PV251 Vizualizace

Jaro 2017

Výukový materiál

---

2. přednáška: Typy vstupních dat

V této přednášce se zaměříme na základní část každého vizualizačního procesu a to jsou vstupní data a jejich charakteristika. Nejdříve se zaměříme na postup, jak vlastně může vizualizace dat probíhat.

Prvním krokem při návrhu vizualizace je prozkoumání vlastností vstupních dat a na základě tohoto šetření je zvolena vhodná vizualizace.

Vstupní data lze získávat mnoha způsoby, například pomocí různých šetření a dotazníků (survey), výčtem ze senzorů, generováním pomocí různých simulací, výpočtů a podobně.

Jak probíhá vizualizace dat

Celý proces se skládá ze tří základních částí. V první fázi dochází k samotnému generování dat, například pomocí měření, simulací, modelování atd. Tento proces může být velmi zdlouhavý (zejména v případě měření či simulace) a může být i velmi drahý (simulace či modelování). Ve druhé fázi jsou data vizualizována. Zde je nutné volit správné vizuální mapování a způsob renderování. Rychlost této fáze závisí zejména na použitém hardware a vlastní implementaci zvolených technik. V poslední fázi dochází k interakci s uživatelem.

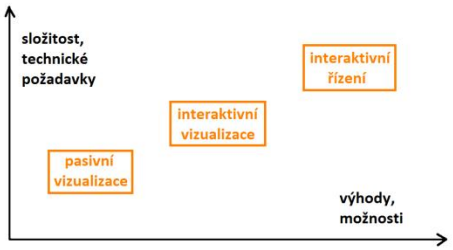
V kontextu těchto tří fází rozlišujeme 3 základní typy vizualizace:

- **Pasivní vizualizace**
  - V tomto přístupu jsou všechny tři fáze striktně odděleny. Po skončení fáze generování dat nastupuje off-line vizualizace, jejímž výsledkem je video nebo animace zobrazující vygenerovaná data. Poslední fází je právě pasivní vizualizace, kdy si uživatel může pouze prohlížet výsledky předchozí fáze.
- **Interaktivní vizualizace**
  - Zde je oddělena pouze fáze generování dat, které probíhá opět off-line. Následuje interaktivní vizualizace, kde jsou data dostupná uživateli pro změnu různých vlastností vizualizace a interakci. Zde je možné například

parametrizovat použité vizualizační techniky. V současné době je toto řešení velmi populární a rozšířené.

- **Interaktivní řízení (steering)**
  - Přístup, který umožňuje uživateli ovlivňovat všechny tři fáze. Data jsou generována „on-the-fly“ a jsou přímo zpracovávána ve vizualizační fázi, která umožní „real-time“ interaktivní náhled na data. Uživatel tedy může ovlivňovat průběh generování dat (simulace, designu při modelování, …). Tento přístup je však velmi náročný na provedení i spotřebované náklady.

Následující graf ukazuje vztah mezi výše uvedenými přístupy.



Data

Data jsou ústředním tématem vizualizace. Vše se vlastně točí kolem dat. Data ovlivňují výběr vhodné vizualizační techniky. Poslední slovo má však většinou uživatel – autor výsledné vizualizace. Při zpracovávání dat je třeba si položit následující otázky:

- Kde data „žijí“? (tj. jaký je jejich datový prostor)
- Jaký je typ dat?
- Jaká reprezentace těchto dat je smysluplná?

Datový prostor, ve kterém se pohybujeme, má různé vlastnosti, jako je jeho dimenzionalita, souřadný systém či region, který ovlivňuje (lokální nebo globální dopad).

Definice vstupních dat

Vstupní data mohou být ve stavu, jak je získáme ze snímačů, výpočtů apod., tedy **raw data** (surová data).

Druhou možností je získání vstupních dat v **předzpracované** podobě – mohlo dojít předem k vyhlazení, odstranění šumu, interpolaci atd.

Definice vstupních dat:

> Množina vstupních dat pro vizualizaci je reprezentována *n* **záznamy** ve tvaru (r_1, r_2, …, r_n). Každý záznam r_i pak obsahuje *m* **proměnných** (**pozorování**) (v_1, v_2, …, r_m). Každá

proměnná může být reprezentována jedním číslem (obecně symbolem) nebo může mít mnohem složitější strukturu (viz dále).

Proměnná:

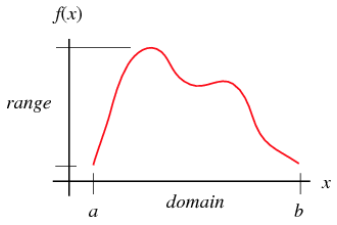
Proměnné klasifikujeme do dvou tříd podle jejich vzájemné závislosti.

- **Nezávislá proměnná iv_i** - proměnná, jejíž hodnota není řízena ani ovlivněna žádnou jinou proměnnou. Příkladem může být proměnná „čas“ v datové množině obsahující časovou sekvenci.
- **Závislá proměnná dv_j** - proměnná, která je ovlivněna jednou nebo více nezávislými proměnnými. Jako příklad lze uvést „teplota“ v daném regionu. Její hodnota může být ovlivněna proměnnými datum, čas a místo.

