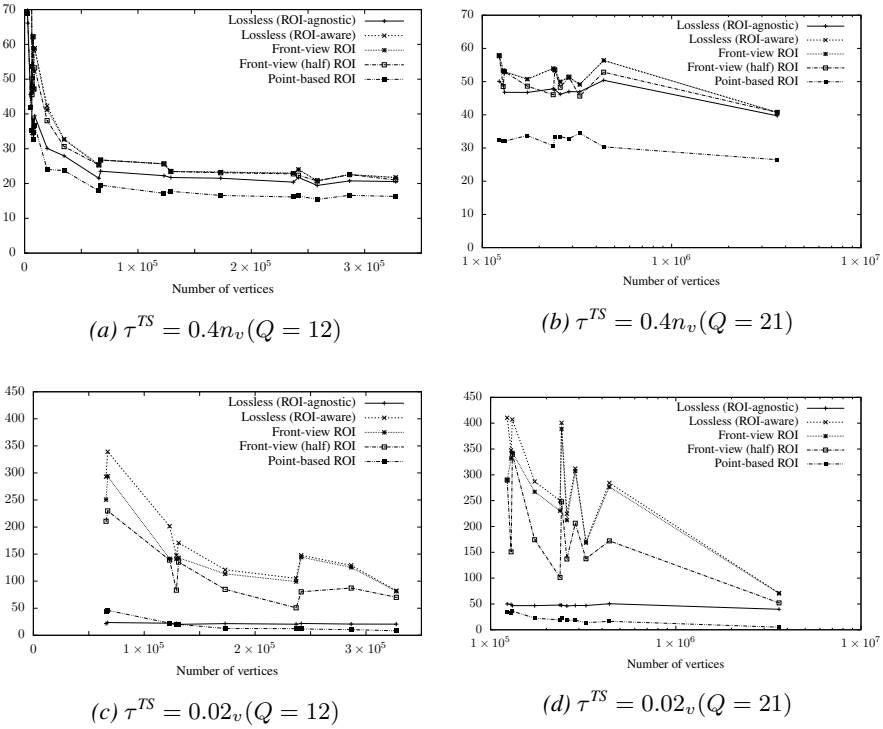
For low values of $p$, tiles are split at a lower resolution in Figure 4.8. Figures

4.19c and 4.19d show the results when splitting at $p = 2\%$. Although higher lossless rate penalties are observed, also observe that these penalties reduce with increasing model sizes. For $\rho = 2\%$, models require $10^6 \sim 10^7$ vertices for sufficiently reduced penalties. Using smaller tile sizes (resulting in more tiles) requires larger models to be efficient. For ROI-decoding, however, much larger gains can be obtained. For smaller models the gains are limited due to the ROI propagation (as was also mentioned in Section 4.6.2.1); for larger models, significant gains are observed. The *dragon* model of $437\,645$ vertices decodes the $21$ bit quantized ROI around a single vertex at $16.4$ bpv (while lossless decoding requires $50.4$ bpv); given that $\tau^{\text{TS}} = 8\,753$ for this model, any ROI consisting of several thousands of vertices will be decoded at a similar bit rate. Similarly, the *Asian dragon* model of $3\,609\,600$ vertices decodes the ROI around a single vertex at $4.96$ bpv (of the $39.7$ lossless bpv); as $\tau^{\text{TS}} = 72\,192$, ROIs consisting of tens of thousands of vertices will be decoded at a similar rate. Finally, remark that, with increasing model sizes, the expected ROI will decrease w.r.t. the total model size, which signifies that the actual rates will move closer towards the *point-based* results.

Notice that decoding the front-view ROI mostly coincides with lossless decoding. This confirms the fact that triangles which face the camera, i.e., front-facing triangles, are not necessarily restricted to the front half of a model, as was seen in Figures 4.16a and 4.16b. Consequently, depending on the tiling granularity, nearly all of the tiles can be required, such that requiring all front-facing triangles at the highest resolution is detrimental for random access. Allowing only front-facing triangles *in the front half* for the ROIs, as depicted by the *front-view (half)* results, still more than half of the tiles are required due to the ROI propagation discussed in Section 4.4.1; these will require lower-resolution triangles in the back half of the model to be upsampled, to ensure proper reconstruction of the front half without blocking artifacts. To allow for efficient streaming, the distance to the camera (i.e., using lower resolutions for far away regions) and the actual visibility (considering, for instance, the view frustum and self-occlusions) need to be taken into account.

