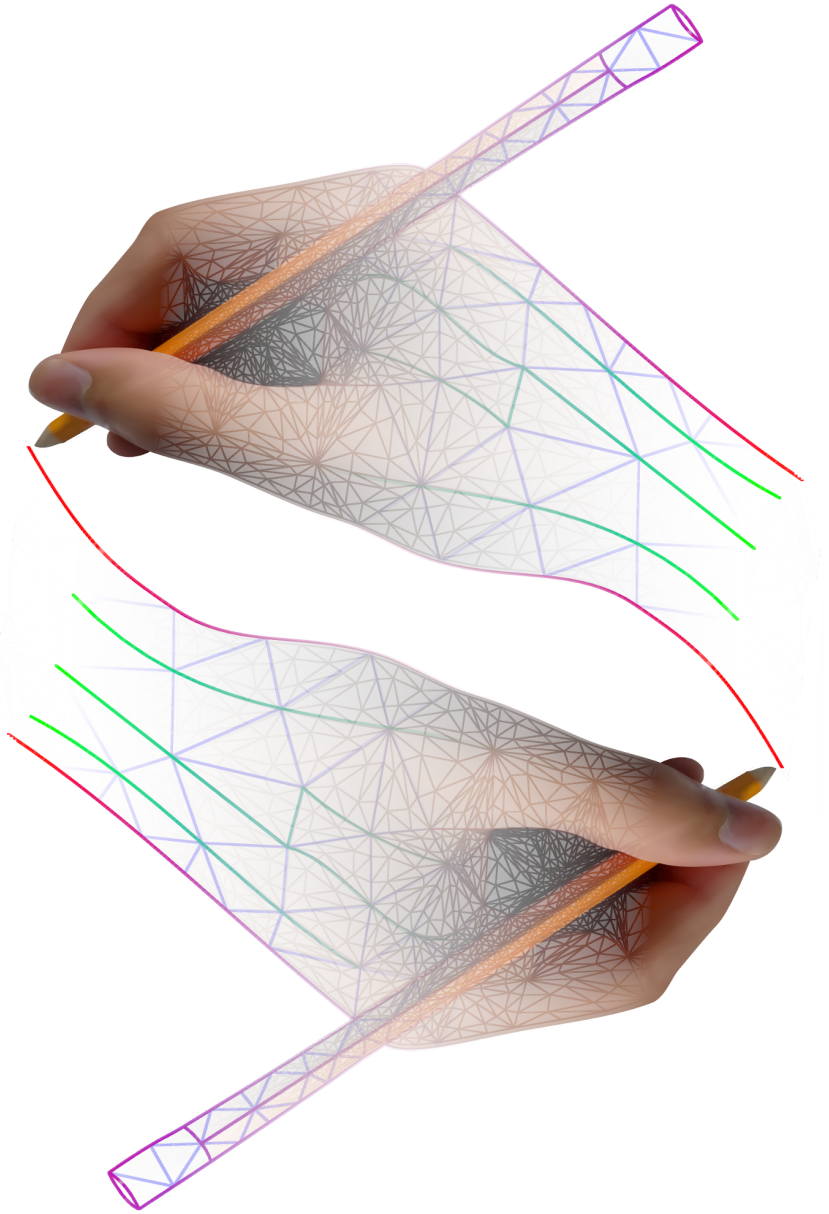
Arbitrary Topology Meshes in Geometric Design and Vector Graphics



Gerben Jan Hettinga

ARBITRARY TOPOLOGY MESHES IN GEOMETRIC DESIGN AND VECTOR GRAPHICS

GERBEN JAN HETTINGA

Cover: Vectorising Hands

Arbitrary Topology Meshes in Geometric Design and Vector Graphics

Gerben Jan Hettinga
Proefschrift

The research for this dissertation was conducted at the Scientific Visualization and Computer Graphics (SVCG) research group, part of the Bernoulli Institute (BI) and the Faculty of Science and Engineering (FSE) at the University of Groningen, the Netherlands. Additional research was conducted at Adobe Research, San Jose, California, United States of America.



university of groningen

Arbitrary Topology Meshes in Geometric Design and Vector Graphics

Proefschrift

ter verkrijging van de graad van doctor aan de
Rijksuniversiteit Groningen
op gezag van de
rector magnificus prof. dr. C. Wijmenga
en volgens besluit van het College voor Promoties

De openbare verdediging zal plaatsvinden op

dinsdag 7 december 2021 om 9:00 uur

door

Gerben Jan Hettinga

geboren op 2 januari 1993
te Sneek

Promotores
Prof. dr. J. Kosinka
Prof. dr. A.C. Telea

Beoordelingscommissie
Prof. dr. J.B.T.M. Roerdink
Prof. dr. K. Hormann
Prof. dr. T. Várady

Quid est Veritas?

— Pontius Pilate

ABSTRACT

Meshes are a powerful means to represent objects and shapes both in 2D and 3D. Nowadays, meshes are typically regarded as having triangular or quadrilateral elements. In addition, there can be restrictions on how these elements can be arranged, i.e. on topology. Even for higher order methods, such as splines, a regular topology is the norm. Meshes of arbitrary topology inevitably contain extraordinary vertices, vertices surrounded by an arbitrary number of faces, and/or extraordinary faces, faces with an arbitrary number of sides. Unfortunately, the traditional spline techniques cannot be applied directly to such topologies. Arbitrary topology meshes, and especially higher-order ones, have many interesting applications in geometric design and (vector) graphics, and can give designers more freedom in designing complex objects. This is exactly where our main contributions lie, as follows.

In computer aided design (CAD), objects are commonly represented as an arrangement of regular spline patches. These patches do not fit nicely together, but are rather arranged in such ways that parts of the surfaces are trimmed off where two or more surfaces intersect. Problems occur in these areas as the exact trimming curve cannot in general be determined, but only approximated. This leads to gaps or overlaps in the surface or the object in these areas. By converting the spline patches into Clough-Tocher elements, the spline patches can be approximated and made watertight at their boundaries. We improve on this conversion method by further ensuring tangent-plane continuity of CAD models by careful boundary curve and normal management, and the use of Shirman-Séquin macro-elements near the trimmed edges. For this we propose three different variants which differ in locality, approximation ability, and visual quality.

B-splines are a powerful surface representation, but in general only admit regular topologies in terms of their control net. We propose a generalisation of the B-spline construction that extends bi-degree uniform B-spline patches to extraordinary regions. The construction is an extension of the generalised Bézier patch that incorporates uniform B-spline basis functions. For this we create special B-spline ribbon surfaces which use extended B-spline basis functions. The resulting smooth multisided patches connect smoothly to other regular or multisided patches. The surfaces visually improve on arbitrary degree subdivision surfaces, but in some configurations might show shape defects, which we alleviate using special basis functions. We also show how to efficiently render multisided patches using the existing triangular and quadrilateral pipeline.

Gradient meshes are powerful vector graphics primitives, which have the ability of representing detailed and scalable images. They are traditionally defined as strictly quadrilateral meshes with a rectangular topology. Recent works have extended them to polygonal meshes using subdivision surfaces and generalised barycentric coordinates. We create two additional formulations of the polygonal gradient meshes using generalised Gregory patches. These patches are adjusted from their original 3D setting to the setting of smooth colour interpolation. We compare the existing and new techniques in terms of visual quality, performance, continuity and editability. The Gregory patches compare favourably to the existing techniques, in terms of rendering performance and quality. The expressive power of gradient meshes is limited to the colours that are assigned at vertices of the mesh. We pair procedural noise functions with gradient meshes to create noisy gradient meshes. We couple three different noise functions, Perlin, Worley, and Gabor noise to the mesh through the use of a shared parametrisation domain. In addition, we create parameters that locally and globally influence the noise by specifying these at vertices and interpolating these along with the geometry. Designers are then able to create spatially varying noise patterns using a very sparse mesh. Additionally, we show how the approach can be extended to displacement mappings applied to regular surface meshes.

The conversion of raster images into vector graphics has been a long standing problem. Past solutions have not seen wide adoption due to various issues such as rendering performance, reproduction quality, editability of the representation or its generality. We present a vectorisation method that proceeds in three steps: feature extraction, mesh generation, and colour transfer. The extracted hard (edges) and soft (shading) image features are vectorised into spline curves, which are in turn used as constraints for the generation of a curved triangular mesh. Colour is represented using mesh colours, a compact way to describe textures on a per-patch basis and an efficient method to transfer colour information to the mesh colours is devised. The combination of triangular patches and mesh colours can be rendered in real-time on commodity hardware. This leads to an efficient vectorisation pipeline that is able to handle a variety of input images such as drawings, designs, paintings, and natural images.

GEARFETTING

Mesken (Ingelsk: meshes) binne in krêftige manier om foarmen en objekten foarm te jaan yn 2D en 3D. Hjoed-de-dei lykje mesken foaral te bestean út trijehoekige en fjouwerkante eleminten. Der binne lykwols ek beheiningen op 'e manieren wêrop de ferskate eleminten regele wurde kinne, mei oare wurden har topology. Sels foar metoaden fan hegere oarder, lykas splines, is in reguliere topology de noarm. It is faaks unûntkomber dat mesken mei willekeurige topology besteane út bûtengewoane hoekpunten, hoekpunten omjûn troch in willekeurich oantal flakken, en/of bûtengewoane flakken, flakken mei in willekeurich oantal kanten. Spitigernôch is it net mooglik om tradisjonele splinetechniken direkt te brûken op sokke topologyen. Mesken fan willekeurige topology, en foaral dy fan hegere oarders, hawwe ynteressante tapassingen yn geometrysk ûntwerp en (fektor)grafyk, en kinne ûntwerpers mear frijheid jaan by it ûntwerpen fan komplekse objekten. Dit is krekt wêr't ús haadbydrage leit.

Yn Computer-aided design (CAD) wurde objekten normaal fertsjintwurdige as in regeling fan reguliere splinelappen. Dizze lappen passe net strak byinoar, mar binne sa arranzjearre dat bepaalde dielen fan 'e flakken ôfsnijd wurde wêr't twa of mear flakken elkoar krúse. Dizze gebieten binne problematysk, om't meastentiids de krekte ôfsnijkromme net bepaald wurde kin, allinnich mar by benadering. Dit liedt ta gatten of oerlappende dielen yn it objekt. Troch de splinelappen te konvertearjen yn Clough-Tocher-eleminten, kinne de orizjinele splinelappen benadere wurde en wetterticht makke oan 'e rânen. Wy ferbetterje dizze konvertearmetoade troch te soargjen dat sels de ynterfaasen kontinu binne troch foarsichtich te wêzen mei de omstannichheden om 'e rânekromme en de normale dêr en troch it brûken fan Shirman-Séquin-makro-eleminten om 'e snijde dielen hinne. Foar dit doel binne noch trije farianten betocht dy't ferskille yn lokaliteit, benader- en fisuelekwaliteit.

B-splines binne in krêftige manier om flakken te fertsjintwurdigjen, mar se binne oer 't algemien allinnich definieare foar in regelmjittich kontrôlenet. Wy stelle in generalisaasje foar fan 'e B-spline-konstruksje dy't unifoarme bi-graad B-splinelappen útwreidet nei bûtengewoane gebieten. De konstruksje is in útwreiding fan 'e generalisearre Bézierrûnte dy't unifoarme basisfunksjes omfettet. Hjirfoar meitsje wy spesjale B-spline lintflakken dy't útwreide basisfunksjes brûke. Dit resultearret yn glêde mearsidige lappen dy't soepel rinne yn oare reguliere en mearsidige lappen. De lappen ferbetterje fisueel op 'e willekeurige graad ûnderferdielingsflakken (Ingelsk: subdivision surfaces), mar yn guon konfiguraasjes kinne der wat defekten sichtber wêze, dy't wy ferbetterje

troch de basisfunksjes oan te passen. Wy litte ek sjen hoe mearsidige lappen effisjint werjûn wurde kinne troch de besteande trijehoekige en fjouwerkante tessellatie piipline.

Gradientmesken binne primitiven fan fektorgrafika dy't de mooglikheid hawwe om ôfbyldings op in detaillearre en skaalbere manier te fertsjintwurdigjen. Tradysjoneel binne se definiearre as fjouwerkante mesken mei in rjochthoekige topology. Resint wurk hat sjen litten dat dizze struktuer ek útwreide wurde kin nei polygonale mesken troch ûnderferdielingsflakken en generalisearre barysintryske koördinaten. Wy meitsje twa ekstra formulearringen foar polygonale gradientmessen basearre op generalisearre Gregory lappen. Dizze lappen binne oanpast fan har orizjinele 3D -omjouwing nei de omjouwing fan glêde kleurynterpolaasje. Wy fergelykje besteande en nije techniken op it gebiet fan fisuele kwaliteit, prestaasjes, kontinuiteit en oanpasberens. De Gregory lappen ferbetterje besteande techniken yn termen fan werjeftesnelheid en fisuele kwaliteit.

De ekspressiviteit fan gradiëntmesken is beheind ta de kleuren pleatst op 'e hoekpunten fan'e mesk. Wy kombinearje prosedurelerûsfunksjes mei gradientmesken om rûsige gradiëntmesken te meitsjen. Wy kombinearje trije ferskillende funksjes, Perlin, Worley, en Gaborrûs troch in dield parameterisaasjedomein. Wy meitsje ek parameters dy't de rûs lokaal en globaal beynfloedzje troch se te spesifisearjen op 'e hoekpunten en de wearden te ynterpolearjen gelyk mei de geometry. Untwerpers hawwe dan de mooglikheid om romtlik wikseljende rûspatroanen te meitsjen mei in tinferspriede mesk. Fierder litte wy sjen dat ús oanpak ek wurket foar ferpleatsingstawizing (Ingelsk: displacement mapping) fan reguliere flakmesken.

It konvertearjen fan rasterôfbylden nei fektorgrafykôfbylden is in al lang besteand probleem. Besteande oplossingen binne noch net wiidferspraat fanwegen ferskate problemen lykas werjeftesnelheid, reproduksjekwaliteit en it oanpasbarheid fan 'e fertsjintwurdiging. Wy presinteare in fektorisaasjemetoade dy't bestiet út trije stappen: kenmerkekstraksje, meskgeneraasje en kleurferfier. De ekstrahearde hurde (rânen) en sêfte (skaad) ôfbyldingskenmerken wurde fektorisearre yn splinekrommen, dy't op har beurt brûkt wurde foar it generearjen fan in kromme trijehoekig mesk. Kleur wurdt fertsjintwurdige troch meskkleuren, in kompakte manier om tekstuer per-lap te beskriuwen en in effisjinte metoade foar it oerdragen fan de kleuren is betocht. De kombinaasje fan trijehoekige lappen mei meskkleuren kin yn realtime werjûn wurde op hardware fan konsuminten. Dit liedt ta in effisjinte fektorisaasjepipeline dy't de mooglikheid hat om in ferskaat oan ynfierôfbylden te behanneljen, lykas tekeningen, ûntwerpen, skilderijen en foto's.

SAMENVATTING

Mazen (Engels: *meshes*) zijn een krachtige manier om objecten en vormen in 2D en 3D vorm te geven. Tegenwoordig worden mazen vooral gezien te bestaan uit driehoekige en vierkante elementen. Er zijn echter ook nog restricties in de manieren waarop de verschillende elementen kunnen worden gerangschikt, met andere woorden hun topologie. Zelfs voor methoden van hogere ordes, zoals splines, is een reguliere topologie de norm. Het is vaak onvermijdelijk dat mazen met willekeurige topologie bestaan uit extraordinaire hoekpunten, een hoekpunt omringd met een willekeurig aantal vlakken, en/of extraordinaire vlakken, een vlak met een willekeurig aantal kanten. Helaas is het niet mogelijk om traditionele spline technieken direct te gebruiken op zulke topologieën. Mazen van willekeurige topologie, en in het speciaal die van hogere ordes, hebben interessante toepassingen in geometrisch ontwerpen en (vector)grafiek, en kunnen ontwerpers meer vrijheid geven in het ontwerpen van complexe objecten. Dit is precies waar onze hoofdbijdrage ligt.

In computerondersteund ontwerp worden objecten normaliter gerepresenteerd als een arrangement van reguliere spline lappen. Deze lappen passen niet goed op elkaar, maar zijn zo gearrangeerd dat bepaalde delen van de vlakken worden afgesneden waar twee of meer vlakken elkaar snijden. Deze gebieden zijn problematisch omdat gewoonlijk de exacte afsnijdkromme niet kan worden vastgesteld, maar slechts kan worden benaderd. Dit leidt tot gaten of overlappende delen in het object. Door de spline lappen in Clough-Tocher elementen te converteren kunnen de originele spline lappen worden benaderd en kunnen deze waterdicht worden gemaakt bij de randen. Wij verbeteren deze conversiemethode door te zorgen dat zelfs de raakvlakken continu zijn door voorzichtig met de condities rond de randkromme en de normaal daar om te gaan en door het gebruik van Shirman-Séquin macro-elementen rond de afgesneden gedeelten. Voor dit doel hebben wij nog eens drie varianten die verschillen in lokaliteit, benader- en visuelekwaliteit.

B-splines zijn een krachtige manier om vlakken te representeren, maar ze zijn in het algemeen alleen gedefinieerd voor een regulier controlenet. Wij stellen een generalisatie voor van de B-spline-constructie die uniforme bi-graad B-spline lappen uitbreidt tot extraordinaire gebieden. De constructie is een extensie van de gegeneraliseerde Bézierlap die uniforme basisfuncties incorporeert. Hiervoor creëren we speciale B-spline lintvlakken die uitgebreide basisfuncties gebruiken. Dit resulteert in gladde meerkantige lappen die soepel overlopen in andere reguliere of meerkantige lappen. De vlakken verbeteren visueel gezien op de willekeurige graad onderverdeel vlakken (engels: subdivision surfaces),

maar in sommige configuraties kunnen er enkele defecten te zien zijn, die we vereffenen door de basisfuncties aan te passen. We laten ook zien hoe meerkantige lappen efficiënt kunnen worden weergegeven door de bestaande driehoekige en vierkante tessellatie-pijplijn te gebruiken.

Gradiëntenmazen zijn primitieven voor de vectorgrafiek die het vermogen hebben om een afbeelding op een gedetailleerde en schaalbare manier te representeren. Traditioneel zijn ze gedefinieerd als vierhoekige mazen met een rechthoekige topologie. Recent werk heeft laten zien dat deze structuur ook tot polygonalemazen kan worden uitgebreid door middel van onderverdeelvlakken en gegeneraliseerde barycentrische coördinaten. Wij creëren nog eens twee additionele formuleringen voor polygonalegradiëntenmazen gebaseerd op gegeneraliseerde Gregory lappen. Deze lappen zijn aangepast van hun originele 3D omgeving naar de omgeving van gladde kleurinterpolatie. We vergelijken de bestaande en nieuwe technieken op het gebied van visuele kwaliteit, prestatie, continuïteit en bewerkbaarheid. De Gregorylappen verbeteren op bestaande technieken op het gebied van weergavesnelheid en visuele kwaliteit.

De expressiviteit van gradiëntenmazen is gelimiteerd tot de kleuren die worden geplaatst op de hoekpunten van de maas. Wij combineren procedureleruisfuncties met gradiëntenmazen om ruisige gradiëntenmazen te creëren. We combineren drie verschillende functies, Perlin, Worley, en Gabor ruis door middel van een gedeelde parametrisatie domein. Tevens creëren we parameters die globaal en lokaal de ruis beïnvloeden door deze te specificeren op de hoekpunten en de waarden te interpoleren met de geometrie. Ontwerpers hebben dan de mogelijkheid om spatieel variërende ruis patronen te maken met een dunverspreide maas. Verder laten we zien dat onze aanpak ook werkt voor verplaatsingstoewijzing (Engels: displacement mapping) en regulierevlakmazen.

Het converteren van rasterafbeeldingen naar vectorgrafiekafbeeldingen is een al lang bestaand probleem. Bestaande oplossingen zijn nog niet wijdverbreid door verschillende problemen zoals weergavesnelheid, reproductiekwalitieit en de bewerkbaarheid van de representatie. Wij presenteren een vectorisatiemethode die uit drie stappen bestaat: kenmerkextractie, maasgeneratie en kleurenoverzetting. De geëxtraheerde harde (randen) en zachte (arcering) afbeeldingskenmerken worden gevectoriseerd tot splinekrommen, die naderhand worden gebruikt voor het genereren van een gebogen driehoekigemaas. Kleur wordt gerepresenteerd door maaskleuren, een compacte manier om textuur de beschrijven op een per-lap basis en een efficiënte methode om de kleuren over te zetten is bedacht. De combinatie van driehoekige lappen met maaskleuren kan worden weergegeven in echte tijd op consumentenhardware. Dit leidt tot een efficiënte vectorisatiepijpleiding die het vermogen heeft om een verscheidenheid aan invoerafbeeldingen te hanteren, zoals tekeningen, designs, schilderijen en foto's.

PUBLICATIONS

This thesis is the result of the following publications:

- *Multisided Generalisations of Gregory Patches.* G.J. Hettinga, J. Kosinka, Computer Aided Geometric Design, Elsevier, 2018
- *A Comparison of GPU Tessellation Strategies for Multisided Patches.* G.J. Hettinga, P.J. Barendrecht, J. Kosinka, Eurographics (Short Papers), 2018
- *Noisy Gradient Meshes: Augmenting Gradient Meshes with Procedural Noise*, G.J. Hettinga, R. van Beckhoven, J. Kosinka, Graphical Models, Elsevier 2019
- *Colour Interpolants for polygonal gradient meshes*, G.J. Hettinga, R. Brals, J. Kosinka, Computer Aided Geometric Design, Elsevier, 2019
- *Conversion of B-rep CAD models into globally G^1 triangular splines*, G.J. Hettinga, J. Kosinka, Computer Aided Geometric Design, Elsevier, 2020
- *A multisided C^2 B-spline patch over extraordinary vertices in quadrilateral meshes*, G.J. Hettinga, J. Kosinka, Computer-Aided Design, 2020
- *Multisided B-spline patches over extraordinary regions*, G.J. Hettinga, J. Kosinka, Smart Tools and Applications for Graphics, Eurographics, 2020
- *Efficient Image Vectorisation Using Mesh Colours* G.J. Hettinga, J.I. Echevarria, J. Kosinka, Smart Tools and Applications for Graphics, Eurographics, 2021

CONTENTS

1 INTRODUCTION 1
1.1 Contributions of this Thesis 3
1.2 Contents and Structure 4

2 PRELIMINARIES 5
2.1 Parametric Curves and Surfaces 5
2.1.1 Parametric Continuity 6
2.1.2 Geometric Continuity 7
2.2 Bézier Curves 8
2.2.1 Derivatives and Continuity 9
2.2.2 Degree Elevation 10
2.2.3 Splitting 11
2.3 Bézier Surfaces 12
2.3.1 Quadrilateral Bézier Patches 13
2.3.2 Triangular Bézier Patches 13
2.3.3 The Method of Chiyokura and Kimura 14
2.3.4 Gregory Patches 15
2.3.5 Generalised Bézier Patches 16
2.3.5.1 Generalised Gregory Patches 18
2.3.6 Gradient Meshes 18
2.4 B-splines Curves and Surfaces 19
2.4.1 Subdivision Surfaces 21
2.4.1.1 Catmull-Clark 22
2.4.1.2 Arbitrary Degree Subdivision 22
2.5 Generalised Barycentric Coordinates 23
2.5.1 Wachspress Coordinates 24
2.5.2 Mean value coordinates 25
2.6 Conclusion 26

I Geometric Design 27

3 INTRODUCTION 29

4 CONVERTING B-REP CAD MODELS INTO TANGENT PLANE CONTINUOUS SPLINES 31
4.1 Introduction 33
4.2 Related work 34
4.3 Preliminaries 35
4.3.1 Continuity Conditions 36
4.3.2 Triangulation of B-rep Patches 36
4.3.3 The Clough-Tocher Construction 36

4.3.4 The Shirman-Séquin construction 38
4.4 Preprocessing 39
4.4.1 G^1-compatible Boundary 40
4.4.2 Other Vertex Configurations 42
4.5 Conversion to Shirman-Séquin Splines 42
4.5.1 An Alternative G^1 Method 43
4.5.2 Full-strip 44
4.5.3 Saw-tooth 45
4.5.4 Saw-tooth: Secondary Split 45
4.6 Results 46
4.6.1 Sphere octants 47
4.6.2 Boundary adjustment 48
4.6.3 Approximation error 51
4.6.4 Performance 52
4.7 Discussion 53
4.8 Conclusion 55

5 MULTISIDED B-SPLINE PATCHES 57
5.1 Introduction 59
5.2 Related Work 60
5.3 Multisided B-spline Construction 60
5.3.1 Extended Cubic Basis Functions 61
5.3.2 B-spline Ribbons and Patch 63
5.4 Multisided B-spline Patches 65
5.4.1 Extraordinary Vertex Patches 65
5.4.2 Extraordinary Face Patches 68
5.5 Continuity 70
5.6 Results 71
5.6.1 Challenging Cases 71
5.6.2 Arbitrary Topology Meshes 71
5.6.3 Effect of Normalisation 71
5.6.4 Shape Adjusted Basis Functions 74
5.7 Discussion 78
5.8 Rendering Multisided Patches 79
5.8.1 Tessellation Strategies 80
5.8.2 Adaptive Tessellation 83
5.8.3 Results & Performance 84
5.9 Conclusion 86

II Vector Graphics 89

6 INTRODUCTION 91
6.1 Raster Images 91
6.2 Vector Graphics 91
6.3 Contents 94

7 NOISY GRADIENT MESHES 95
7.1 Introduction 97
7.1.1 Procedural Noise Functions 98
7.1.1.1 Perlin Noise 99
7.1.1.2 Worley Noise 99
7.1.1.3 Gabor Noise 100
7.1.1.4 Fractal Noise 101
7.2 Noisy Gradient Meshes 101
7.3 Results 103
7.4 Discussion 106
7.5 Conclusion 107

8 COLOUR INTERPOLANTS FOR POLYGONAL GRADIENT MESHES 109
8.1 Introduction 111
8.2 Related Work and Preliminaries 112
8.2.1 Polygonal Gradient Mesh 112
8.2.2 Topologically Unrestricted Gradient Meshes 113
8.2.3 Cubic Mean Value Coordinates 114
8.2.4 Gregory Generalised Bézier Patches 115
8.2.5 The Charrot-Gregory Corner Patch 116
8.3 Local G^1 Colour Interpolation 117
8.3.1 The Method of Chiyokura and Kimura for Colour Interpolation 117
8.3.2 Quadrilateral Gregory Gradient Patches 119
8.4 Multisided Colour Interpolants 120
8.5 Results and Discussion 122
8.5.1 Colour Interpolation on Convex Patches 123
8.5.2 Colour Interpolation on Concave Patches 124
8.5.3 Performance 126
8.5.4 Editability 128
8.6 Conclusion 131

9 IMAGE VECTORISATION WITH MESH COLOURS 133
9.1 Introduction 135
9.2 Related Work 137
9.3 Overview 139
9.4 Feature Extraction 140
9.5 Mesh Generation 142
9.6 Texture Transfer 145
9.6.1 Mesh colour Fitting 147
9.6.2 Rendering 149
9.7 Results 151
9.7.1 Performance 152

9.7.2 Editing 154
9.7.3 Comparisons with Previous Work 155
9.8 Discussion 157
9.9 Conclusion 161

10 CONCLUSION 163
10.1 Future Work 165

BIBLIOGRAPHY 167

ACKNOWLEDGMENTS 179

1 INTRODUCTION

The mesh is a concept that is ubiquitous in computer graphics. The simple concept of connecting points in space to form edges and in turn to connect the edges into loops to form faces is a simple means to represent shape. A collection of connected faces conveys the shape of an object or surface as a discretised set of simple (flat) surfaces. Although the definition of a mesh does not put any restrictions on the valency, the number of edges or sides of a face, a mesh in computer graphics is commonly understood to be either a collection of triangles or quadrilateral faces. Naturally, there are good reasons for this.

The triangle is the simplest polygon one can create and render. Connect any three points in space and a triangle will be the result. The simplicity of this primitive has also seen its use in rendering, where graphics hardware is optimised for rasterising many such triangles. For this reason triangle-based meshes are primarily used in the gaming industry.

The quadrilateral does not in general generate a flat shape when connecting all four points, but it can be trivially triangulated (there are at least two ways to do this) and rasterised afterwards. The main advantage of using quadrilaterals to model surfaces is that it gives designers intuitive control over the shape of the surface. The *edge flow* of a mesh is a topological property of a mesh where the main lines of curvature can be easily inferred from the topology of the mesh, and is a handy guide-line for designers to create nice surfaces. The use of quadrilateral elements is also standard procedure in the animation industry. Here subdivision surface algorithms such as Catmull-Clark subdivision [CC78] are the standard and work best for quadrilateral based meshes.

Most spline definitions, like Bézier patches and B-splines surfaces, are also given by triangular or quadrilateral elements. This poses restrictions in how meshes with such elements can be defined as the topology has to adhere to certain conditions. Only a certain number of faces can be constructed around a vertex or else they cannot be defined well or be smoothly joined together at the vertex. Control point meshes that contain extraordinary vertices, vertices with arbitrary valency, or extraordinary faces, faces with any number of sides, cannot be handled well. This poses restrictions on where these splines can be used and applied, or the way that designers interact with them. Using faces with an arbitrary number of sides in meshes or spline surfaces is not commonly done. Handling such elements comes with a few problems that can complicate traditional rendering and processing pipelines. The rendering of such faces is non-trivial. The faces could be triangulated, but there is a

multitude of ways to do so. For many spline patches there is no generalisation to meshes with arbitrary topology, a mesh with vertices with any valency and faces with any number of sides.

The detail that a mesh can express is often times a function of the number of vertices and faces of the mesh. For highly detailed objects it is required that a lot of vertices are needed to convey a fine amount of detail. This complicates the way objects are designed as all those individual vertices have to be defined. In addition the density of the mesh creates a higher memory footprint, decreasing rendering performance and increasing storage cost. The parametrisation of meshes allows to evaluate functions on the surface of the mesh. This allows for techniques such as texture mapping, where an image or texture is mapped onto a surface, or displacement mapping where the surface is perturbed according to a displacement function.

Objects or scenes of objects can also be represented by their projection on a flat plane commonly known as an image. Digital raster images are represented using a regular raster of pixels, or colour samples. The fidelity of the depicted objects is relative to the resolution, or number of pixels, of the image. Vector graphics provides an alternative representation of images, where it is built out of a collection of arbitrarily placed geometrical entities. The mesh can also be applied in this setting, where in addition to geometrically defining the object, it is also tasked with providing a function describing the colour of the image. Again, the same problems arise here. Dense triangular or quadrilateral meshes are able to convey high-frequency image detail, but are hard to manipulate effectively. Again, splines can bridge the gap here, where sparse meshes can be created to model image regions. Splines are commonly used to model geometry, but they just as well can be used to model colour or colour surfaces. In any case meshes potentially provide a scale independent way to model images.

There are thus restrictions and limitations that are inherent to using meshes with only triangular or quadrilateral elements or splines with only rectangular topology. This limits the freedoms and workflows designers can use to create objects using these elements. Or it can complicate the downstream tasks that can be performed on such meshes such as simulation and analysis. The use of arbitrary topology meshes seems to offer apparent advantages over traditional structures. We thus want to answer the following question:

RQ1: *How do arbitrary topology mesh structures aid, and improve upon, the definition of objects in geometric design and vector graphics?*

Transferring techniques from the original regular setting to arbitrary topology meshes is not easy. Arbitrary topology meshes contain regions that complicate the definition of splines or create situations where smoothness cannot be guaranteed. These regions, known as

extraordinary regions, are areas of a mesh that contain extraordinary vertices or extraordinary faces. In quad meshes, extraordinary vertices are vertices with valencies other than four and extraordinary faces are non-quadrilateral faces. The inclusion of such elements can be seen to divide the mesh into several regular regions. Then extraordinary vertices are the meeting place of several regular regions, meeting in such a way that their parametrisations are incompatible. Extraordinary faces can be surrounded by regular regions, but often times are built out of extraordinary vertices themselves, complicating the situation even further. These areas pose challenging problems to solve. We summarise this in the following question:

RQ2: *How can we adjust existing techniques to work in arbitrary topology situations?*

In their most basic form, meshes represent the surface of an object, but the mesh can be used to do more than that. Traditional techniques such as texture mapping or displacement mapping already allow the evaluation of a texture function or displacement function, respectively. However, the mesh structure could be used to store even more data than just that. We therefore also ask ourselves:

RQ3: *How can we use traditional mesh structures to generate complex geometry or colour surfaces?*

These research questions are answered by the following contributions.

1.1 CONTRIBUTIONS OF THIS THESIS

This thesis examines the use of arbitrary topology meshes in geometric design and vector graphics. We examine ways in which existing spline techniques can be used in combination with arbitrary topology meshes to create smooth surfaces in different settings. We tackle arbitrary topology B-splines surfaces by creating multisided patches that can be fit into otherwise regular surfaces whilst preserving smoothness. We also look at arbitrary topology NURBS-based B-rep models that have been converted to triangular spline surfaces, and devise means to connect them smoothly. In addition, we look for the best ways in which colour can be interpolated smoothly over flat polygonal meshes. We also examine how ordinary surfaces or mesh representations can be augmented with additional data so that complex geometry or colour surfaces can be generated. This is used to create a new vector graphics primitive that can efficiently vectorise images and that adds extra expressivity to traditional gradient meshes.

1.2 CONTENTS AND STRUCTURE

This thesis is divided into two parts, but is preceded by a preliminary Chapter 2 which covers many of the building blocks and theory that is common to the two parts of the thesis. Then in the first Part I, we cover geometric design algorithms for arbitrary topology meshes. The problems of this part of the thesis are introduced in Chapter 3. We then cover the conversion of arbitrary topology B-rep CAD models into tangent plane continuous triangular spline surfaces in Chapter 4. Then, in Chapter 5, we show how to extend uniform bi-degree B-splines to span patches over extraordinary regions in meshes.

In Part II, we look at how arbitrary topology and augmented meshes can be used in vector graphics. First a brief introduction is given on vector graphics and vectorisation in Chapter 6. After which, we first look at how procedural noise functions greatly increase the expressivity of standard gradient meshes in Chapter 7. Then we explore different forms, existing and newly created, of polygonal gradient meshes in Chapter 8. Lastly, we investigate an efficient image vectorisation pipeline in Chapter 9. Finally, the thesis is concluded in Chapter 10 and in addition a few recommendations for future work are given.

2 PRELIMINARIES

In this chapter we set out and explain many of the building blocks, underlying theory and algorithms used in the chapters throughout this dissertation. The dissertation is split into two parts, but this does not mean that there is no overlap in the used constructions. Both the vector graphics and geometric design part make use of parametric curve and surface techniques discussed here.

2.1 PARAMETRIC CURVES AND SURFACES

At the core of many of the techniques used in this thesis are parametric functions. A parametric function is a function

$$\mathbf{y} = f(\mathbf{u}),$$

where $\mathbf{y} \in \mathbb{R}^n$ and $\mathbf{u}$ are parametric values in $\mathbb{R}^m$. Special cases of these are curves in 2D or 3D with $m = 1$ and $n = 2, n = 3$ respectively. Surfaces are defined as $m = 2$ and $n = 2$ or $n = 3$. We will concern ourselves mostly with surfaces in 3D and 2D.

A parametric curve is often represented as a combination of basis functions $B_i(t)$ and some vector valued coefficients $\mathbf{p}_i$

$$\mathbf{c}(t) = \sum_{i=0}^{n} B_i(t)\mathbf{p}_i.$$

The coefficients $\mathbf{p}_i$ are often called control values, or control points when considering values in $\mathbb{R}^2$ and $\mathbb{R}^3$.

For surfaces we need at least two separate parameter variables. Still, we can specify the surface as a combination of basis functions, now parametrised by u or v, for example

$$\mathrm{S}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{n} \mathbf{p}_{ij} B_i(u) B_j(v).$$

This last formulation is known as a tensor product surface. Throughout this thesis we will use many different forms of parametric surfaces which are parametrised in various ways. The most common parametrisations will use the bilinear parametrisation $u \in [0, 1]$ and $v \in [0, 1]$ or the barycentric parametrisation $[u, v, 1-u-v]$, where $u+v+(1-u-v) = 1$ and $u, v \in [0, 1]$.

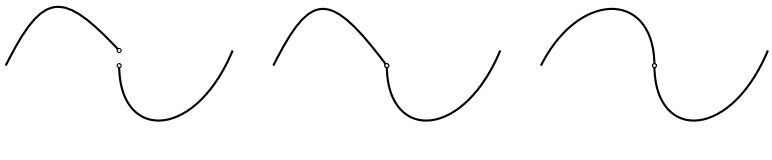


Figure 2.1: *Joining two curves,* $\mathbf{c}_0$ *and* $\mathbf{c}_1$ *with differing levels of continuity at* $\mathbf{c}_0(1)$ *and* $\mathbf{c}_1(0)$.

2.1.1 *Parametric Continuity*

When working with parametric functions to define a curve or surface it is often useful to talk about the continuity of the object as a whole. Usually, objects that are represented by parametric functions are made of many individual curve pieces and surface pieces or patches. The pieces are stitched together and collectively form the whole object. When saying something about the global smoothness or continuity of the object requires to examine the object at the sections where the pieces join together. Of less importance, but not irrelevant, is the continuity of the pieces themselves. This is usually determined by the degree and smoothness of the basis functions it consists of.

To give a sense of different levels of continuity we will first examine the univariate case, of curves, and then go on to surfaces. Given two curves $\mathbf{c}_0(t_0)$ and $\mathbf{c}_1(t_1)$ with $t_0, t_1 \in [0, 1]$ we want to join them together at $\mathbf{c}_0(1)$ and $\mathbf{c}_1(0)$. We can define the continuity C^d as follows

- C^{-1}, discontinuous, the curves are not connected i.e. $\mathbf{c}_0(1) \neq \mathbf{c}_1(0)$
- C^0, continuous, the curves are connected i.e. $\mathbf{c}_0(1) = \mathbf{c}_1(0)$
- C^1, the first derivative is continuous i.e. $\frac{\partial}{\partial u}\mathbf{c}_0(1) = \frac{\partial}{\partial u}\mathbf{c}_1(0)$ and they are C^0
- C^n, the n-th derivative is also continuous i.e. $\frac{\partial^n}{\partial u^n}\mathbf{c}_0(1) = \frac{\partial^n}{\partial u^n}\mathbf{c}_1(0)$ and the curves are C^{n-1}

One must note that when two curves are C^n continuous they are also C^{n-1} continuous. Using this notation we can make statements about the continuity of a collection of curves as a whole.

As with the univariate case we can examine the continuity at a part where the two surfaces are to be stitched together. However, in this case we have to take into account that the patches can be continuous in multiple ways. We can distinguish two types of continuity: *versal* and *transversal*. Versal continuity is 'trivially' satisfied by making the common side of the surfaces coincide positionally or better said make them C^0 continuous. Consider two bilinearly parametrised surfaces $\Phi_a(u, v)$ and

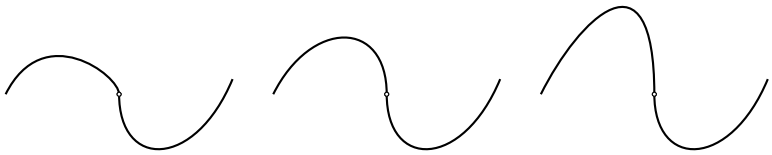


Figure 2.2: *Three different top curves that all connect with G^1 continuity to the bottom curve.*

$\Phi_b(u, v)$. On the common boundary $\Gamma(u)$, $u \in [0, 1]$, of the two surfaces $\Phi_a(u, v)$ and $\Phi_b(u, v)$ one can find the versal derivative $\frac{\partial^k}{\partial^k u}\Gamma(u)$ along the boundary curve and the transversal, or cross-boundary, derivatives $\frac{\partial^k}{\partial^k v}\Gamma_a(u)$ on patch $\Phi_a(u, v)$ and $\frac{\partial^k}{\partial^k v}\Gamma_b(u)$ on patch $\Phi_b(u, v)$.

- C^0, the patches are connected i.e. $\Phi_a(u, 0) = \Phi_b(u, 0)$
- C^1, the first derivative is continuous i.e. $\frac{\partial}{\partial v}\Phi_a(u, 0) = \frac{\partial}{\partial v}\Phi_b(u, 0)$
- C^k, the k-th derivative is also continuous i.e. $\frac{\partial^k}{\partial u^k}\Phi_a(u, 0) = \frac{\partial^k}{\partial u^k}\Phi_b(u, 0)$.

2.1.2 *Geometric Continuity*

Another useful class of continuity is geometric continuity commonly denoted by the symbol G^k. This allows for connecting curves without requiring that they are parametrically smooth, but only geometrically. In other words it is a continuity class that is independent of the parametrization. This means that curves can appear smooth, but when travelling from one curve to another there can be a difference between magnitudes of derivatives. However, when displaying curves it is often not important how fast one travels along the curve as it is displayed in one piece. Of course, the curves can also be made parametrically smooth by reparametrising the curves, i.e. finding another function $f(t)$ that makes a curve $c(t)$ with geometric continuity G^k parametrically smooth by setting $c(f(t))$, i.e. increasing the speed of t increases the magnitude of the derivative. It should be noted that the class of curves and surfaces that are geometrically smooth contains the class of splines that is parametrically smooth, up to pathological cases with vanishing derivatives. It is thus typically *less* restrictive. Figure 2.2 shows an example where three sets of curves are all connected with G^1 continuity. They are all smooth, but connect slightly differently to the bottom curve.

Two curves are G^k continuous if there exists a regular parametrisation that such that they become C^k continuous. Thus in the curves case,

the derivatives differ only in magnitude, but their direction is the same. For surfaces, bilinearly parametrised Φ_a and Φ_b, a sufficient and necessary condition for G^1 continuity is the co-planarity of the three tangent vectors $\partial\Gamma(u)$, $\partial\Gamma_a(u)$ and $\partial\Gamma_b(u)$ along $\Gamma(u)$, i.e.,

$$\det(\partial\Gamma(u), \partial\Gamma_a(u), \partial\Gamma_b(u)) = 0, \quad u \in [0,1].$$

This condition can be expressed in terms of scalar-valued functions $\alpha(u)$, $\beta(u)$ and $\gamma(u)$ such that

$$\alpha(u)\partial\Gamma(u) + \beta(u)\partial\Gamma_a(u) + \gamma(u)\partial\Gamma_b(u) = 0. \tag{2.1}$$

Here, α, β and γ are functions that can aid in satisfying the conditions. Intuitively this means that the derivatives are co-planar and that the surfaces are tangent plane continuous. There exist ways to create conditions for G^2 continuity [Kah83], but we will mostly restrict ourselves to G^1 continuity in this thesis.

Naturally, the mathematical notation for continuity is convenient, but there exist also other terms that convey the same meaning. *Watertightness* is another means to refer to C^0 continuity. Surfaces and curves are touching and should be able to hold a body of water. Tangent plane continuity is another way of stating that the surface is at least G^1 or even C^1 continuous.

We now detail the many parametric curves and surfaces that are used throughout this thesis. We will start with Bézier curves and several of the algorithms that can be applied to them and next to their extension to surfaces and of their variants. Then we will look at B-splines and their generalisations. Finally, we will detail generalised barycentric coordinates and in particular highlight two of their forms which are used throughout this thesis.

2.2 BÉZIER CURVES

The Bézier curve is one of the fundamental building blocks of this thesis. It returns in both the geometric design and vector graphics parts of this thesis. They were first described by Pierre Bézier at Renault in the 1960s as a means of describing in a mathematical way the body of cars. They have since seen their use in many other areas and have subsequently taken his name.

The Bézier curve is a degree d parametric curve of the form

$$\mathbf{B}(t) = \sum_{i=0}^{d} B_i^d(t)\mathbf{P}_i, \tag{2.2}$$

where t is a parameter value $0 \leq t \leq 1$. Here, $\mathbf{P}_i$ are the control points generally viewed as either points in $\mathbb{R}^2$ or $\mathbb{R}^3$. The collection of all control points are often denoted by drawing straight lines segments be-

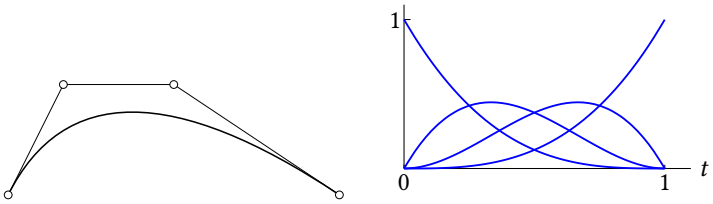


Figure 2.3: *Left: cubic Bézier curve, with control polygon. Right: cubic Bernstein polynomial functions.*

tween them. This creates a polygon which is named the control polygon of the curve. $B_i^d(t)$ are the Bernstein polynomials

$$B_i^d(t) = \binom{d}{i}(1-t)^{d-i}t^i. \tag{2.3}$$

The Bernstein polynomials are powerful basis functions and allow smooth blending of the control points. Figure 2.3, right, shows the four basis functions associated with a cubic Bézier curve. The basis functions partition unity, i.e. $\sum_{i=0}^{d} B_i^d(t) = 1$; this makes the curves affine invariant. A Bernstein polynomial of degree d can be written as a linear combination of polynomials of degree $d-1$

$$B_{i-1}^d = \frac{d-1-i}{d-1}B_i^{d-1}(t) + \frac{i+1}{d-1}B_{i+1}^{d-1}(t).$$

Similarly, the derivative can be written as the difference between two lower degree polynomials

$$\frac{\partial}{\partial t}B_{i-1}^d(t) = d(B_i^{d-1}(t) - B_{i+1}^{d-1}(t)).$$

In Figure 2.3 we show both a cubic Bézier curve in $\mathbb{R}^2$ and the associated Bernstein polynomials.

Bézier curves interpolate the first $\mathbf{P}_0$ and last control point $\mathbf{P}_d$, in addition the curve will always be tangent to the line segment connecting the first and second control point and the ultimate and penultimate control point. This allows them to be used to approximate general functions, by sampling function values and derivatives and provides an intuitive means for designers to shape the curves in whatever way they want.

2.2.1 *Derivatives and Continuity*

The way Bézier curves can be joined together can be determined by looking at their derivatives. A degree d Bézier curve has a first-derivative of the form

$$\frac{\partial}{\partial t}\mathbf{B}^d(t) = d\sum_{i=0}^{d-1} B_i^{d-1}(t)(\mathbf{P}_{i+1} - \mathbf{P}_i). \tag{2.4}$$

This means that the derivative of a degree d Bézier curve is yet another Bézier curve of which the degree is naturally $d - 1$ and the control points are the scaled differences between subsequent control points of the initial curve. The derivatives are thus only dependent on the control polygon of each curve. Therefore, assuring continuity amongst Bézier curves involves the process of determining positions of control points so that continuity conditions are satisfied. We examine the conditions between two degree d curves $\mathbf{b}_0(t)$ and $\mathbf{b}_1(t)$ with control points $\mathbf{Q}_i$ and $\mathbf{R}_i$ respectively.

- C^0 continuity requires that the end point of C_1 must be equal to the begin point of C_2 i.e. $\mathbf{Q}_d = \mathbf{R}_0$.
- C^1 continuity requires that first derivatives match i.e. $\mathbf{Q}_d - \mathbf{Q}_{d-1} = \mathbf{R}_1 - \mathbf{R}_0$
- C^2 continuity requires that second derivatives match i.e. $\mathbf{Q}_d - 2\mathbf{Q}_{d-1} + \mathbf{Q}_{d-2} = \mathbf{R}_0 - 2\mathbf{R}_1 + \mathbf{R}_2$
- C^k continuity requires that k-th derivatives match i.e. $\sum_{i=0}^{k}(-1)^i\binom{k}{i}\mathbf{Q}_{d-i} = \sum_{i=0}^{k}(-1)^i\binom{k}{i}\mathbf{R}_i$.

It is easy to smoothly join a Bézier curve to an existing curve by progressively assuring higher order continuity conditions starting from the C^0 condition. However, joining a curve in between two other curves requires a separation of control points. This is because the continuity conditions at start and endpoint might be conflicting. In these cases a linear curve is enough for C^0 continuity and a cubic curve is enough for C^1. For joining a curve with C^k continuity to two prescribed curves in general a curve of degree $2k + 1$ is needed.

Naturally, we can also join Bézier curves with geometric continuity. The first notable example of this is G^1 continuity, as G^0 also entails continuity in position. Consider again two curves $\mathbf{b}_0(t)$ and $\mathbf{b}_1(t)$ and their associated control points. We can join them with G^1 continuity by requiring that $(\mathbf{Q}_d - \mathbf{Q}_{d-1}) = \alpha(\mathbf{R}_1 - \mathbf{R}_0)$, where $\alpha > 0$. This also reveals why the geometry of the curves is smooth, even though they are not parametrically smooth. As shown before the tangent vectors at endpoints of Bézier curves are determined by the first two and last two control points. In this case the tangents vectors at the start and end points are in the same direction, but they only differ by a scaling factor α.

2.2.2 *Degree Elevation*

Using the property of the Bernstein polynomials that a lower degree polynomial can be expressed as linear combination of higher degree polynomials allows also to express a lower degree Bézier curve as a

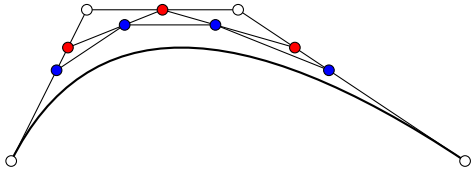


Figure 2.4: *The same exact Bézier curve expressed as cubic (white control points), quartic (red control points) and quintic curve (blue control points).*

higher degree one. Consider a Bézier curve with degree d and control points $\mathbf{P}_i$ which we want to degree elevate to degree $d+1$. This reduces to finding linear combinations of control points

$$\mathbf{Q}_i = \frac{i}{d+1}\mathbf{P}_{i-1} + \frac{d+1-i}{d+1}\mathbf{P}_i,$$

where $0 < i \leq d$. The degree elevation property is often useful as it allows for expressing a curve with additional control points. For instance we can degree elevate a cubic curve to quartic, adding an extra control point in the middle of the control polygon, which can be freely manipulated without ruining the G^1 continuity conditions at the end points of the curve. In Figure 2.4 we show the effect of degree elevating a cubic curve to several higher degree curves. As can be seen the control polygon changes, but the curve does not.

