matfyz

Computer Graphics Charles University

Rekonstrukce izoploch

© 1996-2015 Josef Pelikán, CGG MFF UK Praha

http://cgg.mff.cuni.cz/~pepca/

pepca@cgg.mff.cuni.cz

Rekonstrukce izoploch

- výpočet a zobrazení jedné nebo více **izoploch**
  - prahové hodnoty zadává uživatel
  - aproximace izoplochy sítí rovinných plošek
- **neprůhledné kostky** („cuberille”)
  - Herman: 1979
- **napojování izočar** („contour connecting”)
  - Keppel: 1975, Fuchs: 1977, Ekoule: 1991
- **pochodující kostky** („marching cubes”)
  - Lorensen: 1987

Definice izoplochy

- užší smysl pojmu: množina bodů, na kterých funkce **f(x,y,z)** nabývá dané hodnoty $\mathbf{h_o}$:

$$Izo(h_0) = \{ [x,y,z] \mid f(x,y,z) = h_0 \}$$

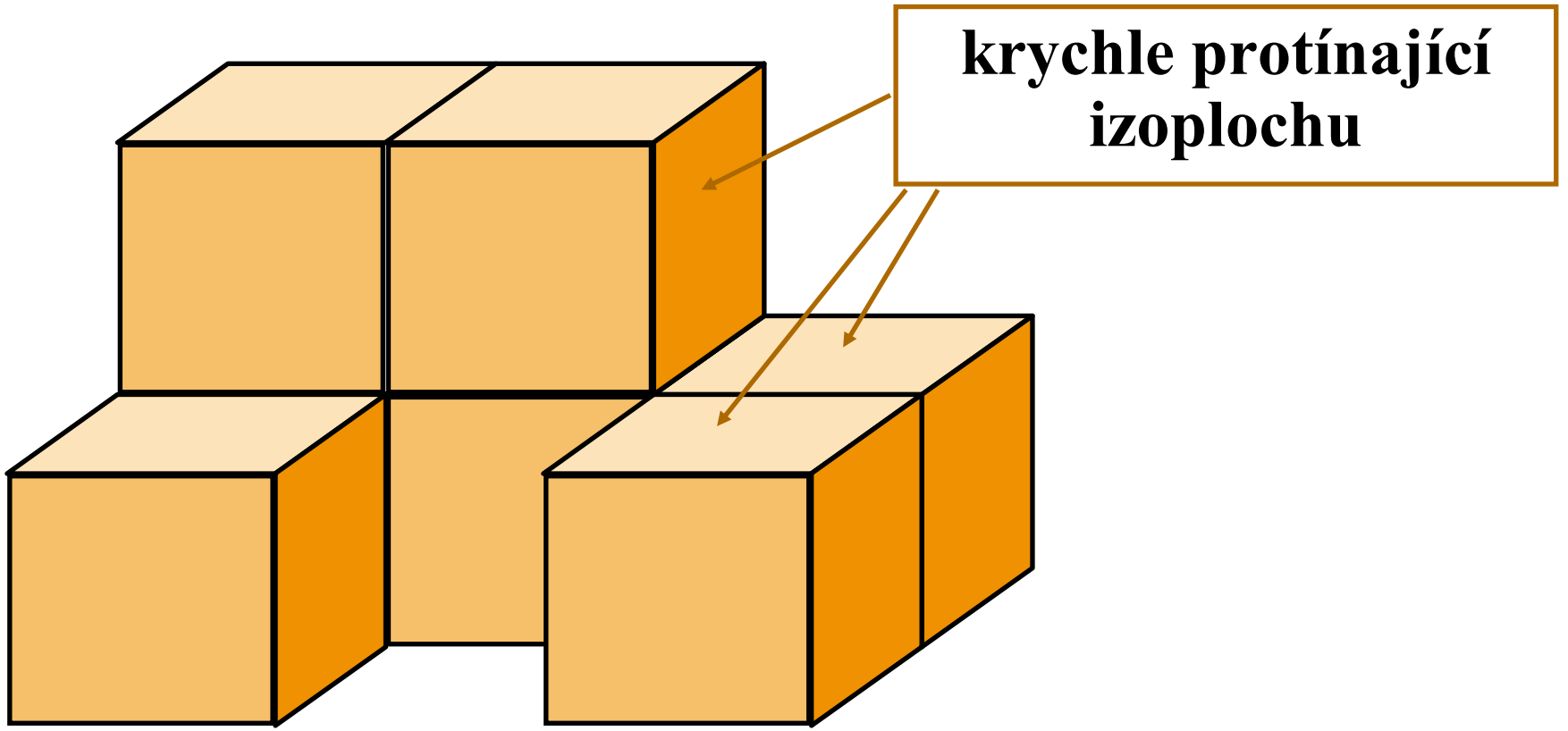
- zobecněná izoplocha: stačí, když v každém okolí bodu **B(x,y,z;ε)** jsou hodnoty větší i menší než práh $\mathbf{h_o}$:

$$Izo(h_0) = \{ [x,y,z]=P \mid \forall\, \epsilon>0 \;\; min_{B(P;\epsilon)} f < h_0 \leq max_{B(P;\epsilon)} f \}$$

vrstevnice na svislé skále ...

Neprůhledné kostky („cuberille”)

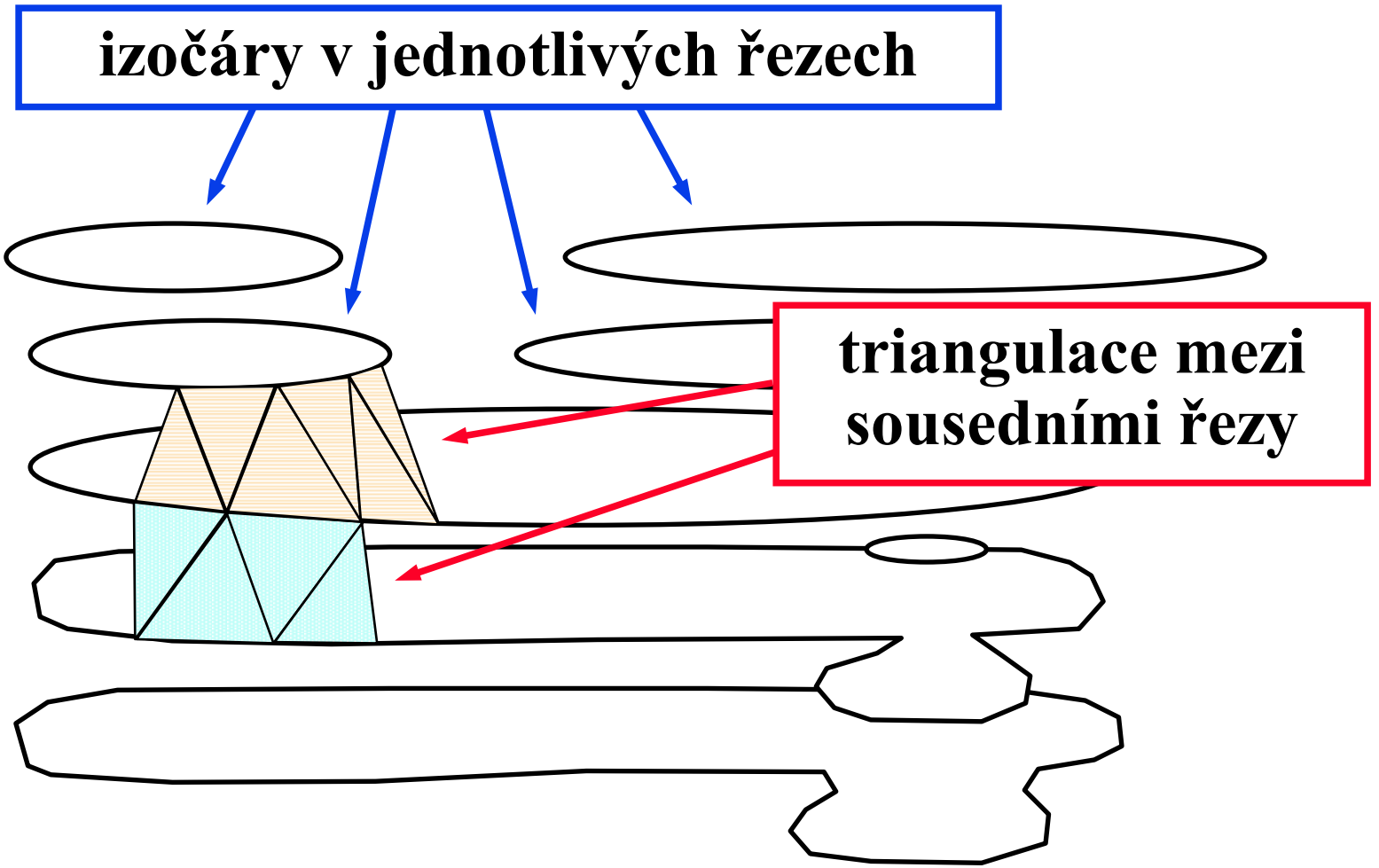
$$\min\{V(P_{ijk})\} \leq h < \max\{V(P_{ijk})\}$$



Neprůhledné kostky

- zobrazení krychlí, které protínají **zadanou izoplochu**
  - podle směru pohledu stačí kreslit pouze tři stěny krychle
- aproximace je příliš **hrubá** (plocha je hranatá)
  - lepší vzhled - **spojité stínování** gradientní metodou (hranaté okraje však zůstávají)
- současné zobrazení **několika izoploch**
  - technika poloprůhledných ploch (kanál alfa)

Napojování izočar



Napojování izočar

1. **výpočet izočar** v jednotlivých řezech
   - izočára se reprezentuje jako lomená čára
   - jeden řez může obsahovat několik uzavřených smyček izočáry
2. **triangulace izoplochy** mezi dvěma řezy
   - trojúhelníky by neměly být příliš protáhlé
   - **topologicky obtížné situace**: několikanásobné větvení (**1:N**, **M:N**), nejednoznačné přiřazení napojovaných izočar