Formálně lze tedy záznam reprezentovat ve tvaru $r_i = (iv_1, iv_2, \ldots, iv_{m_i}, dv_1, dv_2, \ldots, dv_{m_d})$, kde m_i je počet nezávislých proměnných a m_d je počet závislých proměnných. V kombinaci s předchozí uvedenou notací dostáváme $\mathbf{m = m_i + m_d}$.
V mnoha případech však není nutné mít informaci o tom, zda je daná proměnná závislá nebo nezávislá.

Určení závislosti proměnných v souvislosti se získáváním dat

Při generování dat pomocí procesu či funkce tvoří nezávislé proměnné definiční obor dané funkce (domain, osa X) a závislé proměnné obor hodnot funkce (range, osa Y). Přitom platí, že pro každý vstup v definičním oboru existuje právě jeden jedinečný vstup v oboru hodnot. Obecně platí, že vstupní datová množina neobsahuje vyčerpávající seznam všech možných kombinací hodnot proměnných v dané doméně.



Typy proměnných

V závislosti na typu proměnných, se kterými pracujeme, rozlišujeme dvě základní skupiny:

- **Fyzické typy**
  - Charakterizovány vstupním formátem
  - Charakterizovány typem možných operací
  - Příklad: bool, string, int, float,…
- **Abstraktní typy**
  - Popis dat
  - Charakterizovány metodami/atributy
  - Mohou mít hierarchickou strukturu
  - Příklad: rostliny, zvířata, …

Typy vstupních dat

V nejjednodušší formě každé pozorování nebo proměnná reprezentuje samostatnou jednotku informace. Tuto informaci můžeme kategorizovat do dvou skupin:

- **ordinální (numerická)**
- **nominální (nenumerická)**

Každá z těchto skupin může být dále dělena.

Ordinální data dělíme na:

- binární – nabývající hodnot 0 nebo 1
- diskrétní - obsahují pouze celočíselné hodnoty nebo hodnoty z určité podmnožiny (například (2, 4 , 6))
- spojitá - obsahují reálná čísla (např. v určitém intervalu – [0, 5])

Nominální data rozdělujeme na:

- kategorická - obsahují hodnoty vybrány z konečného (často krátkého) seznamu možností (např. RGB)
- setříděná (ranked) - kategorické proměnné, které mají implicitní uspořádání (například malé, střední, velké)
- náhodná (arbitrary) - obsahují proměnné, které mají potenciálně nekonečný rozsah hodnot a jsou bez implicitního uspořádání (např. adresy)

Měřítko (scale)

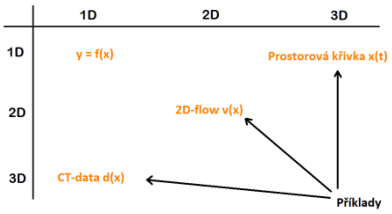
Proměnné lze kategorizovat pomocí matematického konceptu měřítka – scale. Měřítko je důležitý atribut při návrhu příslušné vizualizace, protože každý grafický atribut, který můžeme nastavovat, má v sobě nějaké měřítko obsaženo. V ideálním případě by mělo být měřítko datové proměnné kompatibilní s měřítkem grafické entity nebo vlastnosti, na kterou tuto proměnnou mapujeme.

Měřítko je tvořeno třemi základními atributy:

- **Relace uspořádání na datech.** Tomu odpovídají všechna ordinální data a setříděná nominální data.
- **Vzdálenostní metrika.** Umožňuje počítat vzdálenosti mezi jednotlivými záznamy. Je zřejmé, že tato metrika je přítomna u všech ordinálních proměnných. Naopak u nominálních proměnných není implicitně vůbec definována.
- **Existence absolutní nuly.** V této hodnotě je fixována minimální hodnota proměnné. Používá se zejména při rozlišování jednotlivých typů ordinálních proměnných. Vezmeme-li v úvahu například proměnnou „váha“, je zřejmé, že má nejnižší hodnotu, tedy absolutní nulu. Naopak například „bank balance“, tedy zůstatek na účtu, nejnižší hodnotu zpravidla nemá. Obecně, proměnná vlastní absolutní nulu v případě, že dává smysl na ni aplikovat všechny 4 základní matematické operace (+, -, *, /).

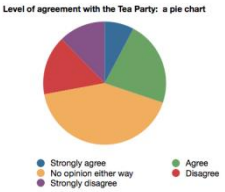
Reprezentace dat

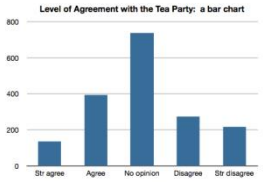
Při volbě správné reprezentace dat vycházíme z jejich závislosti na přítomnosti vlastní prostorové domény a na tom, jak jsou využity jednotlivé dimenze. Zde nás zajímají charakteristiky dat, jaký máme prostor pro zobrazení dat (2D/3D), na kterou část dat se budeme zaměřovat a naopak které části lze abstrahovat.



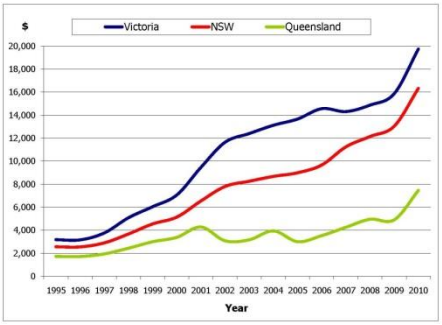
Nyní se zaměříme na několik konkrétních příkladů vstupních dat a jejich mapování na příslušné vizualizační techniky.

