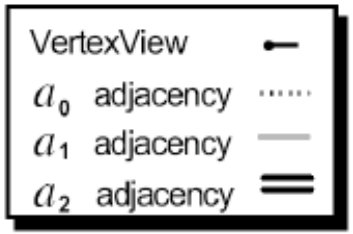
3D Geomodellierung
6. Topologische Repräsentationsmodelle



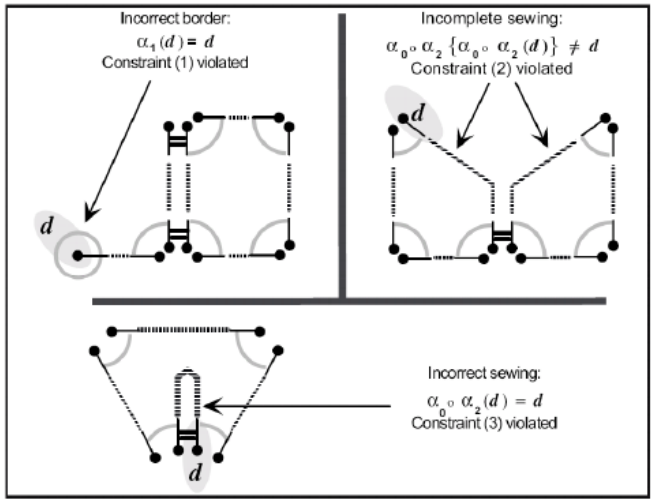


Figure 2.21 *Interpretation of the three constraints on a 2-GMap.*

- Eine *zellulare Zerlegung* ist eine Zerlegung eines (Geo-)Objektes in elementare Zellen „*i*-Zellen“, Vertices, Kanten, Flächen, Volumenelemente …;
- Die *Konnektivität* (connectivity) beschreibt, wie benachbarte Objekte miteinander verbunden sind;
- Die *Einbettung* (embedding) beschreiben die Geometrie und die physikalischen Parameter eines Objektes.

So genannte *Maps* und *Generalized Maps* (GMaps) sind Datenstrukturen, welche es erlauben, die *abstrakte topologische Repräsentation* einer Zerlegung zu realisieren.

Abstrakte topologische Repräsentation …

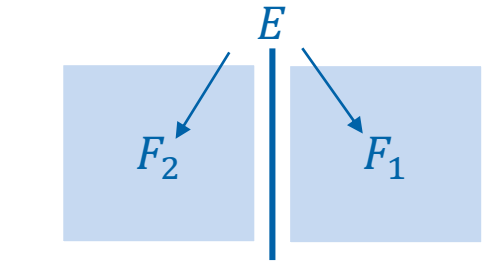
… eines manigfaltigen Objektes A mit einer zellularen Zerlegung $\mathcal{P}(A)$.
Auf Basis eines *Isomorphismus* lässt sich eine abstrakte Repräsentation ohne jegliche Geometrieinformation erstellen, mit

- Abstrakten *i*-Zellen $a^* \in \mathcal{Q}_i(A)$, isomorph zu realen Zellen $a \in \mathcal{P}_i(A)$,
- Der Vereinigungsmenge aller abstrakten *i*-Zellen $\mathcal{Q}(A) = \bigcup_i \mathcal{Q}_i(A)$ und
- Einem Satz von Regeln $\mathcal{I}$, genannt „Inzidenzen“, welche auf $\mathcal{Q}(A)$ operiert.

Das Modell $(\mathcal{Q}(A), \mathcal{I})$ erhält die topologischen Informationen von $\mathcal{P}(A)$.

Definition: „CellView“

Die gemeinsame Kante E zweier Flächen F_1, F_2 lässt sich sowohl über $v_1 = \{E, F_1\}$, als Element der Grenze von F_1 oder über $v_2 = \{E, F_2\}$, als der Grenze von F_2 beschreiben.

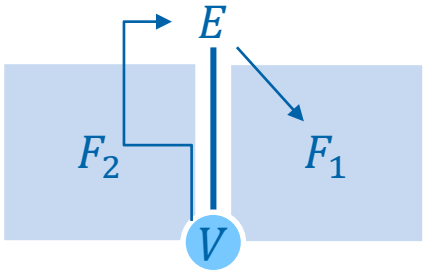


$v_2 = \{E, F_2\}, \quad v_1 = \{E, F_1\},$

Definition: „CellView“

Die gemeinsame Kante E zweier Flächen F_1, F_2 lässt sich sowohl über $v_1 = \{E, F_1\}$, als Element der Grenze von F_1 oder über $v_2 = \{E, F_2\}$, als der Grenze von F_2 beschreiben.

Jeder gemeinsame Vertex V kann analog dazu über $v = \{V, E, F_i\}$, als Element der Grenze von E mit E als Element der Grenze von F_i beschrieben werden.

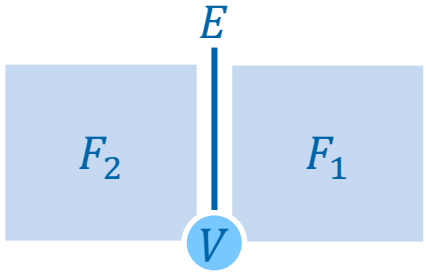


$v = \{V, E, F_1\}$

Definition: „CellView“

Ein so genannter „*i*-CellView“ ist eine Liste von Zellen streng nach aufsteigender topologischer Dimension geordnet, welche eine *i*-Zelle den höher-dimensionalen Zellen zuordnet, zu denen sie „gehört“ oder inzident ist.
Im Allgemeinen besitzt eine *i*-Zelle mehrere CellViews, z. B. für Vertex V:

$$v_1(V) = \{V, E, F_1\}$$
$$v_2(V) = \{V, E, F_2\}$$
$$\dots$$



Definition: „CellView“

Sei $\mathcal{P}(A)$ eine Zerlegungen eines n-manigfaltigen Objektes A und sei v eine nicht-leere Liste von Zellen aus $\mathcal{P}(A)$ der Form $v = \{C_0, C_1, \dots, C_n\}$ mit $C_i \in \mathcal{P}_i(A)$. v wird als CellView von C_0 nach C_n bezeichnet, wenn gilt:

- $C_{i-1} \subset \partial C_i$ und
- $\dim(C_{i-1}) = \dim(C_i) - 1.$

CellView

Der erste Eintrag in der Liste $v = \{C_0, C_1, \dots, C_n\}$ wird als *viewedCell(v)* bezeichnet. Abhängig von der Dimension von *viewedCell(v)* wird v als VertexView (dim = 0), EdgeView (dim=1), FaceView(dim=2) usw. bezeichnet.

CellViews sind grundsätzlich rein topologische Objekte ohne direkten Bezug zur „Geometrie“ des beschriebenen Objektes.

Speziell **VertexViews** spielen eine entscheidende Rolle im topologischen GMap Repräsentationsmodell von Skua-Gocad. $\tilde{\mathcal{P}}(A)$ ist die Liste aller VertexView und jeder VertexView wird als Dart (*Pfeil*) bezeichnet und dargestellt.

VertexViews …

… können als Folgendes angesehen werden:

- Abstrakte Repräsentationen einer zellularen Zerlegung;
- Einzelne Pfade von $\dim(C) = 0$ bis $\dim(C) = n$ im Inzidenz-Graph, der die Zerlegung beschreibt;
- Datentyp
- Eindeutiges Basiselement der Zerlegung genannt „Dart“ 

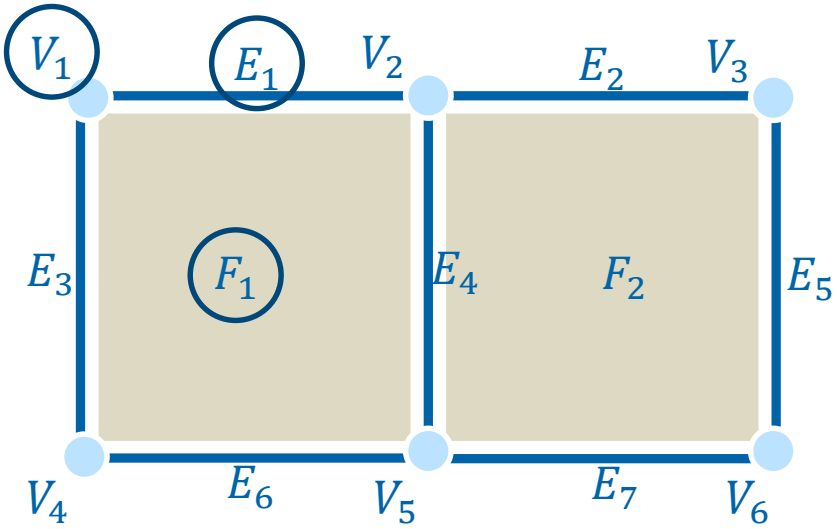
Inzidenz und Adjazenz im Kontex von VertexViews

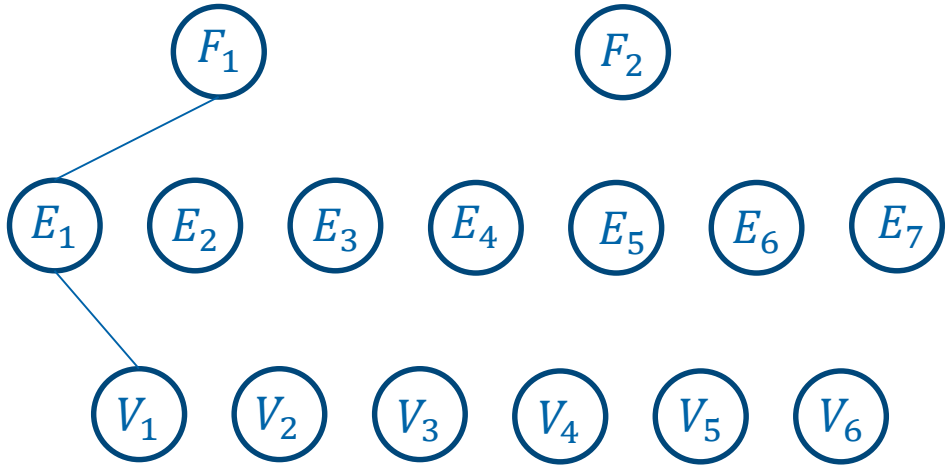
VertexViews lassen sich über Inzidenz- und Adjazenz-Beziehungen beschreiben und vergleichen.

Ein VertexView v und eine Zelle C sind *inzident* zueinander, wenn C ein Element von v ist, d. h. C ist in der Zell-Liste von v enthalten.
Die Menge aller Inzidenz-Beziehungen einer zellularen Zerlegung lässt sich als Graph darstellen. Dieser Graph wird als Inzidenz-Graph bezeichnet.

Jeder VertexView stellt einen Pfad von C_0 nach C_n im Inzidenz-Graph dar.