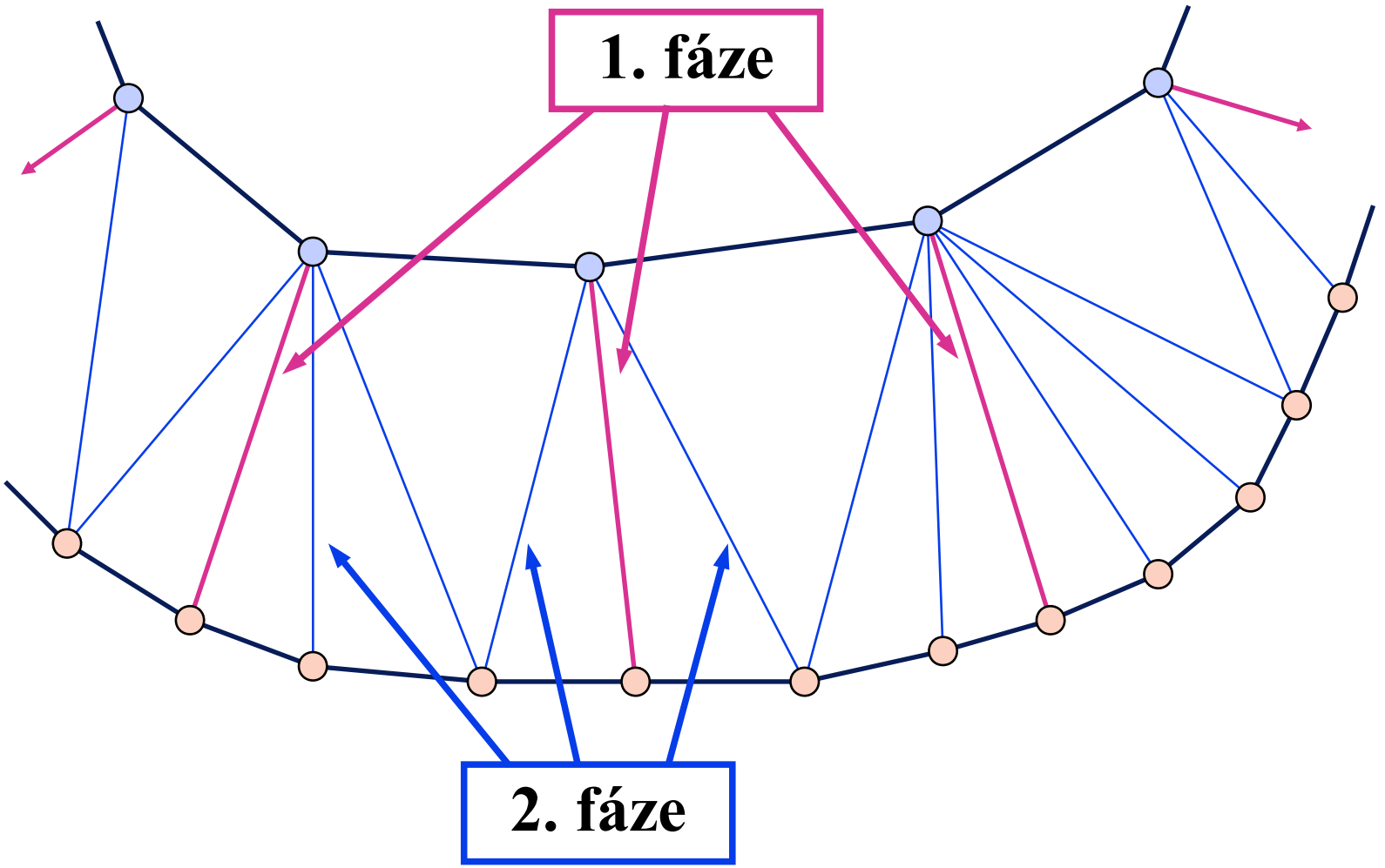
Triangulace (Ekoule, Peyrin, Odet)

- **vlastnosti**:
  - vstupní izočáry mohou mít libovolné tvary
  - generuje dobře vypadající trojúhelníky (nepoužívá dlouhé hrany)
  - uspokojivě řeší větvení **1:N** i **M:N**
- rozdělení na několik geometricky/ topologicky odlišných **případů**:
  - konvexní napojení **1:1**
  - nekonvexní napojení **1:1**
  - větvení **1:N** a **M:N**

Konvexní napojení 1:1

1. pro izočáru s **menším počtem vrcholů**: každý vrchol se spojí s nejbližším bodem naproti
   - efektivní algoritmus pracuje inkrementálně - hledá nejbližšího souseda v okolí naposledy přiřazeného vrcholu
2. zbývající vrcholy z **druhé izočáry** se spojí s nejbližšími protějšky
   - některé vrcholy se musí spojit se dvěma protějšími body (aby vznikla triangulace)