- Diskrétní vstupní hodnoty
  - Vstupem je sada hodnot, vizualizována pomocí sloupcových grafů, koláčových grafů, …

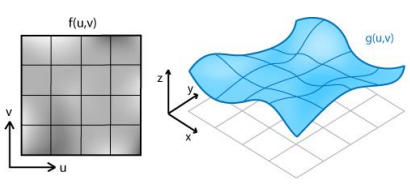




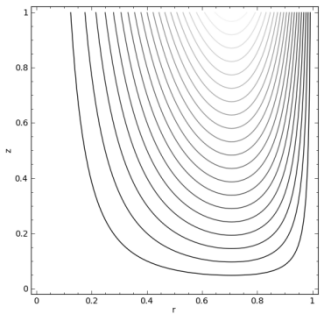
- Spojité vstupní hodnoty
  - Vstupem je funkce, vizualizována pomocí grafů



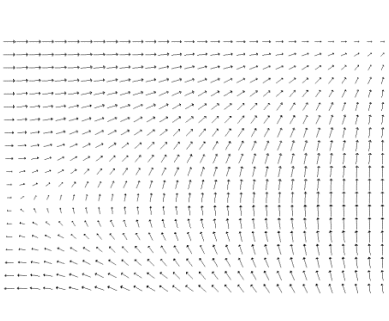
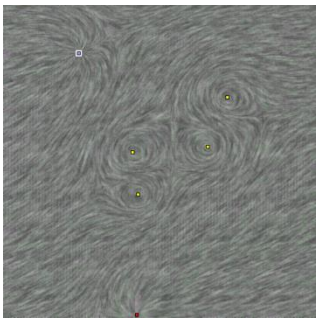
- 2D reálná čísla
  - Vstupem je funkce dvou proměnných, vizualizována pomocí 2D výškových map, kontur ve 2D, …



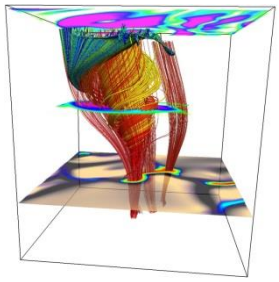
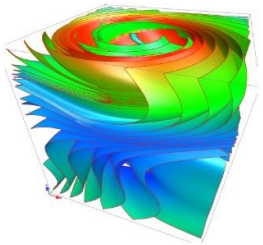
$$g(u,v) \rightarrow (x,y,z) : \begin{cases} x = u \\ y = v \\ z = f(u,v) \end{cases}$$



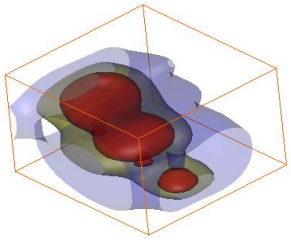
- 2D matice
  - Vstupem jsou 2D vektorová pole, vizualizována pomocí tzv. hedgehog plots, LIC (line integral convolution), streamlets, …

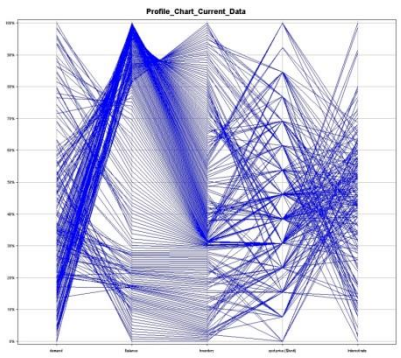

- Prostorová data + čas
  - Vstupem je 3D tok, vizualizován pomocí tzv. streamlines, streamsurfaces

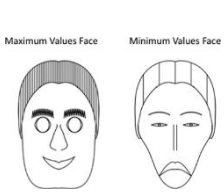



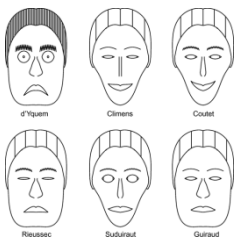
- Prostorová data
  - Vstupem je 3D hustota, vizualizována pomocí izopovrchů, volume renderingu

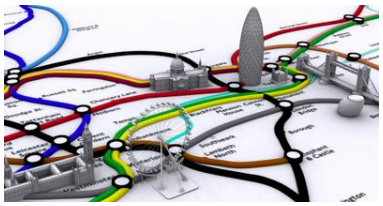



- Multidimenzionální data
  - Vstupem je sada n-tic, vizualizovány pomocí paralelních souřadnic, glyfů, ikon, …









Vnitřní struktura záznamů a vztahy mezi záznamy

Datové množiny mají svou strukturu a to jak ve smyslu jejich reprezentace (syntaxe), tak ve smyslu vztahů uvnitř jednoho záznamu nebo mezi jednotlivými záznamy (sémantika).