2.2.3 *Splitting*

Bézier curves can have any degree and this in turn increases the number of control points, and in some sense the expressiveness of the curves. However, the effect that changing the positions of these control points (save for the first and last) is not major, especially when considering higher degree curves. To get more control over the shape of the Bézier curve it is often better to split the curve so that it becomes a composite curve or spline. Then changing the control points of the spline directly has a much more direct effect on the shape of the curve than it would when changing a control point of a higher degree curve.

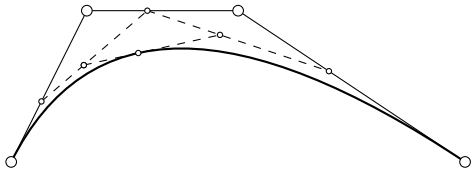


Figure 2.5: *Evaluation of a Bézier curve using De Casteljau's algorithm. The intermediate points generated also constitute two cubic Bézier curves that join together with G^∞ continuity.*

The splitting of the Bézier curve follows the process of de Casteljau's algorithm. This algorithm is another, numerically stable, way to evaluate Bézier curves and was developed by Paul de Casteljau at Citroën in 1959 [dC59]. The process is quite simple and elegant. Given a parameter t and a control polygon with control points $\mathbf{P}_i, i \in 0, \ldots, d$ it can be formalised as the following recurrence iteration

$$\mathbf{P}_i^0 = \mathbf{P}_i, i = 1, \ldots d$$
$$\mathbf{P}_i^j = (1-t)\mathbf{P}_i^{j-1} + t\mathbf{P}_{i+1}^{j-1}, i = 1, \ldots d-j, j = 0, \ldots, d$$

After d iterations of this recurrence the point $\mathbf{P}_0^d$ will be the point on the curve at parameter value t. Figure 2.5 shows the evaluation of a Bézier curve and the intermediate points that are generated.

The intermediate points generated during the recurrence are not without meaning and a subset of these can actually be used as control points of two curves $\mathbf{c}_0(t_0)$ and $\mathbf{c}_1(t_1)$ which cover the domains $[0, t]$ and $[t, 1]$ of the original curve respectively. The curves have control points $\mathbf{P}_0^0, \ldots, \mathbf{P}_0^d$ and $\mathbf{P}_0^d, \ldots, \mathbf{P}_d^0$ respectively. The curves connect with G^∞ continuity and can be reparametrised to have C^∞ continuity.

2.3 BÉZIER SURFACES

Bézier curves can be generalised to surfaces and inherit some of their favourable properties. Like the curves, surfaces consist of a combination of control nets of control points and associated basis functions, which are combinations of Bernstein polynomials. There are many variations of the Bézier surface, all of which share the use of the Bernstein-basis and control points.

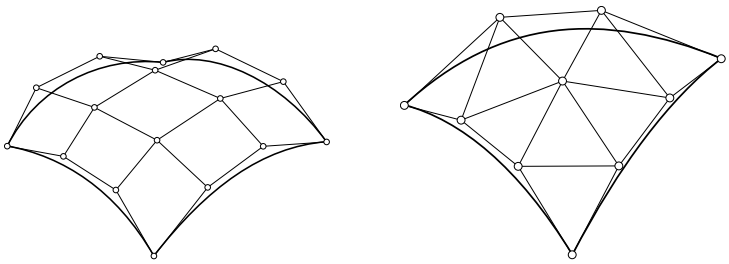


Figure 2.6: *The control net of a bicubic Bézier patch (left) and the control net of a cubic triangular Bézier patch (right).*

2.3.1 *Quadrilateral Bézier Patches*

The simplest type of Bézier surface is the quadrilateral Bézier patch, which is also known as the tensor-product Bézier patch. Like the Bézier curve it possesses a simple control structure of control points, but this time it is not a control polygon but rather a grid of points $\mathbf{P}_{i,j}$ which is known as the control net. Figure 2.6, left, shows the control net of a bicubic Bézier patch. It is parametrised bilinearly by two parameters $u \in [0,1]$ and $v \in [0,1]$ that are used with the Bernstein polynomials $B_i^n(t)$. The definition of the patch is

$$\mathbf{B}^{n,m}(u,v) = \sum_{i=0}^{n}\sum_{i=0}^{m} B_i^n(u)B_j^m(v)\mathbf{P}_{i,j}. \tag{2.5}$$

The equation can be interpreted in various ways. First it can be interpreted as evaluating m degree n Bézier curves in the u direction and subsequently using the evaluated points as control points of a degree m Bézier curve in the v directions. Like the Bézier curve it can also be evaluated through De Casteljau's algorithm by taking linear combinations of four control points at a time. The combination of Bernstein polynomials in the u and v direction can be seen as assigning a special combined basis function per control point. Like the Bézier curve it passes through some of the control points, in this case the four control points at the corners. The boundaries of the patches are Bézier curves.

2.3.2 *Triangular Bézier Patches*

Another generalisation from curves to surfaces can be achieved by using barycentric coordinates and triangular control nets. Barycentric coordinates are coordinates u, v, and w, here $u+v+w=1$, $u,v,w \in [0,1]$. Then a degree d Bézier triangle is defined as

$$\mathbf{B}^d(u,v,w) = \sum_{i+j+k=d, i,j,k\geq 0} B_{ijk}^d \mathbf{P}_{ijk},$$

where $B_{i,j,k}^d(u,v,w) = \binom{d}{i,j,k} u^i v^j w^k$ are the multinomial expansion of the Bernstein polynomials using barycentric coordinates. Like the quadrilateral version, the boundaries of the patch are Bézier curves, and the patch passes through the three corner vertices. Figure 2.6, right, shows the control net of a cubic triangular Bézier patch.

Like Bézier curves, Bézier patches can be degree elevated by generating the control points for a higher degree patch

$$\mathbf{Q}_\mathbf{i} = \frac{1}{n-1}(\mathbf{i}_0\mathbf{P}_{\mathbf{i}+\mathbf{e}_0} + \mathbf{i}_1\mathbf{P}_{\mathbf{i}+\mathbf{e}_1} + \mathbf{i}_2\mathbf{P}_{\mathbf{i}+\mathbf{e}_2}).$$

Here $\mathbf{i}$ is a multi index i,j,k and $\mathbf{e}_i$ are unit vectors along the i-th dimension. Then, the patch can be expressed using the degree elevated control points and basis functions

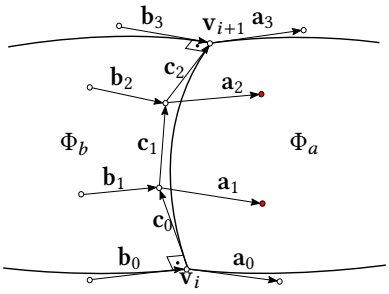


Figure 2.7: *A schematic view of the boundary conditions between the basis patch Φ_b and the actual patch Φ_a. The points created by the method of Chiyokura are depicted in red.*

$$\sum_{i+j+k=d,i,j,k\geq 0} B^d_{ijk}\mathbf{P}_{ijk} = \sum_{r+s+t=(d+1),r,s,t\geq 0} B^{d+1}_{rst}\mathbf{Q}_{rst}.$$

2.3.3 *The Method of Chiyokura and Kimura*

The method of Chiyokura and Kimura [CK83] joins two Bézier patches smoothly by ensuring G^1 continuity at shared edges by taking into account only shared positional and normal data. The method is applied to the common cubic boundary curve of two adjacent Bézier patches. Each boundary curve is defined by a control polygon, which can be used to determine positions of the inner control points of a patch.

Equation 2.1 is used to determine control points so that a cubic patch can be joined smoothly to an auxiliary patch known as a basis patch. Consider the situation depicted in Figure 2.7, where an actual patch Φ_a is connected to a basis patch Φ_b. Vectors $\mathbf{c}_i$ constitute the edges of the cubic control polygon on the common boundary, and $\mathbf{a}_i$ and $\mathbf{b}_i$ belong to Φ_a and Φ_b, respectively. $\mathbf{a}_0$ and $\mathbf{a}_3$ are unit vectors orthogonal to $\mathbf{c}_0$ and $\mathbf{c}_2$, respectively, and lie in the tangent plane of the normal defined at the vertices $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$, respectively. In case of the actual patch being triangular, the first row of control points from the boundary are quartic control points obtained by degree elevation. Equation 2.1 is then rewritten in terms of scalar functions $k(u)$ and $h(u)$:

$$\frac{\partial}{\partial v}\Gamma_a(u) = k(u)\frac{\partial}{\partial v}\Gamma_b(u) + h(u)\frac{\partial}{\partial u}\Gamma(u),$$

so that the actual patch is joined with G^1 continuity to the basis patch. To solve this equation, first express $\mathbf{a}_0$ and $\mathbf{a}_3$ as

$$\mathbf{a}_0 = k_0\mathbf{b}_0 + h_0\mathbf{c}_0, \quad \mathbf{a}_3 = k_1\mathbf{b}_3 + h_1\mathbf{c}_2, \quad k_0, k_1, h_0, h_1 \in \mathbb{R},$$

and set

$$k(u) = (1-u)k_0 + uk_1, \quad h(u) = (1-u)h_0 + uh_1.$$

It is then assumed that $\partial\Gamma_b(u)$ on the basis patch is only quadratic and that vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ can be determined by linear interpolation $\mathbf{b}_1 = \frac{2}{3}\mathbf{b}_0 + \frac{1}{3}\mathbf{b}_3$, $\mathbf{b}_2 = \frac{1}{3}\mathbf{b}_0 + \frac{2}{3}\mathbf{b}_3$. Finally, vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ can be determined as

$$\mathbf{a}_1 = (k_1-k_0)\frac{\mathbf{b}_0}{3}+k_0\mathbf{b}_1+2h_0\frac{\mathbf{c}_1}{3}+h_1\frac{\mathbf{c}_0}{3}, \quad \mathbf{a}_2 = k_1\mathbf{b}_2-(k_1-k_0)\frac{\mathbf{b}_3}{3}+h_0\frac{\mathbf{c}_2}{3}+2h_1\frac{\mathbf{c}_1}{3}.$$

The connection to a basis patch can be done without having explicit knowledge of all control points of adjacent patches; the construction relies only on data from the common boundary. The method is applied to both sides of the boundary individually such that there are two basis patches, one on either side of the boundary. The control points determined for the actual patches are positioned in such a way that the actual patches join with G^1 continuity with the adjacent basis patches. The basis patches themselves join each other with C^1 continuity and therefore the actual patches join each other with G^1 continuity, as desired.

2.3.4 *Gregory Patches*

The Gregory patch is a generalisation of Gregory's ideas [Gre74] to tensor product Bézier surfaces [Chi86]. Consider the control point layout of Figure 2.8, left. The method of Chiyokura and Kimura is applied to all four boundary edges to obtain 8 inner control points (shown in red), one pair per corner. Here, each pair of control points (possibly) denotes a different twist vector. This is known as the twist compatibility problem and rational blending functions can be used to solve it. The functions blend the four pairs of inner control points so that they are interpolated along straight line segments. In particular,

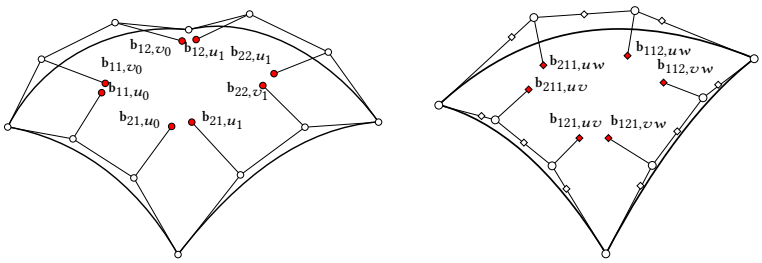


Figure 2.8: *The control net of a quadrilateral Gregory patch (left) and the control net of a triangular Gregory patch (right). The diamonds are quartic control points obtained by degree elevating cubic control points. Red points are the inner control points as constructed by the method of Chiyokura and Kimura.*

$$\mathbf{b}_{11} = \frac{v\mathbf{b}_{11,u_0} + u\mathbf{b}_{11,v_0}}{u+v}, \quad \mathbf{b}_{21} = \frac{(1-v)\mathbf{b}_{21,u_0} + u\mathbf{b}_{21,v_1}}{(1-v)+u},$$
$$\mathbf{b}_{12} = \frac{v\mathbf{b}_{12,u_1} + (1-u)\mathbf{b}_{12,v_0}}{v+(1-u)}, \quad \mathbf{b}_{22} = \frac{(1-v)\mathbf{b}_{11,u_0} + (1-u)\mathbf{b}_{11,v_0}}{(1-u)+(1-v)}.$$

These blending functions are designed such that the blended control points are in the correct position when the patch is evaluated on an edge and thus the correct cross-boundary derivative is maintained.

The same ideas can be extended also to triangular Bézier patches [Lon85] such that a triangular Gregory patch is constructed. Consider the layout in Figure 2.8, right, of a triangle with cubic boundary curves, and the 6 points (shown in red) generated by using the method on the boundary curves after degree elevation from cubic to quartic. The degree elevation step is required to match the cubic cross-boundary derivative, corresponding to a quartic triangular Bézier patch, constructed by the method. These points describe the three inner control points of a quartic Bézier triangle. Again, pairs of these points are blended rationally:

$$\mathbf{b}_{211} = \frac{(1-w)v\mathbf{b}_{211,uv} + (1-v)w\mathbf{b}_{211,uw}}{(1-w)v+(1-v)w},$$
$$\mathbf{b}_{121} = \frac{(1-w)u\mathbf{b}_{121,uv} + (1-u)w\mathbf{b}_{121,vw}}{(1-w)u+(1-u)w},$$
$$\mathbf{b}_{112} = \frac{(1-u)v\mathbf{b}_{112,vw} + (1-v)u\mathbf{b}_{112,uw}}{(1-u)v+(1-v)u}.$$

Both techniques contain simple singularities in the parametrisation at the vertices, but they can easily be handled by analysing the parameter values.

2.3.5 *Generalised Bézier Patches*

Generalised Bézier patches [VSK16], or GB patches, are multisided control structures that combine transfinite and control-point-oriented structures. Generalised barycentric coordinates (see Section 2.5) are used to define local parameters and blending functions. The resulting patches and structure are loosely similar to tensor product Bézier surfaces and they inherit many of their properties.

The patch is a combination of n Bézier ribbons. On each side of the patch a degree d Bézier ribbon is defined by taking into account l rows of $d+1$ control points with $l = (d+1) \bmod 2$. For each ribbon, bivariate tensor-product Bernstein polynomials are used:

$$\mathbf{S}_i^d(s_i, h_i) = \sum_{j=0}^{d} \sum_{k=0}^{l} \mu_{jk}^i \mathbf{b}_{jk}^i B_{jk}^d(s_i) B_{jk}^d(h_i).$$

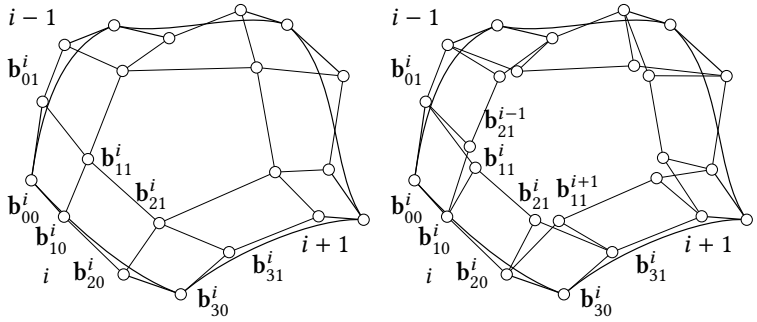


Figure 2.9: *The control net of a cubic generalised Bézier patch. The labelling of the control points corresponding to one side is given, the rest is labelled analogously.*

For ribbon i of a patch, which corresponds to the edge spanned from vertex $\mathbf{v}_{i-1}$ to $\mathbf{v}_i$, local parameter functions can be defined using generalised barycentric coordinates $\boldsymbol{\phi}$:

$$s_i = \frac{\phi_i}{\phi_i + \phi_{i-1}}, \quad h_i = 1 - \phi_i - \phi_{i-1},$$

where s_i is known as the side parameter and h_i as the distance parameter. They are constructed so that they have the same behaviour as ordinary bilinear coordinates. s_i varies from 0 to 1 on side i, and h_i is 0 exactly on side i but increases on the interior of a patch and linearly on sides $i+1$ and $i-1$. $\mathbf{b}^i_{jk}$ are the control points as oriented from side i. The labelling of control points of a single ribbon is given in Figure 2.9. The control points of the remaining sides are labelled analogously and control points in the corners are shared between ribbons. μ^i_{jk} is a scalar function that serves the purpose of assuring locality of the influence of each side, as well as blending control points which are shared by multiple sides such that they are weighted by a linear combination of Bernstein polynomials. It is defined for $d \leq 3$ as

$$\mu^i_{jk} = \begin{cases} \frac{h_{i-1}}{h_i + h_{i-1}} & j < 2 \\ 1 & 2 \leq j \leq d-1 \\ \frac{h_{i+1}}{h_i + h_{i+1}} & j > d-1 \end{cases}. \tag{2.6}$$

For higher degrees, these functions have to deal with certain other constraints [VSK16].

Adding all contributions results in a patch definition that is not necessarily affine invariant, meaning that the weighted Bernstein polynomials of all the sides may not sum to unity. This can be easily fixed by

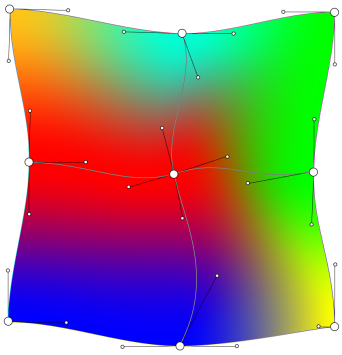


Figure 2.10: *A simple* 2×2 *gradient mesh smoothly interpolating colours and gradients defined at the vertices* $\mathbf{v}_{ij}$ *of the mesh. Gradient handles are denoted by line segments emanating from their corresponding vertices, which are denoted as white circles.*

augmenting the patch structure with another central control point C and its associated blending function

$$B_0^d = 1 - \sum_{i=1}^{n} \sum_{k=1}^{l} \sum_{j=0}^{d} \mu_{jk}^i B_{jk}^d(s_i) B_{jk}^d(h_i).$$

2.3.5.1 *Generalised Gregory Patches*

The Gregory generalised Bézier patch [HK18] or generalised Gregory patch is a simple extension of the ordinary GB patch that generalises the quadrilateral Gregory patch (Section 2.3.4). The boundary curves and ribbons are given as cubic Bézier curves and surfaces, respectively. It removes the need for twist-compatibility of the adjacent Bézier ribbons within a patch such that the patches can join with tangent-plane (G^1) continuity around a vertex with any valency. It does so by removing the requirement that adjacent tangent ribbons explicitly share their inner control points in the first row from the boundary. By allowing this, the patches are able to express different twist vectors at the vertices, and the blending functions μ_{jk}^i then ensure that the control points in patch corners are blended much like the rational blending found in the original quadrilateral version of the Gregory patch [CK83].

2.3.6 *Gradient Meshes*

The gradient mesh primitive is a vector graphics primitive which is used to smoothly interpolate colours defined at the vertices of a regular quadrilateral mesh. Gradient handles defined at vertices can be used to distort the boundary curves of each of the patches of the mesh into the

desired shape. Since colour is interpolated along with the geometry, the gradient handles dictate the spread of colour inside the patches.

Traditionally, a Ferguson patch [Fer64] is the underlying geometric primitive used for a gradient mesh. However, since it is a bicubic patch we can also represent each (quadrilateral) patch as a bicubic Bézier patch instead; see [BLHK18] for an in-depth discussion. The vertices $\mathbf{v}_{ij}$ are defined as the quintuple (x, y, r, g, b), where the first two components represent position, and the last three control its (RGB) colour. The edges of each patch are given by cubic Bézier curves. In the case of the Bézier patch, the gradient handles and the inner control points inherit the colour value (r, g, b) of the logically closest vertex. In this way, colour is interpolated along with the geometry, and, most importantly, the interpolated colour values will not traverse out of the colour gamut. A simple gradient mesh consisting of 4 patches interpolating colours and gradients is shown in Figure 2.10.

2.4 B-SPLINES CURVES AND SURFACES

B-splines refer to a family of piecewise polynomial functions. A B-spline is a function consisting of multiple polynomial pieces that join each other at knots. Although constructed out of multiple pieces the functions are very smooth. A degree d B-spline is in general $d - 1$ smooth. The use of B-splines in parametric functions creates a very powerful means to define curves and surfaces. The high smoothness of the basis functions creates parametric representations that are just as smooth. Like the Bézier variants, they are defined in terms of control points and basis functions.

The B-spline basis functions can be constructed using the following recurrence relation [Cox72]

$$B_{i,0}(t) = \begin{cases} 1 & t_i \leq t \leq t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{i,k}(t) = \frac{t - t_i}{t_{i+k} - t_i} B_{i,k-1}(t) + \left(1 - \frac{t - t_{i+1}}{t_{i+1+k} - t_{i+1}}\right) B_{i+1,k-1}(t),$$

where t_i are the knot values which determine the foot points of the basis functions and $[t_0, \ldots, t_n]$ is the knot sequence. In this thesis we use mostly uniform knot sequences where B-splines are considered. When the relative spacing between knots is irregular the B-splines are known as non-uniform B-splines. Uniform B-splines have knot vectors with equidistant nodes and for brevity we omit the specification of them from now on.

Looking at function $B_{0,d}$ we have defined a single B-spline function or basis function. Other functions like $B_{1,d}$ are simply shifted copies of the

same function. Figure 2.11, left, shows different degree B-spline basis functions on the unit interval. As can be seen the support of the functions increases with the degree. The basis functions have some useful properties

- A degree d basis function has $d - 1$ vanishing derivatives at the ends of their support.
- A degree d basis function can be decomposed into $d + 1$ polynomial pieces.
- A degree d basis function has C^{d-1} continuity at non-multiple knots t_j.
- The functions partition unity everywhere on a knot span $[t_j, t_{j+1}]$.

A B-spline curve is defined by blending many kernels, which are attached to control points, together. Consider the control points $\mathbf{P}_0, \ldots \mathbf{P}_n$. We can attach a B-spline basis function $B_{i,d}(t)$ to each of the control points $\mathbf{P}_i$ to define the curve

$$\mathbf{C}^d(t) = \sum_{i=0}^{n} B_{i,d}(t)\mathbf{P}_i,$$

where $t \in [0, n]$. From the properties of the B-spline basis functions we can find that the curve has the following properties

- The curve is piecewise smooth polynomial and C^{d-1} at knots t_i.
- The curve lies in the convex hull of all $\mathbf{P}_i$.
- Control point $\mathbf{P}_i$ has influence on the spline at interval $[t_i, t_{i+d}]$.

Like the Bézier curve, the B-spline curve has a simple definition using only basis functions and control points. The difference between the Bézier curve and a B-spline curve is that it does not in general interpolate any of the control points. Still it remains intuitive to manipulate the curve through manipulation of the control points. B-spline curves require less control points to define a much smoother curve, but do not provide intuitive control over tangents. The control points only locally influence the shape of the curve as determined by the degree of the basis functions, whereas adjusting control points of a Bézier curve influences the whole curve, up until the endpoints.

A tensor product surface can also be created using B-spline basis functions

$$\mathbf{S}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,d}(u) B_{j,d}(v) \mathbf{P}_{i,j}.$$

This allows for defining smooth surfaces through a grid of control points, which inherit the properties from the B-spline curves.

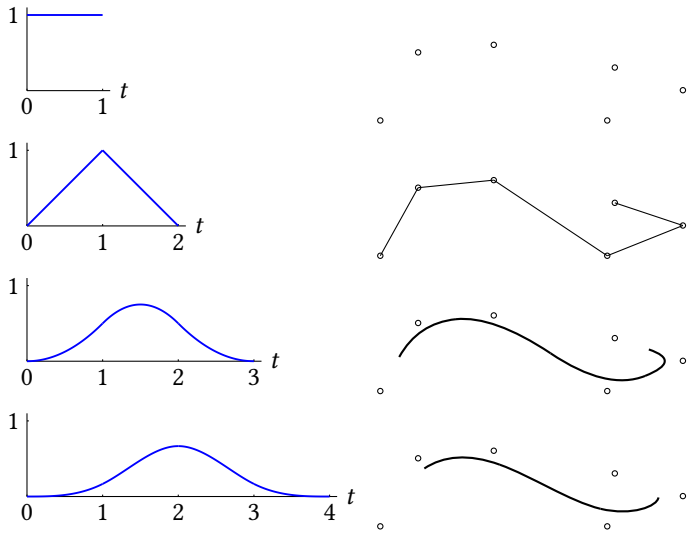


Figure 2.11: *From top to bottom, the constant, linear, quadratic and cubic B-splines. Left: B-spline basis functions, right B-spline curves constructed with their respective basis functions using the same control polygon.*

2.4.1 *Subdivision Surfaces*

Subdivision curves and surfaces are generated by an iterative refinement process that creates a smooth surface from an initial coarse polygon or polyhedron. They provide an alternative means to evaluate B-spline surfaces, but generalise them also to surfaces of arbitrary topology, meaning that they are able to generate surfaces from control meshes that contain extraordinary faces and extraordinary vertices. The structure of subdivision algorithms, the process of generating a surface from an initial control mesh, is derived from the knot insertion algorithm for B-splines. The knot insertion algorithm shows how to insert new control points in the control polygon or mesh without changing the actual curve or surface that it generates. The relations between newly generated points and existing points can then be generalised to encompass situations wherein the control mesh is not regular. This is the place where subdivision surfaces truly shine, as they are capable of generating smooth surfaces in these areas.

Doo and Sabin [DS78] provided one of the first of such schemes that generalised biquadratic B-splines to surfaces of arbitrary topology. In the same year also a generalisation of bicubic B-splines was created by Catmull and Clark [CC78], which we detail in the next subsection. Both these techniques generalise from quadrilateral elements to arbitrary topology, but it can also be done for triangular based B-splines.

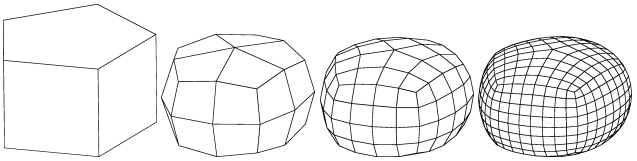


Figure 2.12: *Three steps of Catmull-Clark subdivision applied to an arbitrary topology object containing both extraordinary faces (pentagons) and extraordinary vertices (vertices of valency 3).*

Loop-subdivision [Loo87] generalises quartic box-splines [DBHR93] to arbitrary triangular meshes.

2.4.1.1 *Catmull-Clark*

Catmull-Clark subdivision is arguably one of the most famous and widely used subdivision surface algorithms. It has been the standard of the animation industry [DKT98] for displaying high quality surfaces that can be easily animated. Catmull-Clark subdivision generalizes uniform cubic B-splines to surfaces of arbitrary topology. The evaluation of the subdivision surface is done through a simple recursive algorithm

- Face points $\mathbf{f}_i$ are inserted as the centroids of the faces.
- Edge points $\mathbf{e}_i$ are inserted as the average of the midpoints of the edge and the average of the two adjacent face points.
- Existing vertex $\mathbf{v}_i$ points are updated to be $\frac{F+2E+(n-3)\mathbf{v}_i}{n}$, where F and E are the average of all n face points and n edge points, respectively, surrounding the vertex.

In the limit this process will converge to a smooth surface. The process of four subdivision steps is shown in Figure 2.12. For regular regions this reproduces uniform cubic B-splines and is C^2 continuous. In irregular regions the surface will be G^1 at extraordinary points. The process of subdivision is a global process, but there are means to speed it up by evaluating the regular regions as bicubic B-spline surfaces and the approximating the irregular regions with bicubic Bézier patches [LS08a] or Gregory patches [LSNC09].

2.4.1.2 *Arbitrary Degree Subdivision*

Catmull-Clark subdivision generalises only uniform cubic B-splines. The generalisation of arbitrary degree subdivision surfaces to arbitrary topology can be achieved by restating the subdivision surfaces in terms of a Lane-Riesenfeld split-averaging scheme [LR80].

For surfaces the split operation is generalised into *linear subdivision*, edges and faces are split at their midpoints and connected and the *dual*

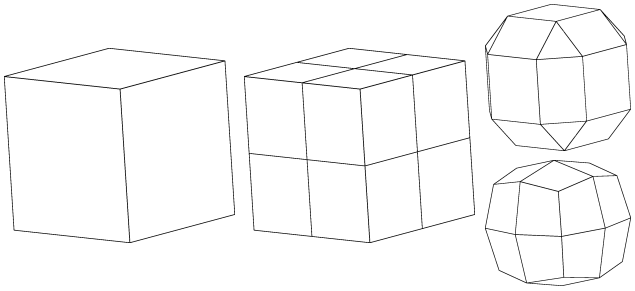


Figure 2.13: *The arbitrary degree subdivision process. The original cube (left) is linearly subdivided (middle) and subsequently, either the dual is computed (right top) to obtain a quadratic subdivision steps from which higher degrees can be obtained by more applications of the even smoothing step, or the linear subdivision is twice odd smoothed (right bottom) to result in a cubic subdivision step.*

mesh operation, whereby the centroid is created for each face and new faces are created by connecting all centroids surrounding original vertices, serves as an averaging step. By applying the linear subdivision step followed by $d-1$ dual mesh operations, any d degree B-spline surface can be reproduced and applied to regions or arbitrary topology [ZS01]. This iterative process works, but can be simplified computationally as the dual mesh operation is quite a heavy operation, due to the changes in topology.

Therefore, Stam [Sta01] simplified the averaging operation by creating an *odd smooth* and *even smooth* steps that do not topologically change the current mesh. The odd smooth step updates positions by a weighted average of surrounding vertex positions, that in the regular case reproduces B-spline stencils. Odd-degree B-spline surfaces then are reproduced by one linear subdivision step followed by $(d-1)/2$ odd smoothing steps. The even smooth step replaces each vertex position with the average of the surrounding face centroids. Then even degree subdivision surfaces are reproduced by a linear subdivision step and a dual mesh operation followed by $(d-2)/2$ even smoothing steps. This means that Doo-Sabin subdivision is already reproduced after the linear and dual mesh operations. Figure 2.13 shows the steps involved in arbitrary degree subdivision for both the odd-degree and even-degree situations.

2.5 GENERALISED BARYCENTRIC COORDINATES

Parametrisation is useful for many different tasks. It allows you to define functions over domains or to interpolate values over regions. Triangular elements are easily parametrised by barycentric coordinates, and for quadrilateral elements a bilinear coordinate system suffices. This

parametrisation suffices to interpolate between three or four samples, such as over triangles or quadrilaterals. For polygons with more vertices than three or four, there exists the option to subdivide the face into triangular or quadrilateral elements, but this complicates the problem of smoothly interpolating over the original polygonal domain. The generalisation of barycentric coordinates to polygons of any valency provides a better mechanism to do this.

A generalised barycentric coordinate system provides a way to express a point on a planar polygon as a weighted combination of the polygon's vertices. Consider a planar polygon ω with vertices $\mathbf{v}_i$, $i = 1, \ldots, n$ and a point $\mathbf{p}$ on ω, see Figure 2.14. Then barycentric coordinate functions ϕ_i (or $\phi_i(\mathbf{p})$) can be determined such that they have the following properties.

- Non-negativity: $\phi_i \geq 0$
- Partition of unity: $\sum_{i=1}^{n} \phi_i = 1$
- Linear reproduction: $\sum_{i=1}^{n} \phi_i(\mathbf{p})\mathbf{v}_i = \mathbf{p}$
- Lagrange property: $\phi_i(\mathbf{v}_j) = \delta_{ij}$, where δ_{ij} is the Kronecker delta.
- Linearity on the boundary: $\phi_i = 1 - \phi_{i+1}$ on $\mathbf{v}_i\mathbf{v}_{i+1}$

In the rest of the thesis we will denote with $\boldsymbol{\phi}$ the vector $\begin{pmatrix}\phi_1, & \ldots, & \phi_n\end{pmatrix}$ of generalised barycentric coordinates. For triangles the coordinates are uniquely defined. This does not hold for polygons with higher valencies, leading to multiple ways of computing generalised barycentric coordinates from a point in the polygon. In this thesis we use two well known forms of generalised barycentric coordinates, Wachspress coordinates and mean value coordinates.

2.5.1 *Wachspress Coordinates*

Wachspress coordinates were developed by Wachspress [Wac75] and were arguably the very first form of generalised barycentric coordinates. Wachspress coordinates are defined using signed triangle areas constructed from $\mathbf{p}$ and three subsequent vertices

$$\phi_i = \frac{w_i(\mathbf{p})}{\sum_j w_j(\mathbf{p})}, \tag{2.7}$$

where $w_i(\mathbf{p})$ is defined as:

$$w_i(\mathbf{p}) = \frac{A(\mathbf{v}_{i-1}, \mathbf{v}_i, \mathbf{v}_{i+1})}{A(\mathbf{v}_{i-1}, \mathbf{v}_i, \mathbf{p})A(\mathbf{v}_{i+1}, \mathbf{v}_i, \mathbf{p})}, \tag{2.8}$$

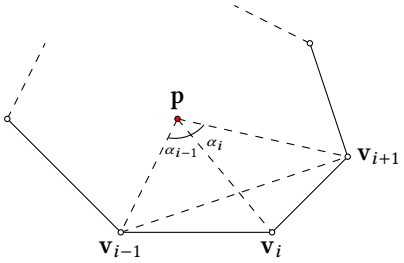


Figure 2.14: *The different triangles constructed with respect to* $\mathbf{p}$ *and* $\mathbf{v}_i$ *on a sector of a polygon.*

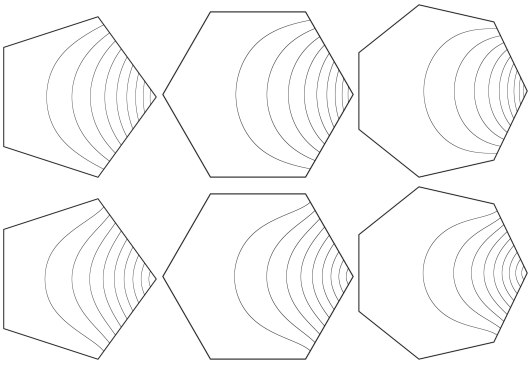


Figure 2.15: *The isolines for* $\phi_1 = j/10, j = 1, \ldots, 10$ *on a regular pentagon, hexagon and heptagon. The top row shows Wachspress coordinates and the bottom row mean value coordinates.*

where $A(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k)$ describes the signed area of the triangle $\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k$. It can be seen in Figure 2.14 how the different triangle areas are constructed. The w_i are normalised to obtain the final generalised barycentric coordinates. The coordinates have limited use because they only work well for convex polygons. However, as we shall see in the rest of this thesis most arbitrary polygons, even those in 3D, can be effectively parametrised using an equivalent regular domain. For this purpose Wachspress coordinates are very effective as they can be computed very efficiently.

2.5.2 *Mean value coordinates*

Mean value coordinates were developed by Floater [Flo03]. They are well defined for a wide range of polygon shapes, most notably for polygons which are not convex [HF06]. Mean value coordinates are also constructed from three consecutive vertices of the polygon. The weights per vertex are this time calculated with:

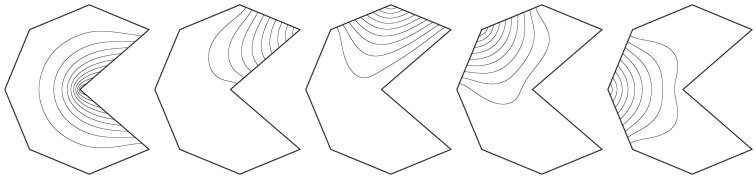


Figure 2.16: *The isolines for* $\phi_i = j/10, j = 1, \ldots, 10$ *on a non-convex octagon for vertices* $\mathbf{v}_i$

$$w_i = \frac{\tan(\alpha_i/2) + \tan(\alpha_{i-1}/2)}{\|\mathbf{p} - \mathbf{v}_i\|}, \tag{2.9}$$

where α_i and α_{i-1} are the angles as shown in Figure 2.14. The weight of the vertex is then dependent on the angle between the point relative to the polygon vertices. The advantage over Wachspres coordinates is that mean value coordinates are well defined outside the polygon and they are even smooth over polygon edges, but not at vertices. However, non-convexity of the polygon means that some of the coordinate functions ϕ_i might become negative. Figure 2.16 shows several of the isolines of the coordinate functions of the vertices of a non-convex octagonal polygon.

2.6 CONCLUSION

Some of the techniques described in this chapter are still only defined for triangular or quadrilateral elements, like gradient meshes or B-spline surfaces, with only regular topology. In these cases we are unable to use them with arbitrary topology meshes as they are either not able to be smoothly defined or cannot even be defined at all. Many of such techniques can be extended to work as multisided versions through the use of generalised barycentric coordinates. In this thesis we will show how we can extend the gradient mesh from a quadrilateral rectangular topology into a polygonal arbitrary topology. Likewise B-spline surfaces are extended so that they can be spanned over extraordinary regions.

Part I

Geometric Design

3 INTRODUCTION

Geometric design or modelling is the study of how shapes can be defined mathematically. There are many ways to define a three dimensional shape or object. The most basic way to define an object is through point clouds (Figure 3.1, top left). A collection of individual points in space, that when viewed as a whole convey an object. There is no apparent relation between the points, but the overall object could be easily determined by human inference when viewed from the right perspective. The shape of the object is not well represented as it is represented discontinuously (C^{-1}).

The concept of images can be extended to higher dimensions to obtain a voxel-based representation of shapes (Figure 3.1, top right). Like the pixel variant, a voxel-based representation can convey a shape fairly well when the resolution is high enough, but high resolutions voxel representations require a lot of storage as the whole volume needs to be described, even the empty space.

A polygon mesh provides a simple way to state where an object exists and also how the points on its surface are related to each other. The mesh is a collection of vertices, faces and edges, that collectively define the shape. A dense enough polygonal mesh is able to define high amounts of detail and convey a surface that appears smooth (Figure 3.1, bottom left). However, handling and designing such dense meshes requires a lot of work. Manipulating a mesh by moving a single vertex at a time will take tremendous time to edit a surface. Modern manipulation and design workflows such as sculpting [Spe11] can manipulate dense meshes more effectively through 3D brushes that manipulate a group of vertices at a time. Although a polygonal mesh is able to approximate smooth surfaces, it remains only a C^0 discretisation of that surface. The individual polygons remain flat.

Splines are a practical way to define an object. Instead of having a very dense representation containing a large number of surface samples, such as with dense polygonal meshes or voxel representations, the surface is represented by a mesh of control points (Figure 3.1, bottom right). Most of the control points do not actually lie on the surface that they define, but they influence a region of the surface of the object. The surface can be generated from the control points by various procedures, such as iterative refinement or using parametric surfaces. The resulting spline surfaces are smooth or piecewise smooth surfaces and provide a compact way to do so. Designing by means of spline surfaces is effortless as a control point influences a large part of the surface. By manipulating the control points a surface can be bent and twisted into shape.

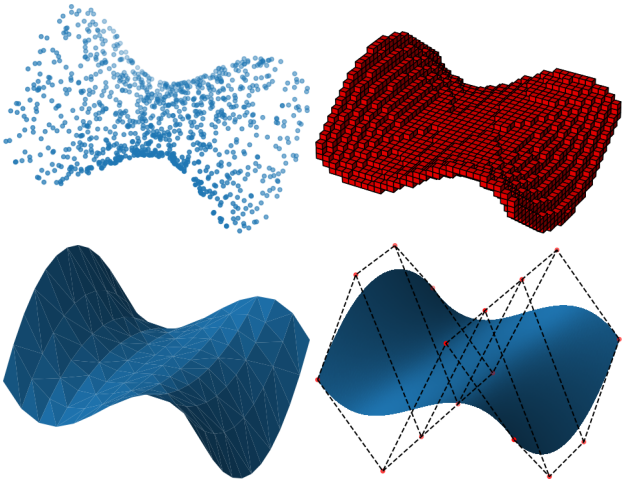


Figure 3.1: *Four different representations of the same surface. Top left: point cloud, Top right: voxels , Bottom left: triangle mesh, Bottom right: Bézier patch with control mesh.*

Visualisation of splines is usually achieved through rendering a polygonal representation that was obtained through a discretisation procedure. This procedure samples the spline surface at a large number of points and in turn connects them into a polygonal mesh. By having a dense enough sampling, a smooth surface can be approximated almost exactly. The surface could be visualised through other means such as raytracing, but the higher-order parametric nature of splines makes ray-surface intersections a costly endeavour [BS93].

Spline surfaces can easily define smooth surfaces, but most are restricted in that they can only exist in a few topologies. Spline surfaces themselves are regular surfaces and to effectively model an object it is often necessary that it should be defined as a collection of spline patches. This poses a couple of challenges, as the spline surfaces have to be fit together. The problem of joining together spline surfaces smoothly might seem simple, but it is deceptively hard. Each of the spline surfaces has its own parametrisation and when arranging multiple of these surfaces together their parametrisations might not align. The challenge exists in using just enough of the data that is shared between neighbouring patches to create smooth transitions.

This part of the thesis investigates two different ways of joining splines surfaces together. In Chapter 4, we look at how arbitrary topology NURBS models, that initially are not even C^0, can be converted into triangular splines that subsequently can be joined together smoothly. Then in Chapter 5, we show how B-spline surfaces of arbitrary topology can be constructed by creating multisided spline patches and how general multisided surfaces can be rendered efficiently.

4 CONVERTING B-REP CAD MODELS INTO TANGENT PLANE CONTINUOUS SPLINES

Parts of this chapter have been published as

- Gerben J. Hettinga, and Jiří Kosinka. "Conversion of B-rep CAD models into globally G^1 triangular splines".

Existing techniques that convert B-rep (boundary representation) patches into Clough-Tocher splines guarantee watertight, that is C^0, conversion results across B-rep edges. In contrast, our approach ensures global tangent-plane, that is G^1, continuity of the converted B-rep CAD models. We achieve this by careful boundary curve and normal vector management, and by converting the input models into Shirman-Séquin macro-elements near their (trimmed) B-rep edges. We propose several different variants and compare them with respect to their locality, visual quality, and difference with the input B-rep CAD model. Although the same global G^1 continuity can also be achieved by conversion techniques based on subdivision surfaces, our approach uses triangular splines and thus enjoys full compatibility with CAD.

4.1 INTRODUCTION

Traditionally computer aided design (CAD) systems present CAD models in terms of their outer shells rather than as solid geometry. These models are called boundary representations or B-reps for short. The representation of the boundary is done using non-uniform ration B-splines or NURBS [PT95]. A NURBS curve is a rational B-spline which can have non-uniform knots:

$$C^d(t) = \frac{\sum_{i=0}^{n} w_i B_{i,d}(t)\mathbf{P}_i}{\sum_{i=0}^{n} w_i B_{i,d}(t)}$$

and can be extended to surfaces by using a tensor-product surface structure and two sets of knots for each parametric direction. NURBS are capable of exactly representing a range of shapes such as conic arcs in the curve case and quadric patches when considering bivariate tensor-product NURBS surfaces. This tensor product structure of the surface only allows NURBS to model a small set of shapes that are topologically equivalent to disks, cylinders or tori. To create more complex geometry, multiple NURBS patches have to be stitched together.

It can often be difficult or impossible to arrange multiple NURBS patches together to form a watertight representation. Usually CAD models are made through the application of several Boolean operations on an initially simple object. The simple objects can be represented exactly as NURBS, but after Boolean operations only certain sections of the original surface remain. The patch can be said to have been *trimmed*, the excess parts have been cut off and only the relevant part of the surface is used. The process of trimming is done in the parameter space of the patches. Consider a tensor product NURBS patch $F(u, v)$, we can trim the patch along the the trimming curve $C^t(u, v)$ defined on the u, v parameter space of the NURBS patch. Then only the part of the surface which is in the region denoted by the trimming curve is used. Figure 4.1 shows the process of trimming and the effect it has on the patch.

Trimming allows to create arbitrary shapes out of an otherwise rectangularly defined patch by imposing trimmed shapes on the parameter domain. One can for instance draw any polygonal shape or curve in the

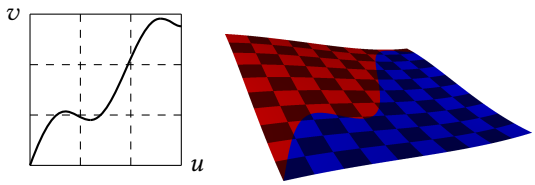


Figure 4.1: *The trimmed uv parameter space (left) of the B-spline patch (right). The trimming curve divides the surface into a trimmed and untrimmed portion.*

parameter domain. The freedom in defining the shape of the patches does not make it easier to compose them together. That is why often the trimming curve is the result of the computed intersection of two or more NURBS patches. Again, this process is not trivial, as the intersection curve of two NURBS patches is not, in general, a NURBS curve. The intersection curve can be approximated, which leads to two trimming curves in the parameter domains of the individual patches. The approximation of the trimming curve leads to non-watertightness, i.e., not even C^0 continuity is easily achieved. The resulting gaps can be kept below well-chosen small tolerances and pose no major problems in the manufacturing process of CAD models. However, for analysis and simulation it may lead to major issues [MH18].

A solution often taken in the finite element field is to process the NURBS patches as a piecewise linear and watertight approximation. However, the rapidly progressing field of isogeometric analysis (IgA) [CHB09] leverages the exact geometry domain of the patches and uses the same basis functions to describe the geometry to also span the solution space. Over time there have been various geometric conversion methods to turn CAD models into higher-order analysis-suitable representations. However, existing conversion methods suffer from one of the following two shortcomings. The resulting global continuity is not better than C^0 [KC15, MvSE18], or the representation of choice is not directly CAD compatible [SKSD14, SKSD16].

Our approach converts B-rep CAD models into globally tangent-plane G^1 continuous triangular splines. Our triangular splines are based on Clough-Tocher [CT65] and Shirman-Séquin [SS91] macro-elements, and lead to CAD compatible converted surfaces as it is only defined using cubic and quartic Bézier triangles. Visually our approach does not increase substantially on the C^0 method of the [KC15], our approach has the added advantage of exact G^1 smoothness. The resulting converted surfaces can be used in downstream applications such as analysis and simulation, a concrete example being enhanced finite element methods [SFMH11].

4.2 RELATED WORK

The handling of CAD models that are represented using trimmed and stitched NURBS patches has prompted a lot of investigations overtime because of the difficulty of handling the trimmed regions [MH18]. This has led to many different representations and conversion techniques.

These conversions are either exact, meaning that the geometry remains intact but also non-watertight, or approximate. The former class of exact techniques is represented by untrimming [EK14, MvSE18]. While this ensures that the converted patches are no longer trimmed, the continuity of the CAD model inevitably does not change.

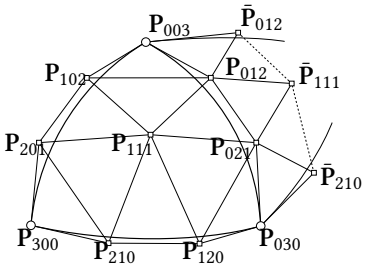


Figure 4.2: *The control net of a cubic triangular Bézier patch P with control point labels. A section of the control net of an adjacent C^0-connected patch $\bar{P}$ is also shown; control points $\mathbf{P}_{003}$, $\mathbf{P}_{012}$, $\mathbf{P}_{021}$, and $\mathbf{P}_{030}$ are shared.*

The latter class of approximate techniques necessarily modify the input geometry, but the resulting conversion error is typically controllable and often confined to regions near the B-rep edges (trimming curves). Existing techniques include conversion to Catmull-Clark [SKSD14] and Loop [SK16] subdivision surfaces. An improved variant in which regions away from the trimming curves can be kept unmodified was recently presented in [SKSD16]. These techniques offer globally G^1 approximations, but are not directly CAD-compatible due to the lack of closed-form representations in irregular regions. A closely related method converts CAD models into T-splines [SFL+08], but again, T-splines rely on subdivision or approximate G^1 conditions near irregular regions.

A promising approach to tackle the trimming problem is based on watertight Boolean operations [UMC+19]. While it produces watertight models, the placement of extraordinary points and feature lines in the method remains a manual task.

CAD models can be converted into collections of curved triangles. This approach was taken in [XQ17], which also offers a volumetric counterpart. However, their construction is based on the Clough-Tocher-Hsieh split [CT65] and parametric continuity, and thus produces globally only C^0 results. An earlier technique based on Clough-Tocher splines appeared in [KC15]. This offers CAD-compatible patches, but in general results in only C^0 continuity across but also near B-rep edges.

To remedy this, we present a technique based on polynomial Bézier triangles and the Shirman-Séquin construction [SS91] to enhance existing Clough-Tocher techniques such that globally tangent-plane continuous models are obtained.

4.3 PRELIMINARIES

We start by introducing basic concepts such as the continuity conditions between Bézier triangles (see Section 2.3.2). Then we discuss the Clough-Tocher construction and the Shirman-Séquin construction.