Konvexní napojení 1:1

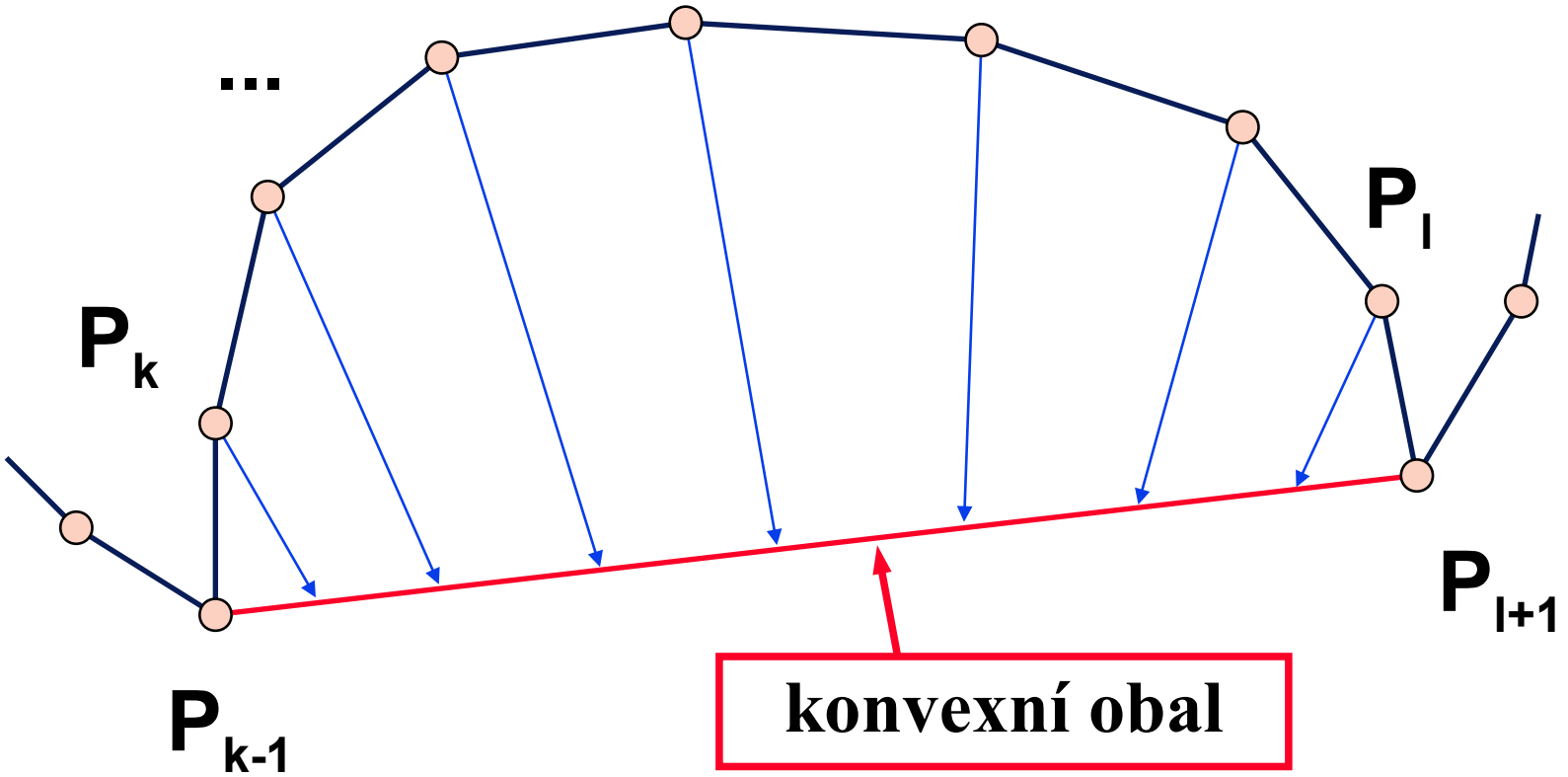


Transformace nekonvex. izočar

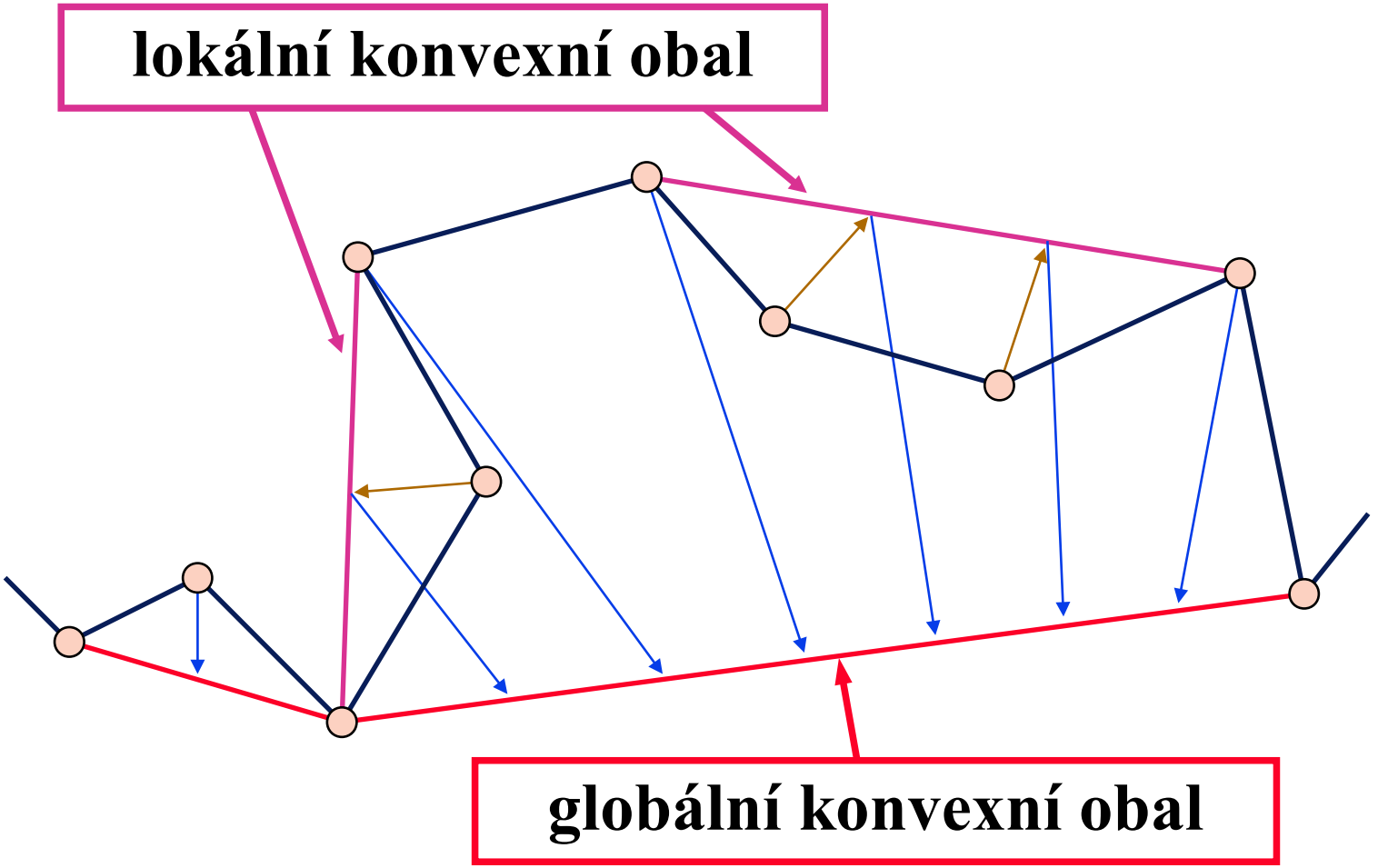
- přenášení vrcholů z nekonvexních úseků na **konvexní obal izočáry**
  - hypotéza: konvexní obaly dvou sousedních izočar si budou více podobné
  - přenášením se zachovává pořadí i relativní vzdálenosti vrcholů

- **triangulace** se provádí na konvexních obalech
  - hranami se potom spojí původní vrcholy izočar

Jednoduchý konkávní výběžek



Složitější nekonvexní izočára



Hierarchická transformace

- převádí obecný mnohoúhelník na **konvexní**
  - zachovává **počet vrcholů** i jejich **relativní vzdálenosti**
  - rekurentní formulace (hierarchický rozklad mnohoúhelníka):

- transformace lomené čáry $\mathbf{P_i} \ldots \mathbf{P_j}$:
  - na začátku se bere celý mnohoúhleník $\mathbf{P_1} \ldots \mathbf{P_N}$

1. výpočet **konvexního obalu (i,j)**
   - vrcholy ležící v konvexním obalu se nepřesunují

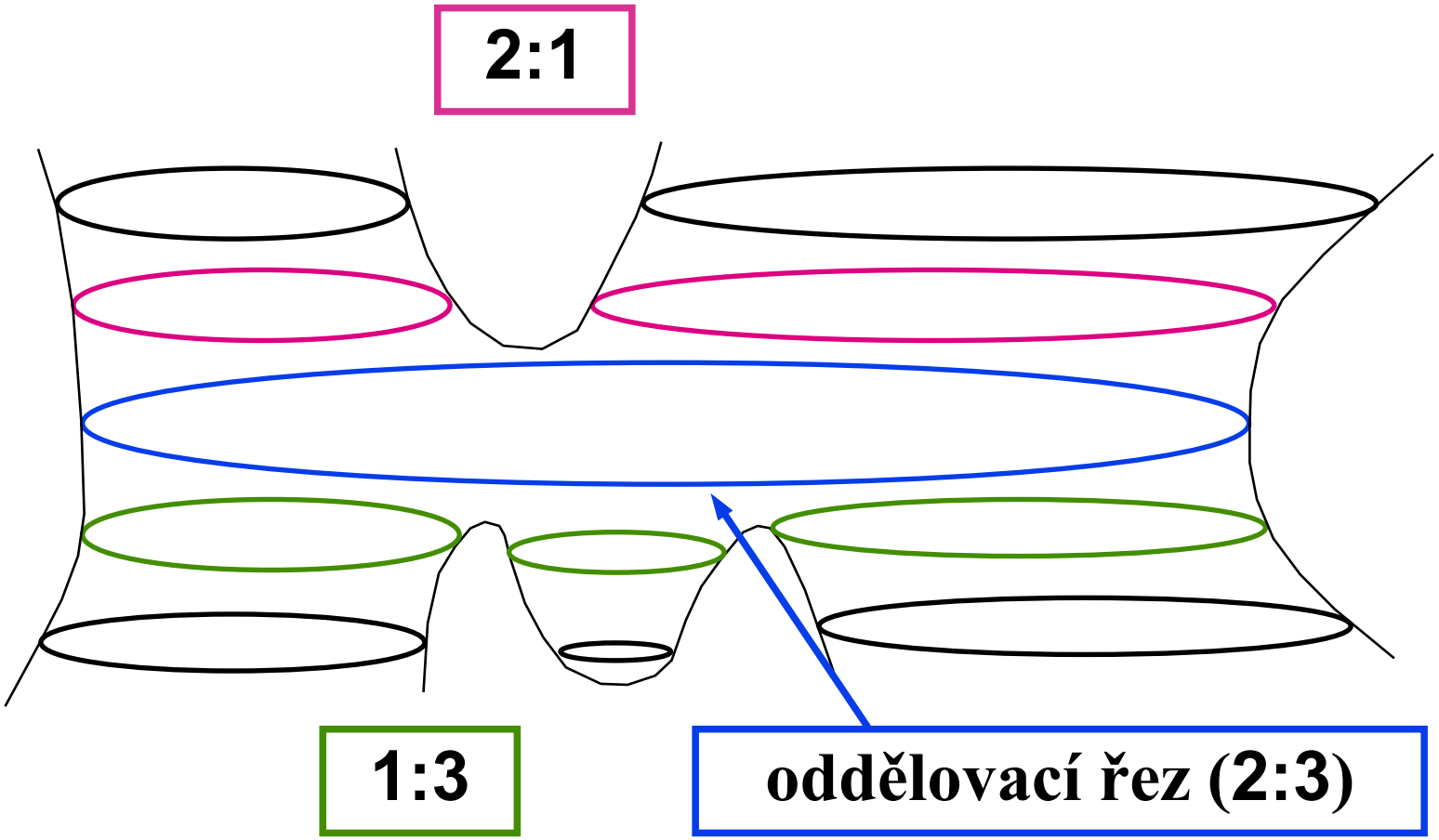
Hierarchická transformace

2. pro každou posloupnost vrcholů $P_k \dots P_l$ $(k \le l)$ neležících v konvexním obalu **(i,j)** se provedou následující kroky:

3. přenesení vrcholů lomené čáry $P_{k-1} \dots P_{l+1}$ do jejího konvexního obalu **(k-1,l+1)**
   - rekurentní vyvolání popisovaného algoritmu

4. přenesení vrcholů P_k až P_l na úsečku $P_{k-1}P_{l+1}$
   - body P_{k-1} a P_{l+1} již leží v konvexním obalu **(i,j)**
   - zachování relativních vzdáleností sousedů