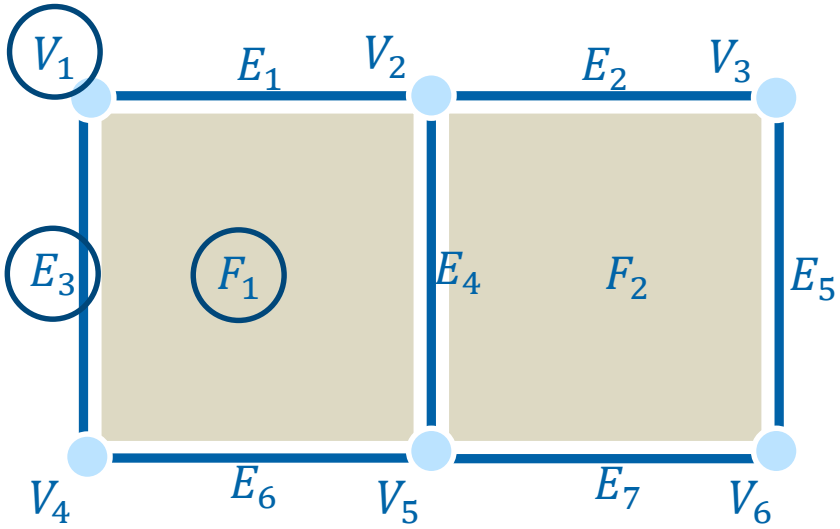
Inzidenz und Adjazenz im Kontex von VertexViews

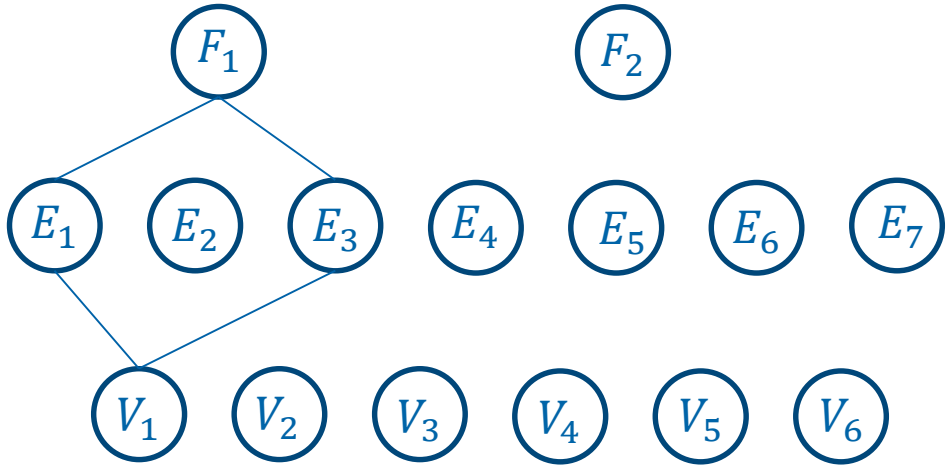




$\{V_1, E_1, F_1\}$

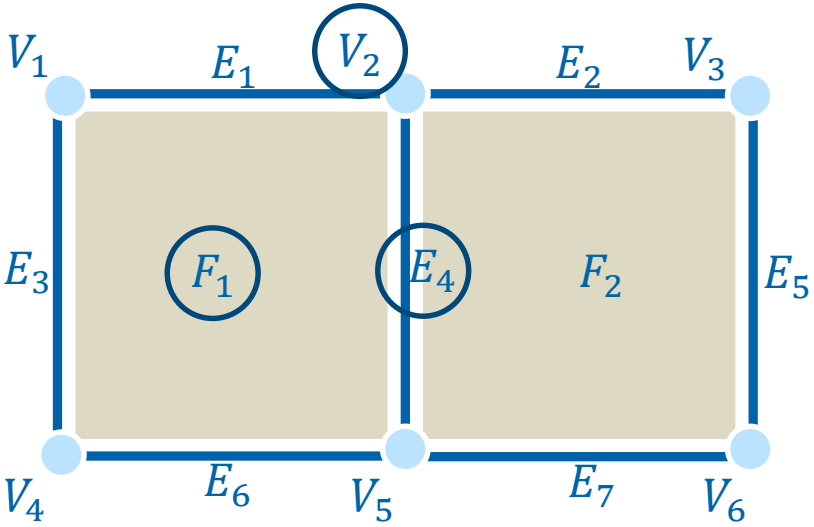
Inzidenz und Adjazenz im Kontex von VertexViews

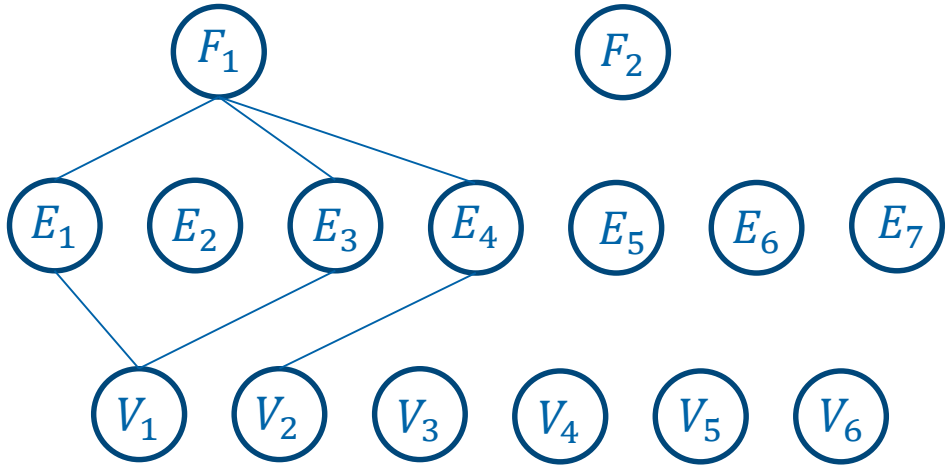




$\{V_1, E_3, F_1\}$

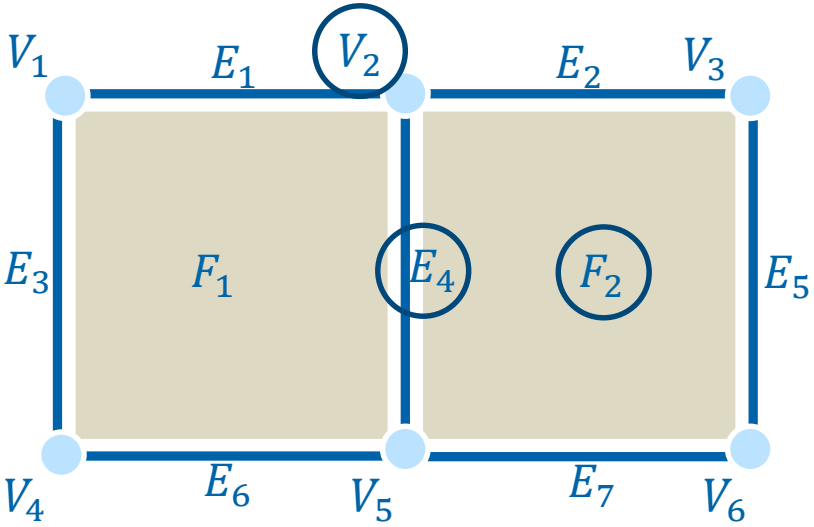
Inzidenz und Adjazenz im Kontex von VertexViews

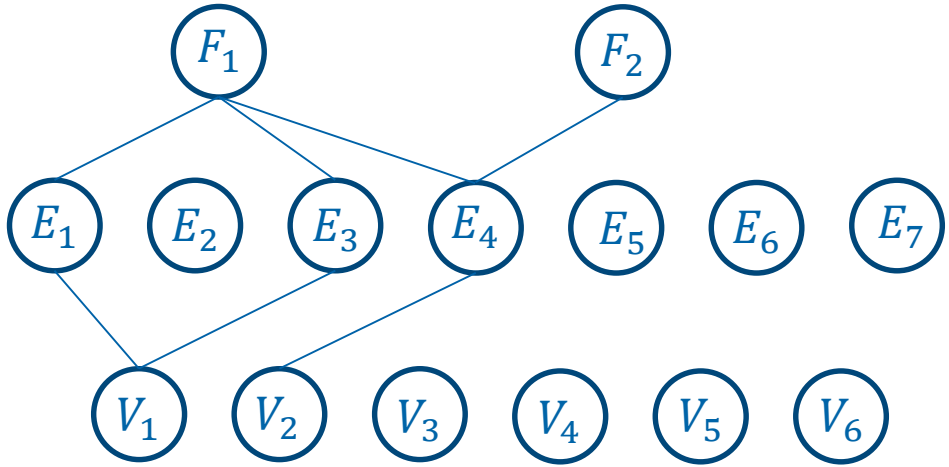




$\{V_2, E_4, F_1\}$

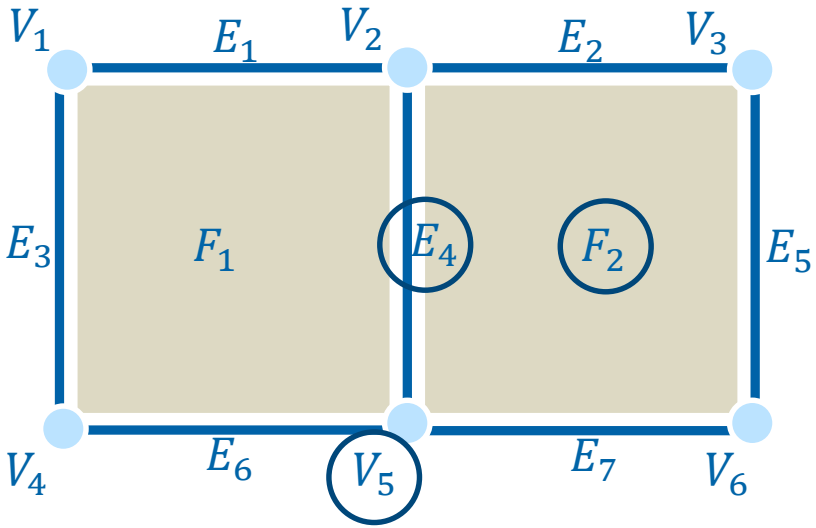
Inzidenz und Adjazenz im Kontex von VertexViews