**Parallel Decoding** An additional advantage of the proposed tiling approach is the unlocked parallel decoding opportunity. Consider an example where a single tile is split into eight tiles at some resolution $R - k$, with $k > 0$. For instance, this is the case for $p = 40\%$ depicted in Figure 4.19a. The results show an average lossless rate penalty of $14.8$ bpv when considering all models, or, by ignoring models with less than $20\,000$ vertices, an average rate penalty as low as $2.5$ bpv.

Earlier, this rate penalty was already justified considering ROI decoding: for large models a relatively small ROI will be required. Consequently, for reconstructing ROI-aware versions of large models, the low value of $\rho$ will result




*(a)* $\tau^{TS} = 0.4n_v (Q = 12)$

*(b)* $\tau^{TS} = 0.4n_v (Q = 21)$

*(c)* $\tau^{TS} = 0.02_v (Q = 12)$

*(d)* $\tau^{TS} = 0.02_v (Q = 21)$

*Figure 4.19: Decoding ROIs with relative tile sizes: bit rates. Bit rates for decoding ROIs, for models encoded with relative tile sizes.*

in reduced bit rates compared to ROI-agnostic coding.

In addition, if sufficient memory, processing power and bandwidth are available and the full model needs to be decoded losslessly, the decoding of the $k$ highest resolution wavelet subbands (i.e., the computationally most expensive subbands to decode) will benefit from a potential eightfold speedup. This is made possible because the individual tiles can be decoded completely independently, even across resolutions. This has not yet been experimented with, however.

## 4.7 Conclusions

In this second part of the dissertation, predefined *Region-of-Interest (ROI)* coding for wavelet-based irregular mesh codecs has been proposed, showing how this can be done in general by scaling wavelet coefficients, similar to the maxshift method of JPEG 2000, and how an ROI-aware inverse wavelet transform can reduce geometric errors and the required amount of vertices. A predefined ROI allows

the encoder to prioritize regions if bandwidth or memory is limited. Furthermore, if the ROI is appropriately selected, lossless visual quality can be obtained at a reduced bit rate and using fewer vertices.

Additionally, an ROI based wavelet transform for irregular triangle meshes has been proposed which allows for varying the resolution of a model over its surface at the finest granularity level. To allow for randomly accessing parts of the data, dynamic tiling in the wavelet domain was proposed where each tile can be independently processed. This allows for adapting the tiling to the sampling densities and the requested ROIs, while also permitting decoding speedups in the lossless case by allowing for parallelism. Despite rate penalties which are unavoidable when adding ROI support, decoding ROIs can be done at the fraction of the lossless rate, for increasingly larger models.

The results show that we are at the turning point where ROI support proves its value. In future work, a more efficient implementation will be able to process models which are several orders of magnitude larger; the main challenge will become the encoder which still needs to process the entire model, not just a selected ROI.

For a truly scalable system, both this ROI coding as well as resolution scalability are required. Together, the available bandwidth and memory usage can be optimized, given a specific camera viewpoint. Viewing a model from far away will result in the model taking up only a small portion of the display; full resolution is not required as many triangles are simply displayed with a single pixel. Resolution scalability in these cases reduces the amount of vertices and triangles, limiting the transmitted data and the memory used for rendering. When viewing the model from nearby such that triangles at the highest resolution can be distinguished, larger portions of the model will no longer be visualizable. ROI decoding will also reduce the amount of vertices and triangles by only decoding the visible regions. Hence, given a viewpoint, the decoded models can be limited to the requested *regions* and their appropriate *resolutions*.

The work presented in this chapter has led to the following publications:

- J. El Sayeh Khalil, A. Munteanu, and P. Lambert. *Rate-Distortion Optimized Wavelet-based Irregular Mesh Coding*. In Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP), volume 1, pages 212–219, Porto, Portugal, 27 February – 1 March 2017.