Větvení 1:N a M:N

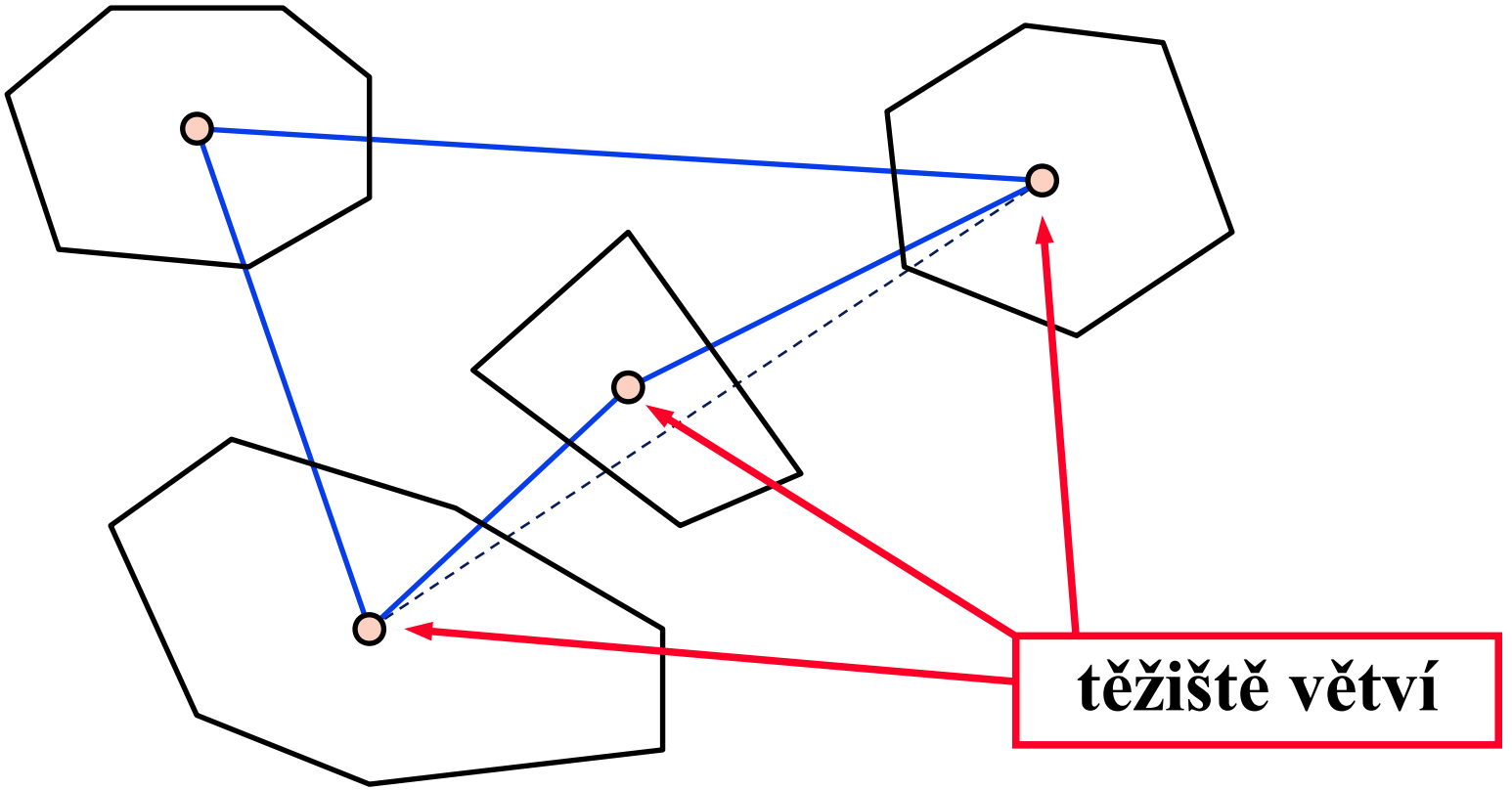


Větvení 1:N

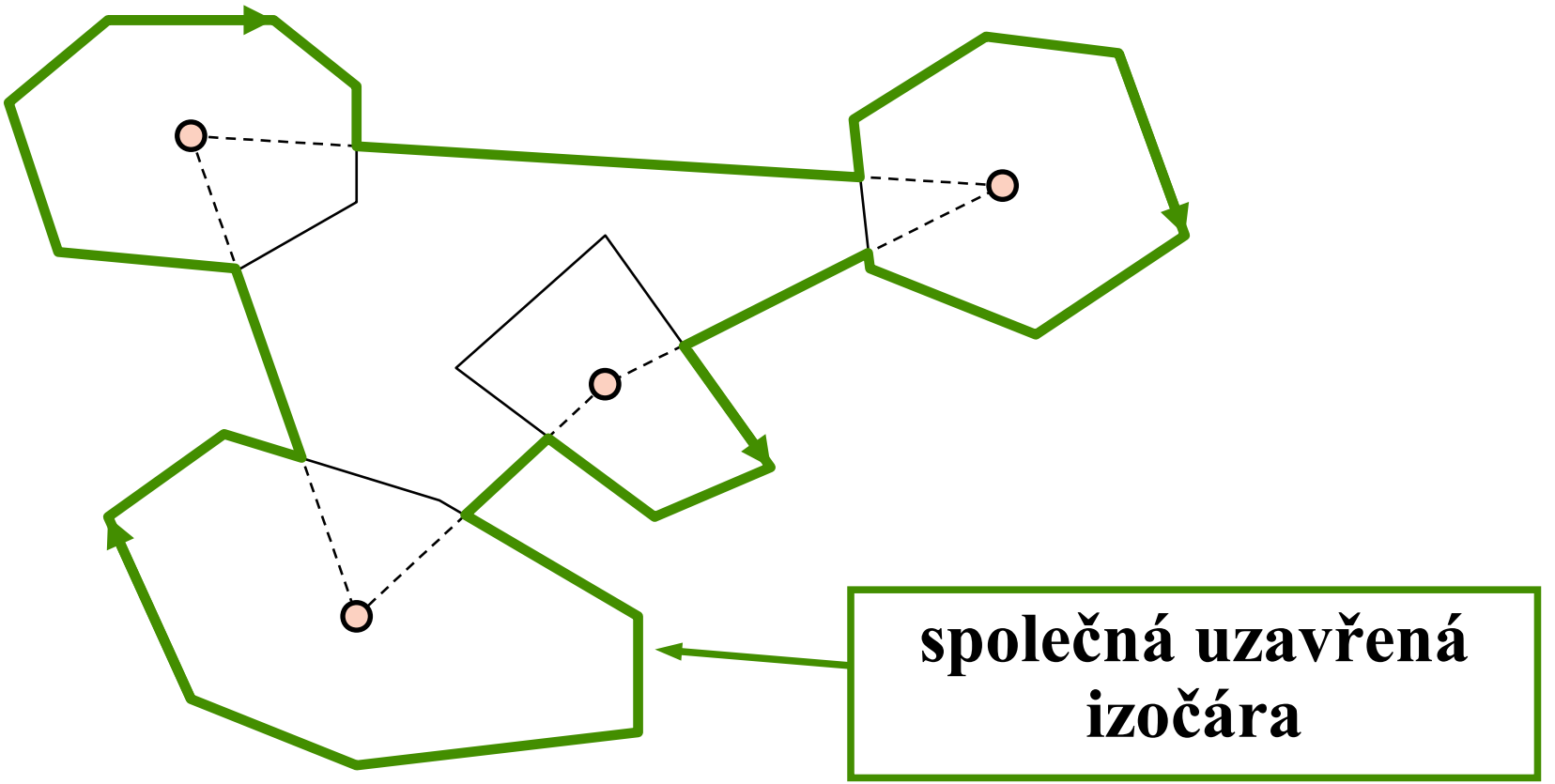
- spočítám **pomocnou izočáru** v polovině mezi oběma řezy
  - konstrukce společné uzavřené izočáry pokrývající všechny spojované větve
  - interpoluje se pomocí prvního kroku algoritmu napojování **1:1**
- **N-krát** aplikuji napojení **1:1** mezi každou větví a pomocnou izočárou