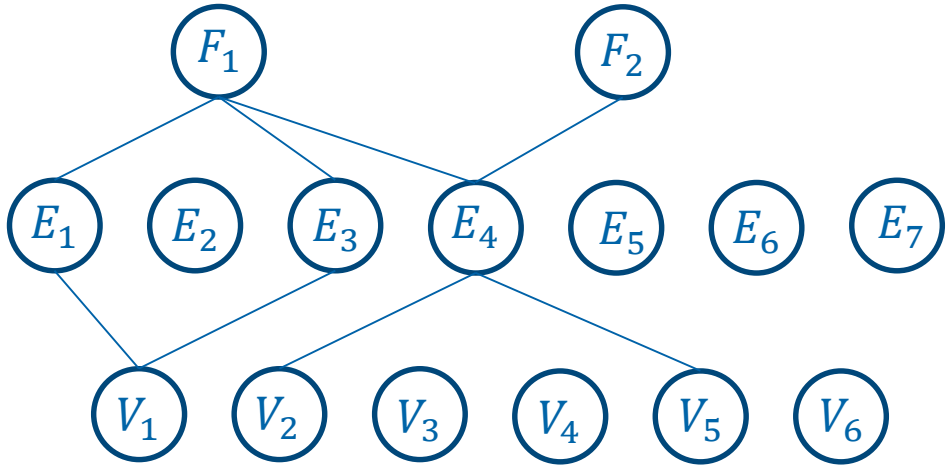




$\{V_2, E_4, F_2\}$

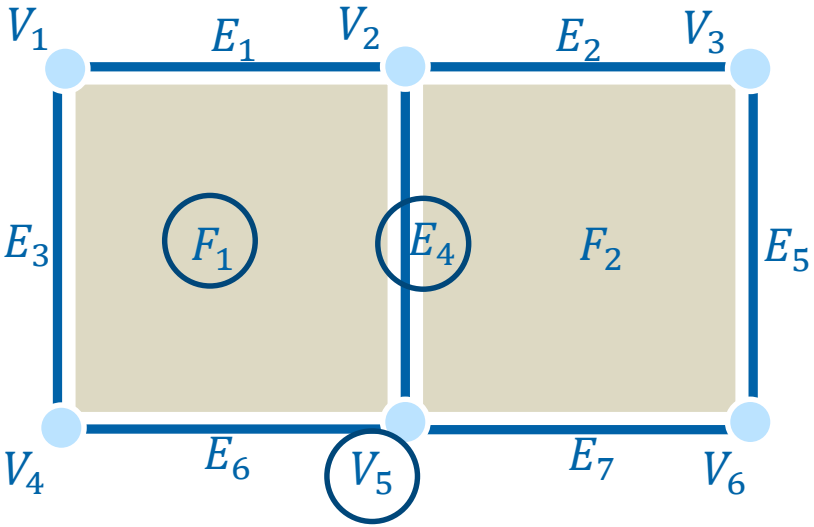
Inzidenz und Adjazenz im Kontex von VertexViews



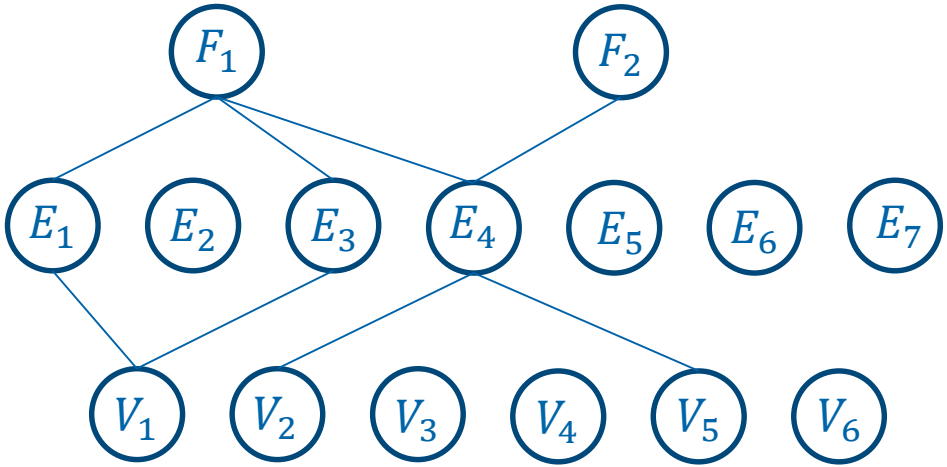


$\{V_5, E_4, F_2\}$

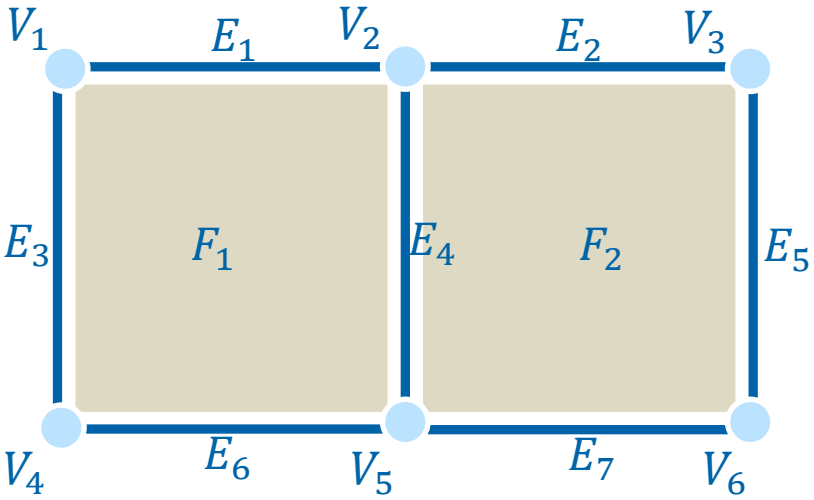
Inzidenz und Adjazenz im Kontex von VertexViews

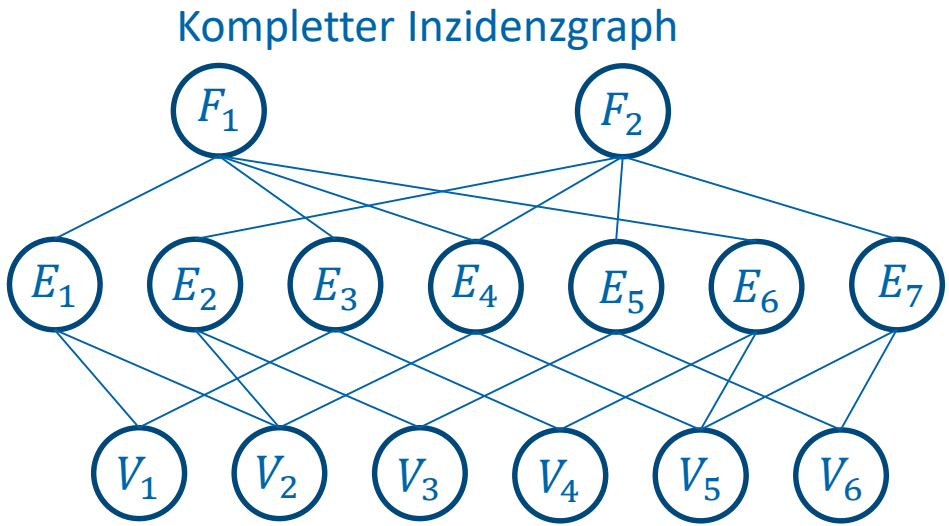


$\{V_5, E_4, F_1\}$



Inzidenz und Adjazenz im Kontex von VertexViews





Inzidenz und Adjazenz im Kontex von VertexViews

Zwei VertexView v und v' werden als *i*-adjazent bezeichnet, wenn ihre Zell-Listen sich nur in der Zelle mit Dimension *i* unterscheiden.

$$v = \{C_0, \dots, C_{i-1}, C_i, C_{i+1}, \dots, C_n\}$$
$$v' = \{C_0, \dots, C_{i-1}, C_i', C_{i+1}, \dots, C_n\}$$

$$C_i \neq C_i'$$

i-Adjazenz

i-Adjazenz beinhaltet die *i*-Äquvalenz-Beziehung

$$\mathcal{A}_i : v_k \mathcal{A}_i v_l$$

wenn die VertexViews v_k und v_l *i*-adjazent sind.

Die Relation $\mathcal{A}_i$ führt zu Äquivalenzklassen. Dabei ist eine *i*-Äquivalenzklasse eine Menge von VertexViews, welche alle *i*-adjazent zueinander sind.

Die *i*-Äquivalenzklasse von $v \in \tilde{\mathcal{P}}(A)$ bestehend aus allen VertexViews, die *i*-adjazent zu v sind, wird als $v^{[i]}$ bezeichnet.

Die Menge aller dieser Klassen wird als $\tilde{\mathcal{P}}^{[i]}(A) = \tilde{\mathcal{P}}(A)/\mathcal{A}_i$ bezeichnet.

i-Adjazenz

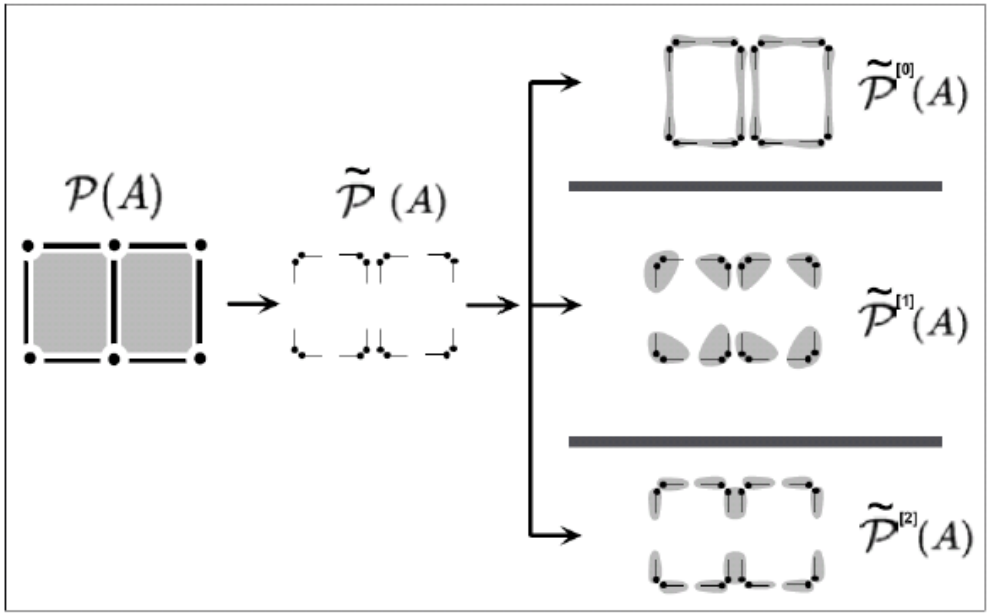


Figure 2.17 *VertexViews (black bullet-headed-segments) of a cellular partition $\mathcal{P}(A)$ and associated classes of equivalences corresponding to i-adjacencies (shaded): it can be seen that, for a given VertexView w, the associated classes $w^{[i]}$ depend strongly on the index i.*

i-Adjazenz

Aus der visuellen Analyse wird klar, dass *i*-Äquivalenz-klassen immer entweder einen oder zwei VertexViews beinhalten:

$$v^{[i]} = \{v, v'\}$$

Oder

$$v^{[i]} = \{v\}$$

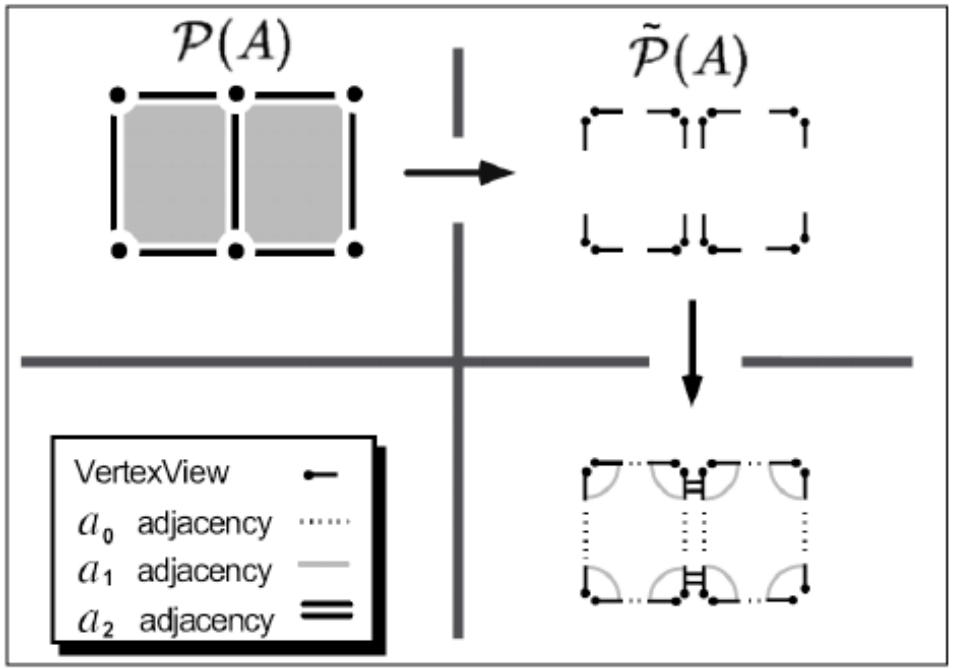


Figure 2.18 *An example of cellular partition $\mathcal{P}(A)$ of a 2-manifold object A and its associated set $\tilde{\mathcal{P}}(A)$ of VertexViews. The adjacency relationships a_i between VertexViews are deduced from the classes of equivalence $\tilde{\mathcal{P}}^{[i]}(A)$.*

i-Adjazenz

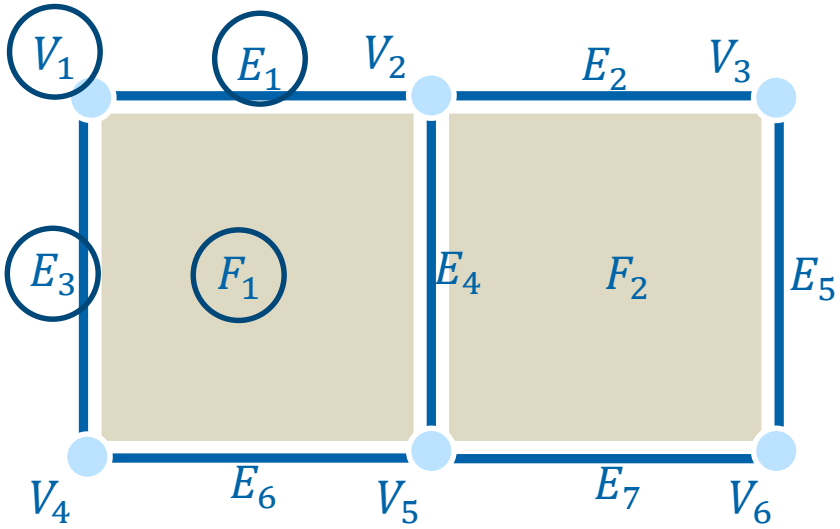
Aus Performance-Gründen wird die Notation der *i*-Adjazenz Beziehung $\mathcal{A}_i$ durch die Notation einer Abbildung (mapping) a_i ersetzt.
Dafür wird eine Abbildung a_i: $\tilde{\mathcal{P}}(A) \mapsto \tilde{\mathcal{P}}(A)$ definiert mit

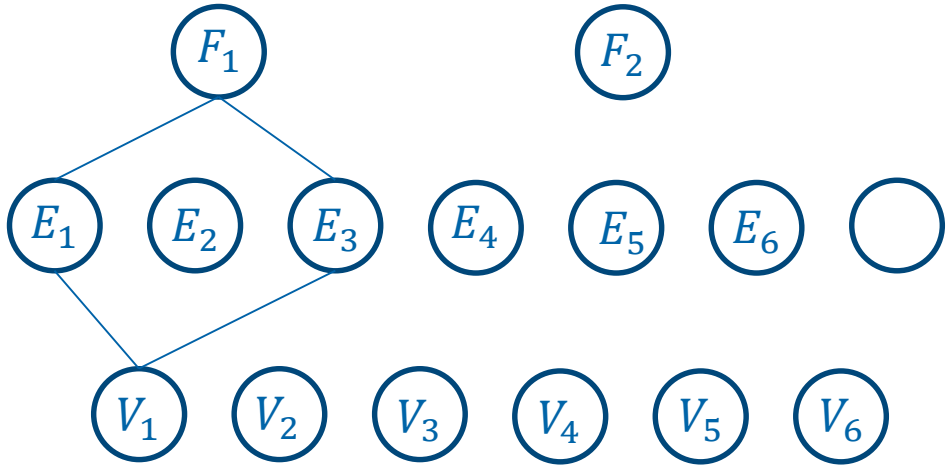
$$a_i(v) = \begin{cases} v \text{ wenn } v^{[i]} \equiv \{v\}^* \\ v' \text{ wenn } v^{[i]} \equiv \{v, v'\} \end{cases}$$

Da gilt $a_i(a_i(v)) = v$, wird a_i als Involution bezeichnet: Die Abbildung ist gleich ihrer Inversen.
a_i verknüpft dabei 2 *i*-adjazente Pfade im Inzidenzgraph, welche sich nur in der *i*-ten Zellen unterscheiden.

* s.g. fixed point (Mallet, 2002)

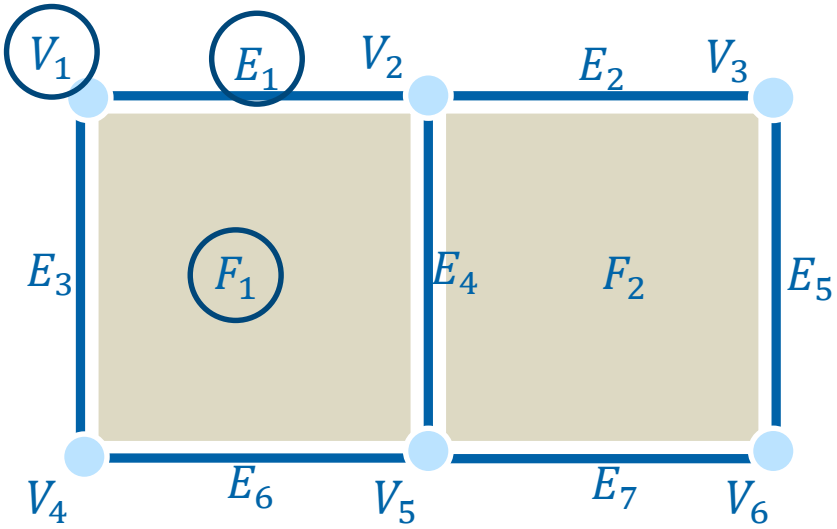
i-Adjazenz

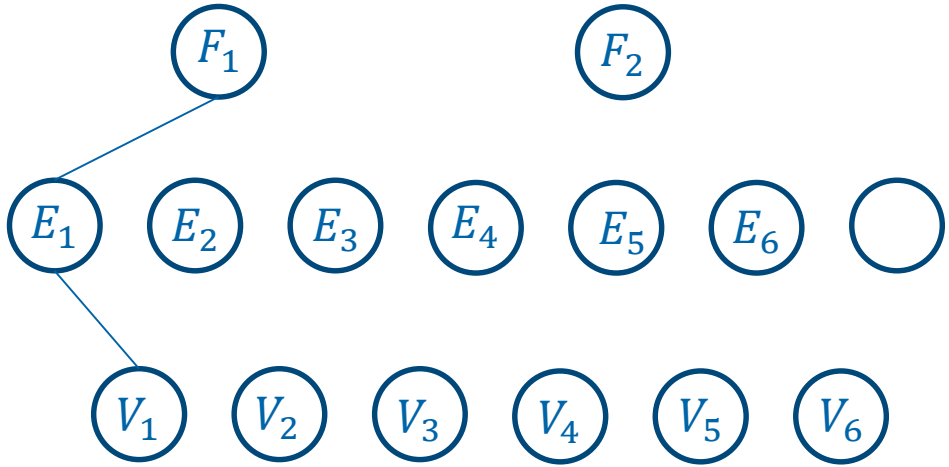




$$v = \{V_1, E_1, F_1\} \qquad a_1(v) = v'$$
$$v' = \{V_1, E_3, F_1\} \qquad a_1(v') = v$$
$$v^{[1]} \equiv \{v, v'\} \qquad a_1(a_1(v)) = v$$

i-Adjazenz – fixed point





$$v = \{V_1, E_1, F_1\} \qquad a_2(v) = v$$
$$v^{[2]} \equiv \{v\} \qquad a_2(a_2(v)) = v$$

Generalized Maps (GMaps)

- In den frühen 1990ern von Pascale Lienhardt vorgestellt
- Zeitgleich wurde von E. Brisson die s.g. „Cell-Tupel-Structure“ beschrieben, welche analog zu GMaps ist

GMaps sind ein präzises algebraisches Framework zur Beschreibung der Topologie einer zellularen Zerlegung über die Menge der zugehörigen VertexViews und der zugehörigen Adjazenz-Beziehungen $\{a_i\}$.

Generalized Maps (GMaps)

Jeder VertexView v wird über einen s.g. Dart d repräsentiert.

Die Adjazenz-Beziehung $\mathcal{A}_i$: $d_k \mathcal{A}_i d_l$ die d_k und d_l verknüpft, wird durch eine Abbildung α_i ersetzt, welche auf der Äquivalenzklasse von d_k und d_l definiert ist mit

$$\alpha_i(d_k) = d_l$$
$$\alpha_i(d_l) = d_k$$

$$\alpha_i(\alpha_i(d_k)) = d_k$$

Abbildungen, für die gilt, dass die zweifache Anwendung der identischen Abbildungen entspricht, werden als Involutionen bezeichnet.