- J. El Sayeh Khalil, A. Munteanu, and P. Lambert. *Scalable Irregular Mesh Coding with Interactive ROI Support*. IEEE Transactions on Circuits and Systems for Video Technology, accepted with minor revisions.

# References

[1] Hugues Hoppe. *Progressive Meshes*. In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 99–108, New Orleans (Louisiana), United States, 4–9 August 1996.

[2] Sébastien Valette, Raphaëlle Chaine, and Rémy Prost. *Progressive Lossless Mesh Compression via Incremental Parametric Refinement*. Computer Graphics Forum, 28(5):1301–1310, 2009.

[3] Michael Lounsbery, Tony D. DeRose, and Joe D. Warren. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. ACM Transactions on Graphics, 16(1):34–73, January 1997.

[4] Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. *Progressive Geometry Compression*. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 271–278, New Orleans (Louisiana), United States, 23–28 July 2000.

[5] Sébastien Valette and Rémy Prost. *Wavelet-Based Progressive Compression Scheme for Triangle Meshes: Wavemesh*. IEEE Transactions on Visualization and Computer Graphics, 10(2):123–129, March/April 2004.

[6] Leon Denis, Shahid Mahmood Satti, Adrian Munteanu, Jan Cornelis, and Peter Schelkens. *Scalable Intraband and Composite Wavelet-Based Coding of Semiregular Meshes*. IEEE Transactions on Multimedia, 12(8):773–789, December 2010.

[7] Joel Askelöf, Mathias Larsson Carlander, and Charilaos Christopoulos. *Region of Interest Coding in JPEG 2000*. Signal Processing: Image Communication, 17(1):105 – 111, 2002.

[8] Hongjuan Zheng, Bo Liu, and Hongbin Zhang. *Region-of-Interest Coding of 3D Mesh Based on Wavelet Transform*. In Proceedings of the 3rd International Conference on Image and Graphics (ICIG), pages 438–441, Hong Kong, China, December 2004.

[9] Enrico Gobbetti, Fabio Marton, Paolo Cignoni, Marco Di Benedetto, and Fabio Ganovelli. *C-BDAM — Compressed Batched Dynamic Adaptive Meshes for Terrain Rendering*. Computer Graphics Forum, 25(3):333–342, 2006.

[10] Lok M. Hwa, Mark A. Duchaineau, and Kenneth I. Joy. *Adaptive 4-8 Texture Hierarchies*. In Proceedings of the IEEE Conference on Visualization (VIS), pages 219–226, Austin (Texas), United States, 10–15 October 2004.

[11] Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio, and Roberto Scopigno. *Planet-sized Batched Dynamic Adaptive Meshes (P-BDAM)*. In Proceedings of the IEEE Conference on Visualization (VIS), pages 147–154, Seattle (Washington), United States, 19–24 October 2003.

[12] Hugues Hoppe. *View-Dependent Refinement of Progressive Meshes*. In Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 189–198, Los Angeles (California), United States, 3–8 August 1997.

[13] Liang Hu, Pedro V. Sander, and Hugues Hoppe. *Parallel View-Dependent Level-of-Detail Control*. IEEE Transactions on Visualization and Computer Graphics, 16(5):718–728, September 2010.

[14] Patrick Gioia, Olivier Aubault, and Christian Bouville. *Real-time Reconstruction of Wavelet-encoded Meshes for View-dependent Transmission and Visualization*. IEEE Transactions on Circuits and Systems for Video Technology, 14(7):1009–1020, July 2004.

[15] Michaël Roy, Sebti Foufou, and Frédéric Truchetet. *Multiresolution Analysis for Irregular Meshes with Appearance Attributes*, pages 80–86. Computer Vision and Graphics: International Confonference, ICCVG 2004, Warsaw, Poland, September 2004, Proceedings (Computer Imaging and Vision). Springer Netherlands, 2006.

[16] Jeffrey Ho, Kuang-Chih Lee, and David Kriegman. *Compressing Large Polygonal Models*. In Proceedings of the IEEE Conference on Visualization (VIS), pages 357–573, San Diego (California), United States, 21–26 October 2001.