  \+ doplním příp. rovinnou triangulaci

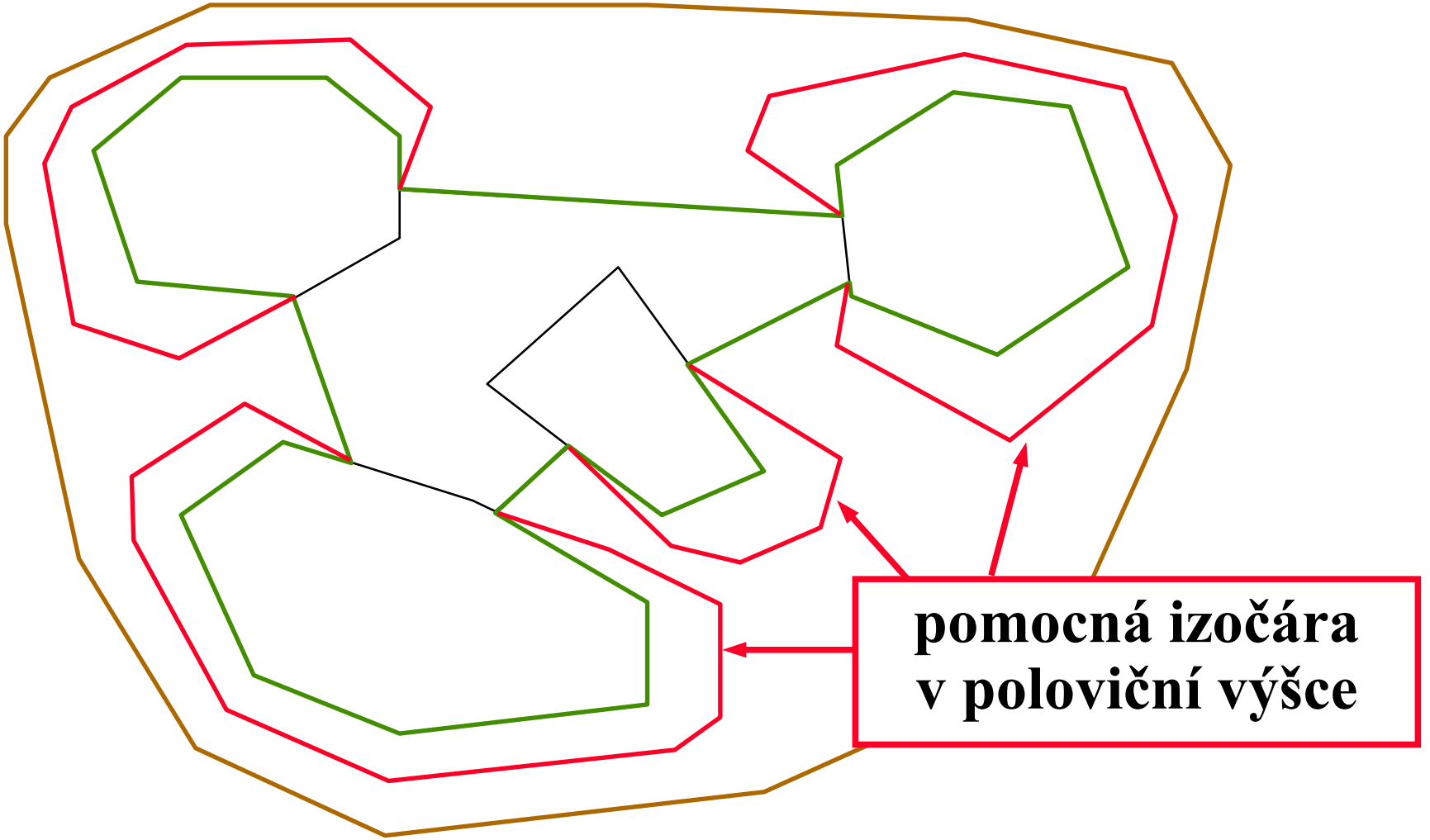
Společná uzavřená izočára



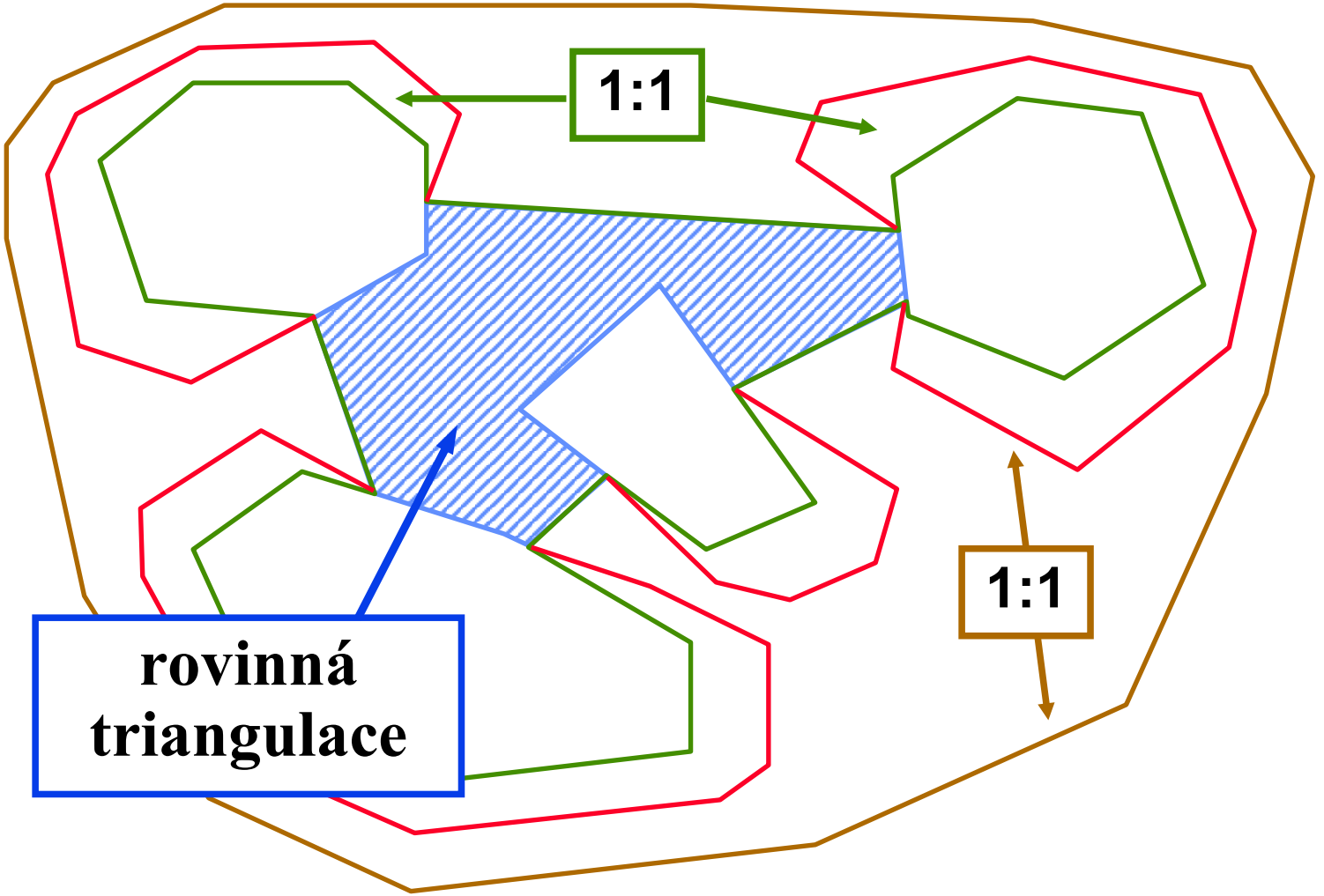
Společná uzavřená izočára



Pomocná izočára



Triangulace

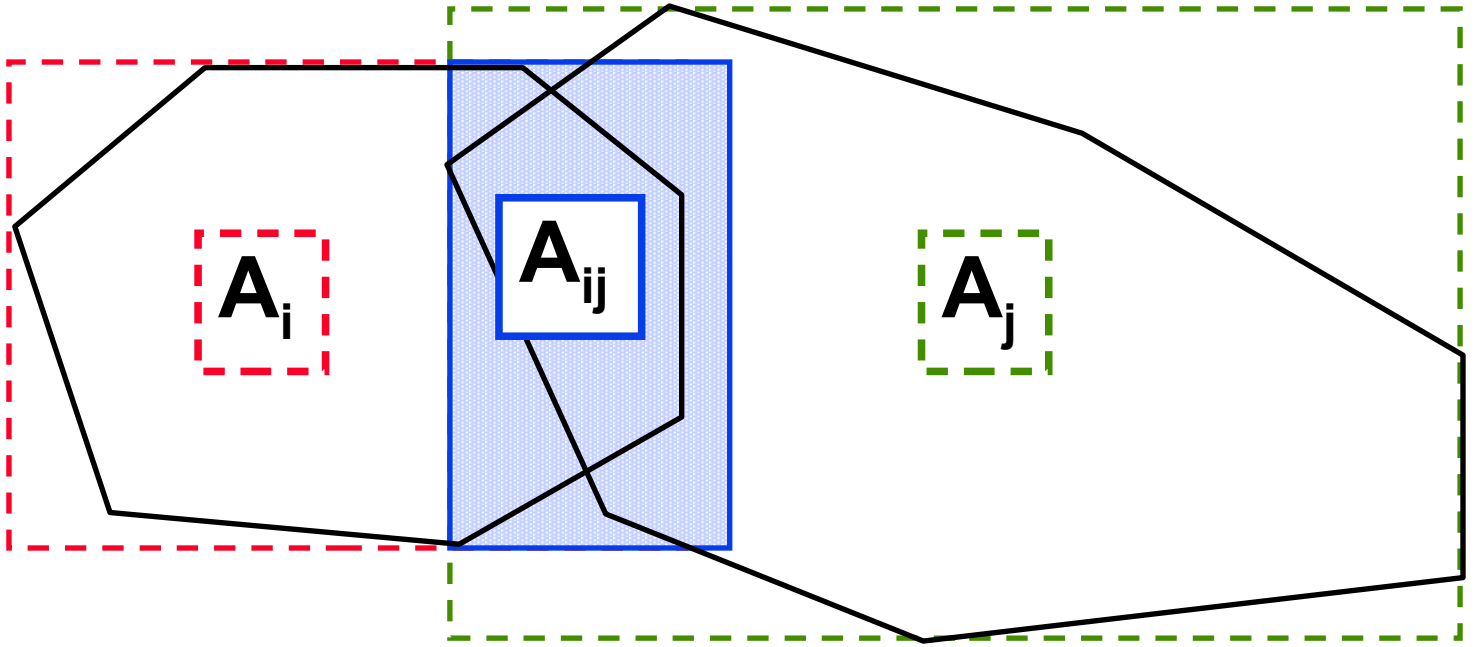


Přiřazování větví soused. řezů

- **automatická procedura** rozhodující o tom, které větve dvou sousedních řezů se napojí
  - jednotlivé komponenty 3D tělesa se mohou v řezech posunovat, měnit tvar nebo rozvětvovat
  - předpokládáme dostatečně tenké řezy (vylučující variantu větvení **M:N**)

- spojení dvou řezů se zredukuje na několik napojení typu **1:1** a **1:N**

Stupeň překrytí dvou izočar



A_{ij} ... přibližná plocha průniku

Koeficient překrytí z **[0,1]**:

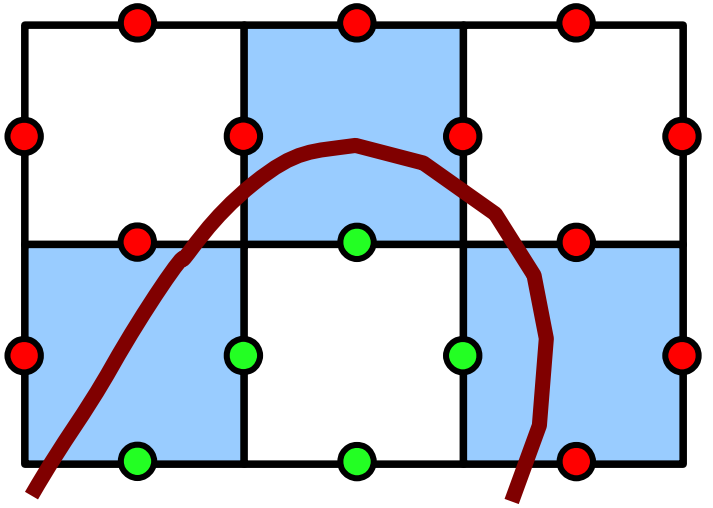
$$S_{ij} = \frac{1}{2}\left(\frac{A_{ij}}{A_i} + \frac{A_{ij}}{A_j}\right)$$

Automatické přiřazování větví

- vhodná **napojovací úroveň c** zadává bipartitní graf přiřazení větví
  - $S_{ij} > c \Rightarrow$ hrana **(i,j)**
- při **dostatečně hustém vzorkování** nedochází k větvení **M:N**
  - v grafu by to ukazovala existence trojice hran **(i,j)**, **(k,j)** a **(k,l)** pro **i≠k** a **j≠l**
- přípustné kombinace odpovídají **napojení 1:1** a **větvení 1:N**

Kreslení izočar v rovině

- Pelikán 1992
- vstupem je reálná funkce na spojitém prostoru f(x,y)
  - diskrétní data se mohou interpolovat/aproximovat
- cílem je nakreslit izočáru v rastrovém prostředí s **pixelovou přesností**
  1. vyhledávání izočar
  2. trasování izočar