4.3.1 *Continuity Conditions*

Consider the situation sketched in Figure 4.2. The C^1 continuity conditions between two cubic Bézier triangles P and $\bar{P}$ and their respective triangles in parameter space $\mathcal{T}(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$ and $\bar{\mathcal{T}}(\mathbf{v}_1, \mathbf{v}_3, \mathbf{v}_2)$ are, along with the trivial C^0 conditions of shared edge control points, given by

$$\begin{aligned} \bar{\mathbf{P}}_{012} &= \tau_0 \mathbf{P}_{102} + \tau_1 \mathbf{P}_{012} + \tau_2 \mathbf{P}_{003}, \\ \bar{\mathbf{P}}_{111} &= \tau_0 \mathbf{P}_{111} + \tau_1 \mathbf{P}_{021} + \tau_2 \mathbf{P}_{012}, \\ \bar{\mathbf{P}}_{210} &= \tau_0 \mathbf{P}_{120} + \tau_1 \mathbf{P}_{030} + \tau_2 \mathbf{T}_{021}, \end{aligned} \tag{4.1}$$

where (τ_0, τ_1, τ_2) are the barycentric coordinates of $\mathbf{v}_3$ with respect to $\mathcal{T}$, i.e., $\mathbf{v}_3 = \tau_0 \mathbf{v}_0 + \tau_1 \mathbf{v}_1 + \tau_2 \mathbf{v}_2$. In other words, the three quadrilaterals $(\mathbf{P}_{102}, \mathbf{P}_{012}, \bar{\mathbf{P}}_{012}, \mathbf{P}_{003})$, $(\mathbf{P}_{111}, \mathbf{P}_{021}, \bar{\mathbf{P}}_{111}, \mathbf{P}_{012})$, and $(\mathbf{P}_{120}, \mathbf{P}_{030}, \bar{\mathbf{P}}_{210}, \mathbf{P}_{021})$ have to be planar and of the same affine shape as the quadrilateral $(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_3, \mathbf{v}_2)$ in parameter space.

4.3.2 *Triangulation of B-rep Patches*

The B-rep patches are triangulated through a triangulation of the parametric domain of the patch. The triangulation of the parametric domain maps to a corresponding triangular mesh defined on the B-rep where the 3D vertex positions lie exactly on the B-rep patch. There are many methods that can be used to construct this triangulation such as [TOC98, BDL09, SBAD16]. The triangulation can be tuned so that it meets certain criteria such tolerances on the triangle shape and approximation quality. All example models used in this chapter were created using the CADfix product [Int19], that creates the triangulation from an imported CAD model. The CADfix triangulation process is similar to the approach taken in [KBT04], which incrementally creates a triangulation based on a constrained Delaunay triangulation where the constraints are the trimming curves of each patch.

The triangulation has vertex positions $\mathbf{P}_i$ in 3D and a corresponding parametric position $\mathbf{v}_i$ in 2D of the B-rep patch $\mathbf{p}$. In addition also the gradient $\nabla \mathbf{P}_i$ at $\mathbf{P}_i$ with respect to the parametrisation of $\mathbf{p}$ is given.

4.3.3 *The Clough-Tocher Construction*

Similar to [KC15] we convert parts of the B-rep patches into collections of Clough-Tocher patches. A Clough-Tocher patch consists out of a macro-triangle, which in turn are made up out of three micro-triangles. The construction, which generates a cubic triangle over each micro-triangle, is generated in three distinct steps.

First, the vertices of the macro-triangle are set according to the $\mathbf{P}_i$ sampled from the B-rep patch. In other words the corners of the macro-

patches lie on the patches $\mathbf{p}$. Using the notation of Figure 4.3, left, according to

$$\mathbf{T}_{ij} = \mathbf{P}_i + \frac{1}{3}\nabla\mathbf{P}_i \cdot (\mathbf{v}_j - \mathbf{v}_i)$$

the sides of the macro-triangles are defined. This creates a cubic curve on each side of the macro-triangle. Then, taking into account C^1 continuity conditions of Equation 4.1, we compute

$$\mathbf{I}_{i1} = \left(\mathbf{P}_i + \mathbf{T}_{i,i+1} + \mathbf{T}_{i,i-1}\right)/3, \tag{4.2}$$

with the index $i \in \{0, 1, 2\}$ understood cyclically (modulo 3).

From here, we can set the positions of the micro-patch central control points $\mathbf{Q}_{01}$, $\mathbf{Q}_{12}$, and $\mathbf{Q}_{20}$. There are varying options for settings the positions of these points according to the many variants of the Clough-Tocher construction. [KC15] gives an overview of the different options. In this chapter we use two variants. One of them augments the input data with additional gradients sampled at the midpoints of edges [KC15, Section 3.6], and the original Clough-Tocher variant [CT65], [KC15, Section 3.1] as specified below.

Lastly, the control point positions of the remaining points can be determined from the C^1 continuity conditions

$$\mathbf{I}_{i2} = \left(\mathbf{I}_{i1} + \mathbf{Q}_{i,i+1} + \mathbf{Q}_{i-1,i}\right)/3,$$

and the macro-patch center point

$$\mathbf{S} = \left(\mathbf{I}_{02} + \mathbf{I}_{12} + \mathbf{I}_{22}\right)/3.$$

Because all control points are set according to the C^1 conditions we maintain this continuity across micro-patch boundaries as well as micro-patch boundaries. In addition the Clough-Tocher patch interpolates positions and gradients of the input B-rep patch. This leads to a good approximation of the patches given a suitably generated triangulation.

Because the construction of Clough-Tocher patches relies on the parametrisation, it does not work for arbitrary manifold topology surfaces in that without a common parametrisation it is unable to ensure C^1 continuity. When B-rep patches with distinct parametrisations meet at a common B-rep edge, for instance constructed from an intersection curve of the two patches, their gradients will not in general match. In addition to not being able to ensure continuity of gradients, it is also not even possible to ensure C^0 continuity at the seam. [KC15] remedied this by moving some of the control points so that they align with the B-rep edge. However, ensuring watertightness sacrifices the C^1 continuity of macro-triangles and micro-triangles that lie on the boundary of the patches. There is no simple way of adjusting the control points to remedy this, not even for lowering the continuity to G^1.

In our approach we use a different construction to obtain G^1 continuity near B-rep edges.

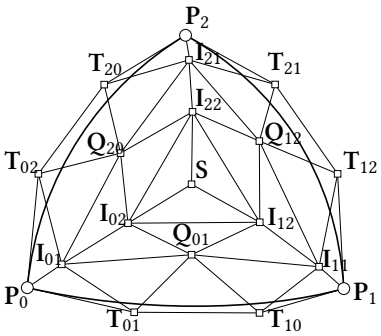


Figure 4.3: *A Clough-Tocher macro-element composed of three cubic Bézier triangles.*

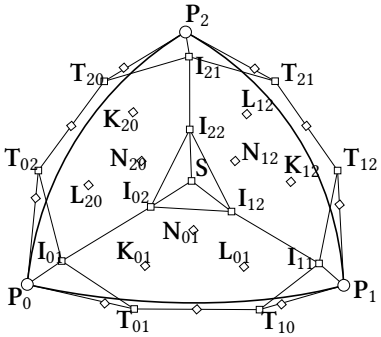


Figure 4.4: *The Shirman-Séquin construction showing a combination of 'cubic' (squares) and 'quartic' (diamonds) control points.*

4.3.4 *The Shirman-Séquin construction*

The Shirman-Séquin construction [SS91, SS87] is another split triangle scheme that is similar to the original Clough-Tocher construction. However the construction of the Shirman-Séquin patches is fully geometric and uses quartic Bézier triangles on micro-triangles instead of the cubic triangles of Clough-Tocher patches. The geometric construction allows it to overcome the parametric and topological restrictions implied by the Clough-Tocher splines. In essence, the construction is a combined construction that uses the method of Chiyokura & Kimura for macro-patch G^1 connections and Farin's method [Far82] to create G^1 joins at micro-edges of the macro-elements.

To construct a G^1 join between macro-triangles the construction makes use of the Chiyokura-Kimura method (CK-method) [CK83] (see Section 2.3.3). The original construction takes as input a triangle with normal vectors defined at vertices. Then the macro-triangle boundary curves are generated to conform to the tangent plane defined by the normal and uses that data for the CK-method. In our context, it is assumed that all cubic edges incident with a mesh vertex meet there with a shared tangent plane, as defined by the gradients of the sampled patches. So

like the Clough-Tocher construction the patches will interpolate positions and gradients.

The patch layout of a Shirman-Séquin patch is shown in Figure 4.4. Generally the positions $\mathbf{T}_{ij}$ are set similar to the Clough-Tocher construction, except in special situations which we will detail later. Similarly, the positions of $\mathbf{I}_{i1}$ are set according to Equation 4.2. From this cubic boundary data the control points for the quartic patches are generated according to process of degree elevation $\hat{\mathbf{I}}_{i1} = (\mathbf{P}_i + 3\mathbf{I}_{i1})/4$ (not shown in the figure), and used in the CK-method to determine all $\mathbf{L}_{i,i+1}$ and $\mathbf{K}_{i,i+1}$. The construction is then completed by computing control point positions from a combination of 'cubic' and 'quartic' control points

$$\begin{aligned} \mathbf{I}_{i2} &= \left(\mathbf{P}_i - 3\mathbf{I}_{i1} + 4\mathbf{K}_{i,i+1} + 4\mathbf{L}_{i-1,i}\right)/6, \\ \mathbf{N}_{i,i+1} &= \left(-\mathbf{I}_{i1} - \mathbf{I}_{i+1,1} + \mathbf{I}_{i-1,1} + 4\mathbf{I}_{i2} + 4\mathbf{I}_{i+1,2} - 3\mathbf{I}_{i-1,2}\right)/4, \end{aligned}$$

and again the macro-triangle split point

$$\mathbf{S} = (\mathbf{I}_{02} + \mathbf{I}_{12} + \mathbf{I}_{22})/3.$$

Then all quartic control points are generated to create a fully degree raised quartic Bézier triangle on each one of the micro-triangles.

The Shirman-Séquin patch is the main ingredient that allows us to construct globally G^1 triangular splines. We will cover the details in the following sections.

4.4 PREPROCESSING

To be able to generate a globally G^1 spline surface from an input CAD model we first preprocess certain parts of it. Consider a CAD model that consists out of one or more (trimmed) B-rep patches. As described in Section 4.3.3 we create a suitable triangulation through an error-driven process to turn the CAD model into a C^{-1} collection of C^1-continuous triangular splines [KC15], where each triangular spline has its own parameter domain.

The B-rep edges are also an important part of the conversion process and are approximated as a C^1-connected cubic Bézier spline. At the endpoints the individual curve pieces coincide with the polyline approximations that were used as constraints for the triangulation. Thus the boundary vertex positions of the triangulation coincide with the B-rep edges. The Bézier curves of the B-rep do not necessarily coincide with the cubic edges of the Clough-Tocher splines that lie on the boundary, as shown in Figure 4.5, left.

In addition we equip each B-rep edge with information that conveys whether it is a sharp edge, i.e. an edge with C^0 continuity across it, or a smooth edge, which should be constructed with G^1-continuity across. If

no such information is provided we attempt to heuristically determine these two classes. With this we rely on normal vector information of the B-rep patches which are incident with that B-rep edge. We look if the angle between normals of the incident patches is below a certain threshold. If it is we deem that vertex to be considered smooth, and sharp otherwise.

Leveraging the most of the Clough-Tocher construction and keeping most of the triangles of the lowest possible degree, we leave the internal Clough-Tocher patches intact. Then we only apply the Shirman-Séquin construction near the B-rep edges. For smooth edges, the boundary data sampled from the CAD model needs to be adjusted to allow the construction to effectively be applied.

4.4.1 G^1-compatible Boundary

In Section 4.3.3 we mentioned that the Clough-Tocher method is only capable of guaranteeing C^0 continuity at B-rep edges. Even after the adjustments to control of micro-triangles near the edges proposed by [KC15]. Even in the case that the gradients at the B-rep span the same tangent plane it is not possible to guarantee C^1 continuity as the parametric domains of the B-rep patches are different. The Shirman-Séquin is able to create a tangent plane continuous-join regardless of the parametrisation, but requires that tangents planes are equal on both sides of the B-rep edge.

Therefore, we must ensure that along the shared B-rep edge we obtain common tangential data. Having the same tangential data translates to the requirement of having a common normal vector defined at vertices of the B-rep edge. To this end we create a procedure that assures equal normal vectors. It proceeds in two steps that first determine an adjusted normal vector for all shared vertices on the B-rep edge and subsequently a step that assures that all curves terminating at such a vertex respect this normal. Different configurations of B-rep edges can occur, but we will first cover the case of two patches meeting at a common B-rep edge, other cases such as junctions are covered in Section 4.4.2.

The situation of two B-rep NURBS patches meeting at a common boundary curve is sketched in Figure 4.5. A vertex $\mathbf{P}$, Figure 4.5, right (shown in grey), has two corresponding positions $\mathbf{P}_1$ and $\mathbf{P}_2$ on the two B-rep patches. The two gradients $\nabla\mathbf{P}_1$ and $\nabla\mathbf{P}_2$ are in general not equal and with that correspond to differing tangent-planes and normal vectors $\mathbf{n}_1$ and $\mathbf{n}_2$. A simple averaging step lets us compute a normal vector $\mathbf{n}_0$, but naturally this normal vector does not have to agree with the B-rep edge at $\mathbf{P}$ in that it is not perpendicular to the tangent of the curve. Another simple projection onto the tangent plane defined by the tangent of the curve at $\mathbf{P}$ lets us create normal vector $\mathbf{n}$ that is perpendicular to the tangents of the curve. Even though two B-rep patches meeting at a sharp B-rep edge do not require a common normal vector,

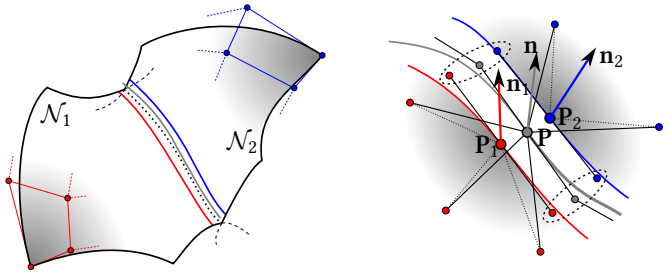


Figure 4.5: *Left: Two NURBS B-rep patches, $\mathcal{N}_1$ and $\mathcal{N}_2$, intersecting along a curve. The true intersection curve (dotted) cannot, in general, be represented as a NURBS curve and is thus approximated by a NURBS B-rep edge (grey). But as this B-rep edge does not, in general, lie on either of the two patches, it is in turn approximated by two trimming curves (red and blue), one on either patch. Right: The situation at a vertex $\mathbf{P}$ of the B-rep edge (grey). Due to the (unavoidable) inexactness of the trimming curves (red and blue), $\mathbf{P}$ corresponds to two, in general distinct, points, $\mathbf{P}_1$ and $\mathbf{P}_2$, one on either of the patches $\mathcal{N}_1$ and $\mathcal{N}_2$. The smaller grey control points are determined using the derivative of the original B-rep edge at $\mathbf{P}$, and similarly for the red and blue control points along the two trimming curves. The other red and blue points come from the constrained triangulations of the two patches and the Clough-Tocher construction on both sides. As the patch normals $\mathbf{n}_1$ and $\mathbf{n}_2$ at resp. $\mathbf{P}_1$ and $\mathbf{P}_2$ do not, in general, agree mutually or with the B-rep edge, they need to be adjusted to $\mathbf{n}$.*

we still apply the projection step to the individual normals $\mathbf{n}_1$ and $\mathbf{n}_2$. In this case no averaging needs to happen beforehand.

We can easily ensure C^0 continuity across the edge by adjusting all red and blue points aligned with the B-rep edge onto the grey points. This is marked as the dotted ellipse in Figure 4.5, right. This exact procedure is also used by [KC15]. However, now that we also have a common tangent plane at boundary vertices $\mathbf{P}$ we can further adjust control points of all edges emanating from $\mathbf{P}$ to conform to the adjusted tangent plane. This procedure is easily achieved by projecting the topologically edge-connected control points onto the plane defined by $\mathbf{P}$ and $\mathbf{n}$. This is a small adjustment in that it respects the original gradients as the projected control points are an affine transformation of this gradient. Having done this second step the surfaces are already G^1 continuous at all vertices $\mathbf{P}$ along the common B-rep edge and we can now apply the Shirman-Séquin construction to also guarantee G^1 continuity along the whole edge.

4.4.2 *Other Vertex Configurations*

Other configurations arise in special cases where multiple B-rep edges meet at a vertex **P**. In these cases the normal adjustment has to proceed differently and needs to conform to multiple B-rep edges. For instance at T-junctions or at X-junctions the normal vector is uniquely defined by the crossing edges, as taken as the perpendicular direction of the incident B-rep directions.

For smooth B-rep vertices that have higher valency, no such unique normal exists. We can still create a common tangent plane by determining a normal **n** by a least squares fit. However, this has the effect that all B-rep edges emanating from this vertex have to be adjusted, by projecting the tangents onto the tangent plane defined by **n**.

4.5 CONVERSION TO SHIRMAN-SÉQUIN SPLINES

Now that the normal vectors at the boundary of the patches have been adjusted we have correctly prepared all the data for G^1 interpolation, or C^0 for non-smooth vertices or edges. We can now apply the Shirman-Séquin construction on triangles at the boundary. It is also possible to replace all Clough-Tocher triangles, but this would be unnecessarily computationally intensive, as the degree of the patches is increased.

However, we still have a few choices of where to apply the construction. We have distinguished three separate strategies of applying it to the triangles at the boundary. Consider the situation as sketched in Figure 4.7. We can categorise the triangles incident with B-rep edges into two groups. 2-connected triangles, are those triangles with two (or more vertices) on the boundary. triangles with only one vertex on the boundary are dubbed 1-connected triangles. All other non-boundary triangles remain as piecewise cubic triangles as constructed from the Clough-Tocher construction.

Here we make the observation that all 1-connected triangles can potentially remain as Clough-Tocher macro-elements. The normal adjustment step does not ruin their internal C^1 continuity. For 2-connected triangles the situation is different as in this case we are dealing with the topological limitations of the technique and we can only resort to the Shirman-Séquin technique. However, this technique cannot also be applied naïvely either, the CK-method only works if both sides of a common edge are constructed using the basis patch technique for determining a cross-boundary derivative. For all 1-connected triangles the cross-boundary derivative is already determined partly by the sampling and fully by the subsequent Clough-Tocher construction. On top of that the cross-boundary derivative of a Clough-Tocher patch is in general different than the one constructed using the CK-method.

Regardless of being either 2-connected or 1-connected, the constructions all take the following initial steps. The boundary of each macro-

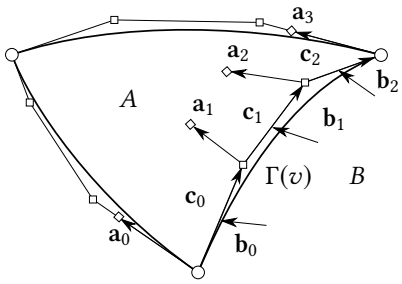


Figure 4.6: *The alternative G^1 method to connect to an existing cubic Bézier triangles. The arrows show the vectors involved in the computation. The points marked by diamonds are 'quartic' control vertices obtained by degree elevation.*

triangle is constructed similarly to the Clough-Tocher construction. This constructs a cubic Bézier curve control polygon for each of the three sides of a macro-triangle. Similarly the control points $\mathbf{I}_{i1}$ (see Equation 4.2) are computed. Having defined these parts of the control nets of the patches we are able to fill the rest in using the Shirman-Séquin construction. However, to join these macro-patches to Clough-Tocher patches we cannot rely on the basis patch construction of the CK-method. Instead we have to devise an alternative method to achieve G^1 continuity.

4.5.1 *An Alternative G^1 Method*

Consider the situation as sketched in Figure 4.6, where an incomplete patch is to be joined to a fully defined cubic Bézier triangle B. Here the patch B is that master in a 'master-slave' situation. With the first two rows of control of B a cross-boundary derivative $\partial\Gamma_B(v)$ is defined along $\Gamma(v)$. We freely choose a direction of the derivative transversal with respect to the edge, but do so in a manner that is independent to vertex indexing and affine reparametrisation. To this end we take the direction from the parametric midpoint of shared edge $(\mathbf{v}_i + \mathbf{v}_j)/2$ and the opposite vertex in parameter space $\mathbf{v}_k$ of the parametric triangle $\mathbf{v}_i\mathbf{v}_j\mathbf{v}_k$. Using this direction we obtain a quadratic transversal derivative given by three control vectors, which we denote by $\mathbf{d}_0$, $\mathbf{d}_1$, and $\mathbf{d}_2$.

We can then use the linear condition of Equation 2.1, which gives the necessary condition for tangent plane continuity between two patches, to obtain

$$\alpha(v)\sum_{i=0}^{3} B_i^3(v)\mathbf{a}_i = \beta(v)\sum_{i=0}^{2} B_i^2(v)\mathbf{d}_i + \gamma(v)\sum_{i=0}^{2} B_i^2(v)\mathbf{c}_i, \qquad (4.3)$$

where $B_i^2(v)$ and $B_i^3(v)$ are the univariate quadratic and cubic Bernstein polynomials, respectively. We choose the functions β and γ to be linear

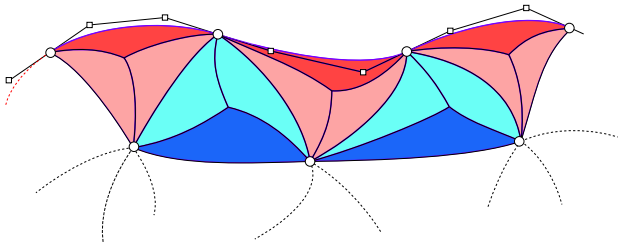


Figure 4.7: *A schematic view of the 1-connected (light and dark blue) and 2-connected (light and dark red) macro-triangles. The control points of the cubic Bézier curves forming the approximated B-rep edge are also shown.*

polynomials, and set $\alpha(v) \equiv 1$. Then we determine the coefficients β_0, β_1 and γ_0, γ_1 by solving Equation 4.3 with $v = 0$ and $v = 1$, i.e., by solving $\mathbf{a}_0 = \beta_0\mathbf{d}_0 + \gamma_0\mathbf{c}_0$ and $\mathbf{a}_3 = \beta_1\mathbf{d}_2 + \gamma_1\mathbf{c}_2$.

The sought-after vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ can then be found via

$$\mathbf{a}_1 = \frac{1}{3}(\beta_1\mathbf{d}_0 + \gamma_1\mathbf{c}_0) + \frac{2}{3}(\beta_0\mathbf{d}_1 + \gamma_0\mathbf{c}_1),$$

$$\mathbf{a}_2 = \frac{2}{3}(\beta_1\mathbf{d}_1 + \gamma_1\mathbf{c}_1) + \frac{1}{3}(\beta_0\mathbf{d}_2 + \gamma_0\mathbf{c}_2).$$

This in turn gives the quartic control points, shown as diamonds in Figure 4.6, of A, such that it connects to B with G^1 continuity.

Note that it is in general impossible to obtain A as a cubic patch. Indeed, modifying the left-hand side of Equation 4.3 to $\alpha(v)\sum_{i=0}^{2} B_i^2(v)\mathbf{a}_i$ admits no solution for generic input data as then only $\mathbf{a}_1$ remains as an over-constrained freedom when $\mathbf{a}_0$ and $\mathbf{a}_2$ are given.

Now having the ability to connect with G^1 continuity to a Clough-Tocher macro patch it becomes apparent that there are two distinct options near B-rep edges. A 'full-strip' strategy where both 2-connect and 1 connected triangles are converted to a Shirman-Séquin spline strip. The other option is to convert only the 2-connected triangles, which creates a saw-tooth-like pattern and is thus dubbed 'saw-tooth'. Figure 4.7 illustrates this. The full-strip pattern contains both blue and red triangles, whereas saw-tooth only contains the red triangles. We will now describe the details of both of these constructions, which assume the boundary adjustment has prepared the data for the G^1 conditions.

4.5.2 *Full-strip*

All 2-connected triangles are constructed as complete Shirman-Séquin patches. Thus the CK-method is used on all three macro-edges of each triangle, especially the side incident to the B-rep edge. As this same step

is applied to the patch on the other side of the edge, and if the normal data is shared, G^1 continuity is ensured.

For the remaining 1-connected triangles we have to proceed differently because we have to join to the 'inner' Clough-Tocher macro-elements of the patch next to the strip. We can use degree elevation to our advantage to degree elevate the boundary data of micro-triangles that connect to these inner macro-elements. In each 1-connected triangle there is one micro-triangle (denoted in dark blue in Figure 4.7) for which we compute the control points according to the Clough-Tocher construction up to the part of the midpoints $\mathbf{Q}_{i,i+1}$ (see Section 4.3.3). This data determines a C^1 cross-boundary derivative which we can degree elevate to quartic, preserving the established C^1 continuity. In principle it is also possible to use the alternative G^1 method instead, however this will lower the continuity to G^1. For the light-blue triangles in Figure 4.7 the inner control points are computed according to the CK-method, as they are adjacent to the 2-connected triangles.

4.5.3 *Saw-tooth*

We can observe that the 1-connected triangles lie on the boundary with one vertex, but the adjustments made in the boundary adjustment step do still let us complete define a Clough-Tocher element. This is because the adjustments respect the gradient at the boundary vertex, only altering it by an affine transformation. In addition, we would like to minimise the number of quartic triangles used and thus only convert 2-connected triangles into Shirman-Séquin patches. Thus all 1-connected triangles emanating from the same boundary vertex are still connected with C^1 continuity.

In this strategy the alternative G^1 method comes into play, as the 2-connected triangles have a light-red micro-triangle adjacent to a light-blue micro triangle which is defined according to the Clough-Tocher construction. We therefore use the method of Section 4.5.1 to construct all inner control points of the light-red micro-triangles. For the dark-red micro triangles the CK-method is used as we are connecting over an actual patch boundary.

We can push this saw-tooth option further, by insisting that only the 2-connected micro-triangles (dark red) are quartic, all other remain cubic. This is possible by performing a secondary split, and we discuss it briefly now for the sake of completeness.

4.5.4 *Saw-tooth: Secondary Split*

In this option, instead of replacing a whole triangle with a Shirman-Séquin element, we first construct a Clough-Tocher element and then replace the boundary micro-triangle with a Shirman-Séquin element. To

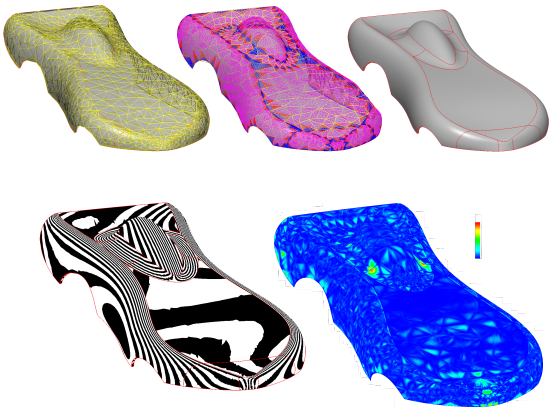


Figure 4.8: *The full conversion process and results. From left to right: Input triangulation of B-rep model, the control net after conversion to a composite Clough-Tocher and Shirman-Séquin triangular spline using the saw-tooth technique, smooth shading and reflection lines on the result, and a visualisation of the error with respect to the original B-rep model (blue is zero error and red is global error maximum of* 1.8168e−4 *relative to the unit bounding box diagonal.). The colour of the control nets signifies the method used to calculate the control data. Purple corresponds to the Clough-Tocher method, blue to the method of Chiyokura & Kimura, and red to the alternative* G^1 *method presented in Section 4.5.1.*

this end we do not perform the adjusting step of the boundary control points to conform to those of the B-rep edge, and construct a Clough-Tocher element. Once constructed we perform the boundary adjustment and afterwards replace the boundary micro-triangle with a Shirman-Sequin element similarly to the ordinary Saw-tooth strategy. In this case we use the alternative G^1 method to join to the light-red micro triangles of the Clough-Tocher element and again the CK-method for the connection across the B-rep edge.

In the corner cases when a macro-triangle is adjacent to two B-rep edges, i.e. a 3-connected triangle, or two 2-connected micro-triangles, then the common micro-triangle boundary also use the CK-method. This behaviour can be clearly observed in the corners of the two different sphere octants depicted in Figure 4.9, fourth column, and is described in Section 4.6.1.

4.6 RESULTS

We summarise once again the pipeline of the conversion of B-rep CAD model into globally G^1 splines in Figure 4.8. In this section we compare and contrast the different strategies and the capabilities of the conver-

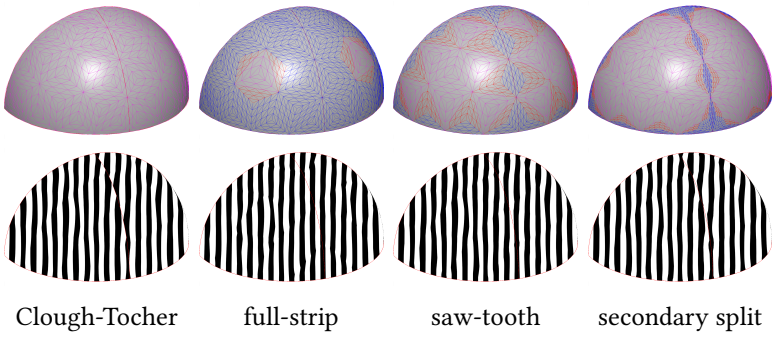


Figure 4.9: *A comparison on a simple model composed of two sphere octants having separate parametrisations. Both octants have been uniformly sampled using* 16 *triangles. In the top row we show Phong shading and Bézier control nets, and in the bottom row we show the reflection lines on the octants after conversion. The colour of the control nets signifies the method used to calculate the control data. Purple corresponds to the Clough-Tocher method, blue to the method of Chiyokura & Kimura, and red to the alternative G^1 method presented in Section 4.5.1.*

sion method. To this end we employ simple examples to illustrate our method, but also use more complicated CAD models to measure the effectiveness in more elaborate situations.

4.6.1 *Sphere octants*

We start with a simple example that is constructed out of two patches. The patches are the connected octants of a sphere, and because of this do not have to worry about the boundary adjustment step as this already guarantees a shared tangent plane at sampled vertices of the common edge. Even though the tangent planes are shared and the octants are fully symmetrical, their gradients at the edge do not agree. In this case the Clough-Tocher based method of [KC15] would yield only C^0 results, even after the edge control point adjustment, as can be seen clearly from the broken reflection line renderings. In that case a maximum normal deviation is observed of approximately 2.65 degrees; meaning that the conversion does not meet the 0.1 degree tolerance of standard CAD systems.

With our conversion method we are able to globally obtain a G^1 results, using any of the conversion variants discussed in Section 4.5. We can observe now that there are no broken reflection lines as is expected of G^1 continuity as seen in the bottom row of Figure 4.9.

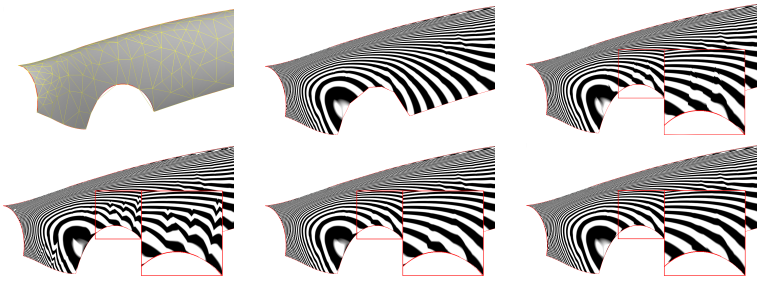


Figure 4.10: *Reflection lines on a trimmed NURBS patch (part of a car model) converted to a triangular spline using the Clough-Tocher variant with additional gradients sampled at edge midpoints. Top row, left to right: The input triangulation of the trimmed NURBS patch, Clough-Tocher patches, Clough-Tocher patches with the boundary adjustment of [KC15]. Note the jumps in reflections lines, indicating that the adjustment ruined some of the internal C^1 continuity. Bottom row, left to right: Full-strip, saw-tooth and secondary split; fully G^1 variants of our conversion method.*

4.6.2 *Boundary adjustment*

The boundary adjustment (see Section 4.4) is a necessary step to achieve G^1 continuity, but it has the effect of changing the geometry near B-rep edges. We will now investigate the effects of the adjustments step on the conversion process.

We will first look at how the adjustment step lets us preserve internal G^1 continuity within a single B-rep patch whilst also allowing B-rep edge interpolation. A single patch of a complicated CAD car model is shown in Figure 4.10, and is compared to the C^0 strategy of Kosinka & Cashman [KC15].

First of all in our method C^0 continuity is guaranteed by the adjustment step of the control point positions of the macro-patch boundary edges, that make them coincide with the control points of the B-rep edge. We first perform the boundary normal adjustment step and assure that the tangent plane of the triangular spline also contains the tangent of the B-rep edge. In this way, C^0 is achieved whilst not distorting the tangent plane at the B-rep edge vertices, which happens with the simpler adjustment step of [KC15]. Now a valid tangent plane is defined at boundary vertices and it allows us to still create G^1 continuity joins internally.

Figure 4.10 shows a single trimmed NURBS patch converted into a Clough-Tocher spline using the extra sampled mid-edge gradients variant [KC15, Section 3.6]. We compare the C^0 adjustment of [KC15] and our three different Shirman-Séquin based methods visually. In this case a visual improvement is observed at the boundaries of macro-elements

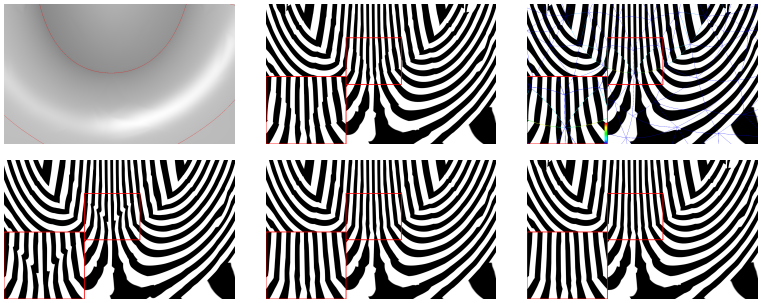


Figure 4.11: *Reflection lines on two trimmed NURBS patches (part of a car model) converted to a triangular spline using the standard Clough-Tocher method. Top row, left to right: A smooth rendering of the trimmed NURBS patches, Clough-Tocher patches with the boundary adjustment of [KC15], Clough-Tocher patches with boundary adjustment of [KC15] and triangulation showing the difference in normal vector; blue indicates no deviation in normal vector and red indicates high deviation. Note the jumps in reflections lines, indicating that the adjustment ruined internal C^1 continuity. Bottom row, left to right: Full-strip, saw-tooth and secondary split; fully G^1 variants of our conversion method.*

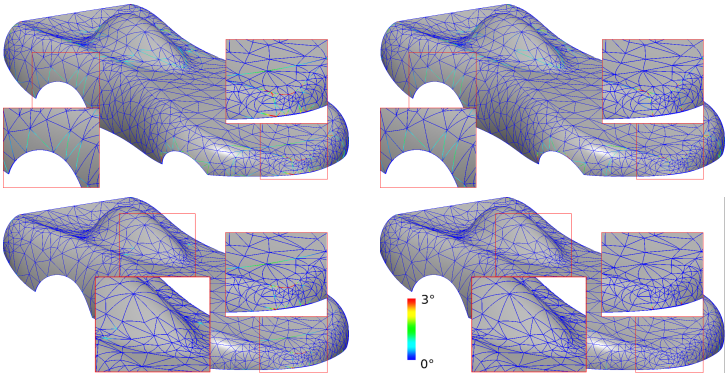


Figure 4.12: *Visualisation of the difference between normal vectors of neighbouring triangles of the spline surfaces along their shared edges. From left to right, top to bottom: Clough-Tocher with control point adjustment, Clough-Tocher with control point adjustment and normal adjustment, saw-tooth Shirman-Séquin and saw-tooth Shirman-Séquin with normal adjustment. The original Clough-Tocher construction was used in these examples.*

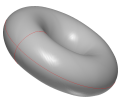
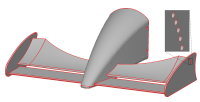
which lie on the boundary. The reflection lines remain unbroken, showing a continuous tangent plane at the rather large triangles connected to the boundary.

We turn our investigation at the situation of two (or more) trimmed patches that ought to join smoothly at along a shared edge. Figure 4.11 we show the back of a car model where patches meet. Along this particular edge the tangent plane of the sampled surfaces do not agree with each other. This leads to G^1-discontinuities when applying the adjustment of Kosinka & Cashman [KC15]. By applying our normal adjustment step along the common edge, we create a shared tangent plane for both spline surfaces on each side of the edge, which can then be used by the Shirman-Séquin macro- and micro-elements to ensure G^1 continuity.

We want to emphasise the necessity of the normal adjustment step by visualising the angle difference between the normal vectors along a common edge of neighbouring triangles in the converted surfaces with and without the normal adjustment and are shown in Figure 4.12. Even with C^0 methods there is a decrease in normal error when applying the normal adjustment. Without the boundary adjustment the G^1 methods are unable to remove the normal error and there are still defects present around B-rep edges. When applying the normal adjustments this is fixed in nearly all situations. In this case there are errors around the T-junctions where the two emanating B-rep edges express different tangent planes. In Section 4.4.1 we have talked about means to also smooth these situations, but this necessitates the adjustment of B-rep edges themselves. In Figure 4.12 we have chosen not to apply this step and, thus give rise to sharp corners.

More complicated situations arise when we want to guarantee G^1 continuity at places where several trimmed patches meet. In Figure 4.13

Table 1: *The approximation error between a (densely triangulated) CAD model (not shown) and its approximation with triangular splines (top row), using different methods at B-rep edges. All models have been scaled to unit bounding-box diameter.*

	 Torus	 Wing	 F1
B-rep patches	1	4	90
Macro-triangles	64	586	3371
Clough-Tocher	3.46e−3	8.10e−4	1.52e−3
Full-strip	4.23e−3	6.90e−4	1.52e−3
Saw-tooth	4.23e−3	6.90e−4	1.52e−3
Secondary split	4.23e−3	8.10e−4	1.54e−3

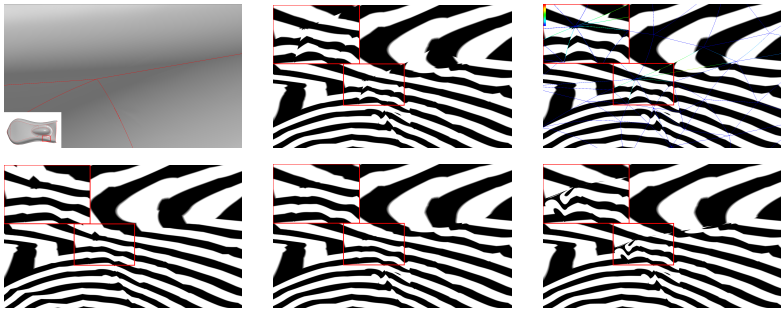


Figure 4.13: *A challenging example: Reflection lines over a meeting point of several trimmed NURBS patches which have been converted to triangular splines using the Clough-Tocher variant with additional gradients sampled at edge midpoints. Top row, left to right: The smooth rendering of the trimmed NURBS patches, Clough-Tocher patches with the boundary adjustment of [KC15] and triangulation showing the difference in normal vector. Bottom row, left to right: The full-strip, saw-tooth, and secondary-split variants of our method.*

we show the situation around a B-rep vertex where several different trimmed NURBS patches meet. All the shown B-rep edges are marked as smooth and subsequently the boundary normal adjustment technique was used to create a common tangent plane for all patches meeting an an edge.

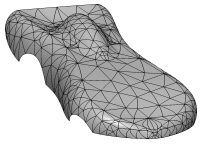
In Figure 4.13, the different strategies show varying levels of preservation of the original C^1 Clough-Tocher spline. Visually it is hard to argue that some of the surfaces actually improve, even though they have exact G^1 continuity, on the approximate C^0 equivalent. Especially in the case of the secondary-split saw-tooth technique they shows rapidly varying reflection lines around the central vertex. The other two techniques show better results and compare favourably to the C^0 technique especially the reflection lines running across the B-rep edges.

Finally, Figure 4.8 shows the result of applying our conversion process to a full CAD model, in this case that of a car composed of 42 B-rep patches. Note that for the generation of the images in this figure the saw-tooth variation was used.

4.6.3 *Approximation error*

As mentioned before, the normal adjustment step alters the geometry around B-rep edges. We therefore need to investigate the effect this has on the approximation error that is incurred during this process. We compare the maximum error between a densely triangulating representation of the input CAD model and the different conversion strategies. In Table 1, we list the results for various CAD models featured in this chap-

Table 2: *The approximation error between a (densely triangulated) CAD model and its two spline approximations differing in triangle counts, using different methods at B-rep edges. All models have been scaled to unit bounding-box diameter.*

		
B-rep patches	13	13
Macro-triangles	532	1420
Clough-Tocher	0.00050614754	0.00017790225
Full-strip	0.00050614754	0.00018168288
Saw-tooth	0.00080641346	0.00018168288
Secondary split	0.00050174044	0.0001821653

ter. The base triangulation of the sphere octant and the torus model is very minimal with only 64 triangles each. The complex Formula 1 front wing model has genus 18 and contains as many as 14 small trimmed bolt holes (see inset, last column) and several fillet patches.

We observe that the approximation error remains approximately the same even when using the G^1 conversion methods. In some cases we observe an even lower maximum approximation error than compared to the method of [KC15]. This could have many reasons that are not readily apparent, but could be because of the increased continuity internally. At the vertices of non-smooth edges the tangent planes of the Shirman-Séquin macro-elements are co-planar with the actual sampled gradients of the underlying NURBS patches, which in turn increases the approximation quality at these vertices with respect to the C^0 technique of [KC15].

By increasing the number of sample points or similarly the density of the triangulation we observe the expected decrease in approximation error. Table 2 shows the approximation errors obtained using two different triangulations of the same B-rep model. We observe a decrease of the approximation error when using each of the three new constructions, and the error remains approximately the same as when using the ordinary Clough-Tocher method.

4.6.4 *Performance*

The increased degree of the micro-elements of Shirman-Séquin macro-elements, quartic instead of the cubic elements of the Clough-Tocher construction, and the necessary steps taken to construct them might

increase the computation time needed in the spline conversion process. We have explored the trade-off between continuity and visual quality with respect to the C^0 methods of [KC15]. However, the different strategies of constructing a globally G^1 spline also affect the efficiency of the methods. For instance the full-strip strategy and the secondary-split strategy are fully local as they can be determined on a per-triangle basis and can thus be constructed fully in parallel.

For the saw-tooth strategy, the Shirman-Séquin elements cannot be constructed concurrently with the Clough-Tocher elements as their construction depends on their neighbouring macro-elements, as they use the alternative G^1 method. It is still possible to construct them fully locally by supplying parametric data of adjacent triangles. However, in that case many of the control points are computed twice to be able to independently compute the cross-boundary tangent function.

To make statements about the relative performance we have measured the performance of the full conversion process using all considered strategies. The performance measurements were captured with respect to the car mesh from Figure 4.8) consisting of 42 B-rep patches. In all cases the B-rep patches have been triangulated into a total of 3233 triangles which determine the topology of the triangular spline surface. On average the construction of the ordinary Clough-Tocher spline surface took 0.09235 seconds, whereas 0.121, 0.108, and 0.123 seconds were the averages recorded for the full-strip, saw-tooth, and secondary split techniques, respectively. Although the new G^1 methods may seem complex in their constructions, they do not have any significant overhead with respect to only using Clough-Tocher macro-elements.

4.7 DISCUSSION

Although we offer the ability to create a smooth join between adjacent patches through the use of the boundary adjustment step, we still need the input CAD models to satisfy certain smoothness conditions. The boundary adjustment only 'slightly' adjusts the models. It is expected that when an edge is marked as smooth that the normal vectors on either side of the edge already approximately match the G^1 conditions, in that they are different only up to a tolerance offered by the CAD system of choice.

The visual and error-wise improvement over the C^0 method of [KC15] is only slight. However, our method creates a globally G^1 triangular spline at little increased computational cost. Our splines are directly usable in isogeometric analysis and only offering G^1 continuity at patch boundaries is not an obstacle [GP15].

As alternative to the Shirman-Séquin macro-element we have considered the triangular Gregory patch [Lon85]. However, the patches provided no direct advantages. These patches are rational as some of the ba-

sis functions are rational polynomials, which complicates computation of derivatives. One advantage is that there is no splitting needed as each triangle can directly be turned into a Gregory patch. Moreover, according to our experiments, they are visually nearly indistinguishable from the Shirman-Séquin macro-elements as their boundary conditions are also set using the method of Chiyokura & Kimura. With this they could be used with the same saw-tooth and full-strip strategies presented in this chapter.

Naturally, the full-strip conversion strategy causes the biggest visual changes with respect to the Clough-Tocher approach, as more triangles are converted to Shirman-Séquin elements and their cross-boundary data is constructed using the CK-method. In the case of saw-tooth variants most of the original cross-boundary derivatives, from the Clough-Tocher constructions, are kept intact. There are possible settings wherein the full-strip produces a noticeably worse quality surface or a better quality one. In general the full-strip strategies have more noticeable changes in directions in their reflection line visualisation, because of the increased patch degree. The secondary-split technique can show even more rapid changes in the reflection lines, especially near B-rep edges, as can be seen in Figure 4.9. These changes are confined to the converted micro-triangles, which are relatively small, compared to surrounding triangles, and the higher quartic degree of the patches and the fact that there is yet another split, leads to rapidly changing surface. In theory the micro-triangle could be split yet again, but this increases these problems even further. The normal saw-tooth strategy provides the most balanced results out of all three strategies. It respects the original (internal) C^1 Clough-Tocher approximation, and at the same time remains visually reasonable, whilst being able to create G^1 continuity over the patch boundary.

The triangulation of the B-rep patches is an important factor in conversion process. The results heavily depend on this initial triangulation. We did not investigate the effect of the triangulation on the conversion process, but rather focused on creating a construction that would allow for G^1 continuity between patches of an arbitrary topology model. There are some other choices that might affect the quality of the conversion process, which is the choice of the split-point of the (macro-) triangles. And of course which variant of the Clough-Tocher variant to use. We made our approach agnostic of the variant of the Clough-Tocher scheme and we have experimented with different variants. With that our finds were in line with those in [KC15]. In any case, no matter which combination is used, it will lead to globally G^1 results.

4.8 CONCLUSION

We have investigated and presented several methods and strategies to improve existing conversion techniques for converting trimmed CAD models into triangular spline surfaces. Our solutions convert the models into a G^1-continuous triangular splines that are mostly C^1 continuous except for areas near B-rep edges. The results are analysis-suitable and compatible with CAD systems.

We require an intermediate preprocessing step that adjusts the boundary normal of B-rep patches that are required to join smoothly.

We have presented three different strategies of employing Shirman-Séquin macro-elements at B-rep patch boundaries. These are used either on a full-strip of 1- and 2-connected boundary triangles, or exclusively on 2-connected triangles, or even only on 2-connected micro-triangles. The three techniques have varying levels of preservation of the C^1 continuity created by the original Clough-Tocher conversion. The full-strip variant is the most invasive, followed by the replacement of 2-connected macro- and micro- triangles. The three techniques are efficient, as they are only applied near B-rep edges and do not require a lot more effort to compute than the ordinary Clough-Tocher techniques, whilst offering comparable approximation quality, even after the boundary adjustment step.

ACKNOWLEDGEMENTS The car, wing, and F1 front wing CAD models used in this chapter have been kindly provided by International TechneGroup Ltd. The triangulations and other sampled data from these models have been obtained using CADfix [Int19].

5 MULTISIDED B-SPLINE PATCHES

Parts of this chapter have been published as

- Gerben J. Hettinga, Pieter J. Barendrecht and Jiří Kosinka. "A Comparison of GPU Tessellation Strategies for Multisided Patches."
- Gerben J. Hettinga, and Jiří Kosinka. "A multisided C^2 B-spline patch over extraordinary vertices in quadrilateral meshes."
- Gerben J. Hettinga, and Jiří Kosinka. "Multisided B-spline patches over extraordinary regions"

We propose a generalised B-spline construction that extends uniform bi-degree B-splines to multisided patches spanned over extraordinary regions in arbitrary topology meshes. We show how the structure of the generalised Bézier patch introduced by Várady et al. can be adjusted to work with uniform B-spline basis functions as well as how they can be spanned over extraordinary faces and vertices. We create ribbon surfaces based on B-splines using special basis functions. The resulting multisided surfaces are C^{d-1} continuous internally and connect with G^{d-1} continuity to adjacent regular and other multisided B-spline patches. We visually assess the quality of these surfaces by comparing them to Catmull-Clark limit surfaces on several challenging geometrical configurations. We design several specialised functions that increase the visual quality of the patches, in both the extraordinary vertex and face settings.

In addition we create an augmentation of the traditional tessellation pipeline with several different strategies that efficiently render multisided patches through the use of generalised barycentric coordinates. The strategies involve different subdivisions of the polygon and the use of textures. In addition, we show that adaptive tessellation techniques naturally extend to some of these strategies whereas others need a slight adjustment. The technique of Loop et al., commonly known as ACC-2, is extended to multisided faces to illustrate the effectiveness of multisided techniques. A performance and quality comparison is made between the different strategies and remarks on the techniques and implementation details are provided.

5.1 INTRODUCTION

Arbitrary topology meshes introduce problems in the definition of B-spline surfaces due to the extraordinary regions on which ordinary B-splines cannot be applied. Dealing with these regions has been a long standing problem in computer graphics and geometric design, and recently also in (isogeometric) analysis. Extraordinary vertices and faces do not posses a regular structure for which tensor-product B-splines are well-defined. This leads to problems on how to define smooth basis functions at these regions. Many solutions have been proposed that try to create basis functions or other smooth surfaces over the extraordinary regions, most of which focus on quad-dominant meshes.

Subdivision surfaces (see Section 2.4.1) tackle the problem in an iterative manner where the smooth surface at extraordinary regions is obtained by a refinement process. This process reproduces B-spline surfaces in regular regions, but still there is no closed-form representation in extraordinary regions. [Sta98] provides a way to evaluate the surface at an arbitrary parametric position, but only in areas with restricted topology of the control mesh. In whatever way subdivision surfaces are defined they are typically only G^1 at extraordinary vertices, and can introduce other undesirable artefacts around this region, especially when considering high-valency vertices or faces. Finding a subdivision scheme that provides both good shape, global continuity higher than G^1 and that remains simple has not been found (yet) [RS19].

The generalised Bézier patch [VSK17] (see Section 2.3.5) generalises the tensor-product Bézier patches to a face with an arbitrary number of sides. It has the ability to interpolate positions and derivatives and can be smoothly joined to other (multisided) Bézier patches. The simple structure remains as intuitive as ordinary Bézier surfaces and is composed out of a combination of ribbons, partial Bézier surfaces. The simple structure of these patches motived us to find means to alter the structure of the patch and the basis functions so that they can be used in combination with B-spline surfaces or integrated into existing B-spline surfaces.

In this chapter we show how we can change the structure of the generalised Bézier patch to incorporate B-spline basis functions, so that we can use them to create surfaces over extraordinary regions in B-spline surfaces. In addition, we show how multisided generalised Gregory Bézier patches can be used to approximate Catmull-Clark surfaces over extraordinary regions using a multisided ACC2 [LSNC09] variant. Finally, we show how multisided surfaces can be rendered efficiently using the GPU hardware tessellation pipeline.

5.2 RELATED WORK

There are many strategies that solve the related problem of using multiple quadrilateral elements around extraordinary vertices, so that they join smoothly with surrounding (B-spline) regions and to each other. However, for extraordinary faces there exists no such special setting without reducing it to the extraordinary vertex case by splitting the face. Catmull-Clark subdivision surfaces have been approximated by Gregory patches [LSNC09], to create G^1 smooth surfaces. For higher-continuity, such as G^2, higher degree patches have been used such as the biseptic patches of [LS08b] and bi-sextic of Karčiuaskas & Peters [KP16]. [Pet19] provides an overview of such constructions.

Our approaches use a single parametric patch instead of a multitude of patches. Over time there have been many different multisided patch structures. The aforementioned generalised Bézier patch is just one of many. Recently an adjusted generalised Bézier patch has been developed that uses B-splines as boundary curves [Vai21]. However, when joining this patch to adjacent B-spline regions the control points of the ribbons have to be determined in Bézier format to guarantee G^2 continuity. Our approach only uses adjusted B-spline basis functions so that the original control points of the control mesh can be used. The S-patch [LD89] is a Bézier patch constructed using a multinomial expansion of the Bernstein basis functions. However, it is not trivial to include these patches in B-spline surfaces. Loop & DeRose [LD90] created a means to include S-patches in special B-spline surfaces such that they join with G^1 continuity to biquadratic and bicubic regions. The patches are of high-degree and have an extremely large number of control points. The corner interpolator patches of Gregory [GLZ90] are able to interpolate arbitrary curves, derivatives and second derivatives, by defining a patch as n blended corner surfaces. These surfaces have been used to create arbitrary topology cubic B-spline surface [ZZZS05a] by first using a number of Catmull-Clark subdivision steps. A related approach generalised biquadratic B-spline surfaces [ZZZS05b] with multisided Zheng& Ball patches [ZB97], but can only be applied to patches with a limited number of sides.

More complicated constructions rely on the theory of manifold surfaces to generalise B-spline surface of arbitrary degree for arbitrary topology meshes [YZ04, NG00]. A partition of unity combines overlapping special charts associated with each element of the mesh. Although, the extraordinary regions can be changed, this technique also inadvertently influences the surrounding regular regions.

5.3 MULTISIDED B-SPLINE CONSTRUCTION

For the multisided B-spline construction we first look at how we can generalise uniform cubic B-splines to multisided regions. First we will

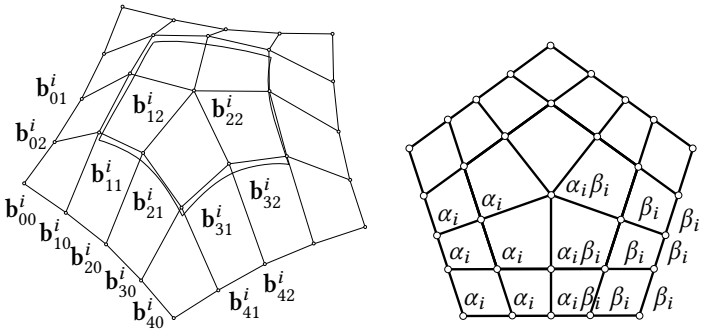


Figure 5.1: *Left: The control net of a pentagonal cubic generalised B-spline patch with control point labels with respect to* Γ_i*. Right: The distribution of blending functions for side* Γ_i *is shown.*

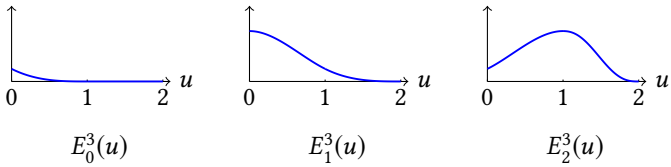


Figure 5.2: *The extended basis functions used in the construction of the multisided B-spline patch. Both* $E^3_0(u)$ *and* $E^3_1(u)$ *are simply the continuation of the uniform B-spline kernel.* $E^3_2(u)$ *is a combination of a cubic and a quintic function.*

look at the regions around extraordinary vertices and their one-ring neighbourhood. For an extraordinary vertex with valency n, a single multisided patch is created, covering the one-ring neighbourhood. The multisided B-spline patch is a combination of n ribbons. The ribbons represent a composite B-spline surface with specially constructed basis functions, which we detail later. The control net for a cubic multisided cubic B-spline surface is shown in Figure 5.1. Regular bicubic patches will leave a multisided hole where each side is adjacent to two regular regions. Therefore, the ribbons of the multisided face must connect smoothly to two patches at the side. Moreover, the patch should be smooth by itself. For this we need to construct appropriate basis functions.

5.3.1 *Extended Cubic Basis Functions*

We want to use B-spline basis functions for the ribbon surfaces, but simply interchanging the Bernstein functions for B-spline functions leads to a few problems. First of all, all ribbons represent now a composite surface and need to extend smoothly towards the inside of the patch. If

not, discontinuities will be created within the patches, along the centre of the patch running outwards towards the edges of the patch. Moreover, unlike the Bernstein basis functions, some of the standard B-spline basis functions have non-zero values and derivatives at certain knots. With the composite ribbons we are dealing with an increased interval over $[0, 2]$ with three knots. Therefore, we need to adjust the standard B-spline basis functions to be C^2 everywhere inside the patch and reproduce the B-spline basis functions at certain other values within the extended interval, and also vanish at the ends of the interval.

We create the functions $E_j^3(u)$ over the extended interval $[0, 2]$ so that they cover the domain of the B-spline ribbon. They are positive everywhere in the interval and remain within the range $[0, 1]$. In addition, on $[0, 1]$ they are equal to the standard uniform cubic B-spline basis functions, i.e., $E_j^3(u) \equiv N_j^3(u)$ for $u \in [0, 1]$. Here $N_j^3(u)$ is the j-th uniform cubic B-spline basis function (see Section 2.4). With this we make sure that these basis functions connect smoothly to the standard B-spline basis functions at the start of the interval and we inherit their favourable shape properties. Then, we we also want to guarantee that the contribution of a ribbon Γ_i vanishes on the distant sides $\Gamma_{i+2}, \cdots, \Gamma_{i-2}$. This means that it does not contribute positionally $E_j^3(2) = 0$. In addition the derivatives should also vanish meaning that the first and second derivative should vanish at $u = 2$, too.

Unfortunately, merely extending the standard cubic basis functions to have an extended support only partly satisfies the conditions, as only some of the endpoint conditions are met. As the functions E_j are fine on the interval $[0, 1]$, we only need to find suitable extensions to $[1, 2]$ that assure the listed conditions for G^2 continuity. However, having this piecewise representation of E_j adds additional constraints. As we do not want to have lower than C^2 continuity at any point on the interval, care must be taken to create a C^2 continuous join at $u = 1$.

With these constraints we can start to design our basis functions. We need to ensure G^2 continuity with adjacent (regular) B-spline regions for each B-spline ribbon. This means that each ribbon needs at least three rows of control points and therefore also three basis functions. In Figure 5.4 we show our three extended cubic basis functions. The first and second basis functions are simply $E_0^3(u) \equiv N_0^3(u)$, and $E_1^3(u) \equiv N_1^3(u)$ with $u \in [0, 2]$. As these functions both vanish at $u = 2$ up to second derivative, they satisfy all properties that we need to guarantee smoothness of the ribbons. The third basis function does not vanish at the end of the interval and we have to create a custom piece on $[1, 2]$ as a quintic polynomial:

$$E_2^3(u) = \begin{cases} N_2^3(u), & u \in [0,1]; \\ \frac{2}{3}(2-u)^5 + \frac{2}{3}5(2-u)^4(u-1) & \\ +\frac{17}{30}10(2-u)^3(u-1)^2 & u \in [1,2]. \end{cases}$$

It is impossible to create a cubic function that reproduces up to the second derivative that of $N_2^3(u)$ at $u = 1$ and also has vanishing derivatives up to order two at $u = 2$. Therefore, a quintic function is needed to join smoothly to $N_2^3(u)$ on $[1, 2]$. The quintic function connects to the cubic B-spline function with C^2 continuity and at the endpoint of the interval it vanishes along with its first and second derivative. The coefficients for the quintic function were obtained by solving the system

$$\begin{bmatrix} 1 & 0 & 0 \\ -3 & 3 & 0 \\ 6 & -12 & 6 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} N_2^3(1) \\ \frac{\partial N_2^3(u)}{\partial u}\Big|_{u=1} \\ \frac{\partial^2 N_2^3(u)}{\partial^2 u}\Big|_{u=1} \end{bmatrix},$$

where c_i are the coefficients for a univariate quintic Bézier polynomial over $[0, 1]$. We only need to determine the first three coefficients c_0, c_1, c_2, as the other three coefficients, c_3, c_4, c_5, can simply be set to 0 to let all of the necessary derivatives vanish. Finally, the piece is shifted from the interval $[0, 1]$ to $[1, 2]$.

Having defined the extended basis functions, we are now ready to define the B-spline ribbons.

5.3.2 *B-spline Ribbons and Patch*

The definition of the multisided B-spline patch is very similar to that of the generalised Bézier patch (see Section 2.3.5). It is also a combination of B-spline ribbons. The local ribbon parameters s_i and h_i are constructed from generalised barycentric coordinates, and are continuously defined over the whole patch. We scale the parameters so that they span the interval $[0, 2]$: we set $\bar{s}_i(\boldsymbol{\phi}) = 2(1 - \phi_i - \phi_{i-1})$ and $\bar{h}_i(\boldsymbol{\phi}) = \frac{2\phi_i}{\phi_i + \phi_{i-1}}$. Then we use the standard uniform cubic B-spline basis functions along $\bar{s}_i$ and the extended basis functions along $\bar{h}_i$.

Next, we have to make sure that control points which are used in multiple ribbons are blended correctly. The distribution of the blending functions for a single ribbon is shown in Figure 5.1, right. All control points have at least one attached blending function and the middle column has two because that column of control points is shared with the B-spline ribbons on Γ_{i-1} and Γ_{i+1}. Each ribbon has at least three rows of control points to be able to join with G^2 continuity to adjacent regions, and thus the overlap is necessary. We can now complete the ribbon definition:

$$\mathbf{R}_i(u,v) = \begin{cases} \sum_{j=0}^{3}\sum_{k=0}^{2} v_j^i \mathbf{b}_{jk} N_j^3(u) E_k^3(v) & u \in [0,1), \\ \sum_{j=3}^{6}\sum_{k=0}^{2} \omega_j^i \mathbf{b}_{jk} N_{j-3}^3(u-1) E_k^3(v) & u \in [1,2], \end{cases}$$

with $v \in [0,2]$. Here, N_j^3 are the uniform cubic B-spline basis functions and E_k^3 are extended cubic basis functions as defined above. The distribution of the blending functions is governed by

$$v_j^i = \begin{cases} \alpha_i^3 & j < 3; \\ \alpha_i^3\beta_i^3 & j = 3; \end{cases}, \quad \omega_j^i = \begin{cases} \alpha_i^3\beta_i^3 & j = 3; \\ \beta_i^3 & j > 3. \end{cases}$$

The ribbon blending functions are similar to the one used in the (enhanced) generalised Bézier patch [VSK17], but instead of squared terms they are cubed. This is necessary to let the right derivatives vanish as we will show in Section 5.5.

The cubic B-spline ribbon is a piecewise C^2 surface where the pieces join at $u = 1$. The blending functions introduce removable singularities in the corners of the patches, due to a zero denominator. These corners correspond exactly with the limit positions, which can be found by applying the regular limit stencil

$$\begin{bmatrix} 1 & 4 & 1 \\ 4 & 16 & 4 \\ 1 & 4 & 1 \end{bmatrix} /36$$

to the one-ring of vertices of the control mesh around the corner vertices, for example $\mathbf{b}_{11}^i$ and $\mathbf{b}_{31}^i$, surrounding the extraordinary vertex.

The ribbons can be combined into the final surface:

$$\mathrm{S}^3(\boldsymbol{\phi}) = \sum_{i=0}^{n} \mathbf{R}_i(\bar{s}_i, \bar{h}_i).$$

Similarly to the generalised Bézier patch, this patch can be made affine invariant either by normalisation by dividing it by the sum of basis and blending functions:

$$f(\boldsymbol{\phi}) = \textstyle\sum_{i=0}^{n} r_i(s_i,h_i), \text{ where}$$
$$r_i(u,v) = \begin{cases} \sum_{j=0}^{3}\sum_{k=0}^{2} \mu_j^i N_j^3(u) E_k^3(v) & u \in [0,1), \\ \sum_{j=3}^{6}\sum_{k=0}^{2} \mu_j^i N_{j-3}^3(u-1) E_k^3(v) & u \in [1,2], \end{cases} \tag{5.1}$$

or by introduction of an additional point that takes care of the excess or deficient weight in the basis functions as is done with the generalised Bézier patch.

The structure of the generalised B-spline patch is simple and requires only the control points of the B-spline control net. This is in contrast to other methods like [ZZZS05b] or [Vai21] that first need to extract the Bézier control points. The surfaces are affine invariant, preserve the convex hull property, and edits to the control mesh have local influence. The internal continuity of the patch is C^2 as the ribbons themselves are piecewise surfaces.

5.4 MULTISIDED B-SPLINE PATCHES

Having seen the example of the cubic generalised B-spline patches we can now generalize this structure to B-spline patches of arbitrary degree and also to use the same structure to also create patches over extraordinary faces. Extraordinary vertex and face patches need to be processed slightly differently as the control nets of the latter are slightly larger.

As we saw in the cubic case, a number of subdivision steps are needed to separate extraordinary regions and to create a sufficient regular region around them to be able to define the ribbons correctly. For bi-degree d B-splines, a separation of at least $d-1$ regular vertices is needed. However, we want our subdivision step to be compatible with the degree of the B-splines we are trying to create. Therefore we employ arbitrary degree subdivision [Sta01] (see Section 2.4.1), until extraordinary regions are separated sufficiently.

5.4.1 *Extraordinary Vertex Patches*

We can generalize the structure of the multisided B-spline patches to other degree B-spline surfaces. We will skip the trivial bilinear B-spline case and consider the next simplest case of biquadratic B-spline surfaces. The main difference with the cubic case is that each ribbon is just a single B-spline surface. They consist of a control net of 3×2 control points. Again, we use mostly standard uniform quadratic B-spline basis functions except for the last row of control points. Thus $E_0^2(u) = N_0^2(u)$, but for the second row we need to again create a custom basis function that vanishes at the end of the interval. In this case we only need that the basis functions connect with C^1 continuity at the start of the interval and vanish positionally and up to the first derivative at the end. For this a simple cubic function is sufficient

$$E_2^1(u) = \frac{1}{2}(1-u)^3 + \frac{5}{2}(1-u)^2 u.$$

The distribution of the blending functions α_i^2 and β_i is also analogous, in that they overlap in the middle column of control points for each ribbon. The degree of each term in the blending functions needs to be just quadratic.

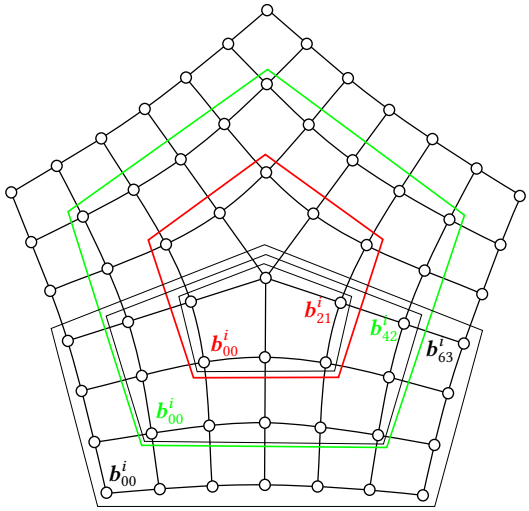


Figure 5.3: *The control point layout for extraordinary vertex patches up to quartic patches for a pentagonal patch (valency five). The red area denotes the control points used in a quadratic patch and the green for a cubic patch, the rest of the control points are used in quartic patches. The black areas enclosed by their respective degrees denote the control points used in a single ribbon.*

A biquartic generalisation would have to be created from B-spline ribbons which consist of three pieces each. Each ribbon consists of four rows of control points with seven columns each. This time the first three rows of control points can be used with standard uniform quartic B-spline basis functions. Again we need to extend the interval and can do so by scaling our local parameters $\bar{s}_i = 3s_i$ and $\bar{h}_i = 3h_i$. For the last row we need to again determine an extended basis function that joins with C^3 continuity at the start of the interval and vanishes up to the third derivative at $u = 3$. We need a septic function to be able to do so.

The generalisation follows a pattern where increasing the degree requires both more pieces per ribbon surface as well as scaling the parametric domain. With this comes the increased separation between extraordinary vertices. Using this we can develop a general formulation of a multisided B-spline of arbitrary degree.

The extraordinary vertex patch ribbons are piecewise surfaces composed out of $d - 1$ polynomial pieces. Each piecewise surface has $d - 1$

rows and d columns of control points, totalling $d \times (d+1)$ points. Then the ribbon definition is

$$\mathbf{R}_i^d(u,v) = \begin{cases} \sum_{j=0}^{d+0} \sum_{k=0}^{d-1} v_{jk}^i \mathbf{b}_{jk}^i N_{j-0}^d(u-0) E_k^d(v) & u \in [0,1), \\ \vdots & \vdots \\ \sum_{j=e}^{d+e} \sum_{k=0}^{d-1} v_{jk}^i \mathbf{b}_{jk}^i N_{j-e}^d(u-e) E_k^d(v) & u \in [d-2, d-1], \end{cases}$$

where $e = d-2$ and $v \in [0, d-1]$. The functions E_{d-1}^d restricted to the last knot-interval $[d-2, d-1]$ can be represented as Bézier polynomials of degree $2d-1$ for which the first $d-1$ coefficients c_i can be found by solving the system $\mathbf{PQx} = \mathbf{b}$, where

$$\mathbf{P} = \begin{bmatrix} 1 & & & \\ & \prod_{i=1}^{1}(2d-1-i) & & \\ & & \ddots & \\ & & & \prod_{i=1}^{d-1}(2d-1-i) \end{bmatrix},$$

$$\mathbf{Q} = \begin{bmatrix} 1 & & \\ (-1)^1 \binom{1}{0} & (-1)^2 \binom{1}{1} & \\ \vdots & \ddots & \\ (-1)^1 \binom{d-1}{0} & \cdots & (-1)^d \binom{d-1}{d-1} \end{bmatrix}$$

and

$$\mathbf{x} = \begin{bmatrix} c_0 \\ \vdots \\ c_{d-1} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} N_{d-1}^{2d-1}|_{u=d-2} \\ \vdots \\ \frac{d^{d-1} N_{d-1}^{2d-1}}{du^{d-1}}|_{u=d-2} \end{bmatrix}.$$

Again, the remaining coefficients can be set to zero so that all derivatives vanish at the end of the interval and finally the function needs to be shifted by $d-1$ intervals. In the quadratic case, the extended basis function is $E_1^2(u) = \frac{1}{2}(1-u)^3 + 5(1-u)^2 u$, and in the quartic case we get

$$E_3^4(u) = \begin{cases} N_3^4(u), & u \in [0,2), \\ \frac{11}{24}(1-v)^7 + 7\frac{65}{168}(1-v)^6 v + & \\ 21\frac{103}{336}(1-v)^5 v^2 + 35\frac{383}{1680}(1-v)^4 v^3 & u \in [2,3], \end{cases}$$

where $v = (u-2)$. Naturally, $E_j^p(u) \equiv N_k^p(u), \forall p \in [0, d-2]$. Figure 5.4 shows the extended basis functions for quadratic, cubic and quartic patches.

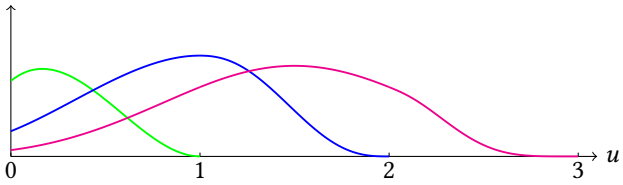


Figure 5.4: *The quadratic (green), cubic (blue) and quartic (magenta) extended basis functions used in the construction of the multisided B-spline EV patches.*

A patch of bi-degree d needs to blend $d-1$ rows and columns of its control mesh functions, and thus the following blending function distribution is applied

$$v_{jk}^{i} = \begin{cases} \alpha_i^d & j < d, \\ \alpha_i^d \beta_i^d & j = d, \\ \beta_i^d & j > d. \end{cases}$$

The complete patch definition is then

$$\mathbf{S}^d(\boldsymbol{\phi}) = \sum_{i=0}^{n} \mathbf{R}_i((d-1)s_i, (d-1)h_i).$$

Internally the patches are C^{d-1} continuous, and connect with G^{d-1} continuity to adjacent B-spline patches (see Section 5.5).

5.4.2 *Extraordinary Face Patches*

We can use the same ideas to span patches over extraordinary faces. However, we have to take into account some key differences. First of all the patches are composed out of an increased number of pieces compared with the vertex patches. This does not have to be a downside as this increased separation leads to an increased parametric interval leading to all the first d basis functions having vanished at the end of it. This means we attach the standard uniform B-spline basis functions for all control points of the patch. An extraordinary face B-spline ribbon consists out of d pieces and has d rows and $2d$ columns of control points. Figure 5.5 shows the control net for these patches for quadratic up to quartic degrees. The ribbon definition is

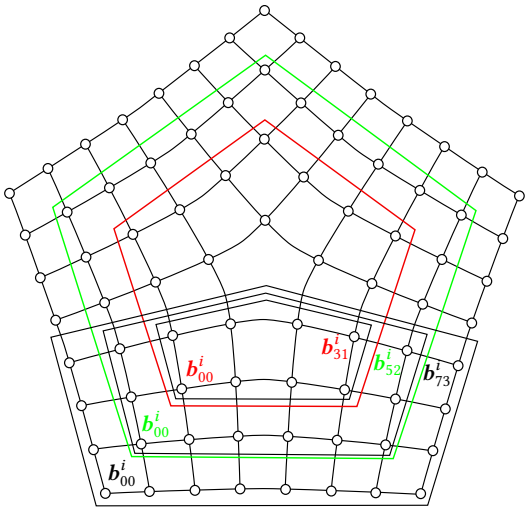


Figure 5.5: *The control point layout for extraordinary face patches up to quartic patches for a pentagonal patch (valency 5). The red area denotes the control points used in a quadratic patch and the green for a cubic patch, the rest of the control points are used in quartic patches. The black areas enclosed by their respective degrees denote the control points used in a single ribbon.*

$$\mathbf{R}^d_i(u,v)=\begin{cases} \sum_{j=0}^{d+0}\sum_{k=0}^{d-1} \omega^i_{jk}\mathbf{b}_{jk}N^d_{j-0}(u-0)N^d_k(v) & u\in[0,1), \\ \vdots & \vdots \\ \sum_{j=c}^{d+c}\sum_{k=0}^{d-1} \omega^i_{jk}\mathbf{b}_{jk}N^d_{j-c}(u-c)N^d_k(v) & u\in[d-1,d], \end{cases}$$

with $c = d-1$ and $v \in [0,d]$. The ribbons use the (same) blending functions

$$\omega^i_{jk} = \begin{cases} \alpha^d_i & j \le d, \\ \beta^d_i & j > d, \end{cases}$$

but this time there is no overlap in the blending functions of the centre column of control points. For a degree d patch the extraordinary faces should be separated by at least $d-1$ layers of regular faces. This time the local parameters s_i and h_i need to be scaled by d to encompass the extended intervals, resulting in the patch definition

$$\mathbf{S}^d(\boldsymbol{\phi}) = \sum_{i=0}^{n} \mathbf{R}_i(ds_i, dh_i).$$

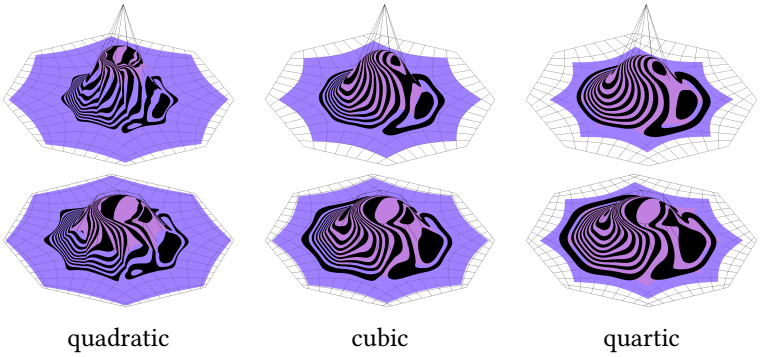


Figure 5.6: *Reflection lines visualisations at an octagonal extraordinary vertex (top row) and face patches (bottom row) for different degrees of multisided B-spline patches.*

5.5 CONTINUITY

The d-degree multisided B-spline patches are G^{d-1} continuous with respect to neighbouring regular regions and other multisided B-spline patches. This can easily be seen from the construction of the B-spline ribbons and their associated blending functions. We want to show that the only ribbon that contributes positionally and differentially towards the patch on the boundary Γ_i is $\mathbf{R}_i$. Let us examine a control point towards the left side of ribbon $\mathbf{R}_i$, which is shared with $\mathbf{R}_{i-1}$, and its associated basis function:

$$\alpha_i^d B(s_i, h_i) + \beta_{i-1}^d C(s_{i-1}, h_{i-1}), \tag{5.2}$$

where

$$\begin{aligned} B(u,v) &= N_j^d(u) E_k^d(v) \quad \text{of} \quad \mathbf{R}_i \quad \text{and} \\ C(u,v) &= N_{d-k}^d(u-1) E_j^d(v) \quad \text{of} \quad \mathbf{R}_{i-1}. \end{aligned}$$

The blending functions α_i^d and β_i^d have the following properties on Γ_i

$$\begin{aligned} \alpha_i^d\big|_{\Gamma_i} = 1, \quad \partial^j \alpha_i^d\big|_{\Gamma_i} = 0, \quad \forall j > 0, \\ \beta_{i-1}^d\big|_{\Gamma_i} = 0, \quad \partial^j \beta_{i-1}^d\big|_{\Gamma_i} = 0, \quad \forall j > 0, \end{aligned}$$

where the derivative is taken in some direction not parallel to s_i. By taking the p-th derivative ∂^p of (5.2) in a particular direction not parallel to s_i on Γ_i, we thus obtain

$$\begin{aligned} &\partial^p(\alpha_i^d) B(s_i, 0) + \alpha_i^d \partial^p(B(s_i, h_i))|_{h_i=0} + \partial^j(\beta_{i-1}^d) C(1, h_{i-1}) + \\ &\beta_{i-1}^d \partial^j(C(s_{i-1}, h_{i-1}))|_{s_{i-1}=1} \\ =&0 \cdot B(s_i, 0) + 1 \cdot \partial^p(B(s_i, h_i))|_{h_i=0} + 0 \cdot C(1, h_{i-1}) + \\ &0 \cdot \partial^j(C(s_{i-1}, h_{i-1}))|_{s_{i-1}=1} \\ =&\partial^p(B(s_i, h_i))|_{h_i=0}. \end{aligned}$$

This is exactly the p-th derivative of the basis functions of the ribbon $\mathbf{R}_i$, which is what we set out to show. This shows that only the ribbon associated with Γ_i contributes to this side, and that the multisided patches will reproduce the derivatives of their input B-spline ribbons. This process also happens analogously with respect to $\mathbf{R}_{i+1}$, and therefore the patches connect with G^{d-1} continuity. Internally the patches are C^{d-1} because the B-spline ribbons they are constructed from are C^{d-1} continuous.

5.6 RESULTS

This section shows various comparisons with existing techniques and explores avenues to decrease some of the deficiencies of the proposed multisided constructions.

5.6.1 *Challenging Cases*

As Catmull-Clark subdivision surfaces are one of the most used representations for B-splines surfaces of arbitrary topology we want to look at how our multisided cubic patches compare to them. In Figure 5.7, we show several challenging extraordinary vertex settings. Each one of these geometries requires a single Catmull-Clark subdivision step to be able to generate a cubic multisided patch, which is then surrounded by a strip of regular cubic patches. We compare the resulting patches with the limit surface obtained through subdivision. Internally, our patches are smooth and show no unexpected changes in the reflection lines, especially around the extraordinary region. In this region we can also see the main difference with respect to the Catmull-Clark limit surface, which is only G^1 in that region and shows a pinching artefact that is noticeable in the reflection lines. Our patches are C^2 everywhere and connect with G^2 continuity to the surrounding regular regions.

5.6.2 *Arbitrary Topology Meshes*

The ability of our patches to create smooth B-splines surfaces is shown for different surface with multiple extraordinary vertices and faces in Figure 5.8. The multisided patches are even able to be joined to each other with expected smoothness given the underlying degree of the B-spline scheme used.

5.6.3 *Effect of Normalisation*

Like the generalised Bézier patch, we use normalisation to create patches that are affine invariant. However, the normalisation has some effect on the shape of the inside of the patches. This is mainly due to the

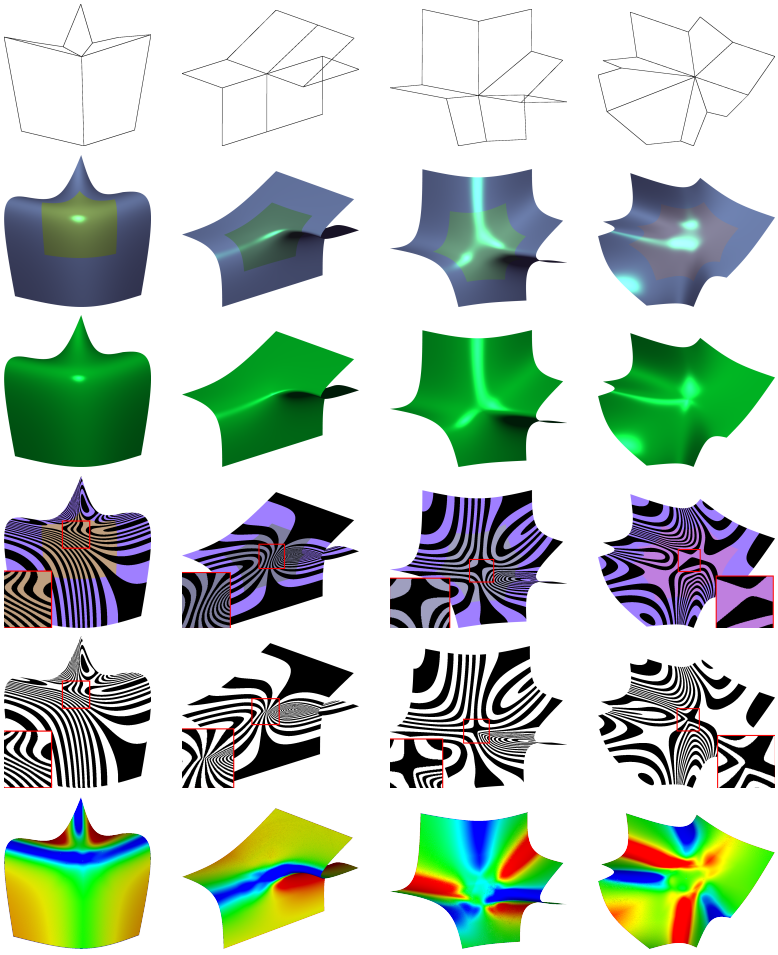


Figure 5.7: *Several different mesh geometries with an extraordinary vertex for which a multisided generalised B-spline patch is generated. From top to bottom: Control mesh, shaded multisided B-spline patch and surrounding regular patches, shaded Catmull-Clark limit surface, reflection lines on the multisided B-spline patch and surrounding regular patches, reflection lines on the Catmull-Clark limit surface, mean curvature of the multisided B-spline patch (from red (negative curvature) via green to blue (positive curvature)). Observe that our construction does not suffer from pinching artefacts near the extraordinary points.*

difference in total weight sum throughout the patches. In Figure 5.6 we show and highlight the effect that normalisation has on the centre of octagonal patches of varying degree. We can observe in the quadratic case that the patches tend to flatten off towards the centre of the patches. For higher degree patches it becomes less prominent, but still notice-

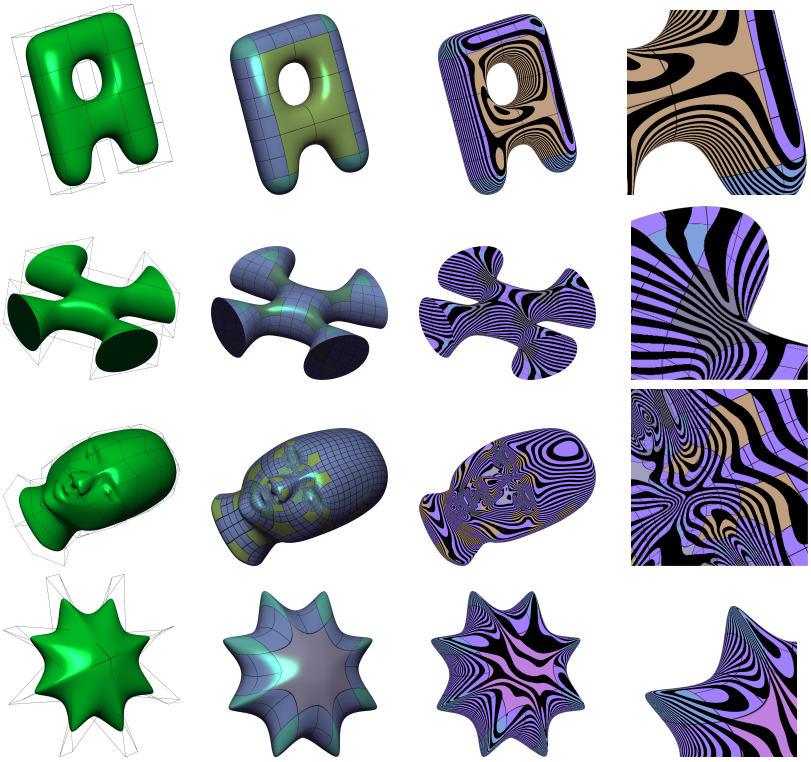


Figure 5.8: *Left: Shading and control mesh. Middle left: Shading with patch boundaries. Middle right: Reflection lines and patch boundaries. Right: Detail of reflection lines and patch boundaries. Top to bottom:* 12 *triangles,* 64 *quadrilaterals,* 12 *pentagons;* 8 *triangles,* 248 *quadrilaterals,* 8 *hexagons;* 84 *triangles,* 1656 *quadrilaterals,* 42 *pentagons,* 18 *hexagons,* 2 *heptagons;* 16 *triangles,* 64 *quadrilaterals,* 2 *octagons.*

able. The effect is more stringent for extraordinary face patches, as can be seen from the bottom row of Figure 5.6.

To reason about why this happens we have to look at the distribution of the weights $f(\boldsymbol{\phi})$ (see Equation (5.1)) over the inside of the patch. In Figure 5.9 we show the sum of weights $f(\boldsymbol{\phi})$ for all points of the patches. We can notice two things, the deficient weight increases both with valency and with degree for EV and EF patches. The deficient weight is a major factor that influences the shape of the patches and it seems that the shape of the patches, or at least the flatness, can be improved by having improved weight functions. Using this observation, we have investigated and designed several special basis functions that strive to improve the shape of the patches.

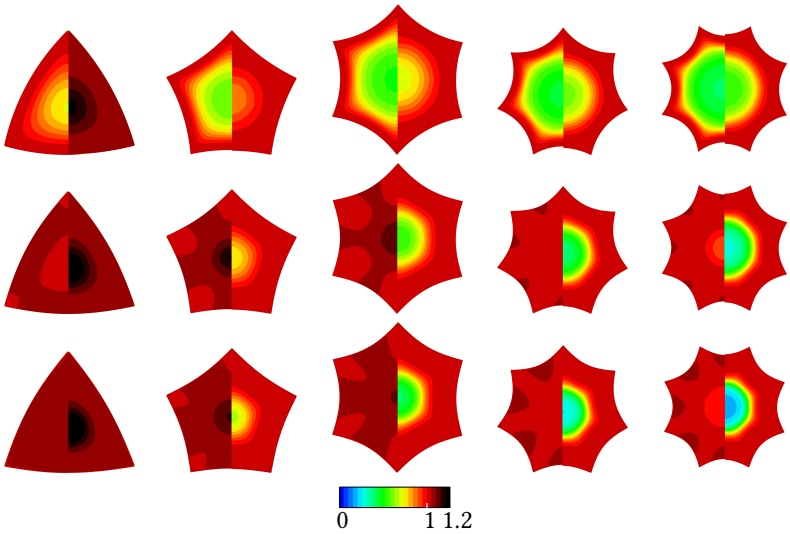


Figure 5.9: Weight deficiency/excess for triangles, and pentagons up to octagons. Left to right column: the quadratic, cubic and quartic case. Each image shows the sum of weights for EV patches on the left and EF patches on the right.

5.6.4 *Shape Adjusted Basis Functions*

The extended basis functions have to adhere to a few requirements, such as the joining with C^{d-1} continuity at the start of their interval and vanishing up to the $d-1$-th derivative at the end. They remain positive everywhere and internally are also C^{d-1}. With these constraints in mind we can design different functions that increase the weight at the centre of the interval. In this way we can increase the contribution of different control points throughout the patch and can tune the shape of the patches, without losing any of the continuity properties internally and externally. As before these adjusted functions are only attached to the h_i parameter of each ribbon, changing only the surface following the parametric direction going towards the inside the patch.

As quadratic patches have only few functions to adjust and the effect of the normalisation is the most prominent, we start with them. We have created a parametrised function that uses ρ, to control added weight to the patch. With certain settings of the parameter it can even lead to excess weight. The centre function for a quadratic EV patch is a C^1 piecewise cubic function on $[0,1]$ (see Figure 5.12), defined as

$$V_1^2(u) = \begin{cases} \frac{1}{2}(1-2u)^3 + 4(1-2u)^2u + 3\rho(1-2u)(2u)^2 + \rho(2u)^3 & u \in [0, \frac{1}{2}], \\ \rho(2-2u)^3 + 3\rho(2-2u)(2u-1) & u \in [\frac{1}{2}, 1]. \end{cases}$$

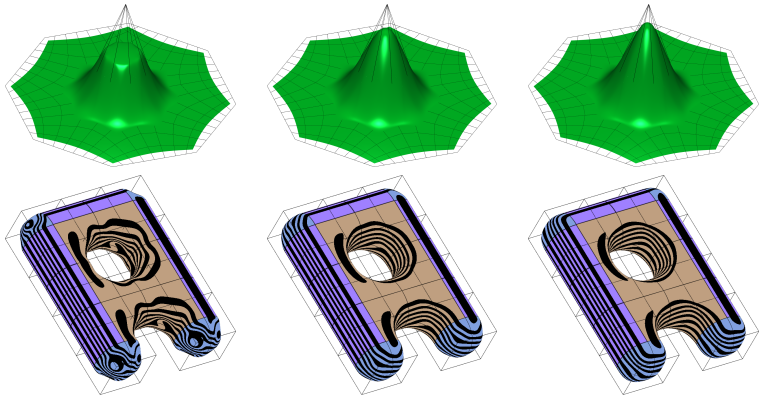


Figure 5.10: *Top row: A quadratic octagonal EV patch. Bottom row: An object with several valency 3 and 5 quadratic EV patches. From left to right: Standard extended basis functions, with centre functions with* $\rho = \frac{3}{4}$*, and with* $\rho = 1$.

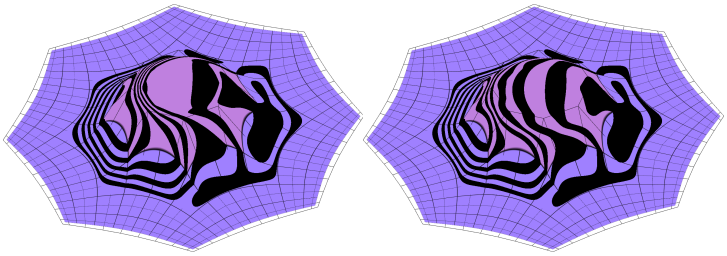


Figure 5.11: *Reflection line renderings of a quadratic EF patch. Left: with standard extended basis functions. Right: using* $F_0^2(u)$ *and* $F_1^2(u)$.

In Figure 5.11, we show the effect of using our hand-crafted functions and varying the values of ρ. We can see that by increasing the parameter value of ρ we can improve the fullness of the quadratic extraordinary vertex patches. Increasing the weight of a function towards the centre of the interval seems like a good means to alleviate flatness. To this end we have also created adjusted functions for higher degree than quadratic extraordinary vertex and faces patches. Again some of these functions are parametrised to adjust the functions at will.

We first define

$$B[a_0, a_1, \ldots, a_d](u) = \sum_{i=0}^{d} a_i B_i^d(u)$$

with $B_i^d(u)$ the Bernstein polynomials of degree d. In other words, the sequence $[a_0, a_1, \ldots, a_d]$ are the control values of the polynomial in the Bernstein-Bézier form.

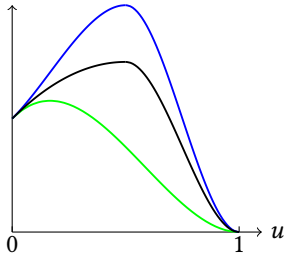


Figure 5.12: *The proposed cubic function $V_1^2(u)$ attached to the central control point in quadratic EV patches. The original $N_1^2(u)$ is shown in green, the adjusted function with $\rho = \frac{3}{4}$ in black, and the adjusted function with $\rho = 1$ in blue.*

We create the following basis functions for quadratic extraordinary faces patches:

$$F_0^2(u) = \begin{cases} B[\frac{1}{2}, \frac{5}{6}, b, a](u) & u \in [0,1], \\ B[a, 2a-b, 0, 0](u-1) & u \in [1,2], \end{cases}$$

and

$$F_1^2(u) = \begin{cases} B[\frac{1}{2}, \frac{1}{2}, \bar{b}, \bar{a}](u) & u \in [0,1], \\ B[\bar{a}, 2\bar{a}-\bar{b}, 0, 0](u-1) & u \in [1,2]. \end{cases}$$

We chose the parameters $a = 0.6, b = 0.8$ and $\bar{a} = 0.1, \bar{b} = 0.23$. No matter what values are chosen for a and b, the functions are always C^1. The functions are this time attached to not only the centrally placed control points, but rather all extended basis functions in the h_i direction are replaced by these new expressions. The functions are shown in Figure 5.13.

We created the following expression for the adjusted extended basis function as a piecewise quintic Bézier function with coefficients for cubic EV and EF patches respectively:

$$V_2^3(u) = \begin{cases} B^5[\frac{1}{6}, \frac{4}{15}, \frac{5}{12}, \frac{55}{100}, \frac{4}{5}, \frac{6}{5}](u) & u \in [0,1], \\ B^5[\frac{6}{5}, \frac{12}{5} - \frac{4}{5}, \frac{12}{5} - \frac{55}{100}, 0, 0, 0](u-1) & u \in [1,2], \end{cases}$$

and

$$F_2^3(u) = \begin{cases} D_2^3(u) & u \in [0,1], \\ B^5[\frac{2}{3}, \frac{2}{3}, \frac{17}{30}, \frac{2}{5}, \frac{2}{5}, \frac{2}{5}](u-1) & u \in [1,2], \\ B^5[\frac{2}{5}, \frac{2}{5}, \frac{2}{5}, 0, 0, 0](u-2) & u \in [2,3]. \end{cases}$$

The functions are shown and compared to the normal extended basis functions in Figures 5.14 and 5.15, respectively.

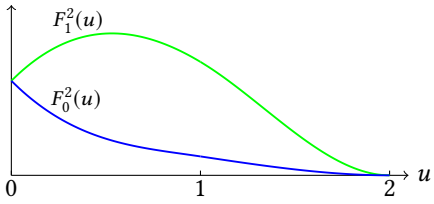


Figure 5.13: *Adjusted extended basis functions for the centrally placed control points of quadratic EF patches: $F_0^2(u)$ (blue) and $F_1^2(u)$ (green).*

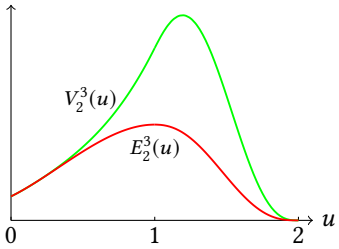


Figure 5.14: *Adjusted extended basis function for the central control point of cubic EV patches. $V_2^3(u)$ (green) and $E_2^3(u)$ (red).*

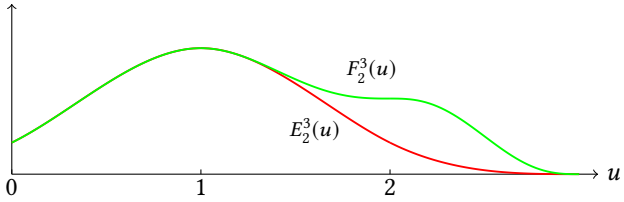


Figure 5.15: *Adjusted extended basis function for the ring of central control points of cubic EF patches. $F_2^3(u)$ (green) and $E_2^3(u)$ (red).*

Naturally, the same process can be done for higher-degree patches as well. However, as their extended basis functions consist out of more individual polynomial pieces, the functions have a more complicated definition, due to a higher number of derivatives that have to be matched. Furthermore, the shape of the functions must be considered carefully, as oscillating or otherwise poorly shaped functions will also show clear changes in the resulting patches.

In Figure 5.11, we show the difference between using the standard extended basis functions or using our hand-picked ones for quadratic EF patches. And Figure 5.16 shows the difference between the standard extended cubic basis functions and our adjusted ones. In all cases the flatness towards the centre is alleviated and improved fullness is achieved.

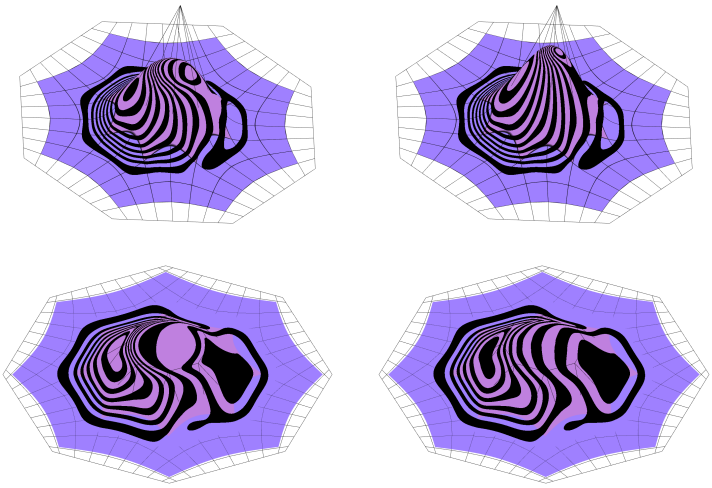


Figure 5.16: *Reflection line renderings of, top row: octagonal extraordinary vertex patch, bottom row: octagonal extraordinary face patch. Left column: without adjusted centre basis functions and, right column, with adjusted centre basis functions.*

5.7 DISCUSSION

We have seen that it is possible to create functions that can adjust the shape of the patches internally without ruining the continuity with respect to adjacent patches or internally. However, even with these functions weight deficiency remains an issue, as the flatness can never be alleviated perfectly, without having small undulations or other perturbations in the surface. Creating these special functions requires careful consideration, and do not provide a unified best setting. In some geometrical settings an adjusted basis function might provide a pleasing surface whereas in other it does not do so. The designer would have to carefully tune parameters to get the best result. This ruins the simplicity of the patches, which we believe should remain as intuitive and simple as standard tensor-product B-spline patches. The use of Bézier functions and a smaller interval for the ribbons of [Vai21] might provide better shape properties than our approach with extended basis functions. Our designed functions provide only one of the possibles choices for these functions. As mentioned in Section 5.6, as long as the functions vanish at the end of the interval and they connect with sufficient smoothness at everywhere else, they are valid functions. By tuning all functions carefully it might be possible to minimise the weight deficiency for all positions on the patch, but this can become a challenging task, due to large number of parameters involved.

Usability of our patches might also be limited by the topological constraints we set for the input meshes, as these require a certain separation of extraordinary regions. Of course, this is simply fixed by applying a few initial steps of general degree subdivision, but may provide to be counter-intuitive whilst designing meshes.

5.8 RENDERING MULTISIDED PATCHES

We have seen that splines provide a very useful means to represent and edit curves and surfaces. An important part in working with these surfaces for designers is the ability to instantly manipulate these surfaces through adjusting their parameters. With this it becomes important to be able to efficiently render these surfaces. The parametric nature of splines allows them to be evaluated efficiently, because many parametric positions on the spline can be evaluated in parallel and afterwards can be connected topologically into a linear approximation of the spline, such as a triangulation. The spline can be said to be *tessellated* into linear triangles

This embarrassingly parallel workload has led to dedicated hardware components on the GPU that generate points efficiently and in parallel. This process is known as hardware tessellation, and is a powerful tool for evaluating parametric curves and surfaces. It has been used in game engines through the techniques of Phong Tessellation [BA08] and PN-triangles [VPBM01]. These techniques augment the standard rendering pipeline, where vertices and normals are sent to the GPU, and Bézier patches are constructed and evaluated on-the-fly. In this way, richer geometry is created without adding any additional data which may congest the CPU-GPU bus. Other ways in which hardware tessellation may help in creating richer geometry is through displacement mapping [SKU08], where a smooth normal field is displaced according to some function (see Section 7.3, where we apply procedural noise functions with displacement functions) or through height values stored in a texture [NL13].

The tessellation pipeline of modern GPUs is fully programmable, meaning that tessellation is only defined in an abstract manner. It is possible to tessellate three different abstract patch types which are *triangles*, *quadrilaterals* and *isolines* and this is reflected in which type of parametric coordinates are supplied by the GPU. These are either barycentric coordinates, for triangles, or (bi)linear coordinates for quadrilaterals and isolines. Tessellation happens in between the vertex and fragment shading stages of the rendering pipeline. There are two parts to tessellation the control stage (TCS) and the evaluation stage (TES), each with their respective shaders. In the Direct3D pipeline these stages are called the hull- and domain-shading stages and are equivalent to their OpenGL counterparts. Typically, the control stage is used to set up a patch, e.g. set up the control points in a set order so that the patch can be evaluated

easily. In addition, the tessellation levels are set. These levels determine the number of generated points for each of the individual edges and the inside of the patch. Then the tessellation primitive generation, a fixed-function stage, will generate points on the patch and connect them into primitives, such as triangles and lines. Each of the generated points is at that point in the pipeline only a parametric position on the patch. The parametric position is the input of the evaluation stage along with the created patch of the control stage. Then a position on a patch, through for instance the de Casteljau algorithm can be computed and output to the fragment shader stage along with anything else that needs to be output. In this sense the tessellation stage of the rendering pipeline is versatile, but it is only geared for those few abstract patch types. For multisided patches, no such abstract patch type is available.

The multisided B-spline patches of the preceding chapters have good visual quality and continuity properties, but if they are not able to be rendered effectively they are of little use. This also holds for other interesting multisided patch schemes that have been developed in the past [LD89] and recent years [VSK16]. Even though, there have been many such techniques, multisided patches have not seen wide adoption, which might in part be due to the inability to render them effectively. Over time, various methods have been proposed to render multisided patches. Instancing and geometry shaders were used to render a special 5-sided patch called the P_m patch [MNP08]. Similarly, geometry shaders were combined with *transform feedback* to render multisided patches [LWZ11]. However, it is known that geometry shaders do not scale well, due to them not being able to be scheduled effectively and ruining parallelism. Phong Tessellation and PN-polygons have been generalised to multisided patches using a simple triangulation of a regular domain polygon and the tessellation pipeline.

This section (Section 5.8) proposes several strategies (see Figure 5.18) to augment the existing graphics pipeline with the ability to efficiently render multisided polygons and parametric patches. In Section 5.8.1 an in-depth explanation of the proposed strategies is given, including how they can be used efficiently in the rendering pipeline. In Section 5.8.2, adaptive tessellation for multisided polygons is discussed. After this, in Section 5.8.3, visual and performance comparisons of the strategies are made, and a multisided generalisation of ACC-2 [LSNC09] is given as an application of our methods.

5.8.1 *Tessellation Strategies*

Rendering multisided polygons and multisided parametric patches requires a few steps. First of all we need to define or generate generalised barycentric coordinates for the multisided polygon. Now this is generally only possible when the polygon is planar. For this we create ω as the planar parametrisation domain of an arbitrary (possibly non-planar)

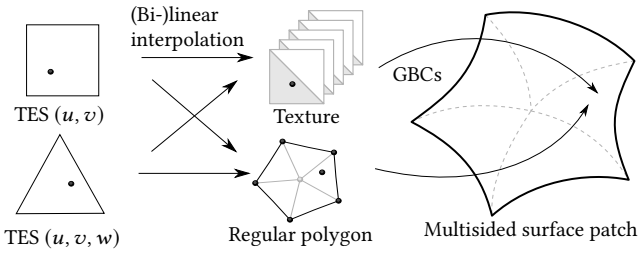


Figure 5.17: *Our tessellation process. Bilinear or barycentric coordinates are used to find a position on a texture or on a regular polygon. From these positions generalised barycentric coordinates (GBCs) can be retrieved or calculated.*

polygon Ω with vertices $\mathbf{v}_i, i = 1 \dots n$. ω has 2D vertices $\mathbf{w}_i$, $i = 1 \dots n$. We can then define a point $\mathbf{p} \in \omega$ for which we can create generalised barycentric coordinates $\boldsymbol{\phi}$. Then we can evaluate a position on Ω using $\mathbf{q} = \sum_{i=1}^{n} \phi_i(\mathbf{p})\mathbf{v}_i$.

The parametrisation domain should be subdivided into polygons that can be processed by the tessellation pipeline. This means either triangulating or quadrangulating the parameter domain, so that each multisided patch can be rendered as a collection of patches. Each one of these sub-polygons can be tessellated through the standard tessellation pipeline, using the regular barycentric or bilinear coordinates to do so. However, this would only create a piecewise approximation to the multisided- polygon or patch. Instead, we use the parametric coordinates to compute a point on a sub-polygon of a regular parametrisation domain and then compute generalised barycentric coordinates for this point using the whole domain, see Figure 5.17. For the subdivision step we created two different pairs of subdivisions, each pair consisting of a triangulation and a quadrangulation. This subdivision step is not done explicitly, but only implicitly through the use of *instanced rendering*. Instanced rendering allows to render the same geometry data multiple times. Then by using the index of the currently rendered instance, we can determine which sub-polygon of the subdivision we are on.

The first strategy is a fanning subdivision of the polygon. This was already described [HK17] for the triangular case. It creates triangles from the first vertex of the polygon towards the other to create $n - 2$ triangles. For polygons with even valencies, such as hexagons or octagons, we can define a fanning quadrangulation so that $n/2 - 2$ quadrilaterals are created. The second strategy makes use of the barycentre to create a symmetrical subdivision around the centre of the parametrisation domain. In this way n triangular sectors or, in the case of a polygon with even valency, $n/2$ quadrilateral sectors are created. A triangular sector is created by connecting two consecutive vertices on the parametrisation

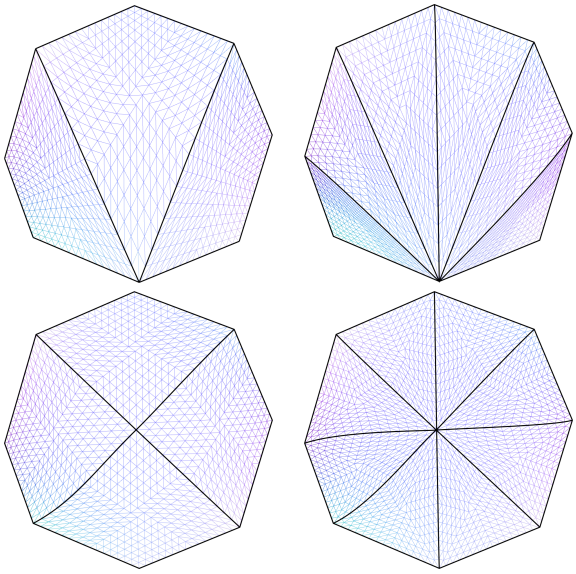


Figure 5.18: *Different tessellation strategies applied to a non-planar octagon. Top row: fanning quadrangulation, fanning triangulation. Bottom row: symmetric quadrangulation, symmetric triangulation.*

domain with the centroid. The quadrilateral case is similar, but takes three consecutive vertices.

As said before we use instancing and the instance index k to determine which sub-polygon should be evaluated. When calculating the generalised barycentric coordinates it means determining the vertices of the sub-polygon using the instance k. When considering triangular subdivisions we can get three vertices of the parametrisation domain using $\mathbf{w}_{(k+i) \bmod n}$, $i = 1, 2$ and $\mathbf{w}_1$ as either the origin of the fan, or the barycentre in the symmetric strategy. For quadrilaterals we can do the same $\mathbf{w}_{(2k+i) \bmod n}$, $i = 1 \ldots 3$ and either $\mathbf{w}_1$ or the barycentre, for the fanning or symmetric strategies, respectively.

The symmetric subdivision strategy has an added advantage, which makes it possible to use textures to store pre-computed generalised barycentric coordinates. Since our parametrisation domain is a regular polygon, it makes the coordinate functions n-fold symmetric. Thus by computing all coordinates for one sector of the polygon, we can determine coordinates on other sectors by shifting coordinate functions. To efficiently store and retrieve these coordinates we make use of 3D textures. Each layer of the texture storing a single coordinate function where a sector is mapped onto the (u, v)-domain of the texture. For triangular sectors this means mapping to the lower part of the domain where $u + v \leq 1$. With this approach is becomes possible to also use more computationally intensive types of generalised barycentric coor-

dinates, such as harmonic coordinates [JMD+07]. However, since the parametrisation domain is only regular Wachspress coordinates will also work great. Then we can again determine the correct order of coordinate functions using the instance number k. We can determine the correct coordinate using $\phi_{(i+k) \bmod n}$.

Unfortunately, it is not possible to have a variable number of input vertices to the tessellation stage. This means we need to create a separate shader program for each valency n of polygon we are rendering. We create a shader for each valency n and store them in an array. Then to efficiently render a model with polygons of different valency we first batch together all vertices of a valency and render these in batches so that we keep the overhead of the switching shaders to a minimum. We do not have to hard-code the parameters valencies into the shader, and allow shaders to be created on demand by replacing valency flags in the shaders when a shader is needed.

5.8.2 *Adaptive Tessellation*

Adaptive tessellation is an additional step one can take to speed up rendering by only tessellating in areas where it will show an actual benefit. Think of the contours of a piecewise linear approximation of a sphere where the individual triangles can clearly be seen, but at the center of the sphere the shading will create a much better illusion of good quality geometry. Techniques such as Phong tessellation and PN-triangles can artificially inflate the geometry through tessellation of Bézier patches, and will be most effective when used around those contours. There are different ways to apply adaptive tessellation to a 3D model. Most often the tessellation factor is determined from the ways vertices, edges, or faces project onto the screen. A commonly-used strategy computes tessellation levels by calculating the projected length of a polygon's edges in screen space [Can11]. This can be used to match the tessellation level with the number of pixels an edge projects to. Other techniques that aim to improve contours of objects can look at the angle between the normal of a vertex and the viewing direction [BA08]. In this way faces that are facing almost away from the viewer are tessellated more densely.

For adaptive tessellation of multisided patches we can use the same principles, but need to proceed differently considering the intermediate subdivision of the patches. Cracks will be generated in the surface by having different tessellation levels for patches that meet at a common edge. Care must be taken that such edges are assigned the identical tessellation levels. For multisided patches this is automatically guaranteed for the boundary edges of the polygon, but not for the spokes of the symmetric subdivision strategies. In that case there exists no centre vertex, as this is implicitly defined by the generalised barycentric coordinates, the subdivision and the patch itself. Therefore we approximate the position of the centre of the multisided patch, by averaging the n vertices

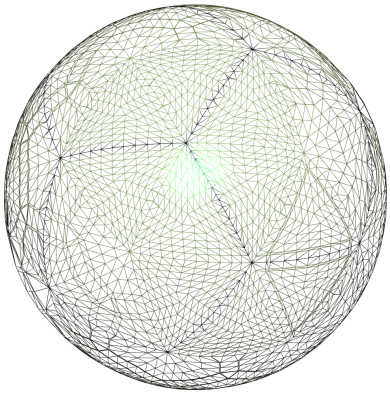


Figure 5.19: *A dodecahedron model using the symmetric triangulation strategy is adaptively tessellated and the wireframe is shown with shading. Difference in tessellation density can be observed, but there are no gaps in the tessellation.*

of the polygon. For higher-order patches, such as the generalised Bézier patch, we can approximate it by taking the average of the most centrally placed control points. It is also possible to evaluate the exact midpoint of a patch. However, at the stage where tessellation values are determined, the control shader stage, it might be difficult to have easy access to all control points of a patch and care should be taken to make them available.

5.8.3 *Results & Performance*

To investigate the quality of the tessellation generated by the four different strategies of subdivision we have computed the discrete mean curvature of the resulting triangulation. In Figures 5.18 and 5.20 we can already see that the two symmetric strategies provide a tessellation that is more uniform. During the calculation of the discrete mean curvature, artefacts are created in regions where the density of the triangulation rapidly changes. This is due to the more variable size of the sub-polygons in the fanning strategies.

PERFORMANCE Although the tessellation pipeline is usually very fast, we introduce more overhead to the rendering process due to the necessity of having to switch shaders between rendering each batch of polygons of the same valency. However, we are still able to render polygons with good performance. In Figure 5.21 we show a performance comparison for the rendering of a complex model containing different multisided polygons. The performance captures were done on a PC with an Intel Core i7-4770K CPU and an NVIDIA GeForce GTX 770 with 4GB

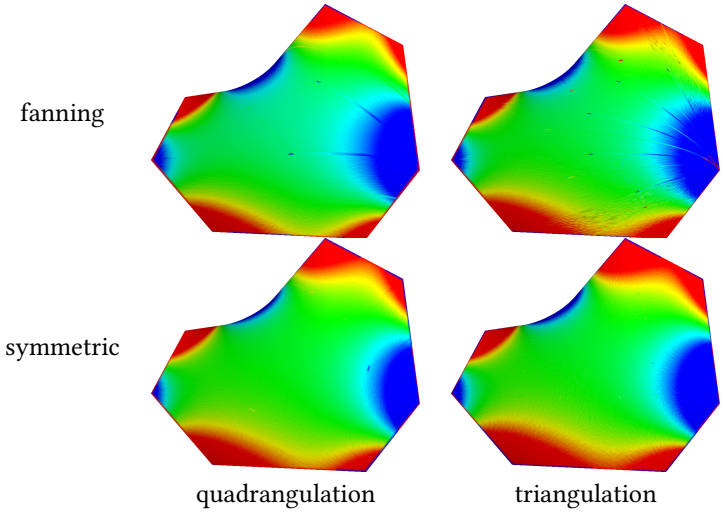


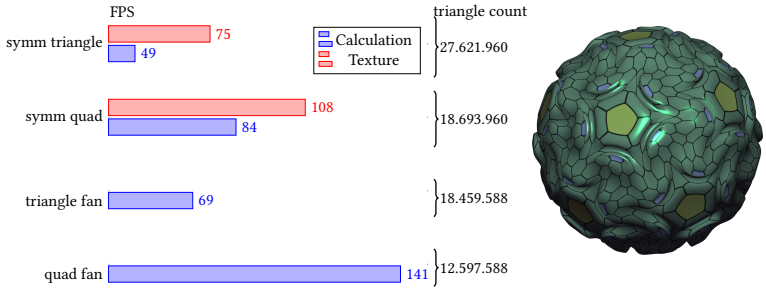
Figure 5.20: Discrete mean curvature [MDSB03] visualised on a non-planar octagon with respect to the different subdivision methods, cf. Figure 5.18. Note that the fanning subdivisions lead to artefacts when estimating discrete mean curvature due to uneven tessellation densities.

of memory running NVIDIA drivers for Ubuntu Linux using OpenGL 4.5. For all captures the maximum tessellation level of 64×64 was used to render the object, no adaptive tessellation was used. As we can see the object is able to be rendered in realtime for all the different strategies.

We would like to examine now how the performance of the different techniques stack up to each other. First of all the results indicate that a substantial performance gain can be achieved when using the texture-based approach of retrieving pre-calculated generalised barycentric coordinates. Secondly the symmetric subdivision of the parametrisation domain performs better than the fanning triangulation. Worst performance is observed when using the fanning triangulation. Naturally, the cause of this difference in performance seems to be caused by the number of generated triangles, which is in turn related to the number of sub-polygons in the subdivision. The symmetric triangulation creates a lot more triangles than the other technique, but is still able to rendered efficiently, especially considering the texture-based approach.

ACC-2 To illustrate the power of multisided patches coupled with tessellation, we have extended the approximate Catmull-Clark scheme ACC2 [LSNC09] to polygonal meshes of arbitrary topology. The original ACC2 scheme approximates Catmull-Clark subdivision surfaces in irregular regions by triangular and quadrilateral Gregory patches. The technique is an efficient means to approximate subdivision surfaces as

Figure 5.21: *Performance comparison in FPS and number of generated triangles: Rendering a model (right) containing 60 quads, 12 pentagons and 750 hexagons using different tessellation strategies at the maximum tessellation level of* 64×64.



the difference between the Catmull-Clark limit surfaces is slight and it can be efficiently evaluated using tessellation shaders. The scheme can easily by extended to higher valency faces by using multisided Gregory patches (see Section 2.3.5.1). The calculation of the vertex, edge and face points remains the same as in the original scheme. Each side of the multisided patches is interpreted as being from a quadrilateral Gregory patch and because of this, G^1 connections can be made in irregular regions. The use of multisided patches removes the need for an extra global Catmull-Clark subdivision step as extraordinary faces can now directly be fitted with approximating multisided patches. Figure 5.22 shows several examples of surfaces that can be generated efficiently that contain many irregularities and extraordinary faces.

5.9 CONCLUSION

The augmentation of the generalised Bézier structure with extended B-spline basis functions have allowed us to generalise it into a multisided B-spline construction for surfaces of arbitrary bi-degree. The surfaces are able to be spanned over extraordinary vertex and also extraordinary faces, by extending the number of intervals used in the cosntruction of the B-spline ribbons. These patches join with G^{d-1} continuity to adjacent regular and other multisided regions and are C^{d-1} smooth internally. The control nets of the patches necessitate the use of a number of steps of arbitrary degree subdivision. The structure of the patches remains true to the their tensor-product counter parts. The patches may flatten off towards the centre of the patches, but we have shown several ways in which the basis functions can be manipulated to create more fullness within the patches. The functions we have determined have only partly alleviated this problem and improving this shape defect remains a challenging problem for future research.

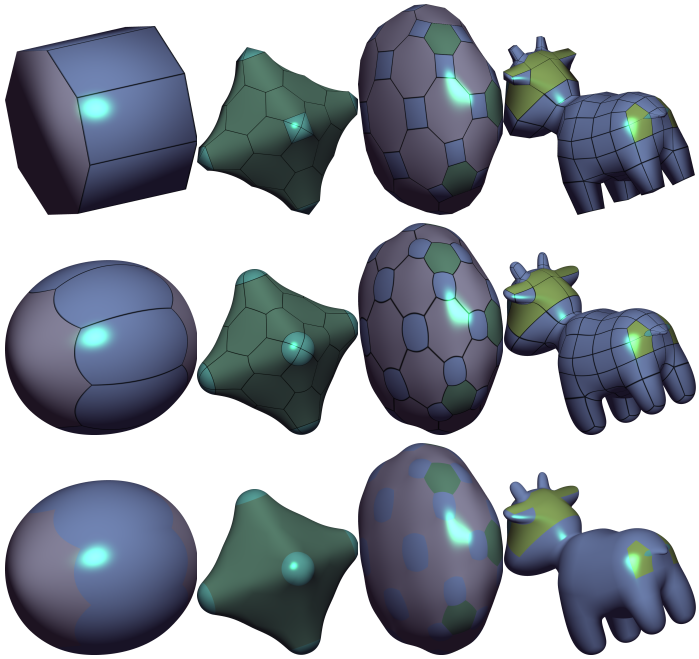


Figure 5.22: *Simple shapes rendered using (generalised) ACC-2 patches where patches have been grouped by colour based on valency. From top to bottom: input mesh, ACC-2 with patch structure, and ACC-2 with Phong shading.*

Multisided patches, such as the aformentioned generalised B-spline patches, can now be efficiently rendered using the existing hardware tessellation pipeline. We have explored different tessellation strategies that subdivide a multisided patch into sub-polygons that can be processed through the modern rendering pipeline. The symmetric techniques create a few more sub-polygons, but are more uniformly tessellated. The fanning techniques present a minimal subdivision of the polygon, but suffer from less uniform geometry. Textures can be used to pre-compute coordinate functions that can be efficiently retrieved during the tessellation process, and improve the speed of it. With this, methods that were previously only defined for traditional patch types such as triangles and quadrilaterals, can be easily extended to multisided patches. We have seen that the ACC-2 technique can be extended to multisided patches and can be rendered in real-time using the modern graphics pipeline. The viability of rendering multisided patches efficiently increases the attractiveness of including multisided faces in polygonal models. This creates freedom for designers and programmers to use such patches in geometric design and even in vector graphics as we will see in Chapter 8.

Part II

Vector Graphics

6 INTRODUCTION

This part of the thesis is about vector graphics. This introduction will first introduce the difference between different ways to represent images, either in raster or vector format. Then we will introduce the problem of converting from raster image format to vector graphics format, and all the requirements and trade-offs associated with it.

6.1 RASTER IMAGES

Most existing digital images are stored in the so-called raster format. Here the image is defined as a grid of picture elements (pixels) [Smi95] where each pixel has a colour. These images can express highly detailed imagery provided that the number of pixels is great enough. An image is reconstructed from the pixels by applying an image reconstruction filter. The most common filters include the nearest-neighbour filter, or linear or cubic filters. There are many different types of digital image formats, but they just provide different ways to encode the raster of pixels.

6.2 VECTOR GRAPHICS

An alternative to these types of images are vector graphics images, which define images in a completely different way. Vector images are represented as a collection of abstract shapes like lines, colour gradients, and other more complex mathematical objects. These objects are called primitives, as they are the building blocks of vector graphics images. The vector primitives are defined relative to a two dimensional Cartesian coordinate system and a colour space. This leads to a resolution independent representation, as for example a line segment can be defined using just two points.

The use of vector graphics is not something new and actually predates wide spread use of digital raster images. Early computer monitors were vector displays or X-Y displays. At the heart of these monitors was the cathode-ray tube (CRT), which could be pointed at desired XY-coordinates of the screen. A ray would fire an electron from the tube and hit the screen on which a fluorescent phosphor-based coating was placed. This coating would then remain lit for a certain time, creating shapes when the beam was moved fast over the screen. By controlling the position of the beam, simple shapes constructed from dots and lines could be drawn. This type of vector display was used to display the pioneering Computer Aided Design (CAD) application called Sketchpad

[Sut64] which was created by Ivan Sutherland for his PhD thesis. The vector display was the favoured display choice up until the mid 1980s, because raster formats where still too expensive to be stored in computer memory at that time. Vector display also saw its use in games, most notably in the arcade game industry. The Vectrex [MB07] was a home video game console with a vector display where people could play an Asteroids-clone (Atari, Inc, 1979) called MineStorm. Most of the arcade games of the time featured vector displays, but the vector display ultimately gave way to the more versatile raster display [Rub98]. Although the CRT was still used it was now used to scan along the screen in a horizontal fashion instead of visiting arbitrary locations.

Vector graphics did not die out with the demise of the vector display. On the contrary, it lived on as the 2D variant of computer graphics, but it did require an intermediate step for displaying images on a raster-based display called rasterisation. Rasterisation is the process of scan-converting vector primitives so that they can be displayed on a raster. For example rasterising lines entails finding which pixels of the screen correspond to the line segment, but it will generate a jagged representation when the line is placed diagonally.

Vector graphics saw its use in the specification of digital documents and most notably in the definition of fonts. These fonts use curves to define the outline of the font, most notably the TrueType uses quadratic Bézier curves and OpenType uses cubic Bézier curves [Cor20]. The letters you are reading right now are defined using the TrueType representation and depending on whether you are reading this in digital format or not you could try and zoom in to check whether the fonts retain the same quality.

Fonts are not the only popular application of vector graphics nowadays. Vector graphics are used everywhere in the graphical design industry because of the aforementioned advantages. The primitives are easy to manipulate and thus facilitate easy editing of images, making them suitable for use by artists, and easy to reuse. The use of these primitives has the advantage that the image can be defined very compactly and the image can be 'scaled' to any resolution. Vector graphics images can be rendered into a raster format at any size, such that high quality images can be retrieved at any resolution. This makes the format very suitable for use in media which have to be printed and displayed at various sizes.

VECTORISATION The vector graphics image format is well-suited to model many different types of image. Unfortunately, there are no cameras available that can capture reality in a vector graphics format and therefore images encountered in the wild are predominantly raster images. A process is needed that converts between a raster based image and a vector image.

The process of conversion from a raster image format to a vector image format is known as image vectorisation. A vector image format is very loosely defined, unlike the raster image which is just a raster of discrete pixels each having a location and a colour. Vector graphics are made up out of vector primitives, and thus vectorisation can be said to be the process of determining a set of vector primitives such that they represent the original image well. Again, the way an image is represented well is also up for debate. There are different degrees and dimensions where we can classify different vectorisations and they depend on what is required of the vectorisation. If we look at the use cases where vector graphics are commonly used we can easily set up requirements for image vectorisation.

Vector graphics should be able to be rendered at any resolution without loss of quality. This is commonly known as resolution independence, and is one of the main advantages of vector graphics over raster images. When displaying a raster image at larger resolution than its base size, the underlying raster structure of the image will become apparent. The use of more advanced image reconstruction filters than simple box-filters [Smi95] might be able to cover some ground, but ultimately they are not able to correctly reconstruct elements such as diagonal lines and curved lines and contours.

Sparseness of the representation is also another factor that comes into play when vectorising images. The sparseness of a vector image can be expressed in the number of primitives needed to represent a raster image. A red disk is easily represented as a position, a radius and a colour whereas the raster version of this image would have to at least cover all red pixels and the pixels not covered by the disk but which are in the bounding rectangle. A vector image is thus deemed to be sparse when it is able to determine the colour of large swathes of pixels of the original image. Sparseness also has effect on other ways in which vector images are an attractive alternative to raster images. The sparseness of the representation influences the file size of the image, making it much more economical to use vector images for example in web-related technologies.

The strength of vector graphics also lies in the ways in which the vector representation can be edited. Take the disk example again, where we can edit the image easily by varying one of the parameters of the disk, e.g. translating the disk or changing its radius. The same operation in the raster version is not easily done and the user is best left to create an entirely new image. It can be important to be able to edit the vector image resulting from a vectorisation. The editability of the vector image is dependent on the number of primitives and how they are ordered with respect to each other. For instance, a pixel curve that is vectorised as a polyline is much harder to edit efficiently than that same curve represented as a B-spline.

Naturally, it is important that the vectorisation captures the likeness of the raster image it tries to represent. This is an important part in assessing the quality of the vectorisation. Due to the complexity of modelling a natural image with vector graphics primitives this can be a hard problem to solve. In the past, due to the limitations of the available vector primitives, shortcuts were taken where parts of images were simply modelled with a single solid colour. This still captures the main pieces of the image, but foregoes on modelling the details of the image. Modelling those would sacrifice the simplicity of the vectorisation as a multitude of small primitives are needed to model tiny image details, jeopardising the sparseness and editability requirements of the vectorisation.

Another important factor in vectorisation is the ability to render the vector image with reasonable performance. A vector image is not an image like a raster image is, and rendering them is often not as straightforward as rendering a raster image. For raster images there exist dedicated texture mapping units on the GPU that make rendering, or mapping them onto objects through texture mapping, very efficient. For vector graphics primitives there exists no such pathways and primitives often have to be approximated through geometric primitives such as triangles, lines and points. It remains important to be able to render the primitives efficiently so that vector images can be used in different forms of media. A vector representation that captures all of the above requirements, but is not able to be rendered efficiently is of little use in interactive environments. With this the performance of the vectorisation process itself is important too. The speed of the process of going from an input raster image to some vector representation can make a method attractive. When this processing time is short it becomes possible to batch vectorise images. In any case the process should not take too long so that vectorised images can quickly be loaded into vector graphics editing software and altered further.

The hard part of vectorisation is combining all these requirements together, as ultimately you are left with a trade-off. A sparse representation can never model all details of an image and similarly a faithfully captured image through the use of a large number of primitives can never be easily edited.

6.3 CONTENTS

In this part of the thesis we first look at how existing vector graphics primitives can be extended to be more expressive. In Chapter 7 we extend the gradient mesh primitive to interpolate additional parameters for procedural noise functions. Then in Chapter 8 we compare and contrast different versions of polygonal gradient meshes. Then finally, an efficient image vectorisation pipeline is described in Chapter 9.

7 NOISY GRADIENT MESHES

Parts of this chapter have been published as

- Gerben J. Hettinga, Rowan van Beckhoven and Jiří Kosinka. "Noisy gradient meshes: Augmenting gradient meshes with procedural noise". In: Graphical Models, 103.

We extend the gradient mesh vector graphics primitive with procedural noise functions. Specifically, we couple Perlin, Worley, and Gabor noise to the gradient mesh. We allow local parameters controlling the noise functions to be defined at the vertices of the mesh. The parameters are interpolated along with the geometry similarly to how colour is interpolated in an ordinary gradient mesh, allowing for spatially varying noise patterns. These noisy gradient meshes facilitate a sparse representation of high frequency regions along with underlying smooth colour gradients. The meshes are easy to edit and efficient to evaluate on graphics hardware, making them a suitable candidate for inclusion in modern vector graphics authoring tools. We demonstrate the utility of our method on gradient meshes with added noise functions. Additionally, we show that the approach can be used in combination with regular surface meshes where noise functions are used to govern their displacement mapping.

7.1 INTRODUCTION

Vector graphics are excellent means for representing images that contain smooth colour gradients and composite objects which are constructed out of simpler shapes which in turn are made out of constant colour or linear gradients. These primitives work great for vectorising image regions that feature constant or slowly changing colour gradients. However, much of the regions depicted in natural images do not conform to this and are instead quite noisy. Many primitives have to be generated to model these regions with some accuracy. This complicates the design or vectorisation process and vector image due to number of primitives needed to manually adjust. A simple primitive which takes care of the main shape of the object and which allows intuitive control over the high-level details within the object is not available.

Procedural noise functions have long been a staple in the computer graphics world to easily add more realism through adding of arbitrary details to simply shaded objects. They have been used to create procedural textures, that model terrain, or to drive displacement mapping, to create a rocky surface [EMP+03]. Most procedural noise functions are easy to define, and require only few parameters to change. In addition, they can be scaled arbitrarily and are thus a great combination with the scalable nature of vector graphics. Procedural noise functions have been combined with vector graphics primitives before. This can be anything from filling a (pathed) region with a procedural noise function or letting procedural noise functions drive blending of primitives. Explicit control for varying the parameters of noise functions over images was approached before [JCW11] where diffusion curves [OBW+08] were used to diffuse parameters of Gabor noise. Likewise diffusion curves were used to smoothly vary parameters for fractal noise [HGA+10] in a terrain generation setting.

In this chapter we combine gradient meshes with procedural noise functions by letting the noise alter the smooth colour gradient generated by the mesh. We also let the meshes smoothly interpolate parameters of different procedural noise functions so that the noise function is smoothly varied over the mesh. This has the advantage that gradient meshes can express arbitrary detail whilst remaining simple geometry-wise. We distinguish local and global parameters, where the local parameters can be varied over the mesh and are assigned to vertices.

The remainder of the chapter is organised as follows. We briefly describe the essentials of a few different procedural noise functions Section 7.1.1. Then, in Section 7.2, we detail the construction of gradient meshes with procedural noise functions. We describe which parameters we use and how they are interpolated. Results of gradient meshes and also of general displacement mapping on surfaces are shown in Section 7.3. In Section 7.4, we discuss limitations and applications. Finally, the chapter is concluded in Section 7.5.

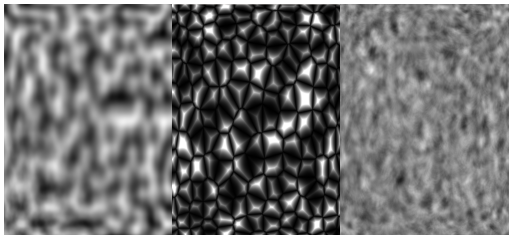


Figure 7.1: *The three different procedural noise functions considered in this paper. Each gives rise to a distinctive noise pattern.*

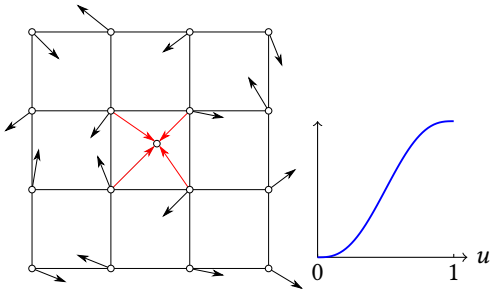


Figure 7.2: *Left: The lattice involved in Perlin noise has pseudo-random gradient vectors defined at lattice points. A point to be evaluated is shown. Right: the quintic function that is used to smooth noise over the edges of the cell.*

7.1.1 *Procedural Noise Functions*

We are going to pair the gradient mesh primitive with three different types of procedural noise functions, Perlin, Worley, and Gabor noise. We chose these three noise functions because they have been mostly standardised, are well-known, have interesting parameters and can all be easily implemented into the fragment shader stage of modern graphics pipelines. Naturally, other noise functions are available [LLC+10], but with these three different types of noise functions we have already covered a lot of what is possible. In Figure 7.1 we show the characteristics of each of these different types of noise function. We now detail how these noise functions are constructed and with that expose some of the parameters which can be useful later.

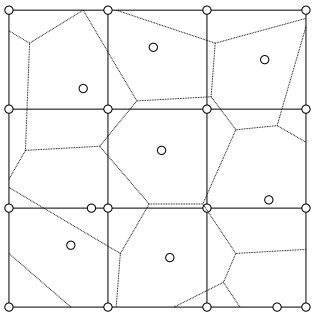


Figure 7.3: *The lattice involved in Worley noise has a feature point within each cell. The pattern created by computing the distance to the closest point is shown with dashed lines (points which are outside the grid have been omitted).*

7.1.1.1 *Perlin Noise*

One of the first true procedural noise functions to be introduced was Perlin noise [Per85], named after its inventor Ken Perlin. It was developed to create a less "machine-like" look for the science fiction movie Tron [Lis82]. The noise function is based on a lattice which is imposed on the uv-plane. For each point of the lattice a pseudo-random noise vector is determined. Figure 7.2 shows a lattice where 2D vectors are shown at lattice points. The process of evaluating the noise function is then done as follows. For each point to be evaluated it is determined which cell of the lattice it resides in. From the cell its four surrounding lattice points and pseudo-random vectors are retrieved. Then the inner product between each gradient vector and a vector pointing from the lattice points towards the evaluated position (the red vectors in Figure 7.2) is computed. These four values are interpolated by a biquintic function (Figure 7.2, right) to create C^2 smooth transitions to neighbouring cells. The frequency of the noise can be increased globally by scaling the uv space of the lattice.

7.1.1.2 *Worley Noise*

Worley noise [Wor96] creates a cellular pattern which is visually similar to Voronoi cells, see Figure 7.1, middle. Although it is a procedural function, it is not exactly a noise function, but rather a texture basis function [LLC+10]. Like Perlin noise it makes use of a lattice, but stores a feature point in each cell instead of a vector at lattice points. Figure 7.3 shows such a lattice with a single feature point per-cell. The placement of feature points in cells happens randomly although a more regular placement of points can be achieved by lowering the *jitter* rate. The

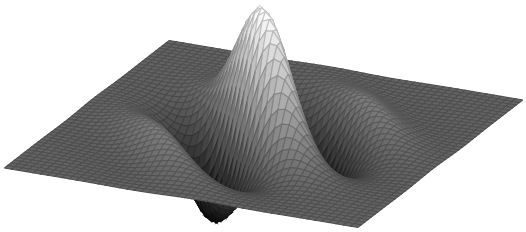


Figure 7.4: *A Gabor noise kernel (see Equation 7.1), which is a Gaussian function multiplied with a 2D cosine function.*

number of feature points can also be varied, but one feature point per-cell is enough to create good looking texture. The basis function F_k are defined as a scalar value based on the k nearest feature points and a distance metric. Figure 7.3 shows the result of F_0 as the dashed contours where $F_0 = F_1$. Different types of shaped textures can be constructed by changing the k value, although the higher the k the more similar the patterns will be. Combinations of different F_k can be used to create even more interesting patterns.

7.1.1.3 *Gabor Noise*

The last type of procedural noise function we are going to cover is Gabor noise [LLDD09]; see Figure 7.1, right. Gabor noise is similar to Worley noise as feature points are also placed arbitrarily into the cells of a lattice. Each feature point is a Gabor kernel and the noise is evaluated as the weighted sum of the kernel functions. Thus the noise in a cell is determined by the kernels in the eight neighbouring cells. The Gabor noise kernels have several useful parameters that influence the appearance of the noise. In addition the parameters can be controlled in ways that influence the statistical properties of the noise allowing it to reproduce different textures [JCW11, GLLD12].

The Gabor noise kernel is a Gaussian kernel multiplied with a 2D cosine function

$$G(u, v) = Ke^{-\pi a^2(u^2+v^2)} \cos(2\pi f(u \cos \omega + v \sin \omega)), \tag{7.1}$$

where K is the magnitude, a is the inverse width of the Gaussian function, f is the frequency, and ω is the orientation of the cosine function. Each of these parameters can be varied over the image plane, leading

for instance to varying frequency of the cosine function. We can change the isotropy ι of the noise by letting the value ω have increased arbitrary deviations, i.e. $\omega = (1-\iota)\omega + \iota(\omega + R)$, where $R \in [0, 2\pi]$ is some random angle.

7.1.1.4 *Fractal Noise*

Gabor noise has a lot of parameters that influence the noise functions. Perlin and Worley noise do not offer as many freedoms or parameters that can be freely varied over the image plane. Therefore, we use fractal Brownian motion, or more commonly known as fractal noise, in combination with the noise functions. Fractal noise combines several octaves of a noise function, by iteratively scaling the noise and diminishing the influence of it. Scaling a noise function leads to an increase in frequency. The persistence value p is used to regulate the prevalence of the successive frequencies of noise. Higher persistence values lead to an increase of visibility of higher frequency components. Figure 7.5 shows the effect of varying the persistence value over a quadrilateral element with Perlin noise, with different numbers of octaves. This parameter was also varied over the image plane in [HGA+10], by diffusing the p value defined at diffusion curves.

7.2 NOISY GRADIENT MESHES

An important step in creating a noisy gradient mesh is to create a parametrisation that can be used to evaluate the procedural noise functions. This parameter domain than contains the lattice for the noise functions. The gradient mesh is a regular rectangular mesh and we can straightforwardly assign texture coordinates to the vertices of the mesh. When considering a gradient mesh of $m \times n$ quadrilateral elements with vertices $\mathbf{v}_{i,j}$, $i = 0, \ldots, m$ and $j = 0, \ldots, n$. We can make this correspond to an $m \times n$ subdivision of a unit square to obtain a parametrised mesh. Thus, the texture coordinates of $\mathbf{v}_{i,j}$ are simply $u_i = i$ and $v_j = j$, which creates a uniform texture space. In this manner the mesh can be glob-

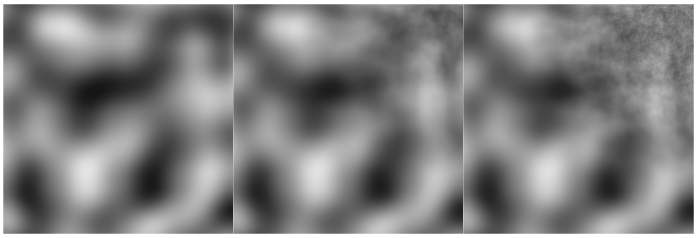


Figure 7.5: *Basic Perlin noise (left). Five (middle) and ten (right) octaves are assigned to the top right vertex of the mesh.*

ally refined without having to reassign texture coordinates and linear interpolation of texture coordinates maintains a continuous parameter domain. The combination of the texture space with the gradient mesh already allows spatial deformation of the noise field. As the noise field now follows the parametric direction of the mesh, changing tangent handles will influence the shape to the patches internally.

To be able to alter the noise field further we expose three local parameters to the user that are defined per vertex of the gradient mesh. Global parameters such as the base frequency of the noise can still be altered by the user, but make no sense on a local scale as they cannot freely be altered over the mesh without artefacts. Three main local parameters are:

- **Persistence:** As mentioned in Section 7.1.1.4, the persistence parameter p regulates the prevalence of higher octaves of noise. By varying this parameter from low to high, an increasing amount of high frequency noise becomes apparent; see Figure 7.5.
- **Opacity:** The opacity value α simply handles the local blending of the noise field with the colour field.
- **Distortion:** We use several secondary functions which manipulate the parameter domain or alter the noise field. The distortion parameter d locally regulates the amount of the applied distortion.

The parameters are attached to the vertices $\mathbf{v}_{ij}$ so that we obtain the tuple $(x, y, r, g, b, p, \alpha, d)$; cf. Section 2.3.6. The handling of the additional parameters happens in the same way as with the colour components of a tradition gradient mesh. The values of the edge and inner control points inherit the value of their logically closest vertex in the bicubic Bézier patch. In this way the noise parameters are also interpolated with C^1 continuity. It is still possible in our case to have sharp transitions also in the noise parameters. The values can be assigned to edges to have a different value for each corner of a surrounding patch.

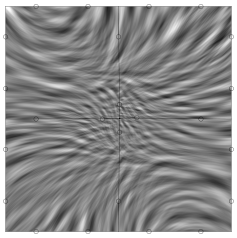


Figure 7.6: *A noisy gradient mesh (4 patches) showing the effect of varying the anisotropy and orientation values for Gabor noise over the mesh.*

The Gabor noise kernels allow for even more parameters to be interpolated. We give the user control over the frequency f and the direction of the kernel ω, see Section 7.1.1.3. The latter can be used to create interesting customisations of the noise field. As the anisotropy value ι is decreased, the noise direction can be more clearly directed with the variable ω on a vertex level. Figure 7.6 show an example of this where the noise field creates patterns reminiscent of vector field visualisations [VW91]. In the simple 2×2 mesh the centre vertex has a higher anisotropy value than the surrounding vertices. The outside vertices have lower anisotropy and are angled towards the centre vertex.

The noise functions are evaluated using the interpolated parameter values and the interpolated (u, v) textured coordinates and one of the three procedural noise functions. The number of octaves of fractal noise can be set for the entire mesh. Worley noise allows also to set the distance metric and which F_k or combinations thereof to use. Again, the noise can be scaled globally by scaling the uv-domain, resulting in visually smaller lattice cells. A global setting for Gabor noise changes the density of kernels in each cell of the lattice, and the width of the kernels. A simple value ranging in $[0, 1]$ is used to control the distortion amount. We experimented with three types of distortion; see Figure 7.8. We use simple coordinate distortions, where the uv parametrisation is distorted by another (noise) function. We can also distort the noise values using the sine or another trigonometric function such that a wave-like characteristic is imposed on the noise field.

Figure 7.7 shows the effect of varying the different local parameters extremely to show how the interpolation affects the noise field. As can be seen in the figure, the noise smoothly varies from low to high frequency noise (left), and also from high amounts of distortion to no distortion at all (right).

Additionally, we allow the noise field to be filtered using standard low-, band- and high-pass filters. We can map colour to the noise values which can additionally be blended using traditional colour blending techniques. In Figure 9.2, we show how the different post-processing effects offer yet another possibility to increase the expressiveness of noisy gradient meshes. In this case, the Perlin noise field values are distorted by a sine function, and a band-pass filter is applied to decrease the width of the marble texture.

.

7.3 RESULTS

The power of the noisy gradient mesh technique is that the mesh can remain simple and the noise function handles the high-frequency details. In Figure 7.9 we show a few such meshes, modelling various objects. The procedural noise functions create tremendously detailed im-

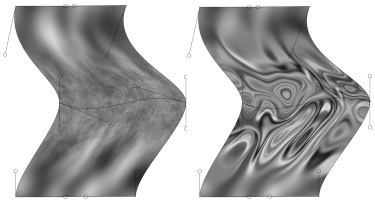


Figure 7.7: *Two noisy gradient meshes showing varying Perlin noise fields. Left: Varying persistence values. Right: Varying distortion amounts. In both examples, the values/amounts are high in the middle, and low at the top and bottom.*

agery that cannot be feasibly modelled with ordinary gradient meshes alone. The detail is also varied over the mesh creating interesting patterns. The meshes can be rendered at any resolution, which would not be possible with (high resolution) raster textures and ordinary texture mapping without showing pixel artefacts. As said before, the noise field follows the geometric directions of the mesh. This can be seen on the dark green bands of the watermelon (Figure 7.9) or on the pattern on the vase (Figure 9.2). This is different that just applying noise on top of the gradient mesh using the evaluated xy position on the mesh as input for the noise function. This would be the same as having a region of a solid noise [LLC+10] which is undesirable. In Figure 7.10, we explicitly show the effect of altering the geometry of the gradient mesh (vertices and gradient handles) on the resulting blended noise and colour field. As can be seen in the figure, the pattern on the top wings is naturally influenced by the change of geometry.

DISPLACEMENT MAPPING We can apply the same principles to generate a noise field that influences the colour of a gradient mesh to influence the geometry of 3D parametric spline patches through displacement mapping. For our base patch we use uniform cubic B-spline patches as the geometry patch, and convert them into a Bézier representation. In this way we can interpolate the parameters in the same way as with gradient meshes. Then the noise parameters are interpolated

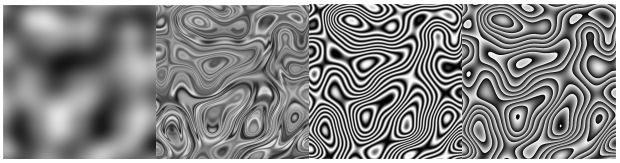


Figure 7.8: *Three different types of distortion applied to the same Perlin noise field. Distortion by a secondary noise field, sine distortion, and wood-like distortion.*

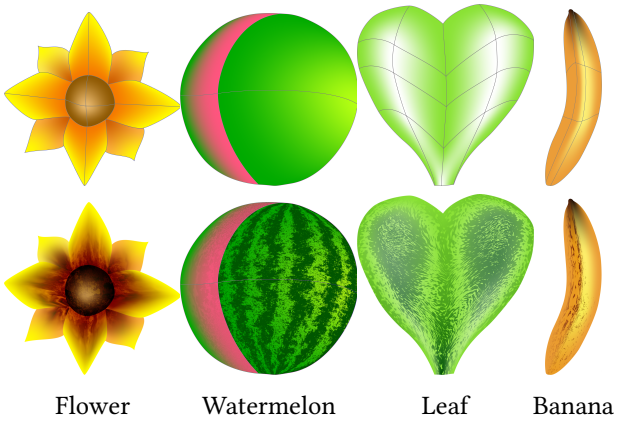


Figure 7.9: *Simple gradient meshes with different types of noise. Top row: Ordinary gradient meshes (with sharp colour transitions). Bottom row: Noisy gradient meshes. The flower model (16 patches) makes use of Worley noise, banana (9 patches) and watermelon (4 patches) meshes make use of Perlin noise, and Gabor noise was used on the leaf (16 patches) mesh.*

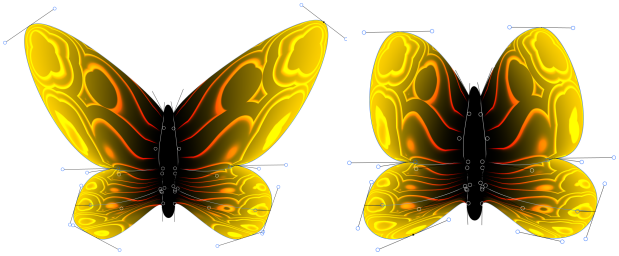


Figure 7.10: *Left: An artistic depiction of a butterfly based on Worley noise. Right: The geometry (i.e., vertex positions and gradients) has been edited to modify the shape of the wings. Note that the noise pattern, whose parameters have not been altered, naturally follows the transformation.*

and are C^1 smooth over the spline, whereas the geometry is C^2 continuous. The interpolation could be relaxed and the noise parameters could also be approximated using the B-spline basis functions, leading to C^2 continuity in parameter values as well. In Figure 7.11 we vary the persistence component over the meshes and the evaluated noise value is used to displace the surface along the normal direction. This creates interesting surface patterns and could for instance be used as a way to control the generation of arbitrary terrains based on a sparse representation of the base geometry. Naturally, other algorithms such as bump mapping or varying characteristics for specularity or transparency can be regulated through the noise fields.

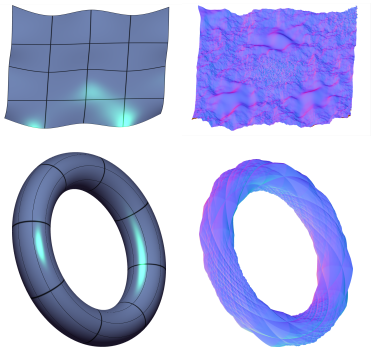


Figure 7.11: *A B-spline surface composed of 16 patches (top row) and a B-spline torus composed of 64 patches (bottom row), both without displacement (left column) and with procedural displacement (right column). The final 'noisy' surfaces are colour-coded according to their normal fields.*

7.4 DISCUSSION

We have presented several straightforward ways to manipulate noisy gradient meshes, but there are many possibilities to explore. Every parameter that influences the noise function can be interpolated, but it depends on the nature of these parameters whether it leads to results that can be intuitively manipulated, or that are usable visually. Node based interfaces or block shaders [AW90] could be used to forge even more complex textures, such as combining different types of procedural noise functions.

We make use of programmable shaders to evaluate the gradient mesh primitives, that handle interpolation of geometry, colour and noise parameters. Fragment shaders are used to evaluate the procedural noise functions, based on those interpolated parameters. This makes our noisy gradient meshes very efficient and they can thus be manipulated in real-time through interactive editing of geometry, colour or local or global noise parameters. We evaluated the performance on a PC with a NVIDIA GTX Titan V graphics card. We can still observe sub-25 millisecond frame times for a full HD pixel-dense rendering of a large gradient mesh, containing 3600 primitives, using Gabor noise with a kernel density of 20. It should be noted that Gabor noise is not continuous when evaluating it using a fragment shader as the Gaussian kernels, with infinite support, are truncated by the lattice boundaries. For Perlin and Worley noise this becomes even sub-10 millisecond rates, for 10 octaves of fractal noise.

The approach can be readily extended to locally refinable gradient meshes [BLHK18] as this still provides C^1 continuity across T-junctions

and the parametrisation will stay intact. It cannot easily be extended to gradient meshes with arbitrary topology [LKSD17] as this requires a more elaborate assignment of texture coordinates to the mesh as this cannot be done through a rectangular subdivision of the uv grid. However, it is still possible to have smoothly varying noise functions if the geometry has at least C^1 continuity.

7.5 CONCLUSION

In this chapter we have presented a simple way to combine procedural noise functions with gradient meshes. The lattice structure of noise functions can be mapped with ease onto gradient meshes, which also have a grid-like structure. By interpolating parameters for the noise functions over the mesh and linking their parametrisation to the gradient mesh, creates abilities to distort and manipulate the noise through the mesh. The user can then create highly detailed and scalable images using a sparse mesh. The noise functions can be evaluated efficiently and can be composed in a multitude of ways through different post-processing, distortion and colour blending modes.

8

COLOUR INTERPOLANTS FOR POLYGONAL GRADIENT MESHES

Parts of this chapter have been published as

- Gerben J. Hettinga, René Brals and Jiří Kosinka. "Colour interpolants for polygonal gradient meshes".

The gradient mesh is a powerful vector graphics primitive capable of representing detailed and scalable images. Borrowing techniques from 3D graphics such as subdivision surfaces and generalised barycentric coordinates, it has been recently extended from its original form supporting only rectangular arrays to (gradient) meshes of arbitrary manifold topology. We investigate and compare several formulations of the polygonal gradient mesh primitive capable of interpolating colour and colour gradients specified at the vertices of a 2D mesh of arbitrary manifold topology. Our study includes the subdivision based, topologically unrestricted gradient meshes [LKSD17] and the cubic mean value interpolant [LJH13], as well as two newly-proposed techniques based on multisided parametric patches building on the Gregory generalised Bézier patch and the Charrot-Gregory corner interpolator. We adjust these patches from their original geometric 3D setting such that they have the same colour interpolation capabilities as the existing polygonal gradient mesh primitives. We compare all four techniques with respect to visual quality, performance, mathematical continuity, and editability.

8.1 INTRODUCTION

The interpolation of colour and the handling of colour in general is an important factor in rendering vector graphics. In essence, interpolating colour (r, g, b), triples of red, green and blue values, with geometry, (x, y) coordinates of the image plane, is done in a 5-dimensional space. However, interpolation in colour space is restricted by its gamut. When going outside of the gamut, colour overflow or clipping occurs, resulting in observable sharp transitions or flat patches of colour. The colours that the user specifies for vector graphics primitives should be interpolated. By only approximating the colours a washed-out and bland approximation of the colours the user specified would be the result instead.

The gradient mesh allows interpolation of colour through a simple mesh structure, with additional gradients at vertices allowing for additional shape manipulation. Various vector graphics creation tools such as Inkscape, Adobe Illustrator and CorelDRAW have adopted the (traditional) gradient mesh. The mesh is limited to rectangular topology and adding detail to the mesh is only possibly globally. This means that refinements will propagate throughout the mesh, quickly increasing the complexity of the mesh. Local refinement has been made available for gradient meshes [BLHK18], but still only works for meshes with rectangular topology.

The restriction in topology limits the ease of use of the gradient mesh primitive. The task of fitting a mesh to an arbitrary shaped region would benefit from doing so with a mesh of arbitrary topology, instead of twisting and tweaking a rectangular mesh. There exists solutions for generalising the gradient mesh for polygonal gradient meshes [LJH13, LKSD17]. Polygonal meshes have some advantages with respect to the ordinary quadrilateral meshes. They can be sparser as they can support patches with any number of sides and more patches can meet at a vertex. Designers have to worry less about the topology of the mesh and can create a more organic workflow for creating meshes. It could also possibly help in image vectorisation, where it helps in fitting to arbitrary shape regions more easily.

In this chapter we examine and create several different variations of the polygonal gradient mesh, each using different multisided colour interpolants, that generalise the attributes of the traditional gradient mesh. We compare two existing polygonal gradient meshes, namely the topologically unrestricted gradient mesh [LKSD17] and the cubic mean value interpolant [LJH13]. In addition we alter the Gregory generalised Bézier patch [HK18] and the Charrot-Gregory corner interpolator patch [CG84] for G^1 smooth colour interpolation.

In Section 8.2 we describe related work and also the techniques for topologically unrestricted gradient meshes [LKSD17] and the cubic mean value interpolant [LJH13] in detail. We also deal with the general constructions for the (Gregory) generalised Bézier patch and the

Charrot-Gregory corner interpolator. Then in Section 8.3 we detail how to adjust these (multisided) patch constructions for fully local colour interpolation with the help of generalised barycentric coordinates. We discuss how the method of Chiyokura and Kimura [CK83] can be adapted to handle colour interpolation and how we increase the utility of these primitives by allowing non-convex corners. Then in Section 8.5 we compare and evaluate the new and known techniques to each other on a range of different meshes, and with respect to performance. The chapter is concluded in Section 8.6.

8.2 RELATED WORK AND PRELIMINARIES

In this section we describe several existing works and some of the building blocks of our new solutions for multisided colour interpolation.

8.2.1 *Polygonal Gradient Mesh*

We want to extend the abilities of the gradient mesh to meshes of arbitrary topology, or polygonal gradient meshes. The traditional gradient mesh has been described in Section 2.3.6. A polygonal gradient mesh takes some of the key elements of the traditional version and makes them suitable for polygons with more than four sides. First of all, the traditional gradient mesh is often constructed as a Ferguson patch [SLWS07, Ado06]. A Ferguson patch is restrictive in that it only allows for defining single partial derivatives and mixed partial derivatives at a vertex. This easily creates a C^1 smooth colour surface as all four patches surrounding a vertex will interpolate these constraints. However, it becomes hard to ensure the same for more than four patches to meet at a vertex. Certainly it is possible to define separate partial derivatives for each patch such that the patches are at least C^0, but the question arises on how to orient the partial derivatives so that they can intuitively be manipulated by the user. In this case it becomes beneficial to convert the Ferguson patch from the Hermite form to a Bézier patch. Both of these are bicubic patches and can be converted from one representation to the other. The resulting Bézier patches define control points which can easily and intuitively be manipulated, and have the same semantics as the gradient handles of the gradient mesh. The inner control points of the Bézier patch are then not exposed to the user. With this it becomes easy to align multiple patches around a vertex in a C^0 manner. The inner control points are then set according to the continuity conditions which are needed. In essence the data for a polygonal gradient mesh is thus polygons with vertices $\mathbf{v}_i$ defined as a position and a colour and tangents $\mathbf{t}_{i+1}$ and $\mathbf{t}_{i-1}$. This data is then used to handle (smooth) colour interpolation by one of the following techniques set out in this chapter.

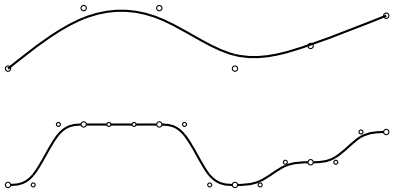


Figure 8.1: *Top: cubic B-spline approximation of the data points. Bottom: Cubic-spline approximation after ternary step applied to the control points. The resulting curve now goes through the original control points.*

8.2.2 *Topologically Unrestricted Gradient Meshes*

The ability of subdivision surfaces to create smooth surfaces from arbitrary topology meshes can also be used to smoothly interpolate colour. The topologically unrestricted gradient mesh [LKSD17] applies Catmull-Clark subdivision (see Section 2.4.1.1), after a mesh preprocessing step, on colours and geometry at the same time. The preprocessing step is a ternary subdivision step, meaning that it divides the edges into three pieces, geometrically. For colours the values of the nearest vertices are assigned to the newly created positions. In Figure 8.1 the ternary subdivision process for a single colour channel is shown for a 1D example, and contrasted with simple cubic B-spline interpolation using only the end points. It can be seen that now the curve passes through the control points and remains within the gamut, whereas the cubic spline only approximates the values. If these were colour values the resulting colours would be washed out representations of the originals and therefore the ternary step is necessary to ensure interpolation. Figure 8.2 shows the ternary step applied to a gradient mesh patch consisting out of vertices, gradients and colour data. A ring of vertices around each vertex $\mathbf{v}_i$ is constructed from its respective outgoing gradient handles $\mathbf{t}_{i,j}$ at $\mathbf{v}_i$ and the centroids $\mathbf{c}_k$ of incident faces F_k. A new vertex is created for each incident edge by simply putting it at the end of the associated tangent handle, $\mathbf{v}_i^j = \mathbf{v}_i + \mathbf{t}_{i,j}$, where $\mathbf{t}_{i,j}$ is the gradient vector along the edge between vertices $\mathbf{v}_i$ and $\mathbf{v}_j$. The face points $\mathbf{f}_k^i$ are constructed by taking into account the adjacent vertices (labelled as $\mathbf{v}_{i-1}$ and $\mathbf{v}_{i+1}$ in Figure 8.2, left) of the current vertex $\mathbf{v}_i$ with respect to the incident face F_k:

$$\mathbf{f}_k^i = (1-d_1)(1-d_2)\mathbf{v}_i + d_1 d_2 \mathbf{c}_k + (1-d_1)d_2\mathbf{d}_{i-1} + d_1(1-d_2)\mathbf{d}_{i+1},$$

where $d_j = 2\frac{||\mathbf{t}_{i,j}||}{||\mathbf{v}_i - \mathbf{v}_j||}$, $\mathbf{d}_j = \mathbf{v}_i + \frac{||(\mathbf{v}_i+\mathbf{v}_j)||}{2}\hat{\mathbf{t}}_j$, $\hat{\mathbf{t}}_{i,j}$ is $\mathbf{t}_{i,j}$ normalised, and $\mathbf{c}_k$ is the centroid of face F_k.

The ternary step does create a flat colour spot at areas corresponding to the control vertices of the control mesh due to the one-ring of vertices inheriting the colour value of the centre vertex. Yet, it is the

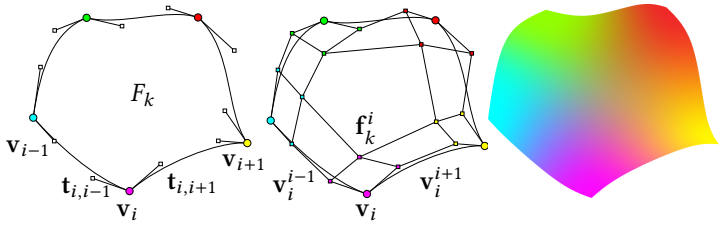


Figure 8.2: *Left: The input data given to the topologically unrestricted gradient mesh (one face, F_k, is shown): vertices and gradients for each edge. Middle: The construction of the new vertices from the data proceeds via a modified linear ternary subdivision step. Notice that the new vertices inherit the colour value of their logically closest original vertex; this ensures colour interpolation. Right: Subsequently, Catmull-Clark subdivision, run in $\mathbb{R}^5$ to accommodate position and colour, is used to obtain the final image.*

only way to simply guarantee interpolation of colour for subdivision surfaces, as after this step the resulting mesh can be further subdivided using ordinary Catmull-Clark subdivision, without losing the interpolation property. Thus the generated geometry and colour surface will be C^2 in regular regions and G^1 at extraordinary regions. Repeated application of Catmull-Clark subdivision of the mesh after ternary subdivision gives rise to several surfaces with denser and denser geometry. By altering these intermediate representations a means for hierarchical editing of the surface is obtained [VK18]. Position and colours of vertices at any subdivision level can be edited, but require that the mesh should be subdivided to at least the level in which the edits are placed.

8.2.3 *Cubic Mean Value Coordinates*

Generalised barycentric coordinates can help in interpolating values defined on arbitrary polygons, but cannot easily interpolate other values such as gradients. The cubic mean value coordinates [LJH13] can be used to interpolate positional data, associated function values and their gradients. It is based on a closed-form version of the Hermite interpolant [FS08], which is in turn based on mean value coordinates [Flo03] (see Section 2.5).

The main components of the interpolant are values f_i defined at vertices $\mathbf{v}_i$ and their gradients f_i^+ and f_i^- which are aligned along its outgoing edge. In addition outgoing normal derivatives h_i^- and h_i^+ are used as cross-edge derivatives. The resulting n-sided interpolant

$$f = \sum_{i=0}^{n-1} a_i f_i + \sum_{i=0}^{n-1} \sum_{s\in\{-,+\}} b_i^s f_i^s + \sum_{i=0}^{n-1} \sum_{s\in\{-,+\}} c_i^s h_i^s.$$

is a weighted sum of its control values and derivatives. The exact forms of weight functions a_i, b_i and c_i are complicated and can be found in the paper [LJH13], which provides freely available code that computes them. The interpolant is piecewise cubic on edges as determined by the derivatives $f_i^{\{-,+\}}$. These derivatives can take on the role of gradient handles as is done with the traditional gradient mesh and can thus interpolate colour values cubically.

With this the interpolant becomes useful for gradient mesh-like manipulation as the tangent vectors can be specified in such a way that the polygon's edges become cubic Bézier curves. The polygons can become non-convex and can cover highly irregular regions with ease through manipulation of vertex positions or gradients. The gradient handles define both the derivatives f_i^+ and f_i^- and their outward normal components h_i^- and h_i^+. The colour components of the gradients are set equal to their respective vertices so that a flat colour field is created at vertices. In this way smooth colour transitions are guaranteed when an arbitrary number of patches meet at a shared vertex. However, the interpolants are only C^0 at these extraordinary points as each outgoing edge specifies an independent gradient. This could be fixed by finding a best fitting gradient to all gradient constraints, but this will involve changing the user-specified gradients at minimal visual difference.

The cubic mean value gradient mesh primitive has been used to simplify dense quadrilateral gradient meshes. Gradient mesh patches are merged on the condition that the resulting cubic mean value patch reproduces the original patches up to some threshold. A greedy approach to merging the patches can be used to coarsen a quadrilateral gradient mesh into many high-valency patches containing possible non-convex regions. For non-convex regions the involved coordinates may become negative and thus the resulting interpolated colour should be clamped to remain within the gamut; see also [LKCOL07].

8.2.4 *Gregory Generalised Bézier Patches*

The generalised Bézier patch (see Section 2.3.5) is able to interpolate n boundary curves and their tangents. In addition a smooth surface is obtained over the polygon. This patch could thus be used easily to extend the traditional gradient mesh to patches with any number of sides. However, to be able to include it into a polygonal mesh with extraordinary vertices it should be able to smoothly connect in those regions. To this end the Gregory generalised Bézier patch (see Section 2.3.5.1) is more suitable, as it is able to create multisided patches that connect with G^1 continuity to adjacent patches. However, for smooth colour interpolation using the constraints given by the polygonal gradient mesh data the way to assure G^1 continuity must proceed slightly differently, as we will see in Section 8.3.1.

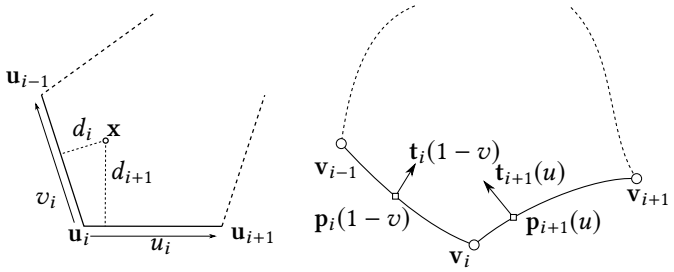


Figure 8.3: *Left: The (regular) polygonal parameter space of the Charrot-Gregory patch with distances to edges used to establish local coordinate systems at each corner. Right: A corner interpolator* $\mathbf{c}_i$ *corresponding to* $\mathbf{v}_i$ *is given by adjacent (cubic) edges and associated cross-tangent fields.*

While (Gregory) GB patches blend edge interpolators (ribbons), it is also possible to approach the problem of constructing multisided patches by making use of corner interpolators, as described in the next section.

8.2.5 *The Charrot-Gregory Corner Patch*

The Charrot-Gregory corner interpolation patch [CG84] is an n-sided parametric patch which blends n corner interpolator functions. Consider a multisided patch bounded by (cubic) curves $\mathbf{p}_i(u)$, $u \in [0,1]$ connecting vertices $\mathbf{v}_{i-1}$ and $\mathbf{v}_i$. Each curve is associated with a differentiable vector function $\mathbf{t}_i(u), u \in [0,1]$ which expresses the transversal tangent field along $\mathbf{p}_i(u)$; see Figure 8.3. At their endpoints, these tangent fields are assumed to agree with their adjacent curves, i.e.,

$$\begin{aligned} \mathbf{t}_i(0) &= -\tfrac{\partial}{\partial u}\mathbf{p}_{i-1}(u)|_{u=1}, \\ \mathbf{t}_i(1) &= \tfrac{\partial}{\partial u}\mathbf{p}_{i+1}(u)|_{u=0}. \end{aligned} \tag{8.1}$$

The Charrot-Gregory corner interpolator combines two adjacent curves $\mathbf{p}_i(u)$ and $\mathbf{p}_{i+1}(u)$, i.e., $\mathbf{p}_i(1) = \mathbf{p}_{i+1}(0) = \mathbf{v}_i$, and their transversal tangent fields $\mathbf{t}_i(u)$ and $\mathbf{t}_{i+1}(u)$ into the corner interpolator

$$\begin{aligned} \mathbf{c}_i(u,v) = -\mathbf{p}_{i+1}(0) - v\mathbf{t}_{i+1}(0) - u\mathbf{t}_i(0) + \mathbf{p}_{i+1}(u) + v\mathbf{t}_{i+1}(u) + \mathbf{p}_i(1-v) \\ + u\mathbf{t}_i(1-v) - uv\frac{v\frac{\partial}{\partial v}\mathbf{t}_i(v)|_{v=1} + u\frac{\partial}{\partial u}\mathbf{t}_{i+1}(u)|_{u=0}}{u+v}, \end{aligned}$$

which interpolates $\mathbf{p}_i(u)$ and $\mathbf{p}_{i+1}(u)$ and their respective tangent fields. The vectors $\frac{\partial}{\partial v}\mathbf{t}_i(v)|_{v=1}$ and $\frac{\partial}{\partial u}\mathbf{t}_{i+1}(u)|_{u=0}$ signify the twist vectors at $\mathbf{v}_i$, which are rationally blended to resolve the twist-compatibility con-

dition. To complete the patch definition, each corner patch is weighted by

$$w_i(\mathbf{x}) = \frac{\prod_{j\neq i,i+1} d_j^2}{\sum_{k=0}^{n-1} \prod_{l\neq k,k+1} d_l^2}, \tag{8.2}$$

where $\mathbf{x}$ is a point in the (regular) polygonal domain of the patch and d_m is the perpendicular distance from $\mathbf{x}$ to the polygon's edge corresponding to $\mathbf{p}_m(u)$, as shown in Figure 8.3. In turn, each corner interpolator is supplied with local variables $u_i(\mathbf{x}) = \frac{d_i}{d_i+d_{i+2}}$ and $v_i(\mathbf{x}) = \frac{d_{i+1}}{d_{i-1}+d_{i+1}}$. The final patch definition then becomes

$$\sum_{i=0}^{n-1} w_i(\mathbf{x})\mathbf{c}_i(u_i(\mathbf{x}), v_i(\mathbf{x})),$$

which interpolates the supplied boundary data positionally and tangentially. We modify this definition in the next section to make the patch suitable for polygonal gradient mesh colour interpolation.

8.3 LOCAL G^1 COLOUR INTERPOLATION

The topologically unrestricted gradient mesh and the cubic mean value interpolant already create smooth interpolation of colour. The construction of Gregory generalised Bézier patches and the Charrot-Gregory corner interpolation patches needs to be adjusted slightly in order to facilitate smooth interpolation. In these cases we aim for a colour surface with at least G^1 continuity and the construction should only depend on the local data of each polygonal patch. In this way we can process these patches efficiently in parallel so that they can be rendered using the modern graphics pipeline (see Section 5.8). As is done in the construction of traditional quadrilateral gradient meshes, the colour component of the gradient handles is defined by their respective vertex.

For the construction of the patches we need to define the Bézier ribbons of the multisided patches. For both patches this proceeds in the same way, as the Bézier ribbon provides a cross-boundary tangent function along the edges of the patch. Thus the control net of the ribbon also defines the tangent functions needed for the corner interpolators. The ribbons are constructed using the method of Chiyokura and Kimura, as is usual with Gregory patches. The ribbons are constructed fully locally using only the data of a single patch.

8.3.1 *The Method of Chiyokura and Kimura for Colour Interpolation*

The method of Chiyokura and Kimura [CK83] (see Section 2.3.3) is able to join together two Bézier patches with tangent plane continuity by only considering shared positional data. We extend this approach so that it also can be used in the setting of colour interpolation.

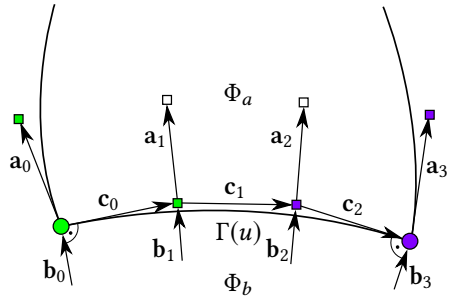


Figure 8.4: *A schematic view of the boundary conditions between the basis patch Φ_b and the actual patch Φ_a. The colour components of the control points are visualised.*

Consider two adjacent quadrilateral gradient mesh patches $\Phi_a(u,v)$ and $\Phi_b(u,v)$ with a shared cubic Bézier boundary curve $\Gamma(u)$, $u \in [0,1]$, with cross-boundary derivatives $\partial\Gamma_a(u)$ and $\partial\Gamma_b(u)$ along $\Gamma(u)$, respectively. Similar to the control points of the patches, the derivatives also live in $\mathbb{R}^{2+3}$.

The task now is to satisfy the sufficient condition for G^1 continuity as described in Equation 2.1 and solve it with the method of Chiyokura and Kimura (see Section 2.3.3). In the original method, an auxiliary patch Φ_b is used, and called the basis patch. This patch is constructed from the common boundary data so that a G^1 connection across $\Gamma(u)$ can be determined fully locally; see Figure 8.4. In the original 3D setting, the basis patch is constructed with the help of normal vectors at the endpoints of $\Gamma(u)$. However, in the colour interpolation setting, the 'normal' vector should be determined in $\mathbb{R}^{2+3}$. In this case the normal vector is the one pointing inwards (or outwards) of the screen as the colour components are all equal and the associated vectors are defined in the image plane. Consequently, the basis patch vectors $\mathbf{b}_0$ and $\mathbf{b}_3$ can be determined simply by rotating the vectors $\mathbf{c}_0$ and $\mathbf{c}_2$ by ninety degrees in the image plane. We can then express the given vectors of the actual patch Φ_a as $\mathbf{a}_0 = k_0\mathbf{b}_0 + h_0\mathbf{c}_0$ and $\mathbf{a}_3 = k_1\mathbf{b}_3 + h_1\mathbf{c}_2$, which determines the values of k_0, k_1 and h_0, h_1.

We then proceed by completing the definition of the basis patch (or rather its ribbon at $\Gamma(u)$ only), by assuming that $\mathbf{b}_1$ and $\mathbf{b}_2$ can be determined by linear interpolation $\mathbf{b}_1 = \frac{2}{3}\mathbf{b}_0 + \frac{1}{3}\mathbf{b}_3$, $\mathbf{b}_2 = \frac{1}{3}\mathbf{b}_0 + \frac{2}{3}\mathbf{b}_3$. Finally, the missing ribbon vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ of the actual patch Φ_a are computed as

$$\mathbf{a}_1 = (k_1 - k_0)\frac{\mathbf{b}_0}{3} + k_0\mathbf{b}_1 + 2h_0\frac{\mathbf{c}_1}{3} + h_1\frac{\mathbf{c}_0}{3}, \quad \mathbf{a}_2 = k_1\mathbf{b}_2 - (k_1 - k_0)\frac{\mathbf{b}_3}{3} + h_0\frac{\mathbf{c}_2}{3} + 2h_1\frac{\mathbf{c}_1}{3}.$$

It is hardly surprising that patches in the plane can be connected with tangent plane continuity at their shared edge, but in our setting of vector graphics, the important part is the colour component, especially that of the internal control points of Φ_a (white squares in Figure 8.4). Since the colour component of the gradients at vertices is zero, the method can be simplified. All the colour components of vectors $\mathbf{c}_0$, $\mathbf{c}_2$, and $\mathbf{a}_0$, $\mathbf{a}_3$ are zero. Therefore the only vector that weighs in the calculation of the colour component of the internal control points is $\mathbf{c}_1$, the middle control polygon leg of $\Gamma(u)$. However, $\mathbf{c}_1$ is simply the difference between the colours at the end points of $\Gamma(u)$. This leads to

$$\mathbf{a}_1^c = 2h_0 \frac{\mathbf{c}_1^c}{3}, \quad \mathbf{a}_2^c = 2h_1 \frac{\mathbf{c}_1^c}{3}$$

in colour space (signified by the superscript c), which enables us to smoothly join together arbitrary gradient mesh patches.

8.3.2 *Quadrilateral Gregory Gradient Patches*

The previous method for establishing colour continuity across edges already has some uses in non-polygonal gradient meshes. The adjusted method of Chiyokura and Kimura for G^1 colour connection makes quadrilateral Gregory patches suitable for use in strictly quadrilateral Gradient meshes, which can have extraordinary vertices. In addition it becomes possible to independently define gradient handles around vertices without sacrificing colour smoothness, i.e. the co-linearity property of gradient handles in the traditional gradient meshes does not need to hold. Moreover, an arbitrary number of patches are now able to meet smoothly around vertices while still guaranteeing a smooth colour surface. This increases the versatility of the gradient mesh primitive and allows the designers to more freely define meshes.

We compare the Gregory gradient mesh with the traditional gradient mesh in a setting where we do not enforce co-linear gradients. Figure 8.5 shows a mesh consisting out of two adjacent quadrilateral patches. The gradient vectors create a curve network which is only C^0 at the vertices. For the traditional gradient mesh it is impossible to create a smooth join between the patches. The Gregory gradient mesh is able to create such a smooth join, even when the gradients are not co-linear. This is because the gradients do not have to be co-linear for Gregory patches, but only co-planar in the $\mathbb{R}^{2+3}$ combined geometry and colour space. The smoothness is highlighted by the visualised colour bands that show the isolines of the individual colour channels. For the Gregory patch these are clearly G^1 smooth whereas the traditional patches show only C^0 continuity.

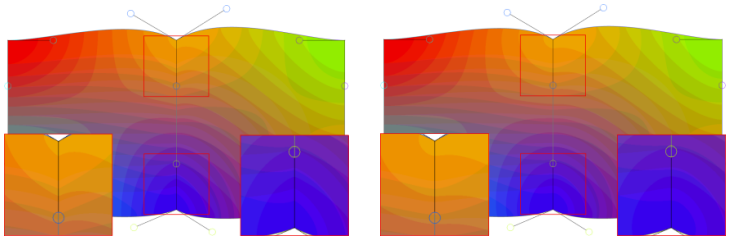


Figure 8.5: *A simple gradient mesh composed of two quads. Left: A traditional gradient mesh (based on Ferguson/Bézier patches) results in non-smooth transitions at the shared edge, which is due to the three (linearly) independent gradient handles at the shared vertices. Right: In contrast, the Gregory gradient mesh produces a smooth image over the entire mesh. The insets highlight the difference in colour transitions, where in the left image the colour isolines are only C^0 at the common edge of the two patches whereas on the right they are smooth (G^1).*

8.4 MULTISIDED COLOUR INTERPOLANTS

To be able to model any polygonal region and thus increase the usability of the polygonal gradient we want to support both convex and non-convex polygons. We have seen before in this thesis that we can use a regular parametrisation domain for an arbitrary polygonal domain. In this case we can choose to use the actual polygonal domain for parametrisation through generalised barycentric coordinates. However, for non-convex polygons Wachspress coordinates, a standard form of generalised barycentric coordinates, are not well-defined. Mean-value coordinates [Flo03] provide a way to parametrise the domain for arbitrary planar polygonal patches. We use this both for generalised Gregory patches and Gregory corner interpolator patches.

We could also use harmonic coordinates instead [JMD+07, SV18], but this would complicate the processing of polygonal meshes in two ways. These coordinates do not have a closed-form solution and require discretisation of the polygonal domain by either rasterisation or triangulation. In addition, this process would be unique for each shape and would require a complicated parametrisation process for each individual polygon.

Mean-value coordinates provide an efficient means to handle any simple polygonal shape. However, these coordinates can become negative in concave polygons, as already mentioned in Section 8.2.3. To remedy this we simple clamp the value of the negative coordinate to 0 for colour interpolation (geometry interpolation proceeds as normal), in the calculation of the local parameters. This leads to a contour of C^0 continuity at the areas where the offending barycentric weight becomes negative. To show the effect of this we have visualised the effect of the clamped coordinate in Figure 8.6 on a concave patch. Although some of the coor-

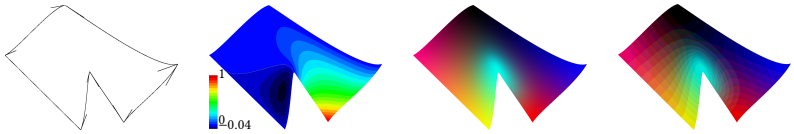


Figure 8.6: *Left to right: Input gradient mesh (non-convex polygon), mean value barycentric coordinate function of the bottom right vertex visualised on-top of a GB patch, and the resulting colour interpolation of the GB patch with non-convex adjustment, and also with colour banding. Negative values in the coordinate visualisation have been scaled to increase their visibility, and the zero-contour is highlighted in grey.*

dinates become negative in different parts of the polygon, the resulting colour surface still appears as smooth.

Using mean-value coordinates in the definition of the generalised Gregory patches does not have to provide severe problems. However, the patches will fold-over when the gradient handles at a vertex make it concave. This occurs when the outward normal components of the gradients lie in the same half-circle. Salvi and Várady [SV18] describe a method that can alleviate the problem. By reversing the direction of the gradients in certain individual ribbons, the surfaces of the ribbons can be made to go into the patch. This technique can even be applied to the end point vectors $\mathbf{a}_0$ and $\mathbf{a}_3$ in the method of Chiyokura and Kimura by having a common direction (into the patch) for both of them. We can do this fully locally, by flipping the tangent vectors as follows. Consider the tangent vectors $\mathbf{t}_{i,i-1}$ and $\mathbf{t}_{i,i+1}$ of some vertex $\mathbf{v}_i$. We can compare the outward normal component $\mathbf{n}_{i,i-1}$ of the tangent vector $\mathbf{t}_{i,i-1}$ with $\mathbf{t}_{i,i+1}$ to determine the sign of $\mathbf{t}_{i,i-1}$ to be used in the method of Chiyokura and Kimura by checking whether $\mathbf{n}_{i,i-1} \cdot \mathbf{t}_{i,i+1} > 0$. We also employ the suggestion of Salvi and Várady [SV18] to adjust the blending functions μ^i_{jk} of (2.6) to use squared terms to increase the smoothness of the surface.

Non-convex domains pose additional problems for the Gregory corner interpolator patch, as this patch constructs its local parameters from a radial sweep-line construction [CG84]. In addition, orthogonal distances are used in the parametrisation of the patch. These parametrisation are not compatible with concave domains as the weighting function and the construction of local parameters (Section 8.2.5) introduce singularities throughout the patch. This can be seen clearly in Figure 8.7, left. Alternative formulations have been suggested previously [VRS11], but are computationally intensive, making them unsuitable for real-time purposes. We propose to construct both weights and local parameters from mean-value coordinates.

To this end, we make use of the same functions that construct the local parameters for the generalised Bézier patch. In particular, we are interested in the parameter $h_i = 1 - \phi_i - \phi_{i-1}$, where the ϕ_i are mean value

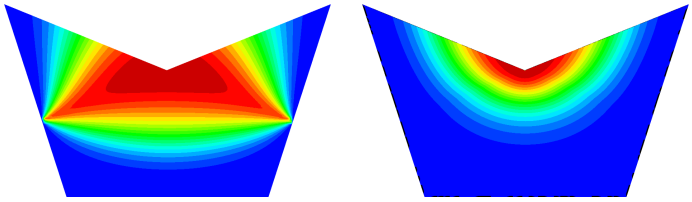


Figure 8.7: *Left: The original parametrisation of the Charrot-Gregory patch that uses perpendicular distances to edges applied to a non-convex polygon. Right: Our technique that uses 'distance' functions based on mean value coordinates provides a better parametrisation.*

coordinates, replacing the edge-distance functions used in the original construction. The functions are zero on the edge $\mathbf{v}_i\mathbf{v}_{i-1}$, and increase linearly on edges $\mathbf{v}_{i-1}\mathbf{v}_i$ and $\mathbf{v}_{i+1}\mathbf{v}_{i+2}$, as desired.

We replace the d_i functions in (8.2) with the functions h_i. For the local parameters u_i and v_i, we substitute the functions s_i and h_i, respectively. This new weight function and local parameters have the same properties as the original functions, i.e., G^1 continuity is still maintained across adjacent patches, but are now suitable for non-convex domains, as illustrated in Figure 8.7.

Unfortunately, we have found that the same adjustment to the tangent vectors cannot be applied to the corner interpolater patches. Flipping directions of these tangents directions will violate a necessary condition (8.1) on the tangent fields along the edge curves. This reduces the usability of the corner interpolator as a gradient mesh primitive, as the internal angle of the tangent vectors must always remain under 180 degrees. Still, the polygonal domain itself can be non-convex as Figure 8.7 and Figure 8.11 (bottom row, second column) show. A full investigation into the injectivity of the resulting planar patches is beyond the scope of this thesis.

8.5 RESULTS AND DISCUSSION

The different forms of the polygonal gradient mesh all interpolate colours in their own way and we would like to compare their properties in a range of ways. This includes their ability of interpolating colours over convex and concave domains. We look at visual fidelity of the resulting colour surfaces and their computational and rendering statistics. Finally we look at the editability of the gradient primitives from a user's perspective.

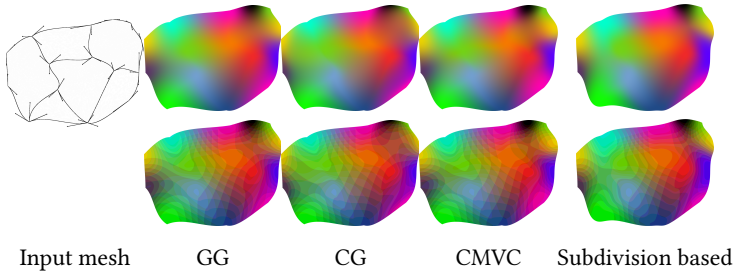


Figure 8.8: *A polygonal gradient mesh (far left) consisting of convex faces of various valencies. Top row: Colour interpolation. Bottom row: Iso-bands of the individual channels of the interpolants. From left to right: Gregory gradient mesh (GG), Charrot-Gregory gradient mesh (CG), cubic mean value coordinate gradient mesh (CMVC), and subdivision-based topologically unrestricted gradient mesh.*

8.5.1 *Colour Interpolation on Convex Patches*

Figure 8.8 shows a simple gradient mesh consisting only of convex polygons with differing valencies. This mesh cannot be modelled with an ordinary gradient mesh, but can easily be used with any of the four different techniques considered in this chapter.

Both the generalised Gregory patch and the Gregory corner interpolator provide visually similar results. This is because they use the same technique for creating G^1 smooth colour transitions between patches and they only differ in how the colour is interpolated throughout the middle of the patches. Even though they only provide a G^1 colour surface, it is still of high visual quality as evidenced from the colour iso-band visualisations. These iso-bands are comparably smooth to those of the much smoother subdivision based technique.

The cubic mean-value interpolant shows the sharpest turns in the colour bands. This could be because the gradient constraints are not C^1 compatible and introduce small visual artefacts around the vertex regions of the polygonal mesh. At these points the continuity is only C^0 because of the inconsistent gradient directions. Nevertheless, along edges the transitions in colours are still smooth. It should be mentioned that the C^1 discontinuity at extraordinary points can be fixed through changing the gradient handles so that they express a common gradient at the vertex. However, this would change these gradient handles and this would diminish the freedom a designer has in the defining them. This trade-off is not present in any of the other considered methods.

If we assess the visual quality of the interpolation methods by the inability to see the underlying mesh structure (a criterion typically employed also in the 3D geometry setting), then it is clear that the topologically unrestricted gradient mesh performs this task in the best manner.

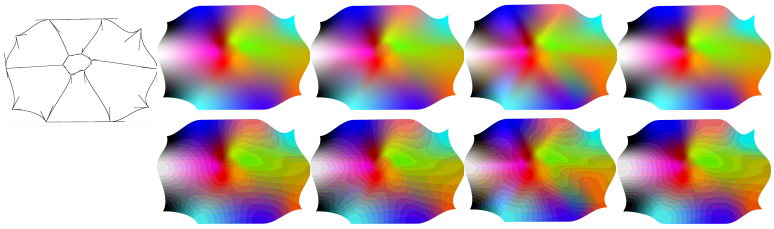


Figure 8.9: *A polygonal gradient mesh (far left) consisting of convex faces of various valencies, but with one relatively smaller hexagon (central polygon). Top row: Colour interpolation. Bottom row: Iso-bands of the individual channels of the interpolants. From left to right: Gregory gradient mesh (GG), Charrot-Gregory gradient mesh (CG), cubic mean value coordinate gradient mesh (CMVC), and subdivision-based topologically unrestricted gradient mesh.*

The increased degree of smoothness, it is C^2 almost everywhere, of the colour surface of this technique removes visual cues that other techniques show around edges.

The introduction of smaller sized polygons into the gradient mesh should not affect the interpolation of colour in a counter-intuitive way. To investigate the effect of different sized polygons we have created a gradient mesh (see Figure 8.9), where a smaller hexagonal polygon is surrounded by larger polygons. In this case both the Gregory GB patch and the subdivision-based gradient meshes perform well, as they do not reveal the underlying mesh. The Gregory-Charrot mesh also performs well although the colour interpolation around the vertices of the central hexagon show sharper transitions in colour as evidenced by the sharper corners in the colour isolines. The cubic mean value interpolant shows the most diverging behaviour as it introduces unexpected colours with non-intuitive propagation in the interior of the polygons as well as sections which reveal the underlying mesh.

8.5.2 *Colour Interpolation on Concave Patches*

When designing a polygonal gradient mesh, it is not unthinkable that concave faces might occur inside the mesh. Automatic image vectorisation or gradient mesh simplification is also likely to create concave faces [LJH13]. In Figure 8.10 we assess the abilities of the different interpolants with respect to concave faces.

It is not surprising that the Charrot-Gregory gradient mesh and the subdivision based gradient mesh do not perform well in non-convex cases. Their definition does not allow them to tackle such cases. They fold over the non-convex regions in increasing manner depending on

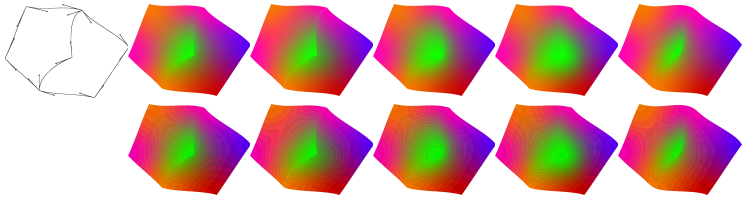
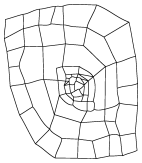


Figure 8.10: *A simple gradient mesh (far left) consisting out of two pentagonal faces, one of which is concave. Top row: Colour interpolation. Bottom row: Iso-bands of the individual channels of the interpolated colours. From left to right: Gregory gradient mesh (GG), Charrot-Gregory gradient mesh (CG), Gregory gradient mesh with non-convexity adjustments (adj. GG), cubic mean value coordinate gradient mesh (CMVC), and subdivision-based topologically unrestricted gradient mesh.*

the concave angle of the edges. The same can be said for the generalised Gregory gradient mesh, but with the non-convexity adjustments to the gradient vectors it is able to successfully handle non-convex corners. The cubic mean value coordinate gradient also performs well in this setting. Looking at the generated iso-bands for both of these techniques we can see that these remain smooth even in the areas where the other patches fold-over. The cubic mean value coordinate gradient mesh creates slightly smoother results, but again they are only C^0 at extraordinary vertices. This shows that these two gradient mesh types have comparable abilities when it comes to colour interpolation on concave faces.

Table 3: *Performance evaluation of rendering a gradient mesh with 65 faces of various valencies (far right). Build time refers to the time it takes to create and gather the data to push it to GPU buffers, whereas render time refers to the time it takes to render the data on screen (using OpenGL) at 1080p resolution. All the gradient meshes were rendered at approximately equal triangulation density except for the last two columns that use the maximum tessellation level of* 64×64.

Method	Build	Render	#△
Gregory GB	1 ms	0.52 ms	343541
Charrot-Gregory	1 ms	1.20 ms	343541
CMVC	491 ms	1.05 ms	281600
Subdivision	590 ms	1.20 ms	330624
Gregory GB	1 ms	1.09 ms	1148928
Charrot-Gregory	1 ms	2.41 ms	1148928



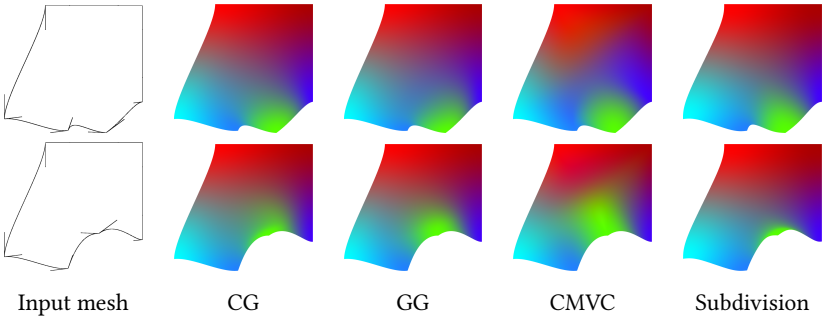


Figure 8.11: *A single hexagonal patch and the effect of translating one of the vertices so that the patch transitions from a convex shape (top row) to a concave shape (bottom row).*

The manipulation of a gradient mesh by manipulating the gradient or vertices is a common operation. The effect of geometrically altering the mesh should not have adverse effects on the colour interpolation, i.e. the spread of colour should be more or less proportional to the polygonal patch. In Figure 8.11 we show an example of a gradient mesh consisting out of a single hexagonal patch. In this example we manipulate the polygon by dragging a vertex around so that an initially convex patch becomes concave. For the subdivision based gradient mesh we can see that the colour spread around vertices remains constant. However, the colour region around the translated vertex does get 'thinner'. For the Charrot-Gregory patch and the Gregory GB patch the colours remain constant even when the polygon becomes concave. For the cubic mean-value based method different colours start to appear and disappear in the interior of the patch as the vertex is moved. This effect is increased even more when one of the edges is shorter than the other edges in the polygon, or due to translating vertices to create such an effect. This effect even happens when the gradients at the vertex are shortened accordingly.

8.5.3 *Performance*

Gradient meshes are often used interactively e.g. they can be the results of a vectorisation that the user manipulates or created from scratch by creating a polygonal mesh. In these cases it is important to have immediate feedback and it is therefore important that the resulting rasterisation can be generated quickly. That is not only important for editing meshes, but for inclusion in webpages or other documents so that it can be displayed with ease at any resolution. We therefore weigh the rendering performance of the different polygonal gradient mesh techniques against each other.

We evaluate the performance of the techniques based on the time it takes to create the mesh structure and the time it takes to actually render the mesh on the screen. For all techniques the basic gradient mesh structure is contained in a half-edge mesh data structure. In this structure all data for vertex positions, and gradient positions and colours are stored as well as the main connectivity of the mesh. This base structure is used as input to each of the techniques.

For the generalised Gregory patch and the Gregory-Charrot corner interpolator patch we use the multisided tessellation technique set out in Section 5.8. We can extract the data for each patch individually and batch them together according to valency. We then render them efficiently using the GPU tessellation pipeline. One problem occurs when using 'highly' concave faces. The triangulation strategies covered in Section 5.8, might not be adequate when considering multiple concave corners of a high valence polygon and may cover areas which lie outside the domain polygon. These patches have to be evaluated on the CPU side.

For evaluating the cubic mean value interpolant-based method we used the publicly available implementation of the authors of [LJH13]. We evaluate it fully on the CPU side, but take advantage of the locality of the patches and construct them fully in parallel. Potentially, the same tessellation mentioned above can be used to evaluate cubic mean value patches, but attempts to port the provided code to be GPU friendly have been unfruitful; instead we used OpenMP parallelisation.

The subdivision-based approach of the topologically unrestricted gradient mesh requires the heaviest computation as both the operations of the ternary step (discussed in Section 8.2.2) and the subsequent Catmull-Clark subdivision operations are fully global. Any adjustment the user makes to gradient mesh structure leads to the full mesh needing to be subdivided again from the ground up, including the ternary subdivision step. At least a few subdivision steps are needed to provide a reasonably smooth colour surface and thus higher subdivision levels lead to decreased performance.

We conducted a test to check the viability of all the techniques in the event of interactively editing a mesh. In this context, interactively editing the mesh is understood as dragging vertices or tangent handles to change the geometry of the gradient mesh, or modifying colours. This is a use-case any designer would go through many times when designing a gradient mesh. We conducted the performance comparison on a computer with two Intel Xeon E5-2630 processors (@2.3 GHz) and an NVIDIA Titan V graphics card with 12 GB of video memory. Our results are reported in Table 3.

We can see that the two newly proposed gradient meshes are very efficient, both in rendering and in the time it takes to transfer their data to the GPU. We rendered them at two different densities, one such that the level of detail is comparable to the other methods and once using

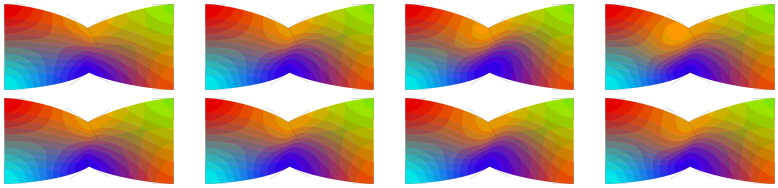


Figure 8.12: *The same Gregory gradient meshes with different values for the parameter s for the common edge of the two quadrilateral faces rendered with showing the iso-bands of the individual colour channels. Top row: Our scaling method. Bottom row: The scaling approach of [SS90].*

the maximum tessellation rate of 64×64 (current hardware limit). The performance of the Gregory gradient mesh is approximately twice as fast as that of the Charrot-Gregory gradient mesh. It shows the viability of using these new primitives in vector graphics authoring software, as they greatly improve the performance with respect to the other methods. With the new primitives, polygonal gradient meshes can be modified in real time, giving the user instant feedback at every step during design. The performance could be increased even further by employing adaptive tessellation strategies [SNK+14] (and see Section 5.8.2).

8.5.4 *Editability*

As already shown all the discussed techniques have comparable capabilities when it comes to smooth colour interpolation when concerning convex polygons, and convex vertices, as shown above. Smooth transitions are clearly important but can only model a range of different situations. To model sharp edges in the colour surface with only smooth transitions requires many patches in close proximity to each other so that the colour surface changes colour fast. This quickly increases the

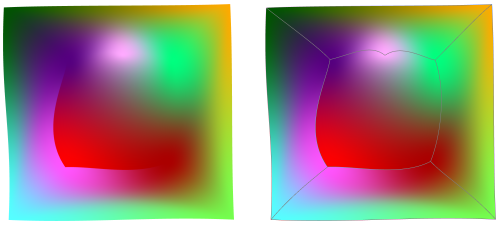


Figure 8.13: *A simple gradient mesh with 2 pentagons and 3 quads showing sharp colour transitions rendered using Gregory GB gradient meshes with the non-convexity adjustment. Notice how the sharp colour transitions blend back into smooth colour transitions along the (hard) edges.*



Figure 8.14: *Three meshes rendered using generalised Gregory gradient meshes with the non-convexity adjustment (face model: 101 faces in total, 12 triangles, 61 quads, 24 pentagons and 4 hexagons; shoe model: 166 faces in total, 20 triangles, 115 quads, 29 pentagons and 2 hexagons; heel model: 195 faces in total, 22 triangles and 173 quads).*

complexity of the mesh, whereas we are after as sparse as possible a representation.

The ability to model sharp features was already mentioned in relation with subdivision gradient meshes [LKSD17] through the use of semi-sharp creases [DKT98]. Semi-sharp creases allow for the user to tag certain edges or chains of edges with a sharpness value. Then depending on the sharpness value, sharp subdivision rules for subdivision are used until the sharpness value, which is decremented in each iteration, is zero where after normal subdivision rules are used. By applying this only to the colour component of the surface creases in the colour surface are obtained. For infinitely hard colour transitions it requires the topological separation of the colour attributes of edges.

Hard transitions are easily modelled by the other variants of gradient meshes. However, semi-sharp creases are much harder to model as it requires mostly altering the meshes geometrically. We determined a simple way to offer a similar ability in the Gregory gradient mesh. The tangent vectors involved in individual Bézier ribbons can be scaled arbitrarily. Salvi et al. [SV18] have shown that there is no apparent restriction on the orientation of the tangent vectors used in the ribbons. The same thing applies to the magnitude of the tangent vectors, as the connection between patches is only G^1-smooth. Therefore, we can scale

the vectors $\mathbf{a}_0$ and $\mathbf{a}_3$ used in determining the inner control points of the Bézier ribbon by an arbitrary constant factor $s > 0$ to increase or decrease the influence a ribbon has on local colour propagation into the patch. The manipulation of the two middle vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ was proposed earlier by Shirman & Sequin [SS90] for quadrilateral Gregory patches. The generalised Gregory patches use blending functions for all control points of a ribbon and thus more freedom is available to vary these.

We have investigated the effect of the parameter s by varying its value for a connection between two quadrilateral Gregory gradient mesh primitives. Figure 8.12 shows that by manipulating the magnitude of s we can influence the spread of colour near the tagged edge. Note that manipulating the magnitude in the case of only using the centre vectors of the basis patch changes the extent in the centre of the ribbon, whereas our adjustment changes it for a wider region of the ribbon. The effect is still quite subtle, but it is possible to see that we can get sharper transitions for $s = 0$ as our method also scales the endpoint vectors of the basis patch. Increasing the parameter value increases the extent and flatness of the colour transition over the boundary. Increasing the extent too much for adjacent edges then leads to fold-overs.

We can also model hard transitions simply by allowing vertices to have multiple colours, one per incident face [BLHK18, VK18]. Figure 8.13 shows a simple polygonal gradient mesh where a sharp transition is created for one of the middle vertices of the mesh. It clearly shows a sharp transition in colour along the user-specified hard edges emanating from this vertex. The sharp colour transition blends back into a smooth colour transition at the end of the sharp edges. This procedure is easily used also for the cubic mean value and corner interpolator variants.

Topological editing of the mesh structure is supported for all mesh types. Adding, removing, and arbitrarily splitting faces and edges can be achieved without loss of smoothness for all the techniques, with the possible exception of cubic mean value coordinates meshes due to the gradient incompatibility issue. The hierarchical editing of topologically unrestricted gradient meshes [VK18] is not easily matched by any one of the other techniques.

Nevertheless, it is possible to create a transition of high detail areas to low detail areas with non-hierarchical methods. For instance, this can be achieved by turning a quadrilateral face into a pentagonal face by subdividing one of its edges. One can see many such situations arising in the realistic gradient meshes in Figure 8.14 rendered using Gregory GB gradient meshes. Notice the complicated topology introduced around the eyes of the ‘face’ mesh and around the shoelaces of the ‘shoe’ mesh.

8.6 CONCLUSION

We have modified the generalised Gregory patch and the Gregory-Charrot corner interpolator patch to make them suitable for colour interpolation in the context of polygonal gradient meshes. The patches can interpolate colour over polygonal meshes similar to how a traditional Ferguson patch can do over gradient meshes with rectangular topology. The patches can be constructed from local data through the use of the adjusted method of Chiyokura and Kimura that now takes into account colour components. The range of polygons that can be modelled has been extended by parametrisation through mean value coordinates, so that also concave faces are able to be used and allow for smooth interpolation of colour. The generalised Gregory gradient mesh has an advantage over the corner interpolator patch as the individual ribbons can be adjusted to allow for concave corners and avoid fold-overs in these areas.

We have compared the patches to the existing cubic mean value interpolant and the topologically unrestricted gradient mesh. We found that in terms of visual quality they are comparable to the subdivision based topologically unrestricted gradient mesh even though the continuity of the colour surface is only G^1. The new techniques are also very efficient as they can be directly rendered using hardware tessellation methods, and are much more efficient than the existing two methods as they require only minimal mesh data. In the case of simple geometric editing of an existing gradient mesh, they behave optimally for interactivity. The new polygonal gradient mesh primitives are good alternatives to the two existing techniques, and increase the viability of the use of arbitrary manifold topology gradient meshes in existing and future vector graphics authoring software.

9 IMAGE VECTORISATION WITH MESH COLOURS

Parts of this chapter awaits publication as

- Gerben J. Hettinga, J.I. Echevarria and J. Kosinka. "Efficient Image Vectorisation using Mesh Colours", STAG 2021.

Image vectorization methods proposed in the past have not seen wide adoption due to performance, quality, controllability, and/or generality issues. We present a new method using mesh colours as a new vector primitive for image vectorisation. We show that mesh colours have clear benefits for rendering performance, texture detail, and editing capabilities. Due to their flexibility, they also enable a simplified and more efficient generation of meshes of curved triangular patches, which are in our case constrained by our new image feature extraction algorithm. The proposed method follows a standard pipeline where each step is efficient and controllable, leading to results that compare favourably with those from previous work. We show results over a variety of input images including photos, drawings, paintings, designs, and cartoons.

9.1 INTRODUCTION

Image vectorisation is a hard problem and the main problems faced when vectorising an image were already set out in the introductory section of this part of the thesis. With image vectorisation a raster image is represented as a vector image by defining the image as a collection of primitives. Manual vectorisation of a raster image is a painstaking process especially when trying to vectorise highly detailed input such as natural images obtained from photographs. It requires expertise and knowledge about the abilities of the vector graphics primitive and immense amounts of time [YCZ+16]. There is thus a need for solutions that automatically vectorise images.

There have been many attempts and solutions at automatic image vectorisation. With this many different types of primitives have been used as means through which vectorisation is done. However, the methods have not seen wide adoption. This is mainly due to the vectorisation lacking in terms of conversion efficiency and/or rendering performance, quality or controllability issues. The limitations of vectorisations have turned into artistic styles of their own like Adobe's *Live Trace* [Ado19b]. Automatically creating complex and realistic looking vector graphics from an arbitrary input image still remains a challenge.

Our proposed automatic image vectorisation method strikes a balance between the requirements for a good vectorisation. We create a vectorisation that is adaptive to the contents of the image, creating a sparse representation in certain areas while increasing detail in others. The adaptivity of the representation and the sparseness makes it possible to easily edit the created representation. Our vectorisation method is agnostic of the contents of the image, but generally is able to create highly accurate vectorisations and can do so on a wide range of inputs, including natural images and stylised design graphics. The method is fast and we have taken careful design decisions in our pipeline to optimally vectorise an image and to be able to efficiently render the image on screen.

The pipeline of our approach follows three main steps: image feature extraction, curved mesh generation and colour fitting/texture transfer. Each step has been designed with quality, performance and control in mind. We use a new and intuitive method for extracting soft image features and use standard techniques to obtain sharp image features. Our mesh generation step then faithfully traces and leverages the extracted features and uses recent advances in mesh generation to create a curved triangular mesh. Likewise, new methods for texture representation are used to create a versatile vector graphics primitive to which we can automatically transfer texture information on a per-triangle basis. Additionally, using the proposed rendering method, our vectorised images can be rendered in real-time on a wide range of hardware, including extra control over the level of detail for the more constrained use cases.

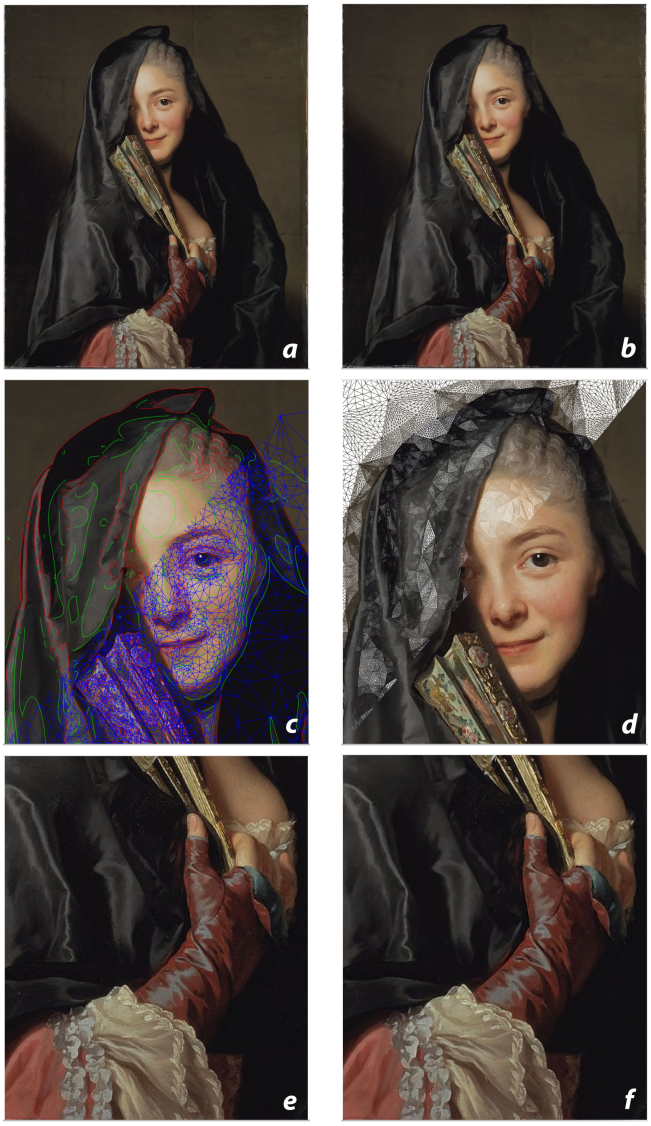


Figure 9.1: *Example of an input image (a) and our vectorized result (b). Our method follows an efficient and controllable pipeline where we initially compute hard (red) and soft (green) image features (c, top). These features are then used to build a curved triangular 2D mesh (c, bottom), where each triangle is equipped with mesh colours (d, top) that can be rendered efficiently in real-time (d, bottom). Insets (e, f), coming from (a, b), show how our method keeps the sharpness around hard features (like the folds), while interpolating colour smoothly everywhere else. Results of this quality can be achieved in just a few seconds with the proposed method.*

In summary, the main contributions of our method are:

- The introduction of mesh colours as an image vectorisation primitive, with an efficient strategy for the automatic fitting of mesh colours;
- A novel way to extract texture information and soft features from the input image;
- An efficient image vectorisation method.

We start with an overview of related work in vector graphics and image vectorisation (Section 9.2). Then we provide an overview of our vectorisation method (Section 9.3), which is followed by a detailed description of its stages: image feature extraction (Section 9.4), mesh generation (Section 9.5), and texture transfer (Section 9.6). To demonstrate the utility of our method, we show the results of applying our pipeline on several types of raster images, edits to our vector images, and compare with previous works (Section 9.7). Finally, we discuss our method before concluding the chapter (Section 9.9).

9.2 RELATED WORK

Early attempts at image vectorisation were mostly limited to the vector graphics primitives of that time. A basic vector graphics primitive is a pathed region, i.e. a region enclosed by curves. The regions can be filled in with flat colour or linear or radial colour gradients. The limited range of these primitives created a stylised representation of the original image. The ArDeco system [LL06] divides the image into polygonal regions with little or no colour variation, to which flat or linear colour gradients are fitted. Similarly, through colour quantisation the quantised iso-curves can be vectorised into paths and the colour of the enclosed region can be grouped and a best fitting colour can be determined for the whole region [Ado19a, LLGRK20]. This simplifies the details of the image and expressiveness as a whole, but is a simple procedure that works reasonably well for natural images depending on the results of the quantisation step. Interactive user guidance has been explored for a means to better preserve salient semantic boundaries of objects in the image [XWLS17, RLMB+14, FLB17].

Images usually feature more complex colour gradients than only linear or radial ones and different primitives have been developed over time to better model these gradients. The gradient mesh primitive which we have seen before (see the previous two chapters in this thesis) is one such primitive. The traditional form of the gradient mesh has featured in several attempts at automatic image vectorisation, where the main difficulty lies in the generation of a mesh for an arbitrary region. Early attempts have used it in combination with (pre-)segmented regions which are embellished with a quadrilateral parametrisation. The

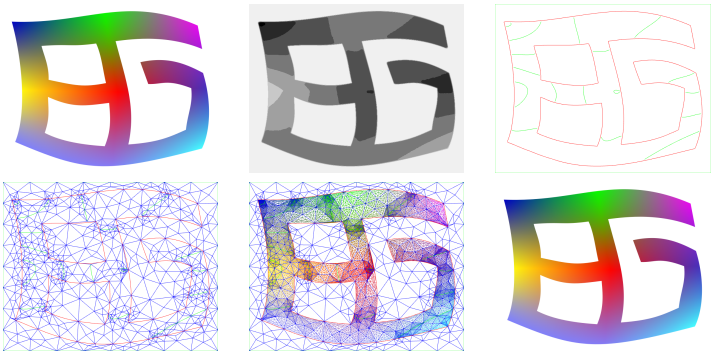


Figure 9.2: *An overview of the steps in our method. Top row from left to right: The input raster image, banded grayscale image to extract soft edges from, extracted hard (red) and soft (green) image features which are vectorised into Bézier curves, Bottom row from left to right: generated curved triangular mesh, mesh colours with colours fitted, and final rasterized result.*

parametrisation is then used to progressively subdivide the patches until they adequately approximate the underlying image region. Finding an optimal position for all the vertices in a gradient mesh is hard to do and requires a lot of tweaking and adjusting. For this purpose automatic means to optimise an existing gradient mesh, both geometrically and colour wise were explored [SLWS07]. A different optimisation approach has been determined with capabilities to fit a gradient mesh to a segmented region with holes [LHM09]. Recently a fully automatic pipeline that generated a gradient mesh for an arbitrary image used frame-field computed from image gradients which is used to create a quadrangulation was presented [WZG+19].

Another primitive able to convey complex colour gradients are diffusion curves [OBW+08]. Diffusion curves describe an image by a set of curves each of which define a sharp transition in colour. The colours of the curve are then diffused over the rest of the image plane. Diffusion curves provide a straightforward way to vectorise images. By detecting edges in images and subsequently representing them as smooth curves the only parameter that needs to be determined is the colours on either side of the edge. Over time some extensions of the diffusion curve have been suggested and different ways they could be used in a vectorisation setting. Hierarchical diffusion curves [XSTN14] are generated from a scale-space feature extraction process and images augmented with depth values have led to depth-aware image vectorisations [LJD+19]. An iterative process of vectorising images has been proposed by fitting diffusion curves on the image residual [ZDZ18]. Diffusion curves and their generalisations are powerful means to represent an image using a sparse set of curves, but their evaluation is expensive as the diffusion

of the colours over the image either involves solving a large linear system [OBW+08] or other complicated methods [BLW11].

The use of subdivision surfaces for vectorisation was approached in combination with triangular meshes. The first such representation [LHFY12] simplifies a pixel level triangulation which conforms to detected edges. The simplified triangulation is fit colours so that the subsequent evaluation using Loop subdivision minimises the colour with respect to the input image. A similar approach is taken by [ZZW14], but in this case the mesh is generated using a polyline representation of the detected edges. Both these approaches are able to generate good results using a fully automatic pipeline, but the evaluation of subdivision surfaces is costly and they do not in general interpolate colours assigned to mesh vertices [VK18]. Both gradient meshes and subdivision surfaces can only define colours at the vertices of their respective meshes, and thus in image areas with high-frequency changes in colour a large number of vertices are needed.

Attempts at tackling this problem have led to the use of a combination of thin-plate splines with cubic Bézier triangles. In [XLY09] a curved triangular mesh is generated by simplifying a pixel-dense triangular mesh, which is then further optimised into Bézier triangles, through a complicated non-linear untangling process. Then each patch is equipped with a thin-plate spline to best fit the underlying image colours. The mesh is able to be sufficiently sparse and the thin-plate splines capture the underlying texture well, but are unable to be smoothly joined to other patches and care must be taken to minimize the discrepancy. The evaluation of the spline is also costly and a special CUDA based algorithm is designed to be able to do so with reasonable efficiency. Chen et al. [CLL+20] create a more efficient kernel to evaluate the splines and pair it with gradient meshes that are generated from a rough manual segmentation of the image. Although, the splines are stated to be able to be rendered in real-time their evaluation is complex and on top of that the splines are not able to preserve sharp features other than those preserved by the segmentation.

In our approach we also create Bézier triangles for our mesh, but directly from detected image edges and soft features. We represent colour inside each Bézier patch as mesh colours [MSY19]. This allows to represent textures in a detailed way and in addition provides a cheap, yet accurate representation of the original image.

9.3 OVERVIEW

Our vectorisation method automatically converts an input image into a vector image. We pose no restrictions on the content and the images can have varying types, from natural images to design graphics. We want to create a vector representation that is editable, which can be rendered efficiently and which is a sparse representation. For this purpose we

design our pipeline so that it can extract image features that capture representative geometry, shading and texture. These features are vectorised into 2D splines and remain as handles for high-level edits. The vectorised features are used as constraints for our mesh generation step, and in the interest of sparsity, we insert curved constraints directly so that a curved triangular mesh exactly encompasses them. The mesh generation step is to remain efficient, but yet creates a good topology that is sparse enough for the user to edit geometrically. Each triangle is then equipped with a mesh colour patch, which is of higher density than the cubic triangles, so that each triangle can capture a lot of texture detail. We fit the mesh colours automatically, using a new automatic texture transfer process. Our vector representation is rasterised efficiently in real-time using tessellation shaders and the efficient evaluation of the mesh colour patches.

Figure 9.2 visually depicts the steps in our pipeline. First, the main features of the image are extracted (Section 9.4). We then generate a mesh (Section 9.5), which is followed by colour fitting of mesh colours (Section 9.6). Figure 9.1 shows the same intermediate steps for a more complex input. In the following we detail the individual steps of our pipeline.

9.4 FEATURE EXTRACTION

We want to extract those image features that represent geometry, shading and texture from the input raster image. These are the features that should be preserved and are the key to successful image vectorisation. We aim to tackle most image content, but we do not want and are not able to make any assumptions about their content. We want our vectorisation to be able to correctly preserve edges of the original image, but also to be able to correctly model the shading and colour gradients in between those regions. To this end we define two types of features: *hard* and *soft* edges. Hard edges are the results of colour discontinuities and capture contours, texture and salient shapes, and should be preserved as much as possible and remain sharp. Soft features do not need to be preserved as sharp, but should preserve shading of the image as best as possible. Detecting such features is still an active topic even after decades of research into edge detection [MA09]. Recent neural approaches have seen increasingly good performance at inferring salient edges at the object level [XT15, LCH+17, HZY+20]. However, these methods are not that well suited to surfacing progressive texture detail.

EDGE EXTRACTION For image vectorisation purposes the edge detection procedure as set out by Canny [Can86] has been sufficiently apt to capture hard image features [XLY09, CLL+20]. We typically set the low and high thresholds to 15 and 100 with respect to the range [0, 255], respectively. However, while Canny is good at detecting hard edges, it



Figure 9.3: *Top Row: Original image and the quantized grayscale image with the extracted hard (red) and soft (green) edges overlayed on top. Bottom row: Our vectorised version without using soft edges (left) and with soft edges (right). Note that soft edges help capturing more detail and to avoid artifacts on the reflections of the statue and in the background.*

fails to pick up soft edges and shading: lowering the thresholds brings in too much noise and/or unwanted texture detail. Too much noise will impact the subsequent steps of the pipeline negatively, as the mesh generation step needs to incorporate all of the detected features. Thus, we propose a new simple procedure to extract soft edges to complement the hard ones found by Canny.

When vectorising images using only hard edges we noticed that the missing soft edges are typically orthogonal to the smooth colour gradients present in the image. Such features are easily exposed using an image quantisation or banding step. We take a greyscale version of the image and quantise it into separate levels (we use 20 by default), and trace the discontinuities of the different bands. This is similar in spirit to the iso-contours used to vectorise brushstrokes in [BDF14] and to the

posterisation strategy to vectorisation as used by popular vectorisation algorithms. We do not directly use the traced iso-contours, but process them further.

EDGE FILTERING Hard and soft edges may overlap in areas where image gradients change quickly. This mostly occurs near hard image features and in those areas there is little room left between contours of the banding. Again this would negatively affect the mesh generation step (Section 9.5) as many triangles would need to be generated. For this purpose we filter the soft edges around the perimeter of neighbouring hard edges. This effectively removes them from areas where hard edges are already present. In practice we achieve this by creating a distance transform of the previously extracted hard edges and filtering the extracted soft features based on their distance to the nearest hard edge. Figure 9.3 shows the result of quantising the greyscale image, and the subsequent soft features extracted from them (Figure 9.3, top right). These features help us better capture soft image details, such as all the blurry background elements in the input image (Figure 9.3, bottom row).

9.5 MESH GENERATION

The goal of the mesh generation step is to create a mesh of cubic Bézier triangles that incorporates the detected features. First we vectorize the detected edges by converting them to cubic Bézier splines, subsequently these splines serve as the constraints for the curved triangulation step.

EDGE VECTORISATION The extracted hard edges are traced and linked into chains of pixels so that they can be easily vectorised into Bézier splines. The splines help us capture curves and remove aliasing present on hard edges later on. We enforce C^0 continuity and G^1 continuity as needed. The actual vectorisation process is done along the lines of [Sch90]. We recursively fit the splines to the chains of pixels by fitting to the whole pixel chain. We keep the fit only if the error between the polyline approximation of the Bézier curve and the pixel chain is half a pixel. If the error is larger than that, we split the pixel chain at the point of maximum error and we perform the same fitting step on the respective halves of the previous pixel chain. We continue until the whole pixel chain is successfully converted.

We could perform the same process of vectorisation on the pixel chain obtained from the tracing of the soft edges, but as these do not represent salient image features there is no need to preserve them as accurately as the hard edges. Therefore the vectorisation approximates rather than interpolates the soft edges, meaning that we do not strictly enforce the start and end points of the individual curves to lie over the actual pixel-chain positions. Figure 9.3 (top, right) shows the vectorised edges obtained from the input image. As can be seen, the hard edges are

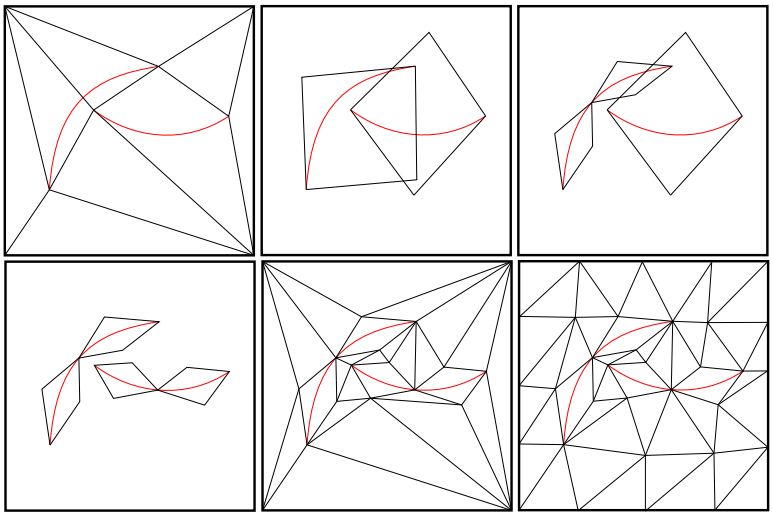


Figure 9.4: *The curved triangulation algorithm by Mandad & Campen. Top row: Left: Degenerate triangulation obtained by using constraints naively. Middle: the initial curves with their guarding triangles intersect. Right: the largest curve is split and results in two smaller guarding triangles, one of which still intersects. Bottom Row: Left: The largest guarding triangle is split and yields again two smaller pairs of guarding triangles. Middle: The curves, as linear constraints, and the guarding triangles are input to the triangulation algorithm. Right: the triangulation is optimised to improve the shape of the faces in the resulting triangulation.*

accurate, whereas the soft edges offer an approximation of the bands from the quantisation, without affecting the quality of the reconstruction. Hard and soft edges are kept separate to be handled differently in later stages of the pipeline, as they require different smoothing conditions.

CURVED TRIANGULATION The features extracted from the previous step represent curved edges in the image. We want to generate a mesh using the curved constraints, but by using standard linear meshing techniques we would run into trouble. Naively inserting the curves by their end points as linear pieces and only afterwards curving them could lead to supporting straight lines segments crossing the curved ones. In addition two curved segments that do not intersect each other might intersect each other when considering only the linear segment connecting their respective start and end points. An example of this can be seen in Figure 9.4, top left. An option to fix this is to subdivide the curved segments until the polyline approximation is accurate enough, but this procedure would quickly increase the number of faces introduced in the subsequent triangulation step. We want to exactly preserve

the curved constraints, leading to a lower number of faces, and create a curved triangulation that incorporates them, ensuring that the curved edges are directly interpolated. Curved triangulations have been used before in a vectorisation settings [XLY09], but the curved triangulation is obtained as the result of an untangling of an initially linear mesh. The generation of our triangular mesh is created in a more efficient manner, by first examining the relations between individual curves.

We follow the example of Mandad & Campen [MC20] who describe an efficient means to create a curved triangulation which guarantees that no degeneracies, crossing segments or flipped triangles, will remain. The technique constructs pairs of guarding triangles around each individual curved segment. The guarding triangles are computed so that the curve itself and the control polygon are completely contained in the guarding triangles. The guarding triangles are meant to be inserted as constraints into a standard constrained Delaunay triangulation procedure. Then by checking intersections of guarding triangles we can determine whether a curve needs to be subdivided further. By subdividing a curve we will also create two new pairs of guarding triangles for the introduced curve pieces. These guarding triangles will cover a much smaller area than the original one. Figure 9.4, top middle, shows a scenario where the two pairs of guarding triangles of two separate curves intersect. The quickest way to resolve intersections further on is to split the curve whose guarding triangles have the largest area. Naturally, the guarding triangle that has the large area can be expected to have the most intersections, and by splitting this curve (see Section 2.2.3) it we gain the most. In the case mentioned earlier we split one of the curves, but one of the resulting guarding triangles still intersects. We again select the curve with larger area guarding triangles and split those resulting into two disjoint sets of guarding triangles. At this point we can safely triangulate the curves and guarding triangles and the resulting triangulation will be completely valid, even with the curved segments included. The triangulation can be improved further, by restricting the maximum length of a segment and by inserting supporting points regularly into the empty portions of the image plane.

The procedure is efficient as it resolves intersection quickly and typically only needs few subdivision of curves to generate a valid triangulation. We accelerate the intersection testing by creating a *bounding volume hierarchy*, where we create bounding volumes first for the complete features and then their individual curve pieces. Alternatively, techniques such as TriWild [HSG+19] could be used to generate the geometry instead. After the triangulation step we have completely determined the topology of our mesh. We can then fix also the geometry and parametrisation of each of the faces.

Since all curved pieces are already given as cubic Bézier curves we can easily convert all supporting straight linear segments to cubic Bézier curves too. Then using the control points of the edges of each

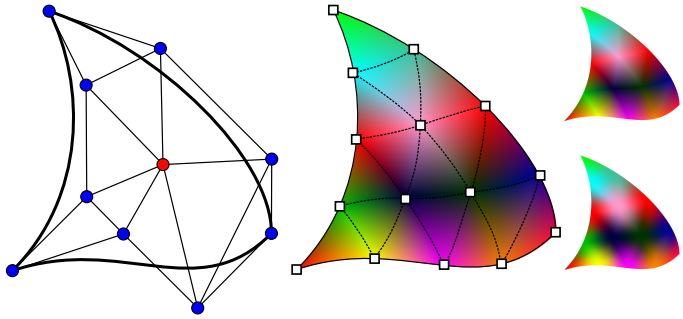


Figure 9.5: *Left: A cubic Bézier triangle with control points (the central one is in red). Middle: Resolution* 4 *patch texture with* $(4+1)\cdot(4+2)/2=15$ *mesh colours mapped on the cubic Bézier triangle. Right: Linearly (top) and quartically (bottom) interpolated mesh colours.*

face we can construct a cubic Bézier triangle (see Figure 9.5, left, or Section 2.3.2). The only missing thing is the central control point (red point) as all blue points are defined already. We define the central control point to be the average of all edge control points. By adding this central control point the parametrisation of each face is now fixed. Additionally, we also keep track which of the edges of the triangles are soft or hard by tagging them. The supporting edges inserted by the triangulation step are always tagged as being soft.

9.6 TEXTURE TRANSFER

The use of Bézier triangles as the vector primitive provides for a way to evaluate our mesh geometrically, but does not provide us with a way to handle varying texture detail effectively. Assigning colour values to the control points will not give enough expressibility to model such texture detail. For this we create a mesh colour patch on each of the triangles of our mesh. Mesh colours [YKH10] have not been considered before in the context of image vectorisation, when not counting the base case of vertex colours or standard linear interpolation on triangles. Mesh colours provide a convenient means to store colour and texture information on a per-patch basis. This removes the need for global texture coordinates, as the colours are defined in the parametric domain of a mesh patch. Therefore, there is no need for complex UV-maps or unwrapping techniques and the generated mesh can be used directly. This has the advantage that texture evaluation becomes robust to transformations and edits. The typical work-flow of using mesh colours is for an artist to paint on the mesh directly [YLT19], and therefore no standard ways of transferring image information to mesh colours is available.

MESH COLOURS We map triangular mesh colours to the cubic Bézier triangles which were created in the mesh generation step (Section 9.5). Figure 9.5, middle, shows a schematic view of mesh colours which are mapped onto the cubic Bézier triangle depicted on the left. Each mesh colour patch is equipped with a resolution r which handles the total number $R = \frac{(r+1)(r+2)}{2}$ of mesh colour vertices $t_{\mathbf{i}}$ per patch, where $\mathbf{i} = (i, j, k)$, $i + j + k = r$ and $i, j, k \geq 0$. The mesh colour patch depicted in Figure 9.5 has a resolution $r = 4$. We follow the procedure of [MSY19] to evaluate the mesh colour patches using barycentric coordinates. For clarity of presentation and to prepare the ground for our proposed mesh colour fitting scheme (Section 9.6.1), in the following we detail the steps for evaluating a mesh colour patch.

The patches are evaluated using their barycentric parametrisation. The barycentric coordinates $\boldsymbol{\phi} = (u, v, w)$ with respect to a triangle $\triangle$ in the mesh are used to determine the three closest mesh colours: $t_{\mathbf{i}}$, $t_{\mathbf{j}}$, $t_{\mathbf{k}}$. These colours together determine a mesh colour (sub)triangle of $\triangle$. From the coordinates $\boldsymbol{\phi}$ we determine the local barycentric coordinates $\overline{\boldsymbol{\phi}}$ inside the mesh colour triangle. We can use these local coordinates to either linearly

$$\overline{\boldsymbol{\phi}} \cdot \begin{pmatrix} t_{\mathbf{i}} \\ t_{\mathbf{j}} \\ t_{\mathbf{k}} \end{pmatrix} = \left(\overline{u}, \overline{v}, \overline{w}\right) \cdot \begin{pmatrix} t_{\mathbf{i}} \\ t_{\mathbf{j}} \\ t_{\mathbf{k}} \end{pmatrix}$$

or quartically

$$\begin{pmatrix} \overline{u}^4 + 4\overline{u}^3(\overline{v} + \overline{w}) + 12\overline{u}^2\overline{v}\overline{w} + 3\overline{u}^2(\overline{v}^2 + \overline{w}^2) \\ \overline{v}^4 + 4\overline{v}^3(\overline{w} + \overline{u}) + 12\overline{v}^2\overline{w}\overline{u} + 3\overline{v}^2(\overline{u}^2 + \overline{w}^2) \\ \overline{w}^4 + 4\overline{w}^3(\overline{u} + \overline{v}) + 12\overline{w}^2\overline{u}\overline{v} + 3\overline{w}^2(\overline{u}^2 + \overline{v}^2) \end{pmatrix}^T \cdot \begin{pmatrix} t_{\mathbf{i}} \\ t_{\mathbf{j}} \\ t_{\mathbf{k}} \end{pmatrix}$$

interpolate the mesh colours. The former results in piecewise linear C^0 colour interpolation and the latter in piecewise quartic C^1 colour interpolated. The quartic interpolation is slightly more expensive computationally and visually seems to drag or flatten out the colour a little more. Overall there is not a lot of difference as can be seen from Figure 9.5, right column), except in cases where the individual mesh colours vary extremely.

The resolution r of each patch can be determined on a per-triangle basis [Yuk16], effectively adjusting the amount of texture detail that can be represented and the storage required for it. Naturally, with an increase in resolution the approximation ability also increases. In Figure 9.6 we show a synthetic example where a static mesh, that is not aligned with image features, approximates the same image with increasing resolutions of mesh colours. Higher resolutions are better able to approximate the input.

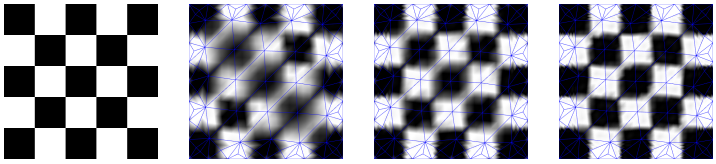


Figure 9.6: *From left to right: Input image and vectorisations using the same mesh but different resolution (2, 4, and 6) of mesh colours. In this test example we made the mesh not to capture the features of the input image correctly, but as can be seen, increasing the mesh colour resolution leads to increasingly better approximations of the input image.*

9.6.1 *Mesh colour Fitting*

Using the parametrisations of the patch we can fit mesh colours to them. Ideally, we would like to fit the whole mesh using a global process, so that we can automatically create smooth transitions between triangles over smooth edges. However, solving this in a global least-squares sense leads to a large system of equations, and would not be practical with regards to memory and performance. We simplify the problem by fitting to each triangle individually, then afterwards we smooth mesh colours across edges at edges that are marked as smooth.

We first fit mesh colours to each cubic Bézier triangle T parametrized over $\triangle$ separately. We sample each T in the mesh uniformly to obtain the pairs $(p_i, \boldsymbol{\phi}_i)$ for every sample position p_i in the image with $\boldsymbol{\phi}_i$ its barycentric coordinates in $\triangle$, i.e., $T(\boldsymbol{\phi}_i) = p_i$. To effectively fit a patch texture, we need to ensure that the number of samples m satisfies $m > R$. Using the image position p_i of each evaluated pair, we look up the bilinearly interpolated colour value $I(p_i) = c_i$ in the input raster image I. Using $\boldsymbol{\phi}_i$, we determine the local barycentric coordinates $\overline{\boldsymbol{\phi}}_i$ of p_i in its mesh colour (sub)triangle. We then minimise the following function on a per-triangle basis, used once per colour channel:

$$\min \sum_i \left(T^c(\boldsymbol{\phi}_i) - c_i\right)^2,$$

where T^c evaluates the colour corresponding to T. This is a standard least squares problem that we solve for the mesh colours of T. The matrix of the system can be reused for fitting triangles with the same mesh colour resolution r, since it is independent of the actual image positions and it uses only parametric positions (expressed in terms of $\boldsymbol{\phi}$), which are generated uniformly for each triangle. We efficiently perform this sample pair creation process in parallel for each triangle using GPU compute shaders.

We can increase the efficiency or quality of the colour fitting step by considering different resolutions of mesh colours. We can use different

resolutions based on the size of the triangle in the original image. We assume that the features extracted during the extraction step are oriented in such a way that the areas in between only contain slowly varying colour gradients and highly varying regions of colour are captured by smaller triangles. Therefore, the smaller the triangle, the less resolution is needed to be able to represent the textured area of the original image. Practically this means that we order the triangles based on their pixel area in the original image into three different bins. The bins correspond to an increasing resolution of mesh colours $r = 1, 2, 4$. This benefits performance as lower resolution textures require less samples to be generated, reducing CPU-GPU congestion and improving the speed of the fitting.

Alternatively we could progressively fit increasing resolutions of mesh colours based on the error with respect to the underlying image region. We can start fitting a patch based on resolution $r = 1$, then depending on the error we could attempt fitting with higher resolutions. Naturally, this process will converge as the mesh colour resolution at one point will exceed the actual pixel resolution. This fitting scheme is less efficient, as potentially multiple fittings should be done, but could be made more efficient by generating enough samples to do at least two fittings with different resolutions. Even this could be accelerated by generating only the samples not covered by lower resolutions.

Optionally, after fitting the resolution of mesh colours can be increased for lower resolution so that each patch has a unified resolution. Additional mesh colours can be generated through linear interpolation of existing mesh colours. This approach allows a unified approach of handling and rendering triangles later on.

OFFSET SAMPLING Bilinear sampling of colours is useful to not let noisy input enter the colour fitting procedure and to be able to access sub-pixel location of the original image. However, sampling near to hard edges causes incorrect sampling, because the bilinearly sampled colour can be a combination of colours on either side of the edge. This causes colour bleeding artefacts to appear on triangles incident to sharp features (Figure 9.7, top). We take the same strategy as set out by Liao et al. [LHFY12], where a one-pixel region around hard edges is padded. When sampling at hard edges, the sampling is offset towards the padded region. In this way we can be sure that the sharp features present in the raster image are preserved by sampling on the correct side of the edge (Figure 9.7, bottom). This leads to crisp edges and features without affecting the interpolation on either side of the edge.

COLOUR SMOOTHING Fitting each triangle individually will lead to small discrepancies between the edge mesh colours on adjacent triangles. Naturally, this is not a problem for hard edges, but soft edges should be soft in that their interpolation is as smooth as possible. This

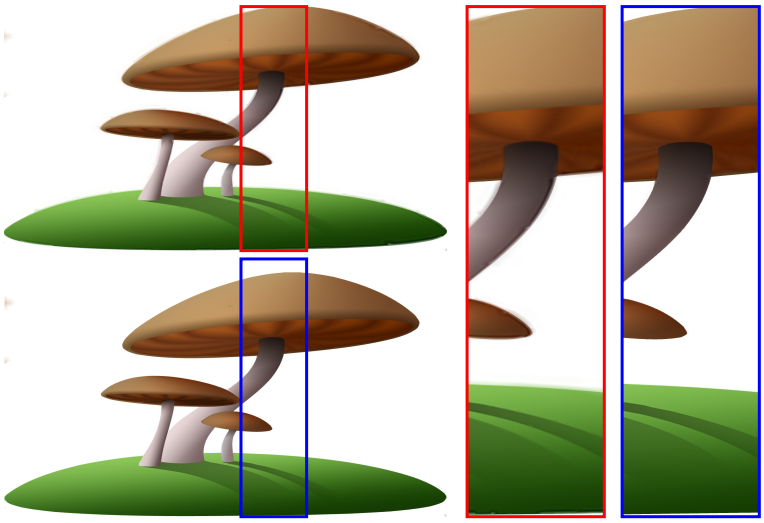


Figure 9.7: *Vectorized image without (top) and with (bottom) offset sampling around hard edges. Insets show the increased sharpness not only on silhouettes, but also hard edges coming from other sources like shading.*

increases the fidelity of our vectorisation as the underlying mesh is not able to be seen from the resulting interpolation. We create a simple procedure where mesh colour values of edges are averaged with respect to their values on adjacent mesh colour patches. The procedure is only done for edges which are tagged as being smooth edges, i.e. supporting edges from the triangulation step (Section 9.5), or the smooth edges from the feature extraction stage (Section 9.4). By averaging the mesh colours on edges we have guaranteed at least C^0 interpolation of colour across smooth edges. Figure 9.8 shows the difference before and after colour smoothing. Before smoothing, the underlying triangulation is clearly visible in some regions. After smoothing, these artefacts vanish and the resulting vector image has C^0 colour interpolation everywhere except at hard features.

9.6.2 *Rendering*

The combination of parametric patches to evaluate geometry and a parametric representation to also evaluate colour lends itself extremely well for efficient rendering. We use tessellation shaders in the modern graphics pipeline to evaluate the cubic Bézier triangles. The mesh colours are stored in textures as proposed by [MSY19] and we evaluate them using the process outlined in Section 9.6 through the use of a fragment shader. Inadvertently this leads to duplication of all mesh colour data stored at

vertex positions or along edges. For modelling of sharp features this is of course necessary, but for smooth features there is duplication.

Recently improvements to the mesh colour techniques [Yuk16, YLT19, MSY20] have been proposed that extend them with increased filtering capabilities, such as mipmapping and anisotropic filtering, and hardware support for efficient texture access and filtering. We have not focussed our efforts into improving the filtering of our representation, but we see no reasons as to why the aforementioned techniques could not be applied to our representation.

Due to the unification of the resolution of the different resolutions of mesh colour patches used in our representation, we are able to render each vector image using a single draw call. We can increase the performance of our rendering even more by employing adaptive tessellation based on the projected edge length of the triangles. Longer edge length calls for increased tessellation levels whereas smaller triangles should not have to be tessellated to such a dense level. By using the edge length as a metric for our tessellation level, we can guarantee that the resulting tessellation is gap-free.

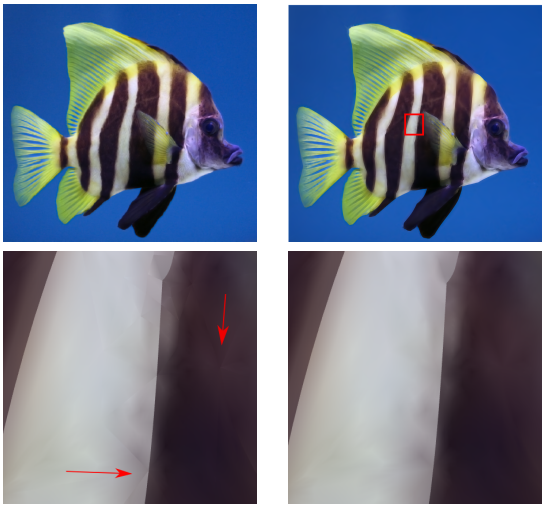


Figure 9.8: *Top: Input image (left) and our vectorised result (right). Bottom: Without smoothing (left) the seams of the mesh become apparent (some examples are highlighted by the red arrows), something especially undesirable at large magnification factors typical for vector graphics. The proposed smoothing of the mesh colour values of neighbouring mesh colour patches removes such seams for a more natural and higher quality result (right).*



Figure 9.9: *Left: Input raster images and their corresponding error maps with respect to our rendered vector images (right). Overall, these examples were vectorised accurately all over the image with a mean squared error of* 2.73, 2.25, 4.69, *respectively top to bottom . Insets in the last row show our increased sharpness (right), plus some extra texture retained from the input image (left). Bottom artwork by Allison Bamcat.*

9.7 RESULTS

In this section we show the results which are obtainable through our vectorisation pipeline. We have already shown several results through-

out the chapter. Figure 9.1 shows an intricate vectorisation of a painting. The intricate detail is nicely preserved by our approach as well as sharp features, specular highlights and folds (Figure 9.1, f) are preserved. Figure 9.2 shows a simple logo that turns into a relatively simple vector image that could be easily edited further to remove the background, for example. Figure 9.3 shows an interesting combination of sharp and soft details, which our image features help capturing accurately. Figure 9.7 shows another example of clean stylised graphics faithfully captured by our representation.

In Figure 9.9 we present an additional set of results that showcase a variety of different image features and textures. We also depict the errors with respect to the reconstructed image from the vector representation. As can be seen most of the errors seem to accumulate along the edges of the original image. This error is mostly likely due to our handling of edges. We vectorise the pixel edges (as mentioned in Section 9.5) and it is likely that our vectorised representation does not perfectly coincide with the pixel edge. In addition, we employ the specialised sampling along hard edges to avoid colour bleeding through the sharp features (Section 9.6), this might slightly alter the results along edges.

9.7.1 *Performance*

The choices we made in the construction of our vectorisation pipeline leads to the vectorisation process being quite efficient. In Table 4 we report the performance of our vectorisation method for several of the results shown in this chapter. We break down the total time it takes to vectorise an input image by the intermediate timings of the individual steps of the pipeline: feature extraction, mesh generation and colour fitting. In addition we also show the rendering times for the rasterisation of the obtained vector representations of the respective input images. We ran the method on my own low-end laptop, which has the following specifications. It has a NVIDIA MX150 GPU with 2GB of VRAM, 8GB of RAM and an Intel i5-8250 CPU.

Our method can vectorise all of the listed images with good performance. It takes at most only just a few seconds to go through the whole vectorisation process. The timings of the individual components are dependent on the content of the image and on each other. The feature extraction step is dependent on the resolution of the input image and also of the number of features in the input image. In turn, the mesh generation step is dependent on the number of extracted features and their relative positioning to each other. Many small curved features that lie within close proximity to each other will generate more triangles than a single curved line. The colour fitting step is then dependent on the number of generated triangles. However, due to the parallelisation and the choice of adaptive resolutions of mesh colours this step can be sped up quite a bit.

The combination of parametric representations for both geometry and colour leads to an efficient means to evaluate our vector representation through the use of tessellation shaders and fragment shaders. Even for the results which generate a large number of triangles, we still achieve real-time rates of performance.

COMPRESSION Our vectorisation provides a sparse representation of the original image. To illustrate this further we have provided compression ratios for select images featured in this chapter and mentioned them in Table 4. We calculated the compression ratio with respect to the raw image data as is common with other vectorisation papers [CLL+20]. We base the size of our representation by the following. First of all we consider a difference between linear triangles $3 \cdot 2 \cdot 4 = 24$ bytes and curved triangles, a triangle with at least one curved edge, with $9 \cdot 2 \cdot 4 = 64$ bytes each. Furthermore, we consider the three different resolutions which have $3 \cdot 3 = 9$, $6 \cdot 3 = 18$, and $15 \cdot 3 = 45$ bytes each. Then the actual raw size of our representation is $\#lineartriangles \cdot 24 + \#curvedtriangles \cdot 64 + \#smallpatches \cdot 9 + \#regpatches \cdot 18 + \#largepatches \cdot 45$ bytes. Our approach is able to achieve comparable compression rates to other vectorisation techniques. The raw data could be compressed even further

Table 4: *The performance of our vectorisation pipeline on several of the results featured in this chapter. The time measurements are shown in seconds except for the rendering time, which is shown in milliseconds, and are split over the elements of our vectorisation pipeline: FE = Feature Extraction, MG = Mesh Generation, CF = Colour Fitting, RT = Rendering Time, CR = Compression Ratio.*

Image	Resolution	$\triangle \cdot 10^3$	FE	MG	CF	Total	RT	CR
Fig 9.1	854×1024	~ 38	0.5	0.8	3.0	4.3	~ 23	0.72
Fig 9.2	785×618	~ 2	0.24	0.1	.5	0.9	< 1	0.07
Fig 9.3 bottom left	848×1280	~ 50	0.5	0.7	4	5.3	~ 25	0.78
Fig 9.7	1280×1181	~ 4	1.7	0.05	1.1	2.9	~ 2	0.06
Fig 9.8	441×441	~ 10	0.03	0.1	1.1	1.3	~ 3	0.89
Fig 9.9 top	1920×1284	~ 22	4.7	0.4	2.7	7.8	~ 25	0.18
Fig 9.9 middle	1280×853	~ 8	0.9	0.7	2.8	4.5	~ 10	0.16
Fig 9.9 bottom	1198×1198	~ 60	0.8	0.9	5.9	7.6	~ 17	0.77
Fig 9.10 left	864×864	~ 147	0.8	18.6	16	35.6	~ 33	2.68
Fig 9.10 right	864×864	~ 20	0.5	1.9	3.5	6	~ 16	0.52
Fig 9.14 left	639×479	~ 22	0.03	0.3	1.8	2.1	~ 7	1.24
Fig 9.16	1113×1291	~ 18	1.5	0.3	2.3	4.1	~ 24	0.26
Fig 9.17	441×631	~ 8	0.1	0.1	1.1	1.3	~ 4	0.56

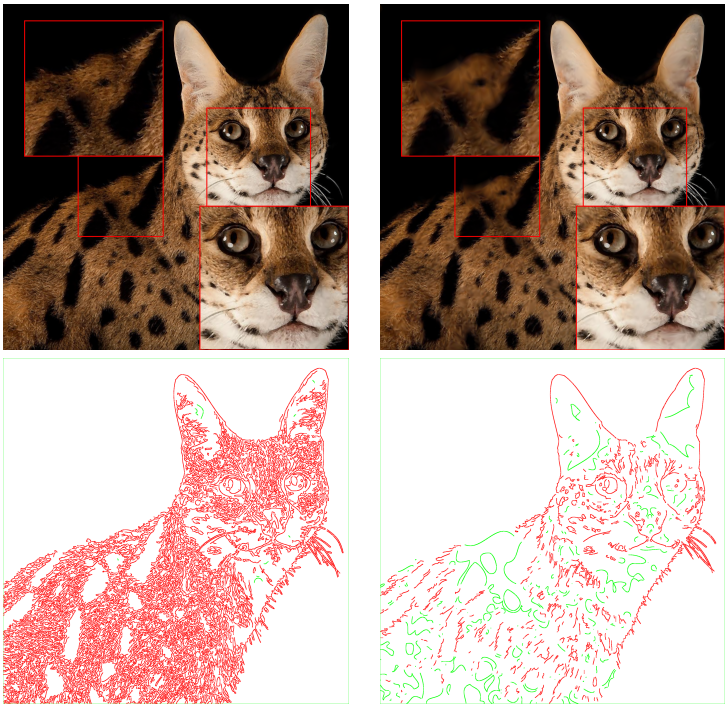
using standard compression techniques such as zip. In addition, the size of the representation could be decreased even further, by not optimising for triangle size, but rather for the content that a triangle depicts. having lower mesh colour resolution for smoother or constant colour image regions.

FEATURE PARAMETERS The selection of features has a big effect on the resulting vectorisation as it influences both mesh generation and colour fitting steps. To investigate the influence we have vectorised an input image with varying levels of edge detection. Figure 9.10 shows two different vectorisations of the same input image. On the left we can see that there is an abundance of detected hard edges in the furry areas of the depicted animal, by increasing threshold values we are able to extract a sparser set of features, which is depicted on the right. In areas that prominently feature small hard features we can see that now the soft features obtained from colour banding take over. Naturally, this blurs the features in those regions as their sharpness is no longer preserved. Both images are still able to create a good approximation of the original image as it correctly extracts and preserves the most salient image features. The large number of features also leads to a large number of generated features, although it is still able to be rendered efficiently, it negatively affects the compression ratio as seen from 4, rows 8 and 9.

9.7.2 *Editing*

Due to the sparseness of the mesh it can easily be manipulated through low-level deformations of the geometric attributes of the mesh. This entails dragging vertex positions and tangent handles of the cubic edges of the Bézier triangles. Higher-level deformations, like manipulating a single curve are also possible. An example of geometrically manipulating the mesh is shown in Figure 9.11. Even more elaborate manipulations such as ARAP deformations [IMH05] or grouping of features as proposed in [LHFY12] are also possible, but we did not implement them in our pipeline. We do not allow the user to manipulate the geometric locations of the mesh colour vertices as these remain linked to the parametrisation of the faces of the mesh. Although, in theory these vertices could be displaced too, but it would raise interesting questions on how to expose such control to the user.

Due to the performance of the pipeline it becomes possible to do interactive vectorisations. We have given the user the ability to draw spline curves on top of a raster image. Like our automatic feature extraction step, the user is able to mark them either as being a hard or soft feature. The created curved features are subsequently used as input to the rest of the pipeline, which generates the mesh and fits the colours. This process is able to be achieved with interactivity as the vectorisation is able to be generated nearly instantly after the user is done creating curves. In



5702 features 1048 features

Figure 9.10: *Two vectorisation of the same image using different edge detection parameters. Left: Canny edge detection with low 15 high 100 and right low 100 high 200.*

addition adding curves is a local operation, and a previously generated mesh can be locally updated and retriangulated and colour be fitted. Figure 9.12 shows an interactively created vectorisation of a raster image. This interactive workflow could be used to clean up automatically vectorised images by putting in or fixing features that were not captured correctly by the feature extraction step.

9.7.3 *Comparisons with Previous Work*

We compare our method and most importantly our choice of primitive against some of the best performing primitives used for image vectorisation to date. These are thin-plate splines (TPS), subdivision-based methods and diffusion curves. We have strived to make the comparison as fair as possible, but we were unable to obtain all the exact same input images or sometimes used an unsegmented version of an image.

Figure 9.13 shows a comparison with the latest TPS-based method [CLL+20]. As can be seen, their method is great at capturing

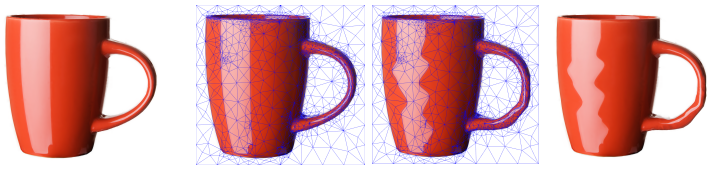


Figure 9.11: *Our vector images are easily editable either by moving mesh vertices or editing the tangent handles of the cubic Bézier triangle edges. From left to right: One of our vectorised results, the mesh of the vector, the edited mesh, and the rendered edited image. Please, see the accompanying video for the editing session.*

fine texture detail, but at the same time it scales similarly to a raster image, thus loosing some sharpness around hard edges. In addition, their vector patches often show seams under magnification. In contrast, our method keeps sharpness around hard edges and does not show texture seams thanks to our colour smoothing around soft edges. Our texture detail is affected both by the feature extraction step and the patch resolution, obtaining less realistic abstracted looks when not sufficient. Because our image quality is often comparable to that of [CLL+20], we chose not to include gradient meshes [LHM09] in our comparisons, as their limitations when capturing highly detailed textures were already demonstrated by Chen and colleagues.

Figure 9.16 shows a comparison against a previous TPS-based method [XLY09], where our simpler and more performant steps achieve comparable results to their more elaborate representation. Figure 9.17 shows a comparison with an apparently similar approach [LHFY12] that uses subdivision surfaces. Our decoupling into a spatial 2D mesh and 3D (RGB) mesh colours allows finer control over tessellation, capturing higher level of detail while achieving comparable smoothness and mesh density.

Figure 9.14 show comparisons with hierarchical diffusion curves [XSTN14]. We found that their image feature extraction translates into a global loss of clarity and detail for photos. For designed graphics, their method extracts cleaner features that produce quality closer to our method.

Finally, Figure 9.15 shows the generality of our method applied over images from [FLB17]. While our method is not meant to produce intuitive semantic layers, it is still able to retain the sharpness and details of the input graphics, through a simple mesh structure not difficult to edit afterwards (as seen before in Figure 9.11). It is also worth mentioning that our precise tracing of hard edges following the contours of the objects allows straightforward cut-outs to remove image backgrounds.

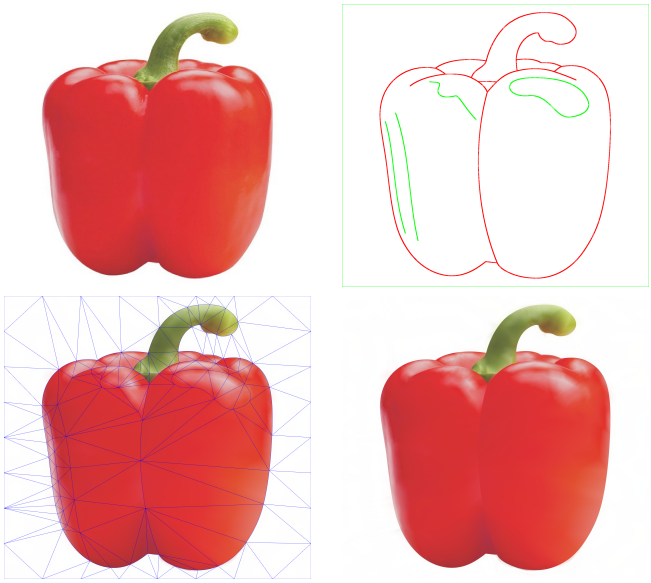


Figure 9.12: *A user guided vectorisation of a bell pepper. The user created 22 separate curve pieces which were used to generate a mesh of 223 triangles. Left to right: original image, sketched curves, generated mesh and resulting vectorisation.*

9.8 DISCUSSION

For existing vector graphics primitives, such as the gradient mesh (see Chapter 8) or diffusion curves, it is often mentioned that it is important that the primitive should smoothly interpolate colour. With this smoothness is meant that it should be at least C^1 or preferably higher. Our vector representation is only C^0 smooth, but given our proposed feature extraction step and the resolution of the mesh colours it is able to express smooth colour gradients without obvious artefacts appearing. Thus it seems that by careful extraction of features there is no need for smoother interpolation strategies like subdivision surfaces or thin-plate splines, which are less efficient in terms of rendering performance than our representation.

The vectorisation is only going to be as good as the features that have been extracted. Although, we use standard techniques for extracting edges from the image this step can easily be changed for another feature extractor. Our pipeline is in this sense independent of the ways features have been extracted and potential future improvements in this area could increase the abilities of our pipeline. Although, our feature extraction step is able to capture most of the salient features in the image there is still room for improvement. We have not focused on per-

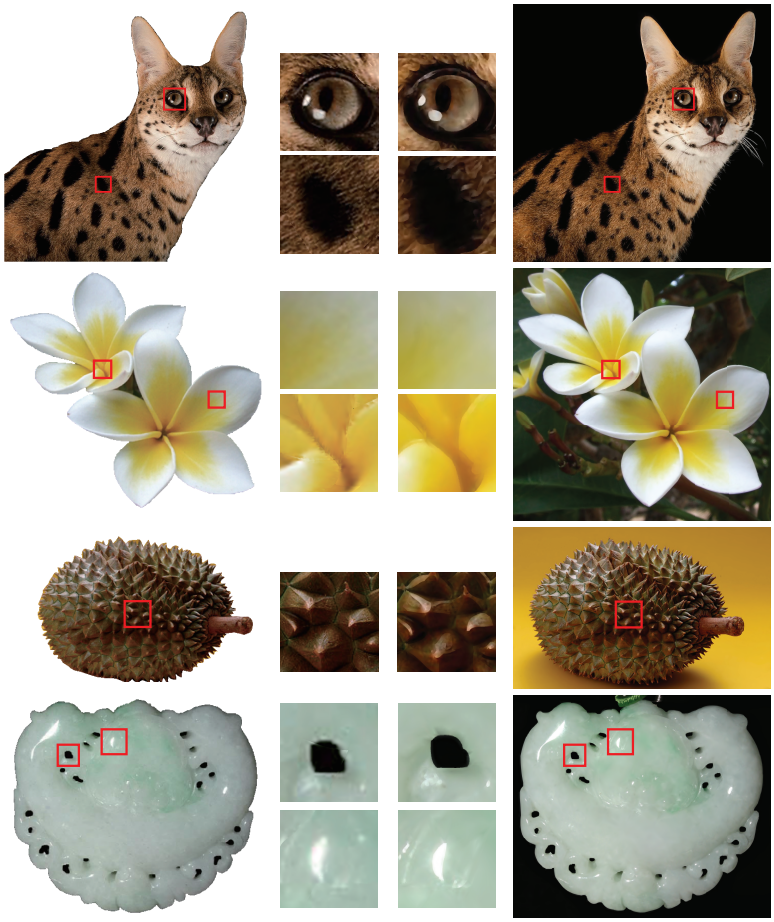


Figure 9.13: *Comparisons between [CLL+20] (left) and our proposed method (right). Our method seems better at preserving sharpness coming from geometric discontinuities (petals in the second row, holes in the last row), which are easily picked up by our extracted image features. However, ours is not that good at capturing extremely detailed textures like fur (first row). Insets also show the relevance of our colour smoothing across patches, absent in [CLL+20] (second and bottom rows). Vectorized backgrounds were not available from [CLL+20], but we included them for completeness.*

fecting the feature extraction step and leave room there for explorations with different filtering mechanisms and perhaps neural models for feature extraction.

The evaluation of our representation using tessellation shaders provides great performance, but it is of course still possible to render the images using only the CPU, or alternatively compute shaders. A piecewise linear version of the representation, at the level of the resolution

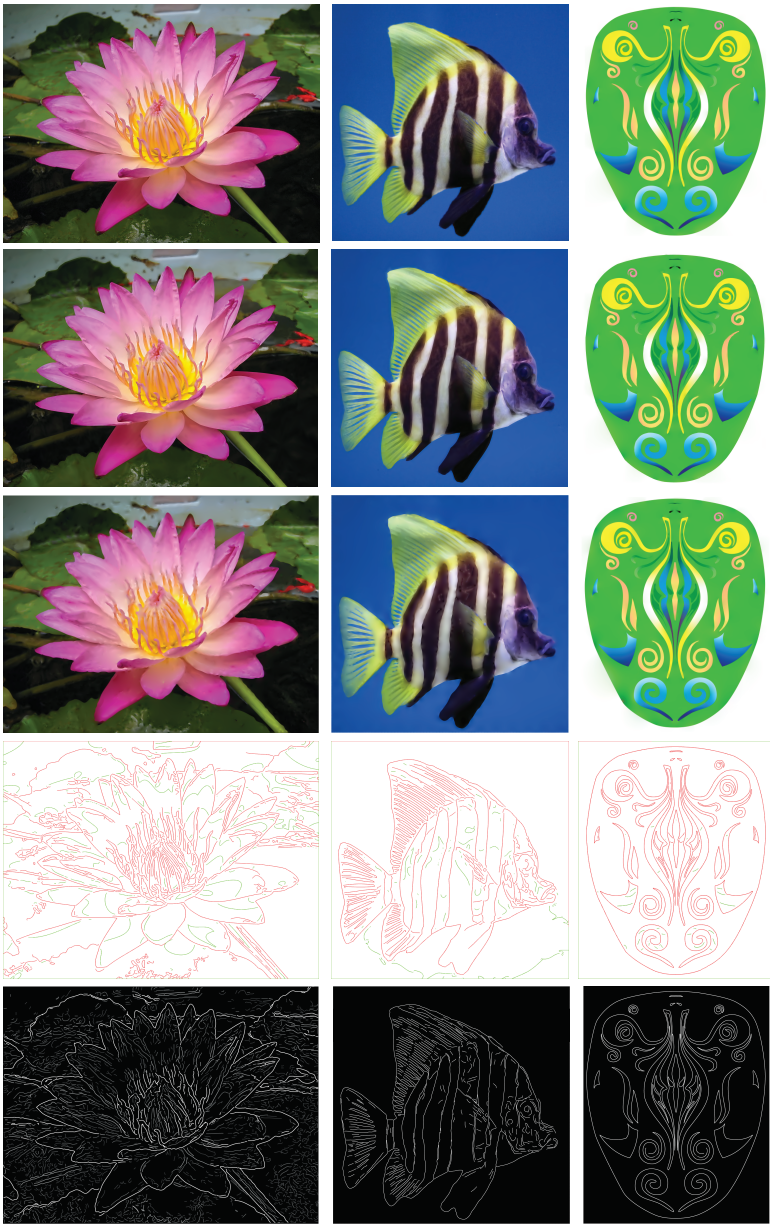


Figure 9.14: *From top to bottom: Input images, our results, results from [XSTN14], our extracted image features and the ones from [XSTN14]. When applied to photos, our method produces sharper and cleaner results across the whole image (please, zoom in for details). For simpler inputs with clearer discontinuities, both methods perform similarly.*

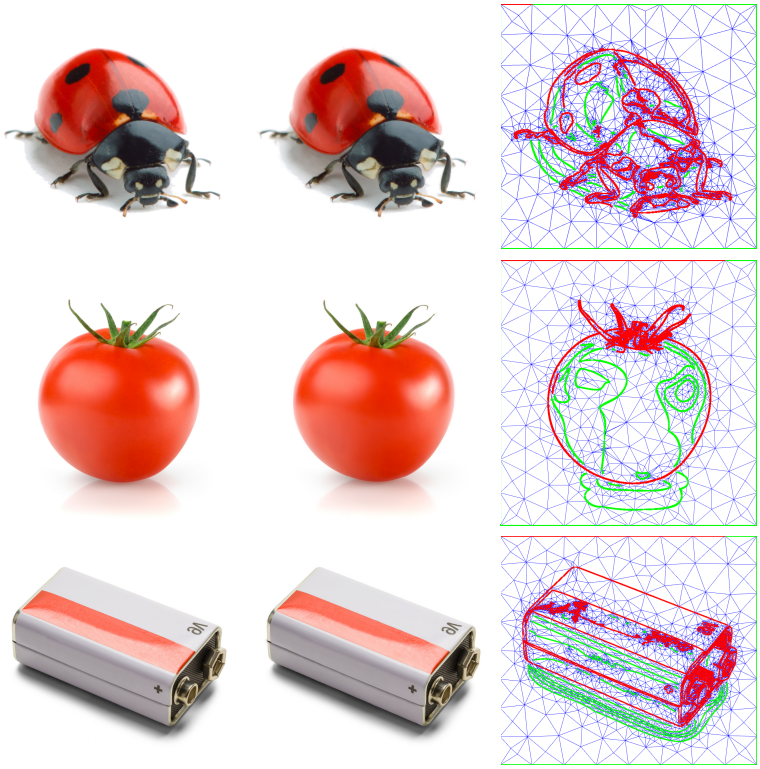


Figure 9.15: *From left to right: Input image, our rendered vector image, and the underlying mesh and image features. Our method does a good job for such simple objects but with precise shapes and materials, outputting an intuitive representation not difficult to edit.*



Figure 9.16: *From left to right: Input image, our results, results from [XLY09], our image features and triangle mesh, and the ones from from [XLY09]. The final quality is very similar between both methods, including sharpness around hard edges due to their feature alignment, and our offset sampling and mesh colours. Main differences lie in the 2D mesh, with theirs being sparser. However, the complexity of their mesh optimization and the evaluation of their thin-plate splines is higher than our simpler but more efficient approach. Note that the vectorized background was not available from [XLY09], but we included it in our result for completeness.*

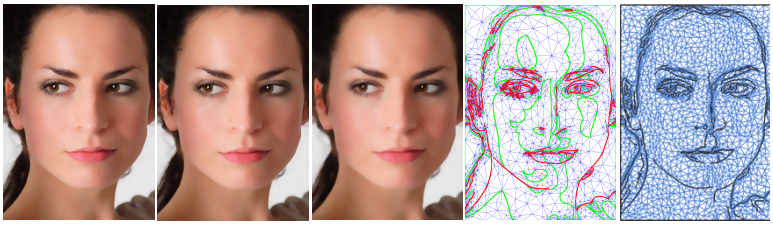


Figure 9.17: *From left to right: Input image, our results, results from [LHFY12], our image features and mesh, image features and mesh from [LHFY12]. While image features and triangle structure look similar, our mesh colour patches provide extra control to capture more detail, even in smooth regions such as under the nose. Our hard edges also preserve sharpness better in important features like the eyes.*

of the mesh colours, can be extracted at the expense of additional memory requirements. This representation could then be used as a bridge between older vector formats such as PDF [Ado06] that do not directly support our proposed vector representation.

Our algorithm strives to adaptively create a curved mesh based on the feature in the input image. Theoretically, we could cover the input image with just two triangles and create high-resolution mesh colour patches on them. Conversely, the image could be represented with a low resolution mesh colour representation that is applied to a pixel-dense triangulation. A balance should be found between these two extremes, and we think that our representation is a good contender for this. Additional optimisations are still possible, such as iterative feedback between different steps of the pipeline or a more error-driven approach to vectorisation.

9.9 CONCLUSION

We have created a fully automatic image vectorisation method that can vectorise a wide range of input images; from natural images to logos and cartoons. The vectorisation process is able to be achieved very efficiently and has good reconstruction accuracy. The use of curved triangles creates a sparse mesh that can exactly incorporate extracted hard and soft image features. The use of mesh colours is then able to efficiently, and faithfully transfer texture detail from the input image to the curved triangular patches. These patches can be efficiently rendered on commodity hardware and because of this it becomes possible to interactively edit the retrieved vector representation. Due to the efficiency of the pipeline it is also possible to interactively vectorise images.

10 CONCLUSION

In this chapter the thesis is concluded. We have presented several methods for dealing with arbitrary topology surfaces in the context of geometric modelling (Part I) and vector graphics (Part II). We will now revisit the chapters in order to present their main conclusions and we will relate them to the research questions posed in the introduction of this thesis (Chapter 1).

Arbitrary topology B-rep CAD models can be converted into triangular spline surfaces (Chapter 4). These surfaces are able to approximate the original NURBS surfaces and are able to exactly integrate the edges of the patches. With the use of Shirman-Séquin patches along the edges of the patches we are able to create smooth joins between patches with different parametrisations. Unfortunately, the surface has to be adjusted slightly along the edges of the patches, but this allows us to use the aforementioned patches to join together smoothly. We created three different variants, where each has varying levels of preservation of the originally sampled B-rep patch. The use of these triangular patches is not more invasive than previous triangular conversion methods and because of the increased smoothness the converted splines can be used effectively for analysis.

B-spline surfaces are powerful spline methods that create smooth surfaces from control points and basis functions. The generalised Bézier patch structure can be easily extended to use B-spline basis functions instead of Bernstein basis functions created from generalised barycentric coordinates (Chapter 5). The only changes that need to be made is to extend the domains of individual ribbons so that they have extended support and thus can use B-spline functions and some of the basis functions need to be adjusted so that they and their derivatives vanish at certain values. This procedure can be applied to any degree B-spline basis functions. The generalised B-spline patch structure can then be used to create surfaces over extraordinary vertices and faces by ensuring they are surrounded by regular regions, which can be ensured by employing arbitrary degree subdivision. The surfaces are able to be joined with G^k continuity to adjacent regular regions and other multisided B-spline patches. Even though the patches are internally C^k smooth, they still might show some shape defects. We can alleviate the shape effects by adjusting the basis functions to have increased weight so that the patches become fuller at the centre, but it requires careful manual adjusting to do so. Multisided patches are not able to be processed by traditional tessellation pipelines. Through the use of instancing and the subdivision of a regular polygon it is possible to efficiently

render multisided patches through tessellation. Even though it requires a few more steps to do so, it is still possible to render models with many polygonal patches in real-time. We demonstrated the utility of rendering multisided patches, by extending existing approximating Catmull-Clark subdivision schemes so that also models containing extraordinary faces can be directly approximated without additional subdivision steps.

The use of procedural noise functions with vector graphics is a great match (Chapter 7). The traditional gradient mesh can only convey as much details as vertices that are available in the mesh. By parametrising the mesh and using the parametrisation as the domain of the procedural noise function we are able to geometrically alter the noise domain by dragging vertices and tangent handles of the gradient mesh. We can give the user even more control by smoothly varying parameters for the noise functions over the mesh by letting the user specify those values at the vertices. The parameters give precise control of the frequencies of three different noise types: Perlin, Worley and Gabor noise. The option to have fine control gives the user the ability to create elaborate patterns with a sparse mesh, which would never be possible with an ordinary gradient mesh.

The extension of traditional gradient meshes to polygonal meshes alleviates the strict topology restrictions that traditional gradient meshes pose (Chapter 8). We created two new versions of a polygonal gradient mesh by altering the method of Chiyokura and Kimura for colour interpolation. This allows us to smoothly join multisided gradient mesh patches. We created two different versions, one based on the generalised Gregory patch and the Gregory-Charrot corner interpolator patch. The use of mean-value coordinates and special conditions for convex corners allows us to effectively use non-convex patches. We compare the two new primitives to cubic mean-value interpolants and the subdivision based topologically unrestricted gradient meshes. Our new representations based on the Gregory patch are just as effective as cubic mean-value coordinates in tackling non-convex patches and have comparable smoothness to subdivision based representation. Our representations are faster to render, because they can be rendered efficiently using tessellation.

We created an efficient image vectorisation pipeline that is able to vectorise a wide range of images, from natural images to design graphics (Chapter 9). We extract hard image features through edge detection and soft image features through quantisation of image colours. The features are vectorised and are embedded into a curved triangulation. On each curved triangle we create a cubic Bézier triangle on which a mesh colour patch is created. We efficiently transfer colour information from the input raster image to the mesh colour patches. Images can be vectorised within seconds and can be efficiently rendered using hardware tessellation. Even though the colour surface is only C^0 continuous the

feature extraction step results in a mesh with only slow variation of colour. The efficiency of the pipeline also allows us to create interactive vectorisation strategies, where the user takes over the task of feature extraction by denoting features by hand. The rest of the pipeline than proceeds and the user can add or adjust drawn features.

We believe that through our contributions and efforts we have advanced the field of arbitrary topology shape modelling and representation, with applications in geometric modelling and vector graphics. The arbitrary topology setting provides unique challenges to solve. The nice and regular structures break down and with that the conditions of multiple regions have to be juggled around in order to satisfy continuity conditions. Our solutions provide means to create smooth surfaces or colour gradients in such settings. In addition we have presented several ways in which the mesh structure, be it regular or arbitrary, can be used not only to present the object itself but to evaluate functions on its surface. These functions provide additional creative control for the designer or can be used to effectively represent higher frequency data. Naturally, further improvements can be made and future work remains, as discussed in the next section.

10.1 FUTURE WORK

We will now detail several recommendations for future work. The work set out in this thesis provides a lot of interesting directions to investigate further. We will give at least one recommendation per chapter, but more pointers could be found in their respective discussion sections.

The conversion of trimmed NURBS patches into triangular elements provides the ability to create smooth connections between different patches (Chapter 4). However, it makes little sense to use triangular elements in the middle of the patches where the surface can be exactly represented using quadrilateral elements. The triangular elements could then be used only at these boundaries and connect smoothly C^1 to the internal patches quadrilateral elements and with G^1 continuity to other patches. Rational methods, such as Gregory patches, could also be explored, removing the need for triangular elements at all. However, the rational nature of these patches will complicate the analysis abilities of this conversion.

The multisided B-splines covered in Chapter 5 are an interesting technique and provide a means to bridge the gaps in otherwise regular B-spline surfaces. However, the shape defects of the surfaces as presented here have not been entirely fixed, merely diminished. The solution proposed by Vaitkus et al. [VVSS21] may provide better shape and weight distribution. Their approach uses the B-spline basis along edges and Bernstein basis functions from the edge towards the inside of the patches. In practice their patches could be used in the same settings of ours by computing the Bézier control points from the input polyhedral

control structure. However, it would be better to somehow preserve the simple structure of the original B-spline setting. It remains to be seen whether such a thing is possible to do.

The combination of procedural noise functions with gradient meshes creates a unique sparse representation of texture (Chapter 7). Gabor noise has been used before to infer texture from an input image. The parameters for Gabor noise could thus be inferred in a completely local way and used in combination with gradient meshes to automatically vectorise images. Textured portions of images could then be sparsely represented by a gradient mesh that interpolates the determined Gabor noise parameters.

The polygonal gradient mesh provides a much more organic way of modelling meshes to the user than the completely regular gradient mesh (Chapter 8). It remains to be seen whether this leads to better ways to vectorise an image. The original regular gradient mesh has been used frequently for image vectorisation purposes. The increased freedom in topology and valency of faces would seem to be able to create a sparser representations than would a strict regular topology. Methods that simplify original gradient meshes could work but will only provide a small decrease in mesh density. Smart methods that build a mesh with the topology freedom in mind could be a better solution. However, we suspect that arbitrary polygonal faces in the polygonal gradient do not provide a clear advantage for vectorisation over using strictly quadrilateral meshes with extraordinary vertices.

The quality of the image vectorisation is largely dependent on the features that have been extracted from the input image. This is an area that can always be improved and will lead to improved vectorisations. The interactive vectorisation framework we have created uses human intuition on where to put hard edges and soft features and with this is able to create a much sparser representation than is currently possible with automatic feature extraction. In the future machine learning approaches could be used to better extract semantic meaning from images leading to improved edge and object detection. This would lead to a sparser set of features and in turn a sparser mesh and vector representation.

BIBLIOGRAPHY

[Ado06] Adobe. Adobe PDF. https://www.adobe.com/content/dam/acom/en/devnet/pdf/pdf_reference_archive/pdf_reference_1-7.pdf, 2006.

[Ado19a] Adobe. Adobe Illustrator: Image Trace. https://helpx.adobe.com/illustrator/using/image-trace.html, 2019. Online; accessed 13 December 2020.

[Ado19b] Adobe. Adobe Illustrator: Meshes. https://helpx.adobe.com/illustrator/using/meshes.html, 2019. Online; accessed 13 December 2020.

[AW90] Gregory D. Abram and Turner Whitted. Building block shaders. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 283–288, New York, NY, USA, 1990. ACM.

[BA08] Tamy Boubekeur and Marc Alexa. Phong tessellation. *ACM Transactions on Graphics*, 27(5):141, 2008.

[BDF14] Mark D. Benjamin, Stephen DiVerdi, and Adam Finkelstein. Painting with triangles. In *NPAR 2014, Proceedings of the 12th International Symposium on Non-photorealistic Animation and Rendering*, August 2014.

[BDL09] Oleksiy Busaryev, Tamal K. Dey, and Joshua A. Levine. Repairing and meshing imperfect shapes with Delaunay refinement. In *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling*, SPM '09, pages 25–33, New York, NY, USA, 2009. ACM.

[BLHK18] Pieter J. Barendrecht, Martijn Luinstra, Jonathan Hogervorst, and Jiří Kosinka. Locally refinable gradient meshes supporting branching and sharp colour transitions. *The Visual Computer*, 34(6-8):949–960, 2018.

[BLW11] John C. Bowers, Jonathan Leahey, and Rui Wang. A ray tracing approach to diffusion curves. *Computer Graphics Forum*, 30(4):1345–1352, 2011.

[BS93] Wilhelm Barth and Wolfgang Stürzlinger. Efficient ray tracing for Bézier and B-spline surfaces. *Computers & Graphics*, 17(4):423–430, 1993.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[Can11] Iain Cantlay. DirectX 11 terrain tessellation. *Nvidia whitepaper*, 8(11), 2011.

[CC78] Edwin Catmull and James Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355, 1978.

[CG84] Peter Charrot and John A. Gregory. A pentagonal surface patch for computer aided geometric design. *Computer Aided Geometric Design*, 1(1):87–94, 1984.

[CHB09] John A. Cottrell, Thomas J.R. Hughes, and Yuri Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley & Sons, 2009.

[Chi86] Hiroaki Chiyokura. Localized surface interpolation method for irregular meshes. In *Advanced Computer Graphics*, pages 3–19. Springer, 1986.

[CK83] Hiroaki Chiyokura and Fumihiko Kimura. Design of solids with free-form surfaces. *ACM SIGGRAPH Computer Graphics*, 17(3):289–298, 1983.

[CLL+20] Kuo-Wei Chen, Ying-Sheng Luo, Yu-Chi Lai, Yan-Lin Chen, Chih-Yuan Yao, Hung-Kuo Chu, and Tong-Yee Lee. Image vectorization with real-time thin-plate spline. *IEEE Transactions on Multimedia*, 22(1):15–29, 2020.

[Cor20] Microsoft Corporation. Opentype specification 1.8.4. https://docs.microsoft.com/en-us/typography/opentype/spec/, 2020.

[Cox72] Maurice G. Cox. The Numerical Evaluation of B-Splines*. *IMA Journal of Applied Mathematics*, 10(2):134–149, October 1972.

[CT65] Ray W. Clough and James L. Tocher. Finite element stiffness matrices for analysis of plates in bending. In *Conference on Matrix Methods in Structural Mechanics*, pages 515–545. Wright Patterson Air Force Base, Ohio, 1965.

[DBHR93] Carl De Boor, Klaus Höllig, and Sherman Riemenschneider. *Box splines*, volume 98. Springer Science & Business Media, 1993.

[dC59] Paul de Casteljau. Outillages methodes calcul. Technical report, Citroën, 1959.

[DKT98] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 85–94, 1998.

[DS78] Daniel Doo and Malcolm Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360, 1978.

[EK14] Gershon Elber and Myung-Soo Kim. Modeling by composition. *Computer-Aided Design*, 46:200–204, 2014. 2013 SIAM Conference on Geometric and Physical Modeling.

[EMP+03] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003.

[Far82] Gerald Farin. A construction for visual C^1 continuity of polynomial surface patches. *Computer Graphics and Image Processing*, 20(3):272–282, 1982.

[Fer64] James Ferguson. Multivariable curve interpolation. *Journal of the ACM*, 11(2):221–228, 1964.

[FLB17] Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. Photo2clipart: Image abstraction and vectorization using layered linear gradients. *ACM Transactions on Graphics*, 36(6), November 2017.

[Flo03] Michael S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

[FS08] Michael S. Floater and Christian Schulz. Pointwise radial minimization: Hermite interpolation on arbitrary domains. *Computer Graphics Forum*, 27(5):1505–1512, 2008.

[GLLD12] Bruno Galerne, Ares Lagae, Sylvain Lefebvre, and George Drettakis. Gabor noise by example. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012)*, 31(4):73:1–73:9, July 2012.

[GLZ90] John A Gregory, Vincent KH Lau, and Jianwei Zhou. Smooth parametric surfaces and n-sided patches. In *Computation of curves and Surfaces*, pages 457–498. Springer, 1990.

[GP15] David Groisser and Jorg Peters. Matched G^k-constructions always yield C^k-continuous isogeometric elements. *Computer Aided Geometric Design*, 34:67–72, 2015.

[Gre74] John A. Gregory. Smooth interpolation without twist constraints. *Computer Aided Geometric Design*, pages 71–87, 1974.

[HF06] Kai Hormann and Michael S Floater. Mean value coordinates for arbitrary planar polygons. *ACM Transactions on Graphics (TOG)*, 25(4):1424–1441, 2006.

[HGA+10] Houssam Hnaidi, Eric Guérin, Samir Akkouche, Adrien Peytavie, and Eric Galin. Feature based terrain generation using diffusion equation. *Computer Graphics Forum*, 29(7):2179–2186, 2010.

[HK17] Gerben J. Hettinga and Jiří Kosinka. Phong Tessellation and PN Polygons for Polygonal Models. In *EG 2017 - Short Papers*. The Eurographics Association, 2017.

[HK18] Gerben J Hettinga and Jiří Kosinka. Multisided generalisations of Gregory patches. *Computer Aided Geometric Design*, 62:166–180, 2018.

[HSG+19] Yixin Hu, Teseo Schneider, Xifeng Gao, Qingnan Zhou, Alec Jacobson, Denis Zorin, and Daniele Panozzo. Triwild: Robust triangulation with curve constraints. *ACM Transactions on Graphics*, 38(4):52:1–52:15, July 2019.

[HZY+20] J. He, S. Zhang, M. Yang, Y. Shan, and T. Huang. Bdcn: Bi-directional cascade network for perceptual edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14, 2020.

[IMH05] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics*, 24(3):1134–1141, July 2005.

[Int19] International TechneGroup Ltd. CADfix version 12.0, 2019. www.iti-global.com/cadfix.

[JCW11] Stefan Jeschke, David Cline, and Peter Wonka. Estimating color and texture parameters for vector graphics. *Computer Graphics Forum*, 30(2):523–532, 2011.

[JMD+07] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. *ACM Transactions on Graphics*, 26(3), July 2007.

[Kah83] Juergen Kahmann. Continuity of curvature between adjacent Bézier patches. *Surfaces in CAGD*, pages 65–75, 1983.

[KBT04] Marcelo Kallmann, Hanspeter Bieri, and Daniel Thalmann. Fully dynamic constrained Delaunay triangulations. In *Geometric Modeling for Scientific Visualization*, pages 241–257. Springer Berlin Heidelberg, 2004.

[KC15] Jiří Kosinka and Thomas J Cashman. Watertight conversion of trimmed CAD surfaces to Clough-Tocher splines. *Computer Aided Geometric Design*, 37:25–41, 2015.

[KP16] Kestutis Karčiauskas and Jorg Peters. Minimal bi-6 G^2 completion of bicubic spline surfaces. *Computer Aided Geometric Design*, 41:10–22, 2016.

[LCH+17] Y. Liu, M. Cheng, X. Hu, K. Wang, and X. Bai. Richer convolutional features for edge detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pages 5872–5881, 2017.

[LD89] Charles T. Loop and Tony D. DeRose. A multisided generalization of Bézier surfaces. *ACM Transactions on Graphics*, 8(3):204–234, 1989.

[LD90] Charles Loop and Tony D. DeRose. Generalized B-spline surfaces of arbitrary topology. *SIGGRAPH Comput. Graph.*, 24(4):347–356, September 1990.

[LHFY12] Zicheng Liao, Hugues Hoppe, David Forsyth, and Yizhou Yu. A subdivision-based representation for vector image editing. *IEEE transactions on visualization and computer graphics*, 18(11):1858–1867, 2012.

[LHM09] Yu-Kun Lai, Shi-Min Hu, and Ralph R. Martin. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Transactions on Graphics*, 28(3):85, 2009.

[Lis82] S. Lisberger. *Tron*. Walt Disney Productions, Lisberger-Kushner Productions, 1982.

[LJD+19] Shufang Lu, Wei Jiang, Xuefeng Ding, Craig S Kaplan, Xiaogang Jin, Fei Gao, and Jiazhou Chen. Depth-aware image vectorization and editing. *The Visual Computer*, pages 1–13, 2019.

[LJH13] Xian-Ying Li, Tao Ju, and Shi-Min Hu. Cubic mean value coordinates. *ACM Transactions on Graphics*, 32(4):126:1–126:10, July 2013.

[LKCOL07] Yaron Lipman, Johannes Kopf, Daniel Cohen-Or, and David Levin. GPU-assisted Positive Mean Value Coordinates for Mesh Deformations. In Alexander Belyaev and Michael Garland, editors, *Geometry Processing*. The Eurographics Association, 2007.

[LKSD17] Henrik Lieng, Jiří Kosinka, Jingjing Shen, and Neil A. Dodgson. A colour interpolation scheme for topologically unrestricted gradient meshes. *Computer Graphics Forum*, 36(6):112–121, 2017.

[LL06] Gregory Lecot and Bruno Levy. Ardeco: automatic region detection and conversion. In *17th Eurographics Symposium on Rendering-EGSR'06*, pages 349–360, 2006.

[LLC+10] Ares Lagae, Sylvain Lefebvre, Rob Cook, Tony Derose, George Drettakis, David S. Ebert, John P. Lewis, Ken Perlin, and Matthias Zwicker. State of the art in procedural noise functions. In *EG 2010-State of the Art Reports*. The Eurographics Association, 2010.

[LLDD09] Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. Procedural noise using sparse Gabor convolution. *ACM Transactions on Graphics*, 28(3):54, 2009.

[LLGRK20] Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics*, 39(6), November 2020.

[Lon85] Lucia L. Longhi. Interpolating patches between cubic boundaries. Technical Report UCB/CSD-87-313, EECS Department, University of California, Berkeley, Dec 1985.

[Loo87] Charles Loop. Smooth subdivision surfaces based on triangles. *Master's thesis, University of Utah, Department of Mathematics*, 1987.

[LR80] Jeffrey M. Lane and Richard F. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1):35–46, January 1980.

[LS08a] Charles Loop and Scott Schaefer. Approximating Catmull-Clark subdivision surfaces with bicubic patches. *ACM Transactions on Graphics*, 27(1), March 2008.

[LS08b] Charles Loop and Scott Schaefer. G^2 tensor product splines over extraordinary vertices. *Computer Graphics Forum*, 27(5):1373–1382, 2008.

[LSNC09] Charles Loop, Scott Schaefer, Tianyun Ni, and Ignacio Castaño. Approximating subdivision surfaces with Gregory patches for hardware tessellation. *ACM Transactions on Graphics*, 28(5):151:1–151:9, 2009.

[LWZ11] Yuen-Shan Leung, Charlie C.L. Wang, and Yunbo Zhang. Localized construction of curved surfaces from polygon meshes: A simple and practical approach on GPU. *Computer-Aided Design*, 43(6):573–585, 2011.

[MA09] Raman Maini and Himanshu Aggarwal. Study and comparison of various image edge detection techniques. *International journal of image processing*, 3(1):1–11, 2009.

[MB07] Bill Loguidice Matt Barton. A history of gaming platforms: The vectrex. *Gamasutra*, 2007.

[MC20] Manish Mandad and Marcel Campen. Bézier guarding: Precise higher-order meshing of curved 2d domains. *ACM Transactions on Graphics*, 39(4), July 2020.

[MDSB03] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, pages 35–57. Springer, 2003.

[MH18] Benjamin Marussig and Thomas J. R. Hughes. A review of trimming in isogeometric analysis: Challenges, data exchange and simulation aspects. *Archives of Computational Methods in Engineering*, 25(4):1059–1127, Nov 2018.

[MNP08] Ashish Myles, Tianyun Ni, and Jorg Peters. Fast Parallel Construction of Smooth Surfaces from Meshes with Tri/Quad/Pent Facets. *Computer Graphics Forum*, 27(5):1365–1372, 2008.

[MSY19] Ian Mallett, Larry Seiler, and Cem Yuksel. Patch textures: Hardware implementation of mesh colors. In *High-Performance Graphics (HPG 2019)*. The Eurographics Association, 2019.

[MSY20] Ian Mallett, Larry Seiler, and Cem Yuksel. Patch textures: Hardware support for mesh colors. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[MvSE18] Fady Massarwi, Boris van Sosin, and Gershon Elber. Untrimming: Precise conversion of trimmed-surfaces to tensor-product surfaces. *Computers & Graphics*, 70:80–91, 2018.

[NG00] J. Cotrina Navau and N. Pla Garcia. Modeling surfaces from meshes of arbitrary topology. *Computer Aided Geometric Design*, 17(7):643–671, 2000.

[NL13] Matthias Nießner and Charles Loop. Analytic displacement mapping using hardware tessellation. *ACM Transactions on Graphics*, 32(3), July 2013.

[OBW+08] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion curves: A vector representation for smooth-shaded images. *ACM Transactions on Graphics*, 27(3):92–1, 2008.

[Per85] Ken Perlin. An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296, July 1985.

[Pet19] Jorg Peters. Splines for meshes with irregularities. *The SMAI journal of computational mathematics*, S5:161–183, 2019.

[PT95] Lesley Piegl and Wayne Tiller. *The NURBS book*. Springer-Verlag Berlin, 1995.

[RLMB+14] Christian Richardt, Jorge Lopez-Moreno, Adrien Bousseau, Maneesh Agrawala, and George Drettakis. Vectorising bitmaps into semi-transparent gradient layers. *Computer Graphics Forum*, 33(4):11–19, 2014.

[RS19] Ulrich Reif and Malcolm A. Sabin. Old problems and new challenges in subdivision. *Journal of Computational and Applied Mathematics*, 349:523–531, 2019.

[Rub98] Owen R. Rubin. Memories of a vector world. *SIGGRAPH Comput. Graph.*, 32(2):44–46, May 1998.

[SBAD16] Jingjing Shen, Laurent Busé, Pierre Alliez, and Neil Dodgson. A line/trimmed NURBS surface intersection algorithm using matrix representations. *Computer Aided Geometric Design*, 48:1–16, 2016.

[Sch90] Philip J. Schneider. *An Algorithm for Automatically Fitting Digitized Curves*, page 612–626. Academic Press Professional, Inc., USA, 1990.

[SFL+08] Thomas W. Sederberg, G. Thomas Finnigan, Xin Li, Hongwei Lin, and Heather Ipson. Watertight trimmed NURBS. *ACM Trans. Graph.*, 27(3):79:1–79:8, 2008.

[SFMH11] Ruben Sevilla, Sonia Fernández-Méndez, and Antonio Huerta. NURBS-enhanced finite element method (NEFEM). *Archives of Computational Methods in Engineering*, 18(4):441–484, Oct 2011.

[SK16] Jingjing Shen and Jiří Kosinka. Conversion of CAD Models to Loop Subdivision Surfaces. In Luis Gonzaga Magalhaes and Rafal Mantiuk, editors, *EG 2016 - Posters*. The Eurographics Association, 2016.

[SKSD14] Jingjing Shen, Jiří Kosinka, Malcolm A. Sabin, and Neil A. Dodgson. Conversion of trimmed NURBS surfaces to Catmull-Clark subdivision surfaces. *Computer Aided Geometric Design*, 31(7):486–498, 2014.

[SKSD16] Jingjing Shen, Jiří Kosinka, Malcolm Sabin, and Neil Dodgson. Converting a CAD model into a non-uniform subdivision surface. *Computer Aided Geometric Design*, 48:17–35, 2016.

[SKU08] László Szirmay-Kalos and Tamás Umenhoffer. Displacement mapping on the GPU—state of the art. *Computer Graphics Forum*, 27(6):1567–1592, 2008.

[SLWS07] Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum. Image vectorization using optimized gradient meshes. *ACM Transactions on Graphics*, 26(3):11, 2007.

[Smi95] Alvy Ray Smith. A pixel is not a little square, a pixel is not a little square, a pixel is not a little square!, 1995.

[SNK+14] Henry Schäfer, Matthias Nießner, Benjamin Keinert, Mark Stamminger, and Charles Loop. State of the Art Report on Real-time Rendering with Hardware Tessellation. In Sylvain Lefebvre and Michela Spagnuolo, editors, *Eurographics 2014 - State of the Art Reports*. The Eurographics Association, 2014.

[Spe11] Scott Spencer. *ZBrush character creation: advanced digital sculpting*. John Wiley & Sons, 2011.

[SS87] Leon A Shirman and Carlo H Séquin. Local surface interpolation with Bézier patches. *Computer Aided Geometric Design*, 4(4):279–295, 1987.

[SS90] Leon A Shirman and Carlo H Séquin. Local surface interpolation with shape parameters between adjoining Gregory patches. *Computer Aided Geometric Design*, 7(5):375–388, 1990.

[SS91] Leon A. Shirman and Carlo H. Séquin. Local surface interpolation with Bézier patches: Errata and improvements. *Computer Aided Geometric Design*, 8(3):217–221, 1991.

[Sta98] Jos Stam. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 395–404, 1998.

[Sta01] Jos Stam. On subdivision schemes generalizing uniform B-spline surfaces of arbitrary degree. *Computer Aided Geometric Design*, 18(5):383–396, 2001.

[Sut64] Ivan E Sutherland. Sketchpad a man-machine graphical communication system. *Simulation*, 2(5):R–3, 1964.

[SV18] Péter Salvi and Tamás Várady. Multi-sided Bézier surfaces over concave polygonal domains. *Computers & Graphics*, 74:56–65, 2018.

[TOC98] Joseph R. Tristano, Steven J. Owen, and Scott A. Canann. Advancing front surface mesh generation in parametric space using a Riemannian surface definition. In *Proc. 7th Int. Meshing Roundtable*, pages 429–445, 1998.

[UMC+19] Benjamin Urick, Benjamin Marussig, Elaine Cohen, Richard H. Crawford, Thomas J.R. Hughes, and Richard F. Riesenfeld. Watertight Boolean operations: A framework for creating CAD-compatible gap-free editable solid models. *Computer-Aided Design*, 115:147–160, 2019.

[Vai21] Márton Vaitkus. *Generating smooth surfaces from discrete pointsets*. PhD thesis, Budapest University of Technology and Economics, 2021.

[VK18] Teun W. Verstraaten and Jiří Kosinka. Local and hierarchical refinement for subdivision gradient meshes. *Computer Graphics Forum*, 37(7):373–383, 2018.

[VPBM01] Alex Vlachos, Jorg Peters, Chas Boyd, and Jason L Mitchell. Curved PN triangles. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 159–166. ACM, 2001.

[VRS11] Tamás Várady, Alyn Rockwood, and Péter Salvi. Transfinite surface interpolation over irregular n-sided domains. *Computer-Aided Design*, 43(11):1330–1340, 2011. Solid and Physical Modeling 2011.

[VSK16] Tamás Várady, Péter Salvi, and György Karikó. A multi-sided Bézier patch with a simple control structure. In *Computer Graphics Forum*, volume 35, pages 307–317. Wiley Online Library, 2016.

[VSK17] Tamás Várady, Péter Salvi, and István Kovács. Enhancement of a multi-sided Bézier surface representation. *Computer Aided Geometric Design*, 55:69–83, 2017.

[VVSS21] Márton Vaitkus, Tamás Várady, Péter Salvi, and Ágoston Sipos. Multi-sided b-spline surfaces over curved, multi-connected domains. *Computer Aided Geometric Design*, 89:102019, 2021.

[VW91] Jarke J. Van Wijk. Spot noise texture synthesis for data visualization. *ACM Siggraph CG*, 25(4):309–318, 1991.

[Wac75] Eugene L. Wachspress. *A Rational Finite Element Basis*. Academic Press, 1975.

[Wor96] Steven Worley. A cellular texture basis function. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 291–294. ACM, 1996.

[WZG+19] Guangshun Wei, Yuanfeng Zhou, Xifeng Gao, Qian Ma, Shiqing Xin, and Ying He. Field-aligned quadrangulation for image vectorization. *Computer Graphics Forum*, 38(7):171–180, 2019.

[XLY09] Tian Xia, Binbin Liao, and Yizhou Yu. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Transactions on Graphics*, 28(5):115, 2009.

[XQ17] Songtao Xia and Xiaoping Qian. Isogeometric analysis with Bézier tetrahedra. *Computer Methods in Applied Mechanics and Engineering*, 316:782–816, 2017.

[XSTN14] Guofu Xie, Xin Sun, Xin Tong, and Derek Nowrouzezahrai. Hierarchical diffusion curves for accurate automatic image vectorization. *ACM Transactions on Graphics*, 33(6):230, 2014.

[XT15] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[XWLS17] Jun Xie, Holger Winnemöller, Wilmot Li, and Stephen Schiller. Interactive vectorization. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 6695—-6705, New York, NY, USA, 2017. Association for Computing Machinery.

[YCZ+16] Ming Yang, Hongyang Chao, Chi Zhang, Jun Guo, Lu Yuan, and Jian Sun. Effective clipart image vectorization through direct optimization of bezigons. *IEEE Transactions on Visualization & Computer Graphics*, 22(02):1063–1075, feb 2016.

[YKH10] Cem Yuksel, John Keyser, and Donald H. House. Mesh colors. *ACM Transactions on Graphics*, 29(2), April 2010.

[YLT19] Cem Yuksel, Sylvain Lefebvre, and Marco Tarini. Rethinking texture mapping. *Computer Graphics Forum (Proceedings of Eurographics 2019)*, 38(2):535–551, 2019.

[Yuk16] Cem Yuksel. Mesh colors with hardware texture filtering. In *ACM SIGGRAPH 2016 Talks*, SIGGRAPH '16, New York, NY, USA, 2016. Association for Computing Machinery.

[YZ04] Lexing Ying and Denis Zorin. A simple manifold-based construction of surfaces of arbitrary smoothness. *ACM Transactions on Graphics*, 23(3):271–275, August 2004.

[ZB97] Jinjin Zheng and Alan A Ball. Control point surfaces over non-four-sided areas. *Computer Aided Geometric Design*, 14(9):807–821, 1997.

[ZDZ18] Shuang Zhao, Fredo Durand, and Changxi Zheng. Inverse diffusion curves using shape optimization. *IEEE Transactions on Visualization & Computer Graphics*, 24(07):2153–2166, jul 2018.

[ZS01] Dennis Zorin and Peter Schröder. A unified framework for primal/dual quadrilateral subdivision schemes. *Comput. Aided Geom. Des.*, 18:429–454, 2001.

[ZZW14] Hailing Zhou, Jianmin Zheng, and Lei Wei. Representing images using curvilinear feature driven subdivision surfaces. *IEEE Transactions on Image Processing*, 23(8):3268–3280, 2014.

[ZZZS05a] Jin Jin Zheng, Jian J. Zhang, Hong Jun Zhou, and L. G. Shen. C^2 continuous spline surfaces over Catmull-Clark meshes. In *Computational Science and Its Applications – ICCSA 2005*, pages 1003–1012. Springer Berlin Heidelberg, 2005.

[ZZZS05b] Jin Jin Zheng, Jian J. Zhang, Hong Jun Zhou, and L.G. Shen. Smooth spline surface generation over meshes of irregular topology. *The Visual Computer*, 21(8):858–864, Sep 2005.

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisors Jiří Kosinka and Alexandru Telea. Thanks especially to Jiří for offering me this PhD position. We have been more or less working together ever since I entered the advanced computer graphics course and started as a teaching assistant for the basic graphics course. Then afterwards when I continued with the research internship and master's thesis. It seemed to be a natural progression between projects and it was then only natural that I would proceed to do my PhD project with you too. And here were are now. Your feedback has always been valuable and your patience immense.

Thank the Scientific Visualisation and Computer Graphics group and all its current and past members. In particular thanks go out to Venustiano, Chengtao, Lorenzo, Pieter, Andre, Youngjoo, Heejun, Lingyun, Xingyu, Jun, Jieying, Mateus, Caio, Lucas, Hamid, Steffen, Renata, Dennis and Kuanhao. I very much enjoyed my time with you and it was truly a shame that we could for a time meet only online.

Thanks to Adobe research and Jose Echevarria for giving me the opportunity to intern in San Jose. Thanks to all my fellow interns at Adobe. Thanks also to the Jaskiewicz family for being so hospitable and letting me stay at their home during my time in California.

Thanks to all the bachelor and master students which I co-supervised on their short programming project, bachelor's or master's thesis: Sietze, René, Rowan, Jeroen, Michiel, Levi, Sarah, Rick, Hein, Tom and Ankur. Your efforts helped create many of the parts of this thesis and have even led to a few of you being co-authors on the papers we published.

Tank oan mien famylje, Heit & Mem, Marjanne & Björn, Rein & Wypkje, Barbara & Cor-Jaap, Julian, Laurens, Manon en Eline. Dankjewel Gijsbert & Helena, Danique & Marco, en Nathalie.

Ongelooflijk veel dank aan Cynthia voor het doorstaan van deze PhD met mij.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Final Version as of October 22, 2021 (`classicthesis`).