[17] Sungyul Choe, Junho Kim, Haeyoung Lee, Seungyong Lee, and Hans-Peter Seidel. *Mesh Compression with Random Accessibility*. In Proceedings of the 5th Korea-Israel Bi-National Conference on Geometric Modeling and Computer Graphics (IK), pages 81–86, Seoul, Korea, 11–12 October 2004.

[18] Sungyul Choe, Junho Kim, Haeyoung Lee, and Seungyong Lee. *Random Accessible Mesh Compression using Mesh Chartification*. IEEE Transactions on Visualization and Computer Graphics, 15(1):160–173, January 2009.

[19] Sung-Eui Yoon and Peter Lindstrom. *Random-Accessible Compressed Triangle Meshes*. IEEE Transactions on Visualization and Computer Graphics, 13(6):1536–1543, November/December 2007.

[20] Martin Isenburg, Peter Lindstrom, and Jack Snoeyink. *Streaming Compression of Triangle Meshes*. In Proceedings of the 3rd Eurographics Symposium on Geometry Processing (SGP), Vienna, Austria, 4–6 July 2005.

[21] Bo Liu and Hong-Bin Zhang. *Wavelet Based Progressive Mesh Compression with Random Accessibility*. In Proceedings of the International Symposium on Intelligent Multimedia, Video and Speech Processing (ISIMP), pages 615–618, Hong Kong, China, 20–22 October 2004.

[22] Céline Roudet, Florent Dupont, and Atilla Baskurt. *Semi-Regular 3D Mesh Progressive Compression and Transmission based on an Adaptive Wavelet Decomposition*. In Proceedings of SPIE – Wavelet Applications in Industrial Processing VI, volume 7248, pages 1–12, San Jose (California), United States, 18–22 January 2009.

[23] Zhi-Quan Cheng, Hua-Feng Liu, and Shi-Yao Jin. *The Progressive Mesh Compression based on Meaningful Segmentation*. The Visual Computer, 23(9):651–660, 2007.

[24] Pierre Alliez and Mathieu Desbrun. *Progressive Compression for Lossless Transmission of Triangle Meshes*. In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 195–202, Los Angeles (California), United States, 12–17 August 2001.

[25] Adrien Maglo, Ian Grimstead, and Céline Hudelot. *Cluster-based Random Accessible and Progressive Lossless Compression of Colored Triangular Meshes for Interactive Visualization*. In Proceedings of the Computer Graphics International Conference (CGI), pages 1–8, Ottawa (Ontario), Canada, 12–15 June 2011.

[26] Ho Lee, Guillaume Lavoué, and Florent Dupont. *New Methods for Progressive Compression of Colored 3D Mesh*. In Proceedings of the 18th International Conference on Computer Graphics, Visualization and Computer Vision (WSCG), pages 199–206, Plzeň, Czech Republic, 1–4 February 2010.

[27] Adrien Maglo, Ian Grimstead, and Céline Hudelot. *POMAR: Compression of Progressive Oriented Meshes Accessible Randomly*. Computers & Graphics, 37(6):743–752, 2013.

[28] Zhiyan Du, Pavel Jaromersky, Yi-Jen Chiang, and Nasir Memon. *Out-of-Core Progressive Lossless Compression and Selective Decompression of Large Triangle Meshes*. In Proceedings of the IEEE Data Compression

Conference (DCC), pages 420–429, Snowbird (Utah), United States, 16–18 March 2009.

[29] Pierre-Marie Gandoin and Olivier Devillers. *Progressive Lossless Compression Of Arbitrary Simplicial Complexes*. ACM Transactions on Graphics, 21(3):372–379, July 2002.

[30] Junho Kim, Sungyul Choe, and Seungyong Lee. *Multiresolution Random Accessible Mesh Compression*. Computer Graphics Forum, 25(3):323–331, 2006.

[31] Clément Courbet and Céline Hudelot. *Random Accessible Hierarchical Mesh Compression for Interactive Visualization*. Computer Graphics Forum, 28(5):1311–1318, 2009.