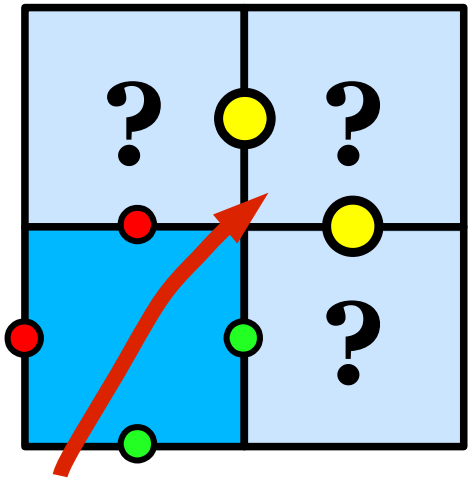


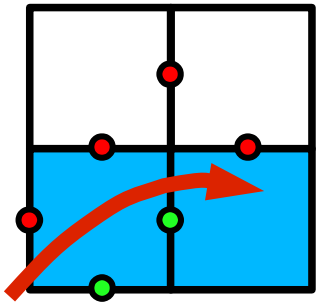
Izočáry v rovině – vyhledávání

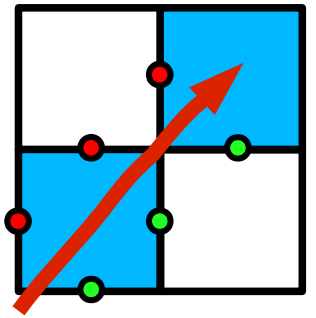
- **„sonda”**
  - **čtyři body uprostřed stran** čtverce (buňky, pixelu)
    (jiný přístup: čtyři rohy čtverce)
- **vyhledávání** izočar
  - průchod mřížkou nebo stromem
  - hledám buňky (pixely) s rozdílnou hodnotou „sondy”
- **urychlování**
  - pro kreslení soustavy více izočar
  - intervalový strom (quadtree nebo „9-tree”)

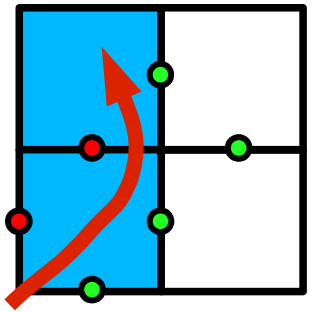
Izočáry v rovině – trasování

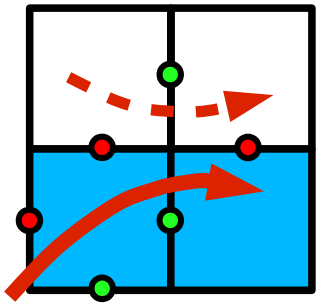
- přechod z aktuálního pixelu izočáry do sousedního
  - konečný počet možností (kombinatorika)
  - již zpracované pixely se značkují
    (… spolupráce s vyhledávacím mechanismem …)











Vektorové izočáry v rovině

- např. vstup pro napojovací alg. (Ekoule)
  - izočára = lomená čára
- **modifikovaný** rastrový algoritmus → generalizace výsledku na lomenou čáru
  - každý k-tý pixel → vrchol
  - podle křivosti čáry (větší křivost → hustší vrcholy)
- **jemnost** hledání a trasování izočar
  - podle požadované přesnosti výsledku
  - souřadnice vrcholů mohou být dopočítány přesněji (dělení čtverce: q-tree, 9-tree)

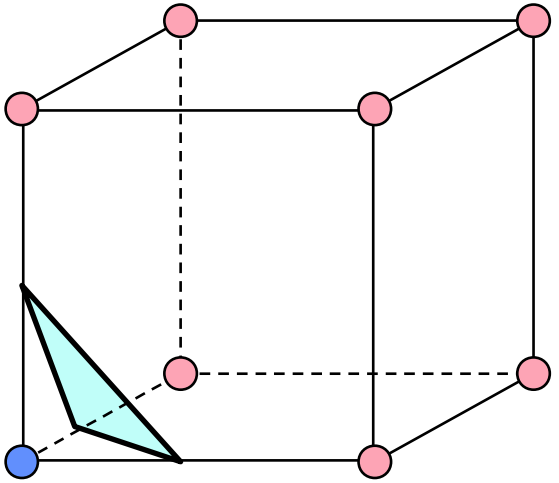
„Pochodující kostky” (Lorensen)

- **jeden z nejpoužívanějších algoritmů**
  - \+ jednoduchá implementace (i HW)
  - \+ generované trojúhelníky nemají dlouhé hrany
  - – velké množství trojúhelníků (i menších než 1 pixel)
  - – mohou se objevit drobné chyby v napojení
- generuje **síť trojúhelníků** v každé buňce
  - – topologie - podle konfigurace vrcholů buňky
  - – vrcholy sítě leží na hranách buňky
- snadno se implementuje **gradientní stínování**

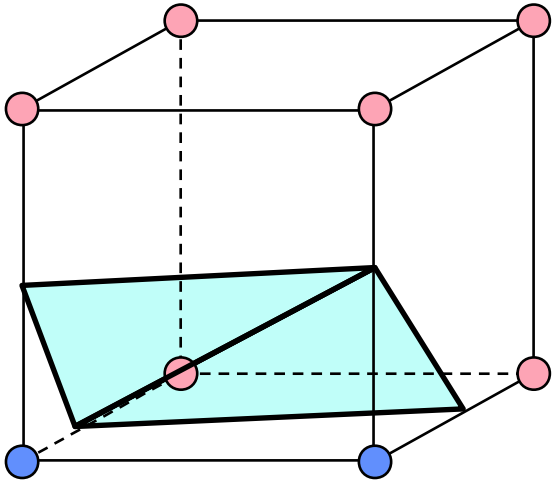
Konfigurace vrcholů buňky

Celkem **14 základních kombinací** (ostatních 240 dostanu pomocí symetrií a otáčení)