Datové množiny mohou nabývat různých struktur:

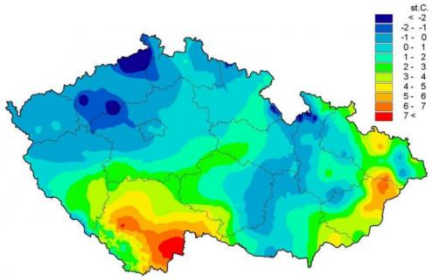
- **Skaláry, vektory a tenzory**
- **Geometrie a gridy**
- **Jiné formy struktur**

Skaláry, vektory a tenzory

- *Skalár* = individuální číslo v datovém záznamu. Skalární hodnoty jsou častým objektem analýz a následných vizualizací. Příkladem skalárních hodnot je například cena výrobku nebo věk.
- *Vektor* = datová položka vzniklá kompozicí několika proměnných do jednoho záznamu. Příkladem je definice bodu v 2D prostoru, který je definován dvojicí hodnot představujících X-ovou a Y-ovou souřadnici. Dalšími příklady mohou být barva (RGB komponenty), telefonní číslo (kód země + číslo). I když každá z komponent vektoru může být zkoumána individuálně, nejběžnější práce s vektorem je jeho interpretace jako celku.
- *Tenzor* = obecnější struktura, skaláry a vektory jsou jeho jednoduchými variantami. Tenzor je definován řádem (rank) a dimenzí prostoru, ve kterém je definován. Bývá reprezentován pomocí pole nebo matice. Skalár je tenzor řádu 0, vektor je tenzor řádu 1. Pro reprezentaci tenzoru řádu 2 v 3D prostoru lze použít matici 3x3. Obecně, tenzor řádu M v D-dimenzionálním prostoru vyžaduje D^M datových hodnot. Příkladem tenzoru, který lze nalézt v datových záznamech, je transformační matice definující lokální souřadný systém.

Geometrie a gridy

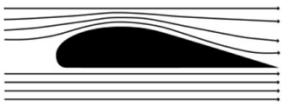
V datových množinách lze běžně nalézt jejich geometrickou strukturu, zejména v takových, které pocházejí za vědeckých experimentů. Nejjednodušší metodou, jak zahrnout geometrickou strukturu do dané datové množiny, je mít explicitní souřadnice pro každý datový záznam v množině. Příkladem může být teplotní mapa republiky, kde je kromě naměřené hodnoty v jednotlivých měřicích stanovištích zaznamenána i jejich poloha, například jako zeměpisná šířka a délka. Dalším příkladem je modelování 3D objektů, kde je jejich geometrie dána souřadnicemi jednotlivých vrcholů, ze kterých se skládá.

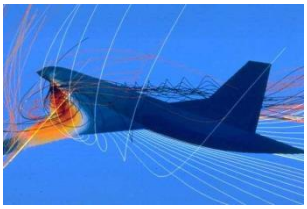


V určitých případech lze geometrickou strukturu odvodit. To lze tehdy, pokud jsou data rozložena na mřížce (**gridu**). Tak můžeme předpokládat, že datové záznamy sousedící s aktuálním záznamem jsou umístěny na sousedních lokacích na mřížce. Proto pokud máme uniformní rozložení mřížky, není nutné uchovávat pro každý záznam jeho souřadnice – ty mohou být odvozeny ze startovní pozice, orientace a velikosti kroku v horizontálním a vertikálním směru.

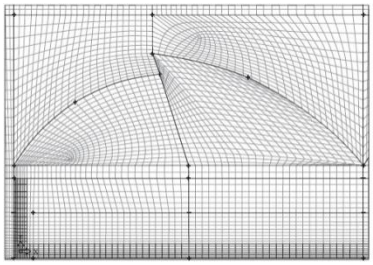
Pro gridová data mohou být použity různé typy souřadných systémů – např. kartézská, sférická nebo hyperbolická souřadná soustava. Její volba je často závislá na způsobu získání dat a na struktuře prostoru, ve kterém jsou data definována. Obecně pro konverzi pozice ze souřadného systému dat do souřadného systému zobrazovacího zařízení (monitor) lze jednoduše použít transformační matici.

Velmi běžná je kromě pravidelné geometrie i ta **nepravidelná**, **neuniformní**. Příkladem je simulace proudu vzduchu kolem křídel letadla. Je totiž potřeba mít větší koncentraci datových záznamů kolem křídel, zatímco záznamy reprezentující proud dále od povrchu letadla mohou být řidší.





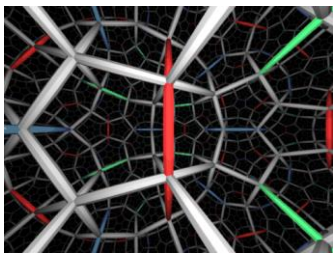
U nepravidelné geometrie je nezbytné mít uchovány souřadnice všech záznamů – nejdou z ničeho odvodit. Podobně je na tom použití neuniformního gridu. Je zřejmé, že v těchto případech významně narůstá výpočetní složitost.

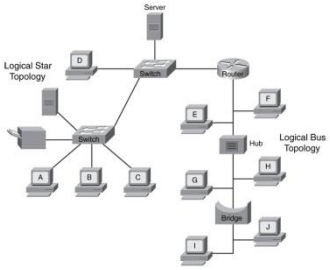


Jiné formy struktur

Čas je jedním z nejvýznamnějších atributů, který lze asociovat s datovými záznamy. Čas má navíc pravděpodobně největší rozsah možných hodnot – často přirozeně využíváme vyjádření v pikosekundách nebo naopak ve tisíciletích. Čas může být rovněž vyjádřen absolutně nebo relativně. Datové množiny obsahující atributy týkající se času mohou být rozloženy rovnoměrně (například vzorkování jevu spojitého v čase) nebo nerovnoměrně (například databáze peněžních transakcí – s ohledem na dobu jejich přijetí či zpracování).