GMaps einer zellularen Zerlegung $\mathcal{P}(A)$

Sei $n \geq 0$ eine gegebene Ganzzahl und sei $\mathcal{G}(D, \alpha_0, \alpha_1, \dots, \alpha_n)$ ein $n+2$ Tupel mit

$$\mathcal{G}(D, \alpha_0, \alpha_1, \dots, \alpha_n) = \begin{cases} D \text{ endliche Menge abstrakter Elemente genannt "Darts"} \\ \{\alpha_i\} \text{ Involutionen auf } D \end{cases}$$

$\mathcal{G}(D, \alpha_0, \alpha_1, \dots, \alpha_n)$ ist eine „Generalized Map“ der Dimension n, oder n-GMap, wenn folgende Bedingungen erfüllt sind …

GMaps einer zellularen Zerlegung $\mathcal{P}(A)$

1. Die Involutionen $\{\alpha_0, \alpha_1, \dots, \alpha_{n-1}\}$ weisen in keinem Fall einen s.g. *fixed point* auf

$$\alpha_i(d) \neq d \; \forall \begin{cases} d \in D \\ i < n \end{cases}$$

2. Die Transformationen $\{\alpha_i \circ \alpha_{i+2+k}\}$ sind immer Involutionen für jedes $i \geq 0$ und jedes $k \geq 0$

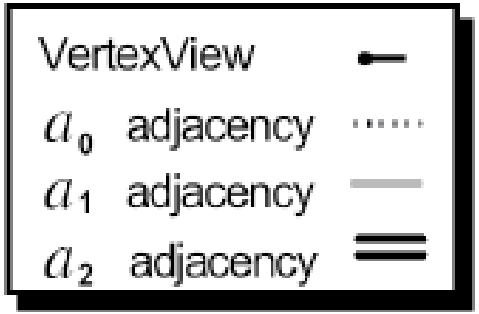
$$\alpha_i \circ \alpha_{i+2+k}\{\alpha_i \circ \alpha_{i+2+k}(d)\} = d \; \forall \begin{cases} d \in D \\ i \geq 0 \\ l \geq 0 \end{cases}$$

GMaps einer zellularen Zerlegung $\mathcal{P}(A)$

3. Die Transformationen $\{\alpha_i \circ \alpha_{i+2+k}\}$ weisen keinen *fixed point* auf für jedes $i \geq 0$ und jedes $k \geq 0$ mit

$$\alpha_i \circ \alpha_{i+2+k}(d) \neq d \; \forall \begin{cases} d \in D \\ i \geq 0 \\ l \geq 0 \end{cases}$$

GMaps einer zellularen Zerlegung $\mathcal{P}(A)$



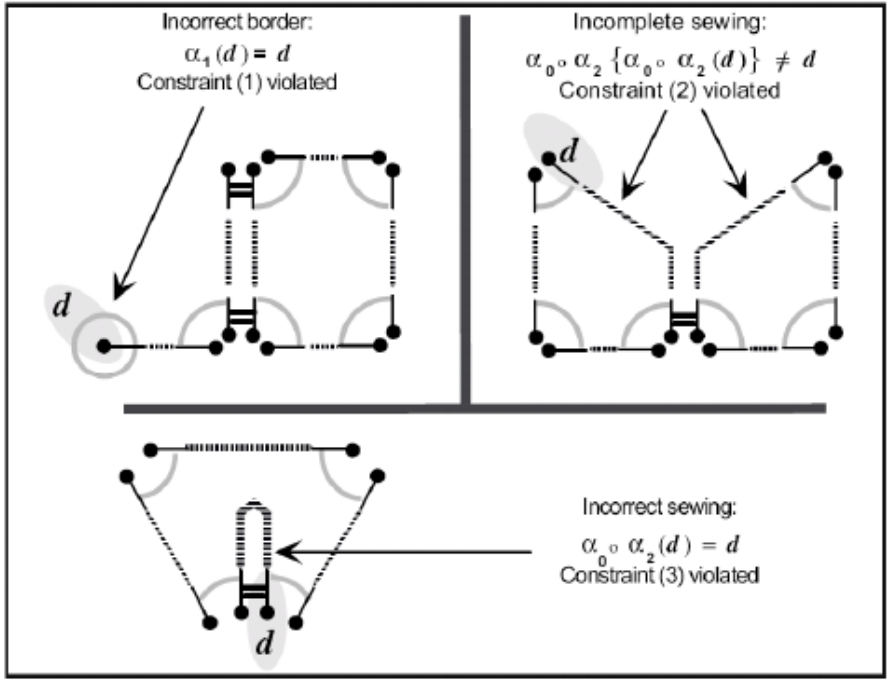


Figure 2.21 *Interpretation of the three constraints on a 2-GMap.*

GMaps einer zellularen Zerlegung $\mathcal{P}(A)$

Die Zerlegung eines manigfaltigen Objektes in Zellen wird über deren Adjazenz-Beziehungen beschrieben, welche wiederum über abstrakte Darts (und damit VertexViews) und Involutionen repräsentiert wird.

Dieser Prozess ist reversibel, d.h. und eine zellulare Zerlegung für ein geometrisches Objekt kann aus der abstrakten Repräsentation durch eine GMap abgeleitet werden.

GMaps sind ein Isomorphismus zwischen geometrischen Zellen und ihrer abstrakten Repräsentation durch Darts.

GMaps einer zellularen Zerlegung $\mathcal{P}(A)$

Daher stellen GMaps einen Isomorphismus zwischen zwei zellularen Zerlegungen

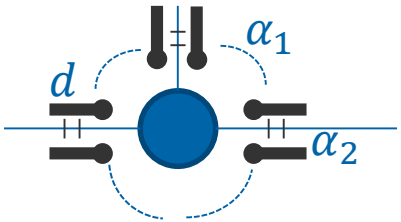
1. Eine initiale Zerlegung mit geometrischer Bedeutung;
2. Eine abstrakte Zerlegung, welche sehr gut geeignet ist, präzise und effizient numerische behandelt zu werden für digitale geometrische Modellierung.

Ein VertexView besteht immer aus einer sortierten Liste von Zellen. Ein Dart repräsentiert eine solche Liste, besteht aber nicht daraus …

GMap-Operationen – Orbits

Alle Darts, welche sich auf eine spezifische *k*-Zelle beziehen, können identifiziert werden, in dem alle Kombinationen der Involutionen α_i mit Ausnahme der Involution α_k angewendet werden;
die Reihenfolge der angewandten Abbildungen wird als *k*-Orbit bezeichnet.

Sei d ein Dart, welcher sich auf einen Vertex (0-Zelle) bezieht, dann ist $< \alpha_1, \alpha_2 > (d) =< \cancel{\alpha_0} > (d)$ die Menge aller Darts, welche sich auf diesen Vertex beziehen: $< \alpha_1, \alpha_2 > (d) = \{\alpha_1(d), \alpha_2(d), \alpha_1\big(\alpha_2(d)\big), \alpha_2\big(\alpha_1(d)\big)\}$



GMap-Operationen – Orbits

Analog dazu ist
$< \alpha_0, \alpha_2 > (d) =< \alpha_1 > (d)$
die Menge aller Darts, welche sich auf die Kante (1-Zelle) beziehen, auf die sich auch d bezieht.

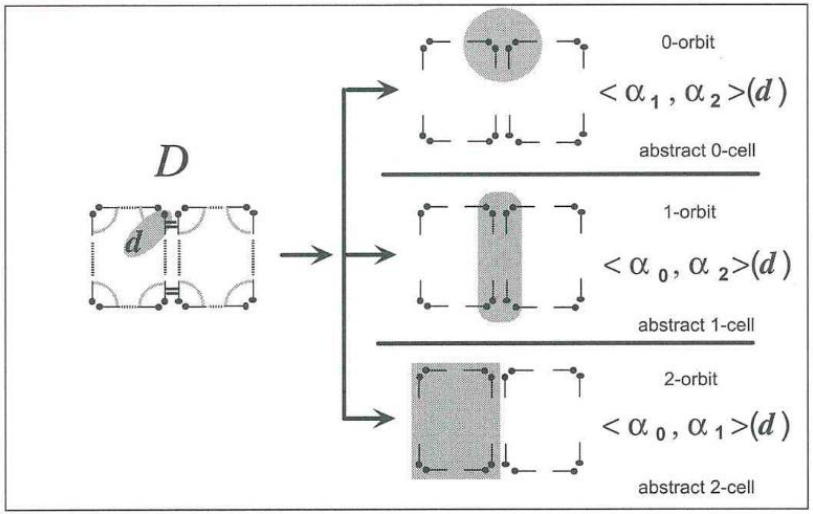


Figure 2.21 *Examples of i-orbits (grey areas) related to a given Dart d in the case of a 2-GMap. From top to bottom of this figure, these i-orbits correspond to an abstract 0-cell, an abstract 1-cell and an abstract 2-cell incidents to the Dart d, respectively.*

GMap-Operationen – Orbits

Orbit-Operation stellen ein generisches Mittel bereit, um auf einer GMap zu operieren.

Über die Orbit-Operationen lassen sich sehr effizient zum Beispiel Nachbarschaften von Zellen identifizieren.
In den VertexViews, die in $< \alpha_1, \alpha_2 > (d)$ enthalten sind, befinden sich ALLE 1- oder 2-Zellen, welche inzident zum zu d gehörigen Vertex sind.

GMap-Operationen – Orbits

Es gibt eine Analogie zwischen einem *i*-Orbit einer GMap und einer *i*-Zelle einer zellularen Zerlegung. Deshalb können *i*-Orbits als abstrakte *i*-Zellen angesehen werden:

$$< \alpha_i > (d) \equiv \text{abstrakte } i\text{−Zelle inzident zu } d$$

Es existiert eine 1-zu-1 Beziehung ϕ zwischen *i*-Orbits und abstrakten *i*-Zellen, welche Darts und VertexViews komponentenweise mittels ϕ_i verknüpft.
ϕ_i assoziiert *i*-Orbits mit *i*-Zellen.

$$\alpha_i(d) = d' \Rightarrow a_i\big(\phi(d)\big) = \phi(d')$$
$$a_i(v) = v' \Rightarrow \alpha_i\big(\phi^{-1}(v)\big) = \phi^{-1}(v')$$
$$d \in D, \phi(d) = \{\phi_0(d), \phi_1(d), \dots, \phi_n(d)\}, v = \{C_0, C_1, \dots, C_n\}$$

Vernähte (sewn) und freie (free) Darts

Sei $G = \mathcal{G}(D, \alpha_0, \alpha_1, \dots, \alpha_n)$ eine gegebene *n*-GMap und $d \in \mathrm{D}$ sein ein gegebener Dart:

- d wird als „vernäht“ (sewn) bezeichnet, wenn gilt: $\alpha_n(d) \neq d$
- d wird als „frei“ (free) bezeichnet, wenn gilt: $\alpha_n(d) = d$

In einer validen GMap können Darts nur in der Involution $\alpha_n(d)$ frei sein.