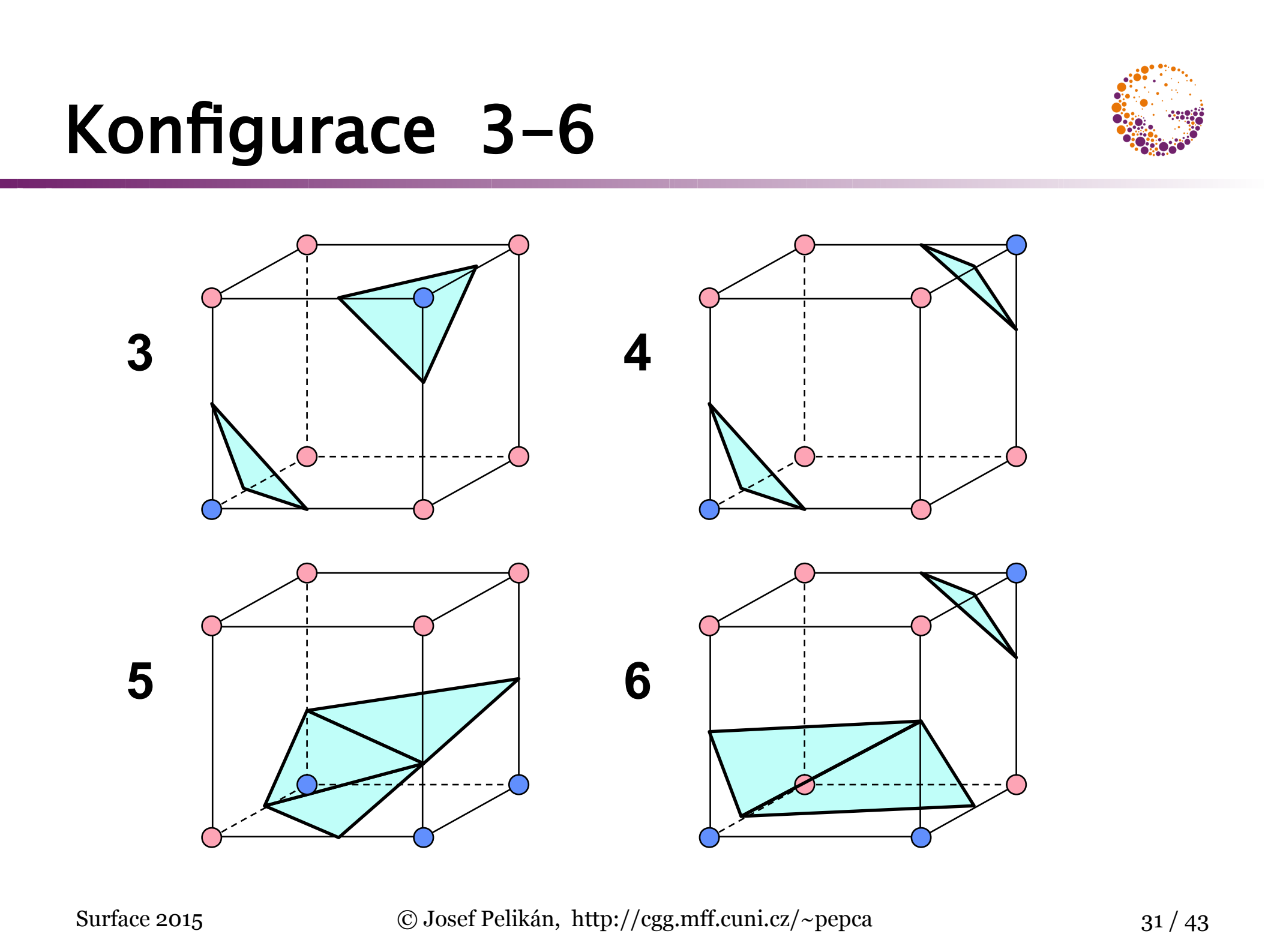
1



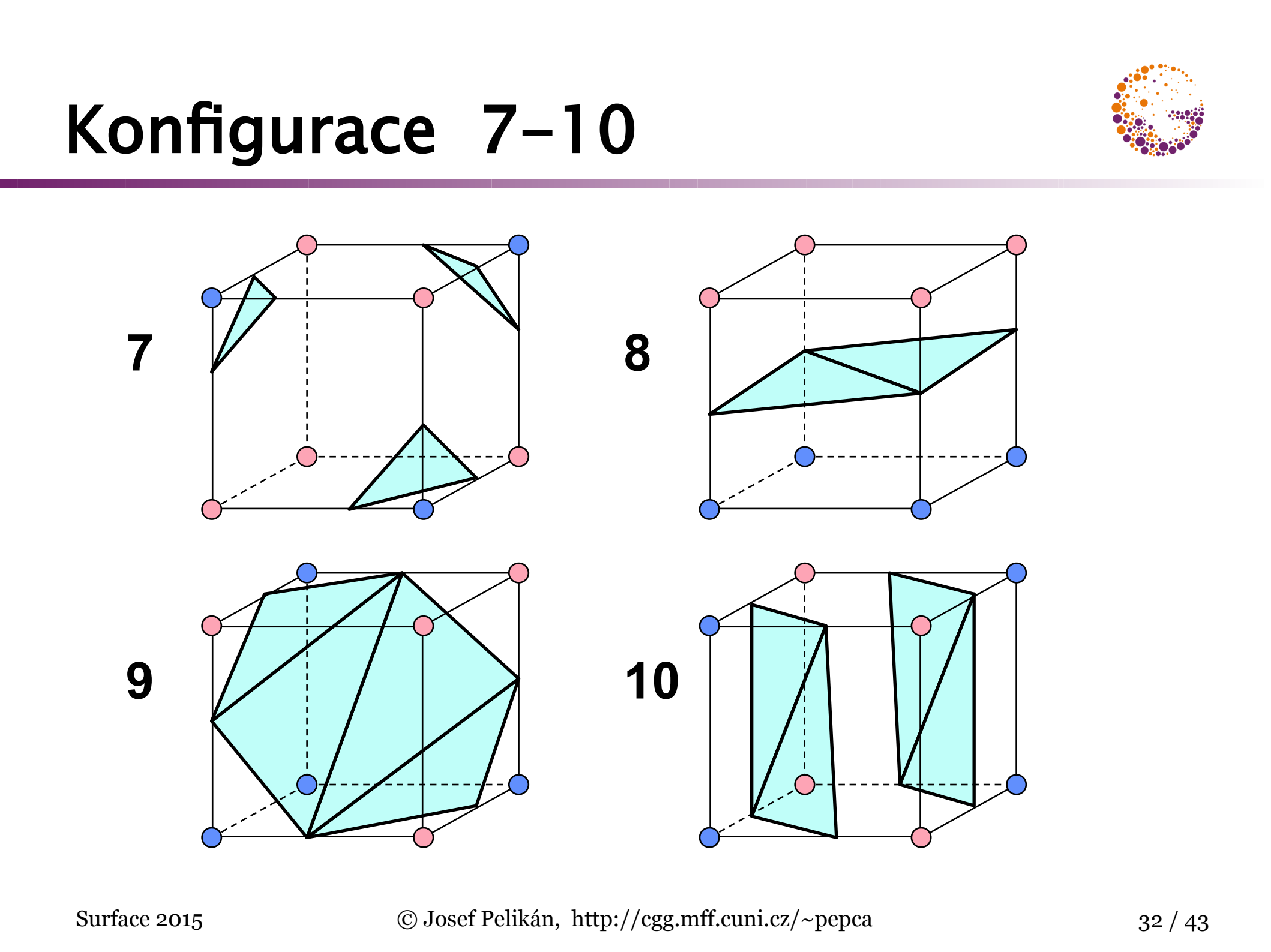
2



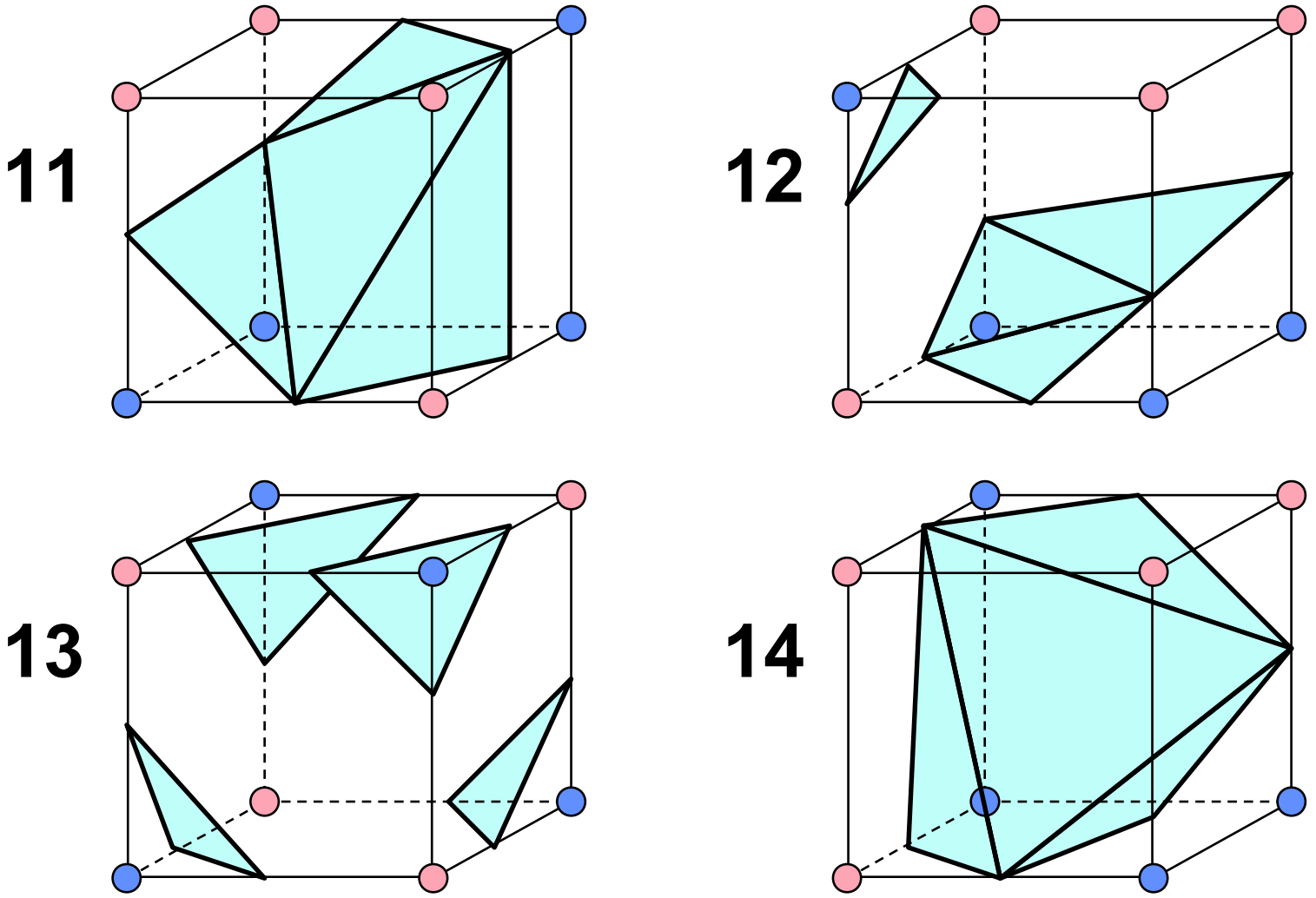
Konfigurace 3–6



Konfigurace 7–10



Konfigurace 11–14



Základní algoritmus

1. **4 sousední řezy** se načtou do paměti
2. pro každou buňku mezi dvěma prostředními řezy se spočítá síť:
3. podle konfigurace hodnot ve vrcholech buňky se z **tabulky** přečte topologie sítě
   – informace o vrcholech a hranách trojúhelníků
4. vrcholy sítě se spočítají **lineární interpolací** podle hodnot ve vrcholech buňky

Základní algoritmus

...

5. **normálové vektory** ve vrcholech buňky se spočítají pomocí diferencí a lineárně se pak interpolují do vrcholů sítě

6. **síť trojúhelníků** se stínováním se zapíše **na výstup**
   - předpokládá se HW podpora viditelnosti a Gouraudova stínování