Další velmi důležitou formou je **topologie**, která se nachází v mnohých datových množinách. Definuje tzv. **konektivitu** = jak jsou jednotlivé datové záznamy mezi sebou propojeny. Konektivita tedy definuje určitý vztah mezi záznamy.
Příkladem je povrch objektu definovaný sadou vrcholů (geometrie), které jsou vzájemně propojeny pomocí hran (topologie). Dalším příkladem může být graf (např. strom) reprezentující hierarchii uzlů. Zde jsou topologií jednotlivá propojení mezi uzly.





Konektivita hraje zásadní roli při procesech převzorkování (resampling) či interpolace.

Další příklady strukturovaných dat:

- Magnetická rezonance (magnetic resonance imagery) – hustota dat (skalár), 3 prostorové souřadnice, 3D grid pro konektivitu.
- CFD – computational fluid dynamics – 1 časový a 3 prostorové atributy, 3D grid pro konektivitu (uniformní nebo neuniformní).

- Finančnictví – žádná geometrická struktura, *n* většinou nezávislých komponent, nominální a ordinální, s časovým atributem.
- CAD (computer-aided design) systémy – tři prostorové atributy s konektivitou v podobě hran a polygonů, vlastnosti povrchu.
- Sčítání lidu – sada vstupních polí všech typů, prostorové atributy (např. adresy), časový atribut, konektivita dána podobnostmi v polích.
- Sociální sítě – uzly složeny ze sady vstupních polí všech typů, různé typy konektivity – prostorová, časová nebo závislá na jiných atributech (např. přihlášení do určité skupiny).

Taxonomie – 7 typů dat

Na základě dimenzionality vstupních dat je určeno, jaký typ objektů data reprezentují.

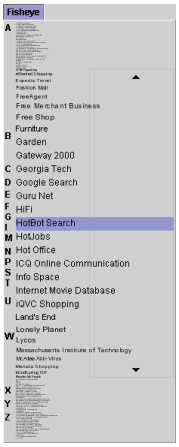
- 1D (lineární sady a sekvence)
- 2D (mapy)
- 3D (objekty, tvary)
- nD (relační)
- Stromy (hierarchie)
- Sítě (grafy)
- Temporální

Pro jednorozměrná data dostáváme reprezentace různých sekvencí. Příkladem dvoudimenzionálních dat jsou například mapy. Ve 3D pracujeme s objekty, zejména s jejich tvarem. Další dimenze reprezentují data s různými relačními vztahy mezi nimi (tzv. multidimenzionální data). Speciálním případem dat jsou stromy, které umožňují budovat hierarchii, dále sítě vytvářející grafy. Poslední významnou skupinu tvoří temporální data sloužící k uchovávání dočasných informací.

Lineární data

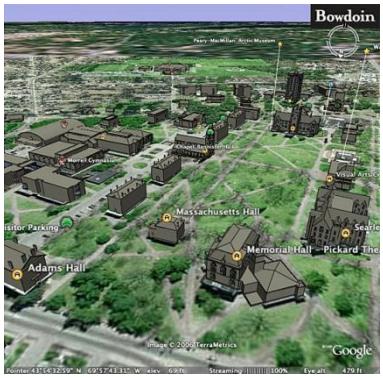
Pro vizualizaci lineárních dat se dá využít několik různých technik. Základní techniky pracují s dlouhými seznamy vstupních položek – například menu nebo zdrojové kódy. Příkladem software pro řešení druhého případu, tedy přehlednou práci se zdrojovými kódy, je program Seesoft (http://dl.acm.org/citation.cfm?id=141348). Ten umožňuje pracovat až s 50000 řádky kódu najednou pomocí mapování každého řádku kódu na tenkou čárku určité barvy. Ta signalizuje „zajímavost“ daného řádku v rámci celého kódu. Například červené řádky signalizují, že byly naposledy měněny a naopak modré řádky nebyly měněny již dlouhou dobu. Seesoft dokáže zpracovávat data z různých zdrojů, jako například z verzovacího systému, který zaznamenává věk, jméno programátora a účel kódu.

Další možností vizualizace lineárních dat jsou metody na principu rybího oka, tzv. fisheye displays.



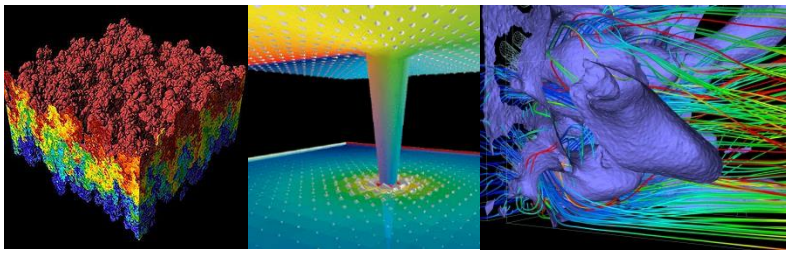
2D data – mapy

Pro vizualizaci 2D dat reprezentujících mapy se využívají geografické informační systémy. Příkladem jsou populární Google maps nebo aplikace Google Earth. GIS umožňují zejména pokládat prostorové dotazy a rovněž analyzovat prostorová data.
http://www.wimp.com/unusualplaces/



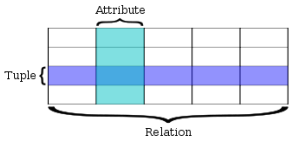
3D data

Vizualizace dat umístěných v 3D prostoru je jednou z nejpopulárnějších oblastí ve vizualizaci. Je mnoho oblastí, kde se tento typ vizualizace využívá. Obrovskou přidanou hodnotu má v tzv. scientific visualization.