GMap der Grenze

Sei $G = \mathcal{G}(D, \alpha_0, \alpha_1, \dots, \alpha_n)$ eine gegebene *n*-GMap.
Per Definition ist die *(n-1)*-GMap $\partial G = \mathcal{G}(D', {\alpha'}_0, {\alpha'}_1, \dots, {\alpha'}_{n-1})$ die so genannte „GMap der Grenzen“ von G, wenn für D' und $\{{\alpha'}_i\}$ gilt:

- $D' = \{d \in \mathrm{D}: \alpha_n(d) = d\}$
- Für jede Ganzzahl $i \in [0, n-2]$ ist eine Involution ${\alpha'}_i$ ist die Beschränkung von α_i auf D'
- Für jeden Dart $d \in D'$ ist die Involution ${\alpha'}_{n-1}$ folgendermaßen definiert:

$$ {\alpha'}_{n-1}(\mathrm{d}) = \{\mathrm{D} \cap < \alpha_{n-1}, \alpha_n > (\mathrm{d})\} - \{\mathrm{d}\} $$

GMap der Grenze

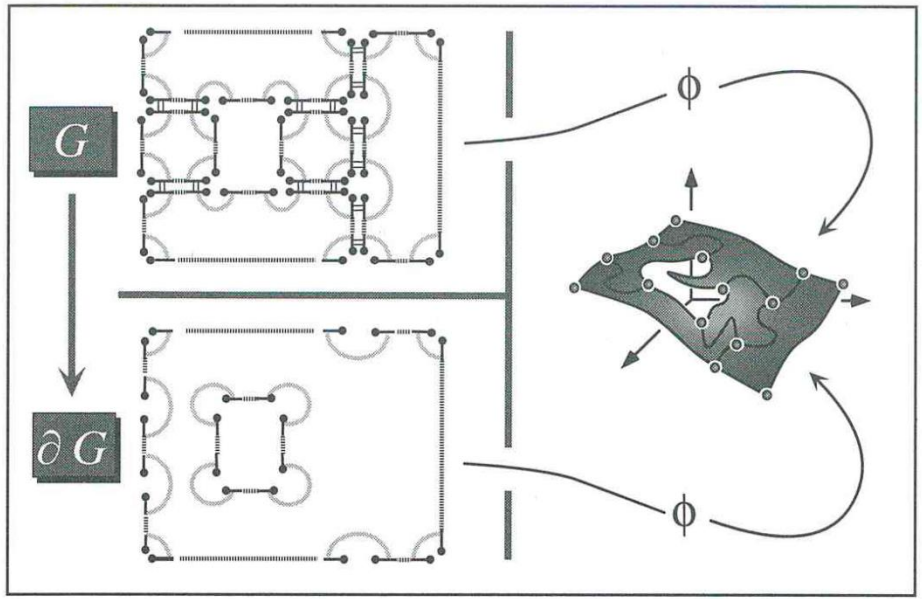


Figure 2.20 *An example of cellular partition $\mathcal{P}(A)$ of a 2-manifold surface A associated with its 2-GMap G and with the 1-GMap ∂G of its boundaries.*

Offene und geschlossene GMaps

Sei $G = \mathcal{G}(D, \alpha_0, \alpha_1, \dots, \alpha_n)$ eine gegebene *n*-GMap.

- G wird als „geschlossen“ bezeichnet, wenn gilt: $\alpha_n(d) \neq d \; \forall d \in D$
- G wird als „offen“ bezeichnet, wenn gilt: $\exists d \in D: \alpha_n(d) = d$

  d. h. eine GMap G ist „offen“, wenn für ihre Grenz-GMap ∂G gilt $D' \neq \emptyset$

  oder einfacher, wenn G eine Grenze aufweist.

Duale GMaps

Sei $G = \mathcal{G}(D, \alpha_0, \alpha_1, \dots, \alpha_n)$ eine gegebene *n*-GMap. Die duale GMap $\mathrm{G} = \mathcal{G}(D^*, \alpha_0^*, \alpha_1^*, \dots, \alpha_n^*)$ ist ebenfalls eine *n*-GMap für die gilt:

- $D^* = D$
- $\alpha_i^* = \alpha_{n-i}$

Die duale GMap lässt sich sehr einfach und effizient ableiten, da nur Involutionen vertauscht werden müssen.

Take-home questions:

1. Bei einer beliebigen Triangulation handelt es sich um eine zellulare Zerlegung. Was sind die zugehörigen *i*-Zellen und welche Art von Einbettung im $\mathbb{R}^2$ ist mindestens erforderlich?
2. Gegeben sei VertexView $v = \{\mathrm{V_a}, \mathrm{E_b}, \mathrm{F_c}, \mathrm{S_d}\}$. Ist v inzident zur Zelle $\mathrm{E_b}$ und 2-adjazent zu VertexView $\mathrm{w} = \{\mathrm{V_a}, \mathrm{E_b}, \mathrm{F_f}, \mathrm{S_g}\}$?

Institut für Geophysik und Geoinformatik
Dr. Peter Menzel
Gustav-Zeuner-Str. 12
09599 Freiberg
Tel. +49(0)3731 39-3815

Take-home questions:

3. Für die Implementierung eines Dart-Objektes soll jedes dieser Objekte eine Liste besitzen, mit der auf mit diesem Dart mittels Involution verknüpfte andere Darts verwiesen werden soll. Jedes Dart-Objekt darf dabei aber nur EINMAL im Speicher vorliegen. Welche Informationen könnte diese Liste enthalten?

Institut für Geophysik und Geoinformatik
Dr. Peter Menzel
Gustav-Zeuner-Str. 12
09599 Freiberg
Tel. +49(0)3731 39-3815

Circular Incident Edge Lists (CIEL)

- CIEL ist eine topologische Datenstruktur, um sehr allgemeine unstrukturierte Gitter abzubilden
- Im Gegensatz zu GMaps auf Anwendung im $\mathbb{R}^2$ und $\mathbb{R}^3$ optimiert
- Ziel ist eine optimiertes Volumenrendering

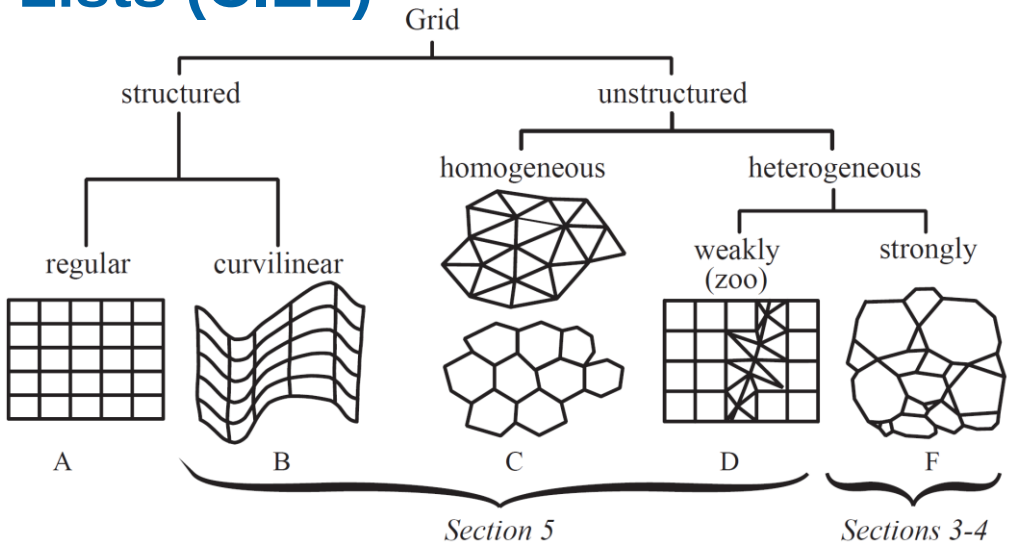
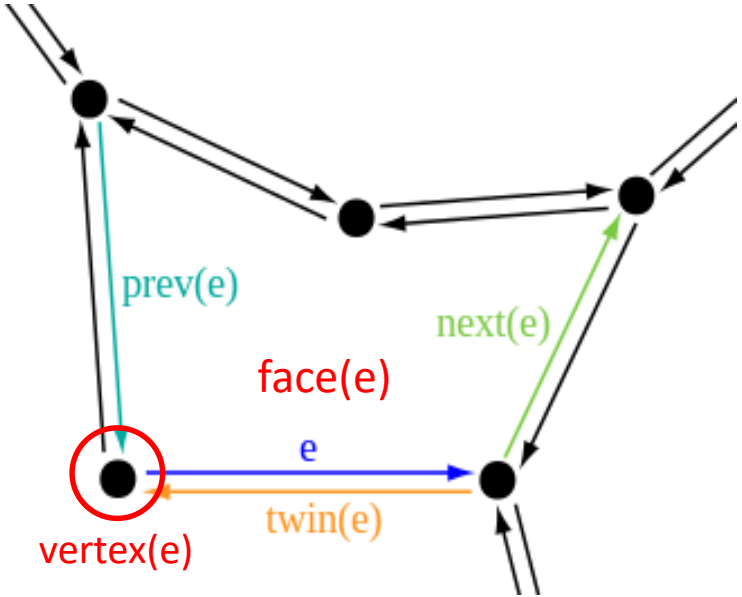


Fig. 1. Grid taxonomy. This article presents a unified method for visualizing all types of grids encountered in Geosciences. We consider the general case of strongly heterogeneous grids, then present optimizations for the more special cases of zoo and curvilinear grids.

Caumon et al. (2005)

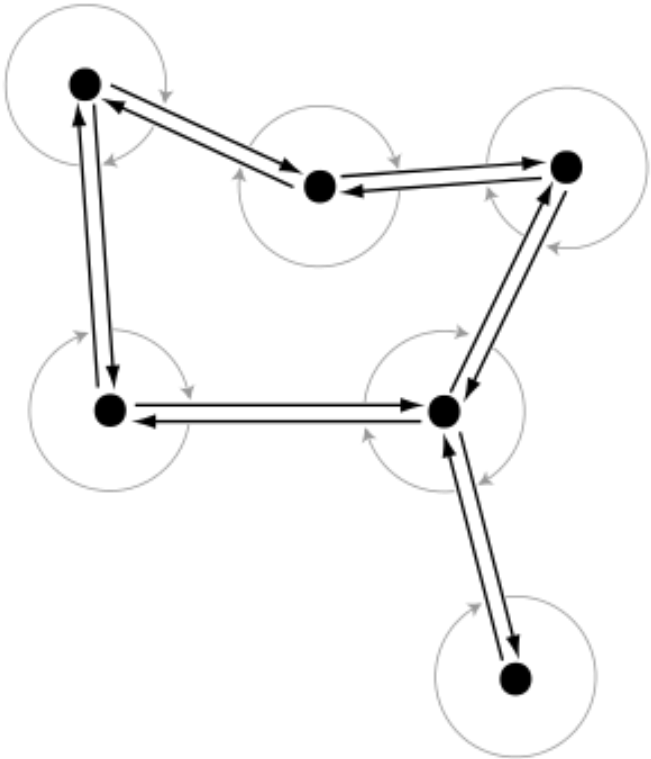
Konzept einer Halbkante (Half-Edge)