[32] Costa Touma and Craig Gotsman. *Triangle Mesh Compression*. In Proceedings of the Graphics Interface Conference (GI), pages 26–34, Vancouver (British Columbia), Canada, 18–20 June 1998.

[33] Jerome M. Shapiro. *Embedded Image Coding using Zerotrees of Wavelet Coefficients*. IEEE Transactions on Signal Processing, 41(12):3445–3462, December 1993.

[34] David S. Taubman and Michael W. Marcellin. *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers, Norwell (Massachusetts), United States, 2001.

[35] Adrien Maglo, Clément Courbet, Pierre Alliez, and Céline Hudelot. *Progressive Compression of Manifold Polygon Meshes*. Computers & Graphics, 36(5):349–359, 2012.

# 5
# Overall Conclusions

Over the last couple of decades, the portion of three-dimensional (3D) content within the vast amount of digital media has been growing at an enormous pace. Applications which render interactive virtual scenes can benefit from vastly increased processing power and available memory in commodity PCs. Moreover, mobile devices such as tablets, smartphones and smart glasses are becoming more ubiquitous, allowing for even more ground to be covered by 3D media. However, despite such a wide spread of digital 3D media, numerous limitations need to be considered in order to provide good quality. Firstly, despite advances in network bandwidth, memory capacity and CPU processing power, both the quantity and quality of 3D models is growing at such a pace that compression techniques remain a necessity. Secondly, models in interactive applications have been represented using levels of detail to cope with system limitations. Compression techniques hence should consider forms of scalability in order to provide similar functionality in an efficient way. This has become an even more pronounced necessity given the larger diversity in storage, bandwidth, memory and processing power available to a more diverse set of devices. Good quality is then subjectively determined by both the actually rendered quality as well as by the allowed interactiveness.

Issues related to 3D model storage and representation have been studied more deeply in Chapter 2. 3D models are often modeled as polygon meshes; for real-time rendering, triangle meshes are conventionally used. These digital representations are obtained using the same two key processes found in any analogue-to-digital conversion: *sampling* and *quantization*. The effect of sampling *regularity* has been discussed: while regular meshes more succinctly describe

mesh samples and allow for a more compressed storage *per vertex*, irregular meshes allow for describing a mesh surface using fewer samples and consequently allow for a reduced memory footprint when rendering. This idea has been explored in the wavelet-based mesh representation presented later in Chapters 3 and 4. Chapter 2 further described measures that are used for comparing mesh coding systems. A conventional comparison considers the remaining distortions when specific bit rates are spent, resulting in *rate-distortion (RD)* curves. In this dissertation, the RD curve comparison was expanded upon by considering the average rate difference to summarize a comparison using a single number. This allows for reporting a codec comparison over a larger set of models. In addition to the classical RD comparison, this dissertation also considered the memory requirements *after* decoding a model. This rendering performance, denoted as the *triangle-distortion (TD)* comparison, considers the remaining distortion given a specific percentage of triangles used, which is directly related to the memory used for rendering. Again, an average triangle percentage difference describes the difference between two curves as a single number, allowing for the evaluation over a larger test set to be reported.

Chapter 3 introduced a feature-preserving irregular mesh codec. The first part of this chapter described all components of the proposed resolution-scalable irregular mesh codec, which is composed of a wavelet transform which preserves geometric features in an implicit way without any preprocessing or any thresholding, and octree-based coders which allow for exploiting spatial correlations via *successive approximation quantization (SAQ)*, significance coding and bit plane coding. The wavelet transform iteratively converts a high-resolution mesh into a low-resolution mesh and a wavelet subband by partitioning the vertices of the high-resolution mesh into odd and even vertices, where each odd vertex is surrounded by even vertices. The odd vertices are removed, resulting in patches which are then retriangulated while preserving geometric features. A single parameter allows for detecting patches where multiple geometric features coincide, which allows for preserving such key vertices. To obtain wavelet coefficients, a two-mode prediction is employed by considering the dihedral angles within the low-resolution patches. If a large angle is found, surpassing a second threshold parameter, the prediction is based solely on the corresponding edge; if not, the entire patch is considered for making a prediction. An octree-based coding approach then efficiently encodes the resulting wavelet coefficients. This same octree-based coding approach is also proposed for connectivity encoding: by assigning a single bit per edge to indicate which edges are inside a patch, the decoder is able to detect all patches for the inverse transform. The results showed that the codec outperforms the state of the art for irregular and feature-rich models while only performing suboptimally at the final, *lossless* rate when, for 12 bit quantized models, average rates of 25 bits per vertex (bpv) are required,