Multidimenzionální data

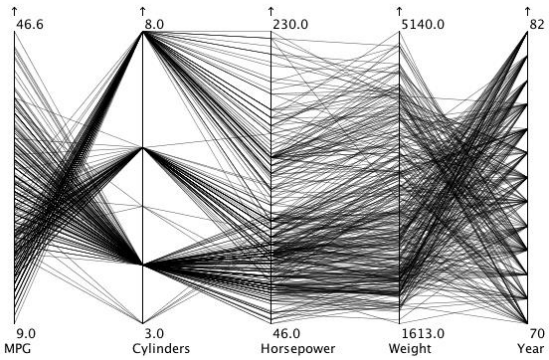
Multidimenzionální data jsou reprezentována n-ticemi proměnných, které jsou v určité relaci. Příkladem mohou být záznamy v relační databázi.



Zmíníme dvě možná řešení. Prvním je vykreslení všech možných párů proměnných ve 2D grafu. Tato metoda je jednoduchá, nicméně pro zobrazení dat jako celku je nepoužitelná.

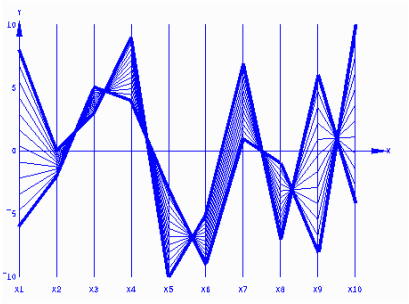
Druhým řešením je využití tzv. **paralelních souřadnic**.

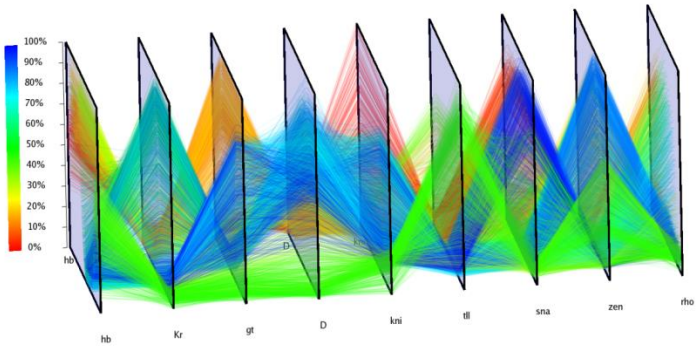
Paralelní souřadnice jsou běžnou metodou vizualizace multidimenzionálních dat, která umožňuje rovněž jejich analýzu. Bod v n-dimenzionálním prostoru je reprezentován lomenou čarou, jejíž vrcholy jsou umístěny na paralelních osách, které jsou typicky vertikální a rovnoměrně rozloženy. Pozice vrcholu na i-té ose odpovídá i-té souřadnici zobrazovaného bodu.



Nevýhodou je, že pozorovateli může zabrat delší dobu, než dokáže zobrazená data správně interpretovat.

Paralelní souřadnice se poprvé objevily v roce 1885, kdy s nimi přišel Maurice d'Ocagne. Znovu objeveny a popularizovány byly v roce 1959 Alfredem Inselbergem a od roku 1977 začaly být systematicky studovány a vyvíjeny jako samostatný souřadnicový systém. Používají se v mnoha odvětvích, jako například data mining, počítačové vidění, řízení letecké dopravy atd.





Zobrazení 3D paralelních souřadnic zachycujících expresi devíti vybraných genů Drosophily.

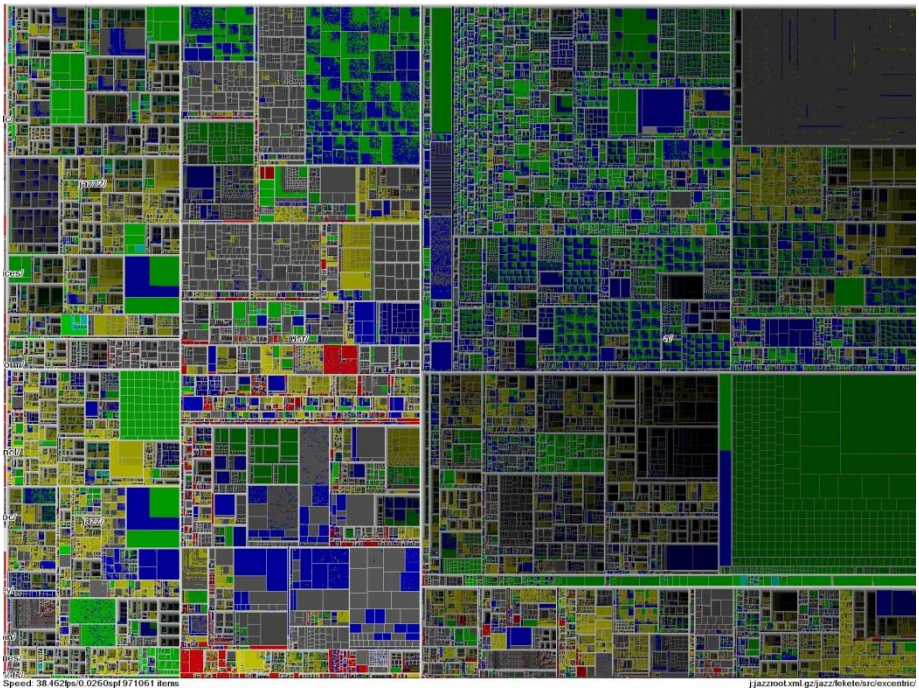
Stromy

Data reprezentovaná stromovou strukturou je nutné nejen zobrazit, ale musíme vizualizovat i jejich strukturu reprezentující jejich vzájemné vztahy. Příklady dat, která jsou předurčena pro zobrazení pomocí stromové struktury, jsou rodokmen nebo file systém na disku.

Typickou vlastností stromových dat je, že při sestupování do nižších vrstev rapidně roste počet dat.

Dalším způsobem zobrazení stromových dat jsou takzvané **Tree Maps**. Zobrazují hierarchická data pomocí vnořených obdélníků. Další informace o daném uzlu jsou reprezentovány pomocí velikosti a barvy příslušného obdélníka.

Potomci ve stromu jsou reprezentováni obdélníky vnořenými uvnitř rodičovských obdélníků. Všechna data jsou viditelná na jediné obrazovce bez nutnosti „scrolování“. Na obrázku je příklad Tree Mapy, která v sobě obsahuje milion různých záznamů.

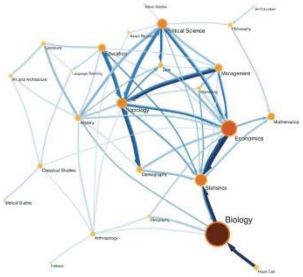


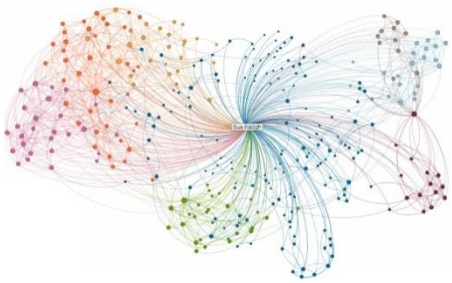
Sítě

U vizualizace sítí jsme v podobné situaci jako u stromů. Kromě samotných dat je nutné vizualizovat i jejich strukturu. V tomto případě však hrany mezi jednotlivými uzly mohou obsahovat cykly (na rozdíl od stromů). Síť se skládá z uzlů – data a hran – jejich vzájemný vztah.

Při návrhu sítě bychom se měli soustředit na splnění následujících kritérií:

- Minimální průniky hran
- Minimální délky hran
- Minimální ohýbání hran

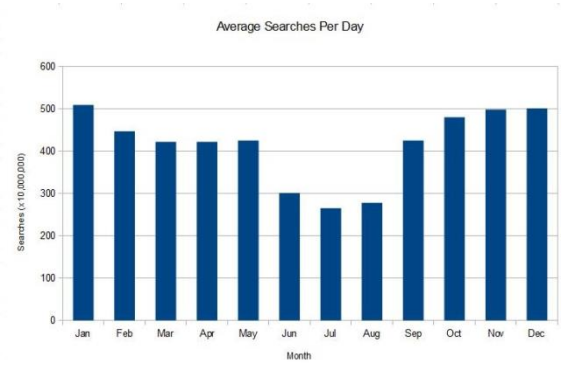




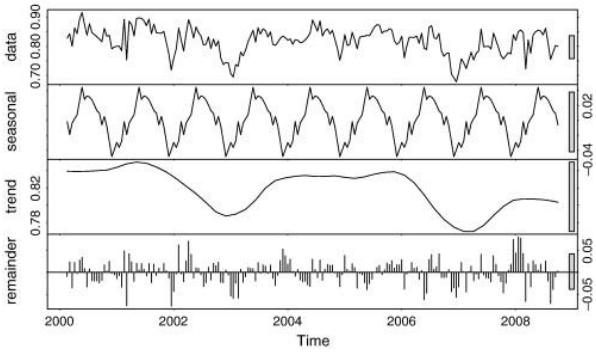
V určitých případech se tato pravidla, často z estetických důvodů, obcházejí.

Temporální data

Temporální data, tedy data mající časovou složku, se tradičně zobrazují pomocí tzv. trend (vývojových) grafů nebo sezónních grafů.



Časové úseky mohou být reprezentovány pomocí tří složek, trendové, sezónní a „zbytku“.



Příkladem nástroje, který zobrazuje temporální data, je aplikace LifeLines. Je to nástroj zobrazující medicínské záznamy pacientů.
Pro každého pacienta:

- Horizontální (časová) čára reprezentuje problém pacienta, jeho hospitalizaci a nasazenou léčbu.
- Ikony na těchto čarách reprezentují události, jako například testy či odborné konzultace.

Tímto způsobem můžeme všechny údaje o daném pacientovi zobrazit na jediné obrazovce.

Předzpracování dat

Většinou je preferováno zobrazování originálních dat bez jakýchkoliv modifikací. Hlavním důvodem je obava ze ztráty důležité informace nebo naopak přidání nežádoucích artefaktů. Příkladem je medicínská vizualizace. Zobrazení přímo hrubých dat často identifikuje problémová místa v datových záznamech, jako chybějící data nebo značně odlišné záznamy, které mohou být známkou chybného výpočtu.

Na druhou stranu pro jiné typy dat je aplikace některé z metod preprocessingu nutností.

Techniky předzpracování dat:

1. Metadata a statistiky
2. Chybějící hodnoty a „čištění“ dat
3. Normalizace
4. Segmentace
5. Vzorkování a interpolace
6. Snižování dimenze
7. Agregace dat
8. Vyhlazování a filtrace
9. Konverze rastrových dat do vektorových

1. Metadata a statistiky

Informace týkající se datových množin – metadata a statistická analýza mohou přinést neocenitelné informace pro předzpracování dat. Metadata poskytují informace, které mohou pomoct při interpretaci dat (např. formát jednotlivých záznamů). Mohou navíc obsahovat referenční bod, ze kterého vychází při různých měřeních datových polí záznamů, jednotka takového měření, symbol či číslo používané pro indikaci chybějící hodnoty a rozlišení, v jakém byla data pořízena. Takovéto informace mohou být důležité pro výběr vhodné metody předzpracování a nastavení jejích parametrů.

Různé metody statistické analýzy mohou poskytnout užitečný náhled na data.

- *Detekce chybných záznamů* (outlier detection) může indikovat data s chybnými datovými poli.
- *Analýza pomocí klastrů* (cluster analysis) může pomoci v segmentování dat do skupin podle jejich podobnosti.
- *Korelační analýza* (correlation analysis) umožňuje odstranit redundantní pole nebo zvýraznit asociaci mezi dimenzemi, která by jinak nebyla zcela zřejmá.

2. Chybějící hodnoty a „čištění“ dat

Častým jevem při analýze a vizualizaci reálných datových množin je absence některých datových záznamů nebo jejich chybná hodnota. Důvodem absence dat může být například chyba ve snímači dat nebo nevyplněné políčko v dotazníku. Důvodem chybných dat je nejčastěji lidský faktor a je obtížné je detekovat. V obou případech je však při analýze takovýchto dat nutné zvolit nějakou strategii pro vypořádání se s těmito chybami. Na některé z těchto strategií, zejména ty, které se vážou k vizualizaci, se nyní zaměříme.

1) **Odstranění špatného záznamu.** Tato zdánlivě drastická metoda, kdy odstraňujeme všechny chybějící nebo špatná pole, je ve skutečnosti jednou z nejpoužívanějších. Samozřejmě kvalita takto zobrazované informace je diskutabilní. Může totiž vést k významné ztrátě informace, protože je uváděno, že v některých odvětvích až 90% záznamů obsahuje alespoň jedno chybné pole. Přitom záznamy s chybějícími daty mohou být jinak jedněmi z nejzajímavějších.

2) **Přiřazení definované hodnoty.** Každé proměnné v datové množině určíme konstantní hodnotu, kterou jí přiřadíme v případě, že nalezneme spornou hodnotu v záznamu. Například pro proměnnou s rozsahem hodnot mezi 0 a 100 můžeme zvolit pro chybný nebo chybějící hodnotu např. -5. Poté při vizualizaci takovýchto dat na první pohled vidíme, která data byla problematická. Je zřejmé, že pro následnou statistickou analýzu je nutné tyto uměle přidané hodnoty vynechat.

3) **Přiřazení průměrné hodnoty**. Jednoduchá strategie vypořádání se s chybnými nebo chybějícími daty je jejich nahrazení průměrnou hodnotou dané proměnné nebo dimenze. Výhodou tohoto přístupu je minimální ovlivnění celkové statistiky spočtené pro danou proměnnou. Nevýhodou tohoto přístupu je, že zvolená hodnota nemusí být vždy „správná“. Další nevýhodou je skutečnost, že takto se zbavíme potenciálně zajímavých míst v datech.

4) **Přiřazení hodnoty odvozené od nejbližšího souseda**. Lepší aproximací je nalezení co nejpodobnějšího záznamu. Podobnost je přitom založena na analýze rozdílů ve všech ostatních proměnných. Princip je následující. Vezměme v úvahu záznam A, ve kterém chybí vstup pro proměnnou *i.* Dále vezměme záznam B, který je blíže k A než všechny ostatní záznamy v datové sadě, přičemž při porovnávání blízkosti bereme v potaz všechny ostatní proměnné kromě *i*. V takovém případě do proměnné *i* v záznamu A dosadíme hodnotu proměnné *i* ze záznamu B. Problémem tohoto řešení je skutečnost, že proměnná *i* může být závislá pouze na malé podmnožině ostatních proměnných (dimenzích) záznamu, proto strategie „nejlepšího souseda“ nemusí vždy najít skutečně nejlepší řešení.

5