- Kombinatorisches Objekt
- Gerichtetes Linienelement einer Fläche; verweist auf:
  1. Gegenläufigen Zwilling der benachbarten Fläche
  2. Nachfolgende Halbkante um die Fläche
  3. Vorhergehende Halbkante um die Fläche
  4. Startvertex und zugeordnete Fläche



https://de.wikipedia.org/wiki/Half-Edge-Datenstruktur

Konzept einer Halbkante (Half-Edge)



https://de.wikipedia.org/wiki/Half-Edge-Datenstruktur

Konzept einer Halbkante (Half-Edge)

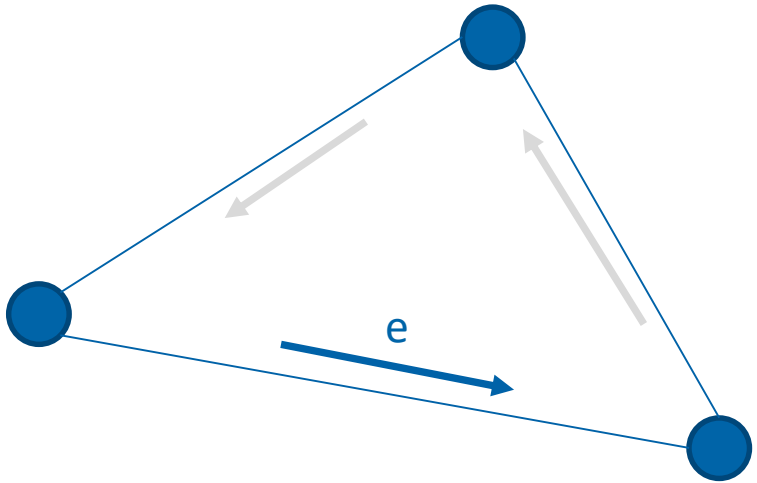
- Jeder Vertex verweist auf EINE beteiligte Halbkante
- Jede Fläche verweist auf
  - Eine Halbkante der äußeren Grenze
  - Eine Halbkante pro innerer Grenze (Loch)

- Optimiert für 2D unstrukturierte Gitter
  Doubly-Connected-Edge-List (DCEL)
- Erweiterungen für 3D möglich, z.B. CIEL
  zusätzlicher Verweis auf
  Gegenläufigen Zwilling des benachbarten Volumens bei gleicher Kante

Operationen mittels Halbkanten

Around-Face
Erlaubt die Bestimmung aller
- Vertices
- Kanten
- (Volumen)

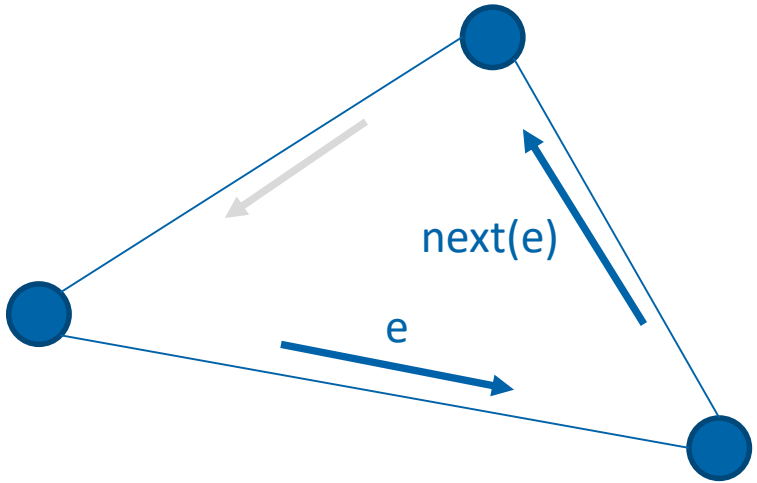
inzident zu einer Fläche



Operationen mittels Halbkanten

Around-Face
Erlaubt die Bestimmung aller
- Vertices
- Kanten
- (Volumen)

inzident zu einer Fläche



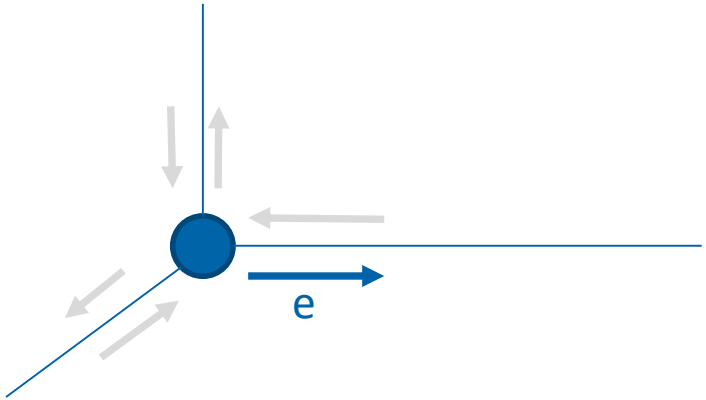
Operationen mittels Halbkanten

Around-Vertex

Erlaubt die Bestimmung aller

- Flächen
- Kanten
- (Volumen)

Inzident zu einem Vertex



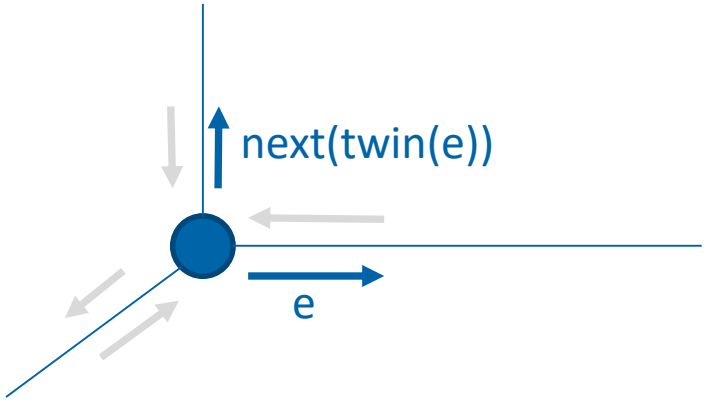
Operationen mittels Halbkanten

Around-Vertex

Erlaubt die Bestimmung aller

- Flächen
- Kanten
- (Volumen)

Inzident zu einem Vertex



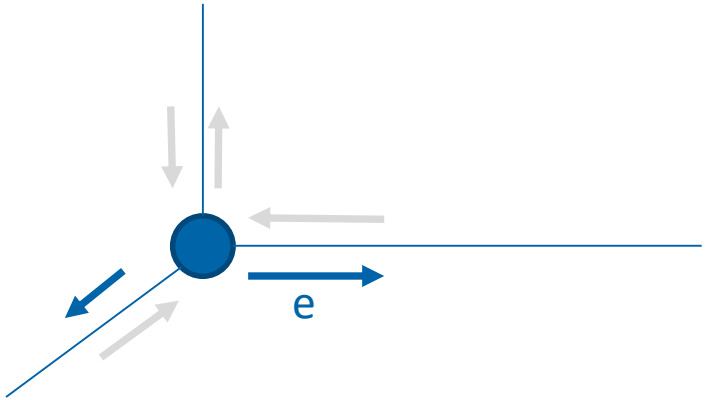
Operationen mittels Halbkanten

Around-Vertex

Erlaubt die Bestimmung aller

- Flächen
- Kanten
- (Volumen)

Inzident zu einem Vertex



next(twin(next(twin(e))))

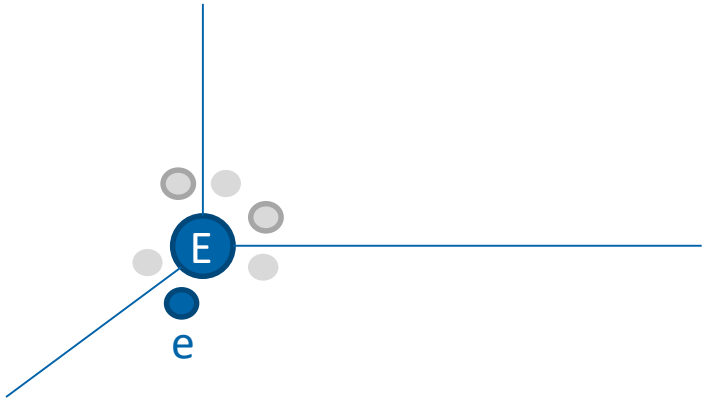
Operationen mittels Halbkanten

Around-Edge

Erlaubt die Bestimmung aller

- Vertices
- Flächen
- (Volumen)

inzident zu einer Kante



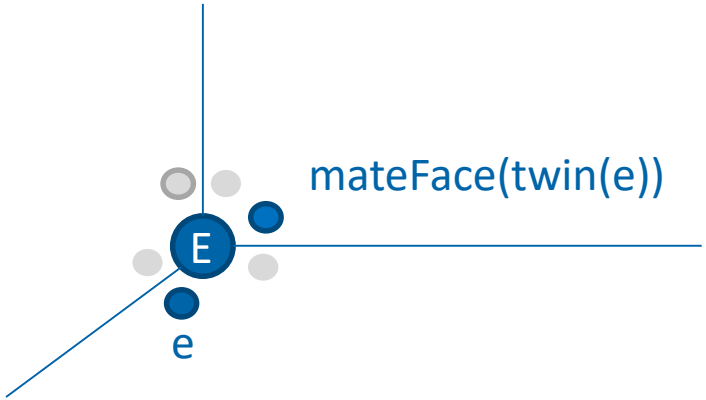
Operationen mittels Halbkanten

Around-Edge

Erlaubt die Bestimmung aller

- Vertices
- Flächen
- (Volumen)

inzident zu einer Kante



CIEL

Beschreibung der Topologie eines (unstrukturierten) Gitters über orientierte Halb-Kanten und 4 Verknüpfungen zw. Halbkanten:

1. `next_around_face`
2. `mate_face`
3. `mate_cell`
4. `next_around_vertex`

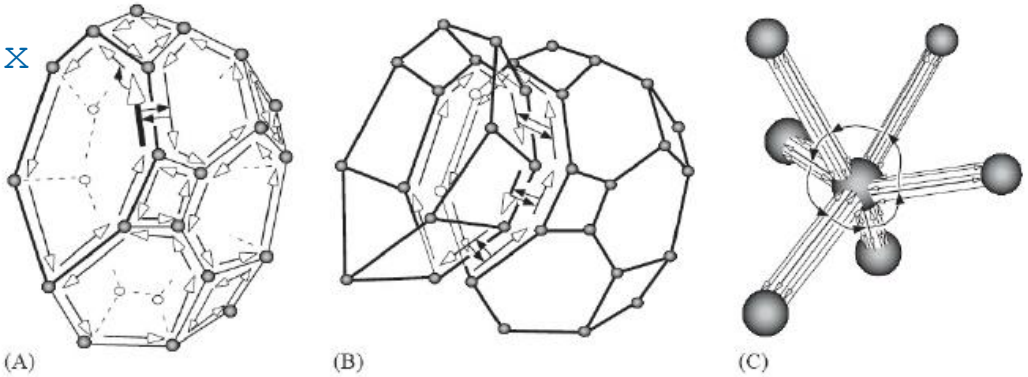


Fig. 4. The notion of half-edge (white-headed arrows) plays a central role in the CIEL data structure. Each half-edge has a pointer to its successor `next_around_face` in the polygon, and a pointer `mate_face` to a half-edge of adjacent polygon (black-headed arrows in (A)). A half-edge also has a pointer `mate_cell` enabling adjacent polyhedron to be retrieved (B). In addition, one half-edge per edge starting from the same vertex appears in a circular linked list implemented by the `next_around_vertex` pointer (C).