of which 8 bpv are needed for connectivity information. Visually pleasing results are often obtained at under 10 bpv, where more densely sampled models can cope with even lower bit rates. Furthermore, real-time rendering was considered by investigating the distortion w.r.t. the amount of triangles instead of the amount of bits, i.e., using the TD measure. This revealed that the proposed codec outperforms the state of the art in an even more pronounced way, signifying that the proposed codec obtains similar quality using less graphics memory, or vice versa that better quality is obtained with the same amount of memory. The trade-off mentioned above clarifies this aspect. Using irregular meshes allows for a more adaptive (albeit more expensive) distribution of samples; consequently, the cost per sample is higher but the same quality can be obtained using fewer samples. This ensures a competitive RD performance while outperforming the state of the art considering graphics memory usage for real-time rendering.

In the second part of Chapter 3, quality scalability was further investigated. By using SAQ, quality scalability was already offered *per resolution*, i.e., the smallest decodable part allows for refining a single bit plane within a resolution. However, this does not allow for scaling the wavelet coefficient quality *across resolutions*. Given a resolution which is not fully decoded, decoding parts of the next resolution results in drift as the encoder and decoder use a different mesh for embedding the samples within the octrees. Such drift is avoided by introducing a template mesh per resolution. This template mesh has the same connectivity as the real mesh, but does not rely on decoded geometry data. As it is synchronized at both the encoder and decoder side, the template mesh is used for embedding samples and allows for decoding without drift, even if geometry information is incomplete. Results have shown that the impact on the coding rate is minimal with increments up to 0.2 bpv observed, while allowing for an arbitrary storage and transmission order. More importantly, this allowed for a rate-distortion-optimized storage and transmission order. This entails that data is encoded such that the distortion is reduced optimally: either a new resolution of vertices is added, or the quality of any existing resolution is improved, whichever introduces the highest quality increase at the lowest rate. This optimization showed clear additional improvements in the low-to-midrange bit rates for feature-rich models.

The codec introduced in Chapter 3 offers resolution and quality scalability. Both are global properties, that is, the resolution and wavelet coefficient quality can be set for an *entire* model. This suffices for models where all details can be observed when the models are displayed *entirely*. However, models currently have such high resolutions that the finest details can only be observed from nearby, when large parts of the models are no longer displayed. Efficiently handling such models demands a new form of scalability. Region-of-Interest (ROI) scalability considers the issue of handling specific regions within a model in order to optimize bit rates and memory consumption further. This was addressed in Chapter 4.

A first part of this chapter discussed encoder-side ROIs. An encoder-side ROI is used for prioritizing specific regions in a model, e.g., ensuring that the details in the face of a virtual human character are visualized prior to the details in the clothes. ROIs need to be defined per resolution, where each lower-resolution ROI needs to supply sufficient information to allow for decoding higher-resolution ROIs. An ROI propagation method was proposed which ensures this. These ROIs, defined in the spatial domain, are then mapped to ROIs in the wavelet domain. With these wavelet-domain ROIs, any wavelet-based codec which encodes wavelet coefficients in a bit-plane-by-bit-plane fashion can be enhanced, without altering either the wavelet transform or the encoding step. This is done by a process called *wavelet coefficient boosting*: upscaling wavelet coefficients within the ROI prior to encoding, and downscaling these after decoding and prior to the inverse transform. Irregular meshes allow for even further optimization: regions in the background (BG) can be kept smooth by preserving a reduced resolution, as opposed to upsampling and smoothing the resulting BG samples as is required for semi-regular mesh coding. Evaluation is tightly coupled with the considered ROIs: for a given camera position, all *camera-facing* regions were considered as the ROI in this dissertation. If the decoder only needs to render from a specific camera position and the encoder accurately predicts this position, the results showed that only half of the vertices and triangles are required. Additionally, the required bit rate is reduced by over 3 bpv without introducing any visual artifacts. Reducing the selected ROI further reduces this bit rate and this amount of vertices and triangles, while the introduced distortions remain unnoticeable. This has been explored for angles with the viewing direction up to $45°$ (instead of the visually lossless $90°$) where 6 bpv are saved, and only $30\%$ of the triangles are used.