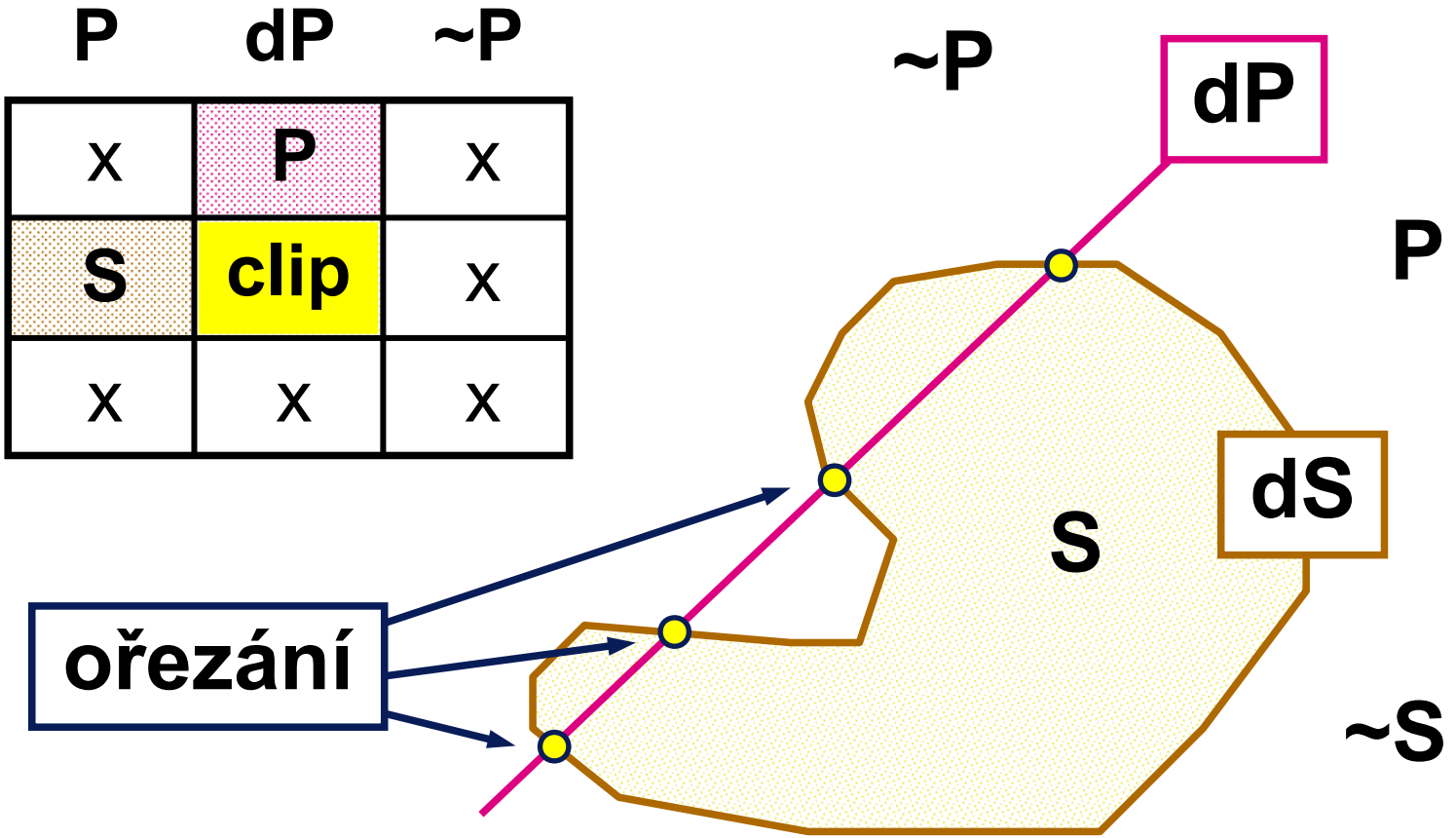
Implementace, vylepšení

- spočítané **vrcholy sítě** a **normálové vektory** se ukládají do „cache” paměti
  - v sousedních buňkách se již nemusí počítat
- náhrada souvislé sítě trojúhelníků v buňce **jediným mnohoúhelníkem** (nerovinným)
  - průmět buňky je často malý (řádově několik pixelů)
  - větší efektivita při kreslení
- **množinové operace**, řezy rovinou
  - paralelně se provádí výpočet několika izoploch

Množinové operace

Tabulka pro **průnik P ∩ S**:

	P	dP	~P
S	x	P	x
dS	S	clip	x
~S	x	x	x



„Dělení kostek” (Cline 1988, patent)

- **efektivnější** varianta algoritmu (nekreslí polygony)
- každá **hraniční buňka** se před zpracováním **promítne**
  - důležitá je velikost projekce v pixelech
- buňky s plochou menší než 1 pixel se kreslí jako **povrchové body: [x,y,z, barva, průhlednost]**
  - Z-buffer (náhodné pořadí)
  - poloprůhledné body se musí třídit podle hloubky (implicitní pořadí dané průchodem)
- větší buňky se **rekurzivně dělí**

„Pochodující čtyřstěny”

- Bourke, 1994
- místo kvádrů se prostor rozdělí na **čtyřstěny**
  - použitelné i v obecnějších topologiích buněk
  - jednodušší kombinace uvnitř čtyřstěnu (jen 14 netriv.)
  - v jednom čtyřstěnu: 1 až 2 trojúhelníky
- z pravidelné sítě vznikne **větší množství čtyřstěnů**
  - 5, 6 nebo 24 čtyřstěnů
  - 5 čtyřstěnů – kvůli spojitosti se musí hlídat i sousední buňky a přepínat mezi 2 konfiguracemi..

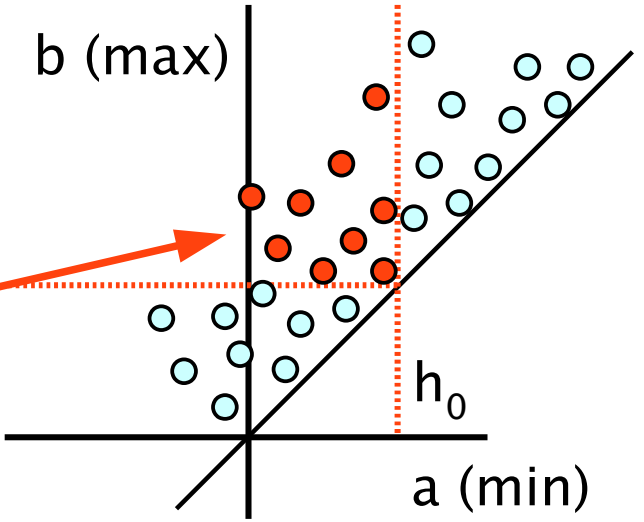
Urychlovací techniky

- metody založené na dekompozici **prostoru hodnot**
  - „span space", intervalové stromy
- „**span space**" – jiný pohled na objemový soubor
  - geometrický prostor ***G*** $G \subset \mathbb{R}^p$
  - prostor hodnot ***V*** $V = \mathbb{R}$
  - množina vzorků ***D*** $D = \{ [g_i, v_i] \} \quad [g_i, v_i] \in G \times V$
  - extrakce izoplochy $S(v) = \{ g_i \mid f(g_i) = v \}$
  - buňka $\mathbf{c_i}$ a její vrcholy $c_i \in C \quad \ldots \quad \{ v_j \}_i$
  - extrémy v buňce $c_i \in C \quad \ldots \quad a_i = min_j(f(v_j)), \; b_i = max_j(f(v_j))$

Intervalový strom

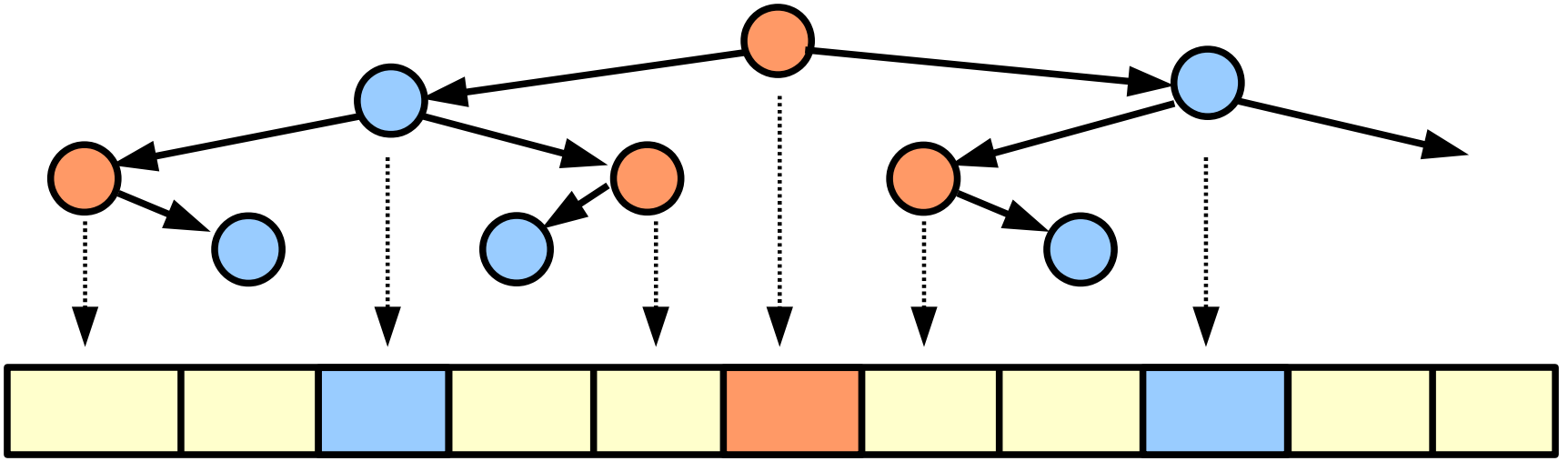
- každá buňka je reprezentována **trojicí** $[c_i, a_i, b_i]$
- buňky se mohou hierarchicky sdružovat do skupin
  - → „**octree**“ – i skupiny obsahují min a max: $[a_i, b_i]$
- nebo se ukládají např. do **KD-stromu**
  - → dělí se ve 2D podle $[a_i, b_i]$ jen polorovina: $\mathbf{a_i \leq b_i}$

lokalizace izoplochy pro h_o



Implicitní KD-strom (bez ukazatelů)

- kompletně **vyvážený KD-strom** uložený v 1D poli
- ukazatel na potomka je implicitně daný
  - $\mathbf{i \rightarrow 2i, 2i+1}$ (à la halda)
  - $\mathbf{h+i \rightarrow \lfloor h+i/2 \rfloor, \lfloor h+3i/2 \rfloor}$ (uzly setříděné zleva-doprava, „cache-friendly“), musí být jednoznačně definováno zaokrouhlení



Literatura

- T. Elvins: *A Survey of Algorithms for Volume Visualization*, CG, August 1992, 194-201
- A. Ekoule et al.: *A Triangulation Algorithm from Arbitrary Shaped Multiple Planar Contours*, ACM TOG, April 1992, 182-199
- W. Lorensen, H. Cline: *Marching Cubes: A High Resolution 3D Surface Construction Algorithm*, CG, July 1987, 163-169
- J. Pelikán: *Raster Algorithms for Contours Extraction*, WSCG '92
- C. Hansen, C. Johnson: *The Visualization Handbook*, Elsevier 2005, pp. 39-82