CIEL

1. Eine polygonale Fläche eines Polyeders besteht aus einer Schleife von über `next_around_face` verknüpften Halbkanten.

2. Zwei Fläcen eines Polyeders mit gemeinsamer Kante werden durch zwei Halbkanten über `mate_face` verknüpft.

3. Zwei Polyeder mit gemeinsamer Fläche werden durch zwei Halbkanten über `mate_cell` verknüpft.

4. Alle Halbkanten inzident zu einen gegeben Vertex lassen sich sukzessiv über `next_around_vertex` verknüpfte Halbkanten ableiten.

CIEL

- Datenstruktur mit Objekten für Halbkanten, Vertices und Zellen
- Jede Halbkante verweist gemäß der 4 Verknüpfungen auf 4 andere Halbkanten, sowie auf **EINE** inzidenten Vertex (Startvertex) und **EINE** inzidente Zelle
- Jeder Vertex speichert Informationen zu Geometry und physikalischen Eigenschaften, sowie den Verweis auf **EINE** inzidente Halbkante
- Jede Zelle speichert Verweis auf **EINE** inzidente Halbkante

```
struct HalfEdge {
  HalfEdge* next_around_face;
  HalfEdge* next_around_vertex;
  HalfEdge* mate_face;
  HalfEdge* mate_polyhedron;
  Vertex* vertex;
  Cell* cell;
  bool is_marked;
};
```

```
struct Vertex {
  float geometry[3];
  float property;
  float gradient[3];
  HalfEdge* half_edge;
};
```

```
struct Cell {
  HalfEdge* edge;
  bool is_marked;
};
```

Fig. 5. The CIEL data structure: half-edge, cell and vertex definitions.

CIEL - Aufbau

- Unstrukturierte Gitter basieren klassischer Weise über Listen für Vertices, Flächen und Zellen;
- Die Flächen-Liste enthält eine Sequenz von Vertex-Ids;
- Die Zell-Liste enthält eine Sequenz von Flächen-Ids;
- Die Squenzen basieren auf einem vorgegebenen Schema;

Vertex_ID	X	Y	Z	…
v_1	x_{v_1}	y_{v_1}	z_{v_1}	…
v_2	x_{v_2}	y_{v_2}	z_{v_2}	…
v_2	x_{v_3}	y_{v_3}	z_{v_3}	…
…	…	…	…	…

Face_ID	Vertex_List
f_1	$(v_i, v_j, \dots)$
f_2	$(v_k, v_l, \dots)$
f_3	$(v_m, v_n, \dots)$
…	…

Cell_ID	Vertex_List
c_1	$(f_i, f_j, \dots)$
c_2	$(f_k, f_l, \dots)$
c_3	$(f_m, f_n, \dots)$
…	…

CIEL - Aufbau

- CIEL verwendet eine solche Listen Repräsentation als Ausgangsdatenstruktur;

1. Zuordnung von Vertex-Ids (wenn nicht schon vorhanden)

Vertex_ID	X	Y	Z	…
v_1	x_{v_1}	y_{v_1}	z_{v_1}	…
v_2	x_{v_2}	y_{v_2}	z_{v_2}	…
v_2	x_{v_3}	y_{v_3}	z_{v_3}	…
…	…	…	…	…

Face_ID	Vertex_List
f_1	$(v_i, v_j, \dots)$
f_2	$(v_k, v_l, \dots)$
f_3	$(v_m, v_n, \dots)$
…	…

Cell_ID	Vertex_List
c_1	$(f_i, f_j, \dots)$
c_2	$(f_k, f_l, \dots)$
c_3	$(f_m, f_n, \dots)$
…	…

CIEL - Aufbau

2. Für jeden Polyeder werden alle Polygone erstellt. Dabei werden neue Halbkanten erstellt und direkt mit bereits erstellten Halbkanten für diesen Polyeder und eventuell bereits bestehende Nachbarpolyeder verknüpft. Dabei wird die Halbkante-Liste basierend auf ihren vorhandenen Einträgen erweitert.

CIEL - Aufbau

3. Basierend auf diesen Informationen lässt sich CIEL in linearer Zeit mit $\mathcal{O}(kn)$ aufbauen mit $k = fve$:

- n: Anzahl der Polyeder
- f: mittlere Anzahl der Flächen pro Polyeder
- v: mittlere Anzahl der Vertices pro Polygon
- e: mittlere Anzahl der Kanten inzident zu einem Vertex

- Tetraedergitter: $k = 192$
- Hexaedergitter: $k = 96$

CIEL für Volumenrendering

CIEL wurde explizit dafür entworfen, um Volumengitter effizient und in Echtzeit zu visualisieren.
Über *inkrementelles Slicing* wird eine Menge aufeinanderfolgender Flächen entweder

- Parallel zur Projektionsebene

oder

- Als auf Parameterwerten basierende Iso-Flächen

erzeugt, welche denn mittels *Alpha-Blending* übereinander dargestellt werden.

Über CIEL lassen sich sehr effizient alle zu schneidenen Kanten eines Polyeder traversieren, wenn man eine Schnittkante kennt.

CIEL für Volumenrendering

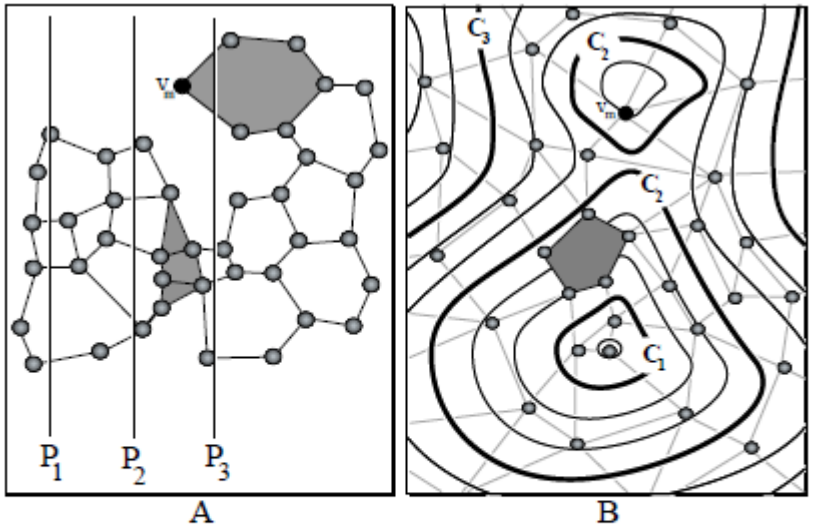


Fig. 2. The algorithm computes a series of slices (A) or isosurfaces (B) incrementally, by maintaining a list of active elements. Small cells, such as the shaded triangle and quadrilateral between P_2 and P_3 in Figure A, and the shaded polygon between C_1 and C_2 in Figure B, are used for the mesh-driven propagation, even if they are skipped by the slicing surfaces. Local minima (v_m in Figures A and B) need to be taken into account when propagating.

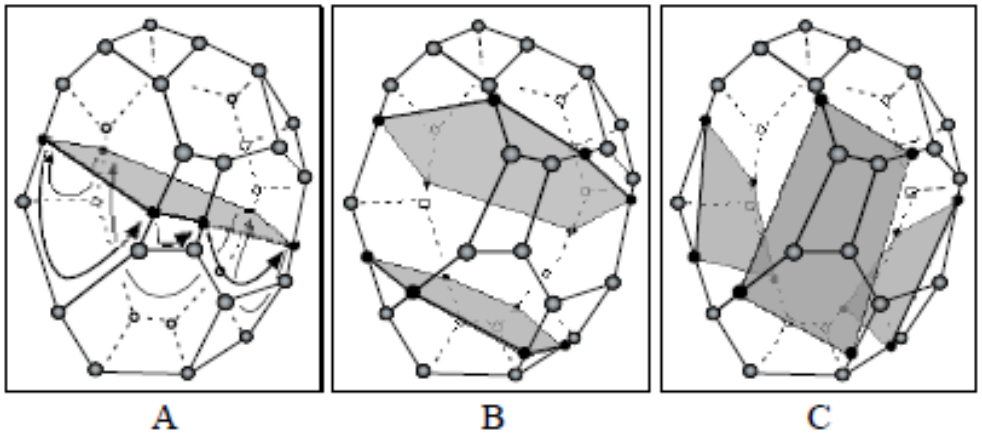


Fig. 3. A: Starting from an intersected edge in a given polyhedron, all the intersections are retrieved by "turning around" the faces of the polyhedron; B and C: two possible interpretations for a given configuration. If cells are convex, slicing cannot produce ambiguities. Ambiguities may only arise if the polyhedron is concave or if the property is not monotonous within the cell. In this case, turning around polygon faces may not lead to the correct order.

Take-home questions:

1. Wie lässt sich über Halbkanten-Operationen der End-Vertex einer Kante e ermitteln?
2. Angenommen, eine Ebene schneidet ein unstrukturiertes Gitter und eine Schnittkante ist bereits bekannt:
   a) Wie lässt sich die nächste geschnittene Kante des zugehörigen Polygons ermitteln?
   b) Wie lassen sich alle weiteren geschnittenen Zellen ohne aufwändige Suche ermitteln?

Institut für Geophysik und Geoinformatik
Dr. Peter Menzel
Gustav-Zeuner-Str. 12
09599 Freiberg
Tel. +49(0)3731 39-3815

Literatur

- Mallet, Jean-Laurent (2002) *Geomodelling*. Oxford University Press.
- Lévy, Bruno, Caumon, Guilliaume, Conreaux, Stéphane und Cavin, Xavier (2001) *Circular Incident Edge Lists : a Data Structure for Rendering Complex Unstructured Grids.* Conference: Proceedings of IEEE Visualization 2001, October 24-26, 2001, San Diego, CA, USA
- Caumon, Guilliaume, Lévy, Bruno und Paul, Jean-Claude (2003) *Combinatorial Data Structures for Volume Rendering Unstructured Grids.* Technischer Bericht.
- Caumon, Guilliaume, Lévy, Bruno, Castanié, Laurent und Paul, Jean-Claude (2005) *Visualization of grids conforming to geological structures: a topological approach.* Computers & Geosciences, 31, pp. 671-680

Institut für Geophysik und Geoinformatik
Dr. Peter Menzel
Gustav-Zeuner-Str. 12
09599 Freiberg

Tel. +49(0)3731 39-3815