The second part of Chapter 4 discussed a novel ROI decoding approach with *dynamic tiling* and *random access*. ROI decoding requires an adaptive inverse wavelet transform which no longer considers an entire wavelet subband but only a selection of wavelet coefficients. Due to the interactive selection of ROIs per resolution, one cannot ensure that sufficient data is decoded for all future, higher-resolution ROIs. Hence, the adaptive inverse wavelet transform needs to recursively upsample lower-resolution meshes. This results in meshes which have their resolution adapted to the requested ROIs while only providing additional samples outside these ROIs to ensure a proper topology. While this addresses issues related to graphics memory while rendering, this still requires decoding all wavelet coefficients. To avoid this, a dynamic tiling approach was proposed. These tiles are dynamically adapted to sampling densities per resolution instead of being fixed at the base resolution. Only the tiles corresponding to the requested ROIs need to be streamed and decoded. This tiling furthermore allowed for refining the RD optimization discussed in Chapter 3: within each resolution, the individual tiles can be considered to obtain the largest quality increase at the lowest rate. The

evaluation of ROI decoding considered two scenarios: either decoding all regions that face a specific camera location similar to the encoder-side ROI evaluation, or decoding a randomly selected vertex as ROI. The former is an estimation of the maximally required data, while the latter estimates the minimal amount of data required for decoding. Evaluating the adaptive inverse wavelet transform showed that the decoded amount of vertices and triangles is well adapted to the selected ROI. Only for lower-resolution models, relatively high amounts of vertices are reconstructed in the BG as well to ensure a proper topology, resulting in up to $80\%$ of vertices being reconstructed when requesting half of the models as ROIs. At higher resolutions, these additional vertices are negligible and on average $50\%$ of the vertices are reconstructed. Evaluating the dynamic tiling revealed that sufficiently large models in the order of $10^6 \sim 10^7$ vertices are required to benefit from tiling, in order to overcome rate penalties. For these large models, the results obtained for any realistic ROI will approach the results found for selecting a single vertex as ROI, and will show large rate gains when only decoding these ROIs while the lossless rate penalties becomes insignificant. A single tile of a $21$ bit quantized $3\times10^6$-vertex model which is losslessly encoded using $40$ bpv can now be decoded at $5$ bpv, reconstructing tens of thousands of vertices.

In conclusion, in this thesis a scalable feature-preserving irregular mesh codec was proposed with three main forms of scalability. Resolution and quality scalability allow for scaling respectively the amount of vertices and the reconstruction quality of the vertices. Spatial scalability comes in the form of ROI coding. This allows for an encoder to prioritize regions for encoding, while it allows for a decoder to select which parts of a model to upsample. Additionally, ROI decoding is enhanced by allowing for random access via dynamic tiling. Combining all forms of scalability results in a truly scalable way of visualizing 3D models.

**Future work** The work described in this dissertation addressed several fundamental challenges in 3D mesh representations, storage and transmission. While the codec already competes with the state of the art, and outperforms it for feature-rich models, the performance will further benefit from in-depth optimization of each part of the coding system. This includes optimizing the vertex partitioning to obtain better odd vertices per resolution, reconsidering the retriangulation by trading off representation quality and connectivity coding performance, investigating larger wavelet supports for better predictions, improving the geometry coder by transforming wavelet coefficients to local reference frames, and investigating techniques to reduce the connectivity cost.

Furthermore, the codec needs to be extended to encode arbitrary vertex attributes in addition to geometry. This will allow for the joint coding of these attributes, further optimizing the coding performance by exploiting correlations

across vertex attributes. The relationship with texture data, finally, also requires further investigation: scalable mesh representations which allow for both arbitrary vertex attributes and scalable texture data will prove invaluable for real-time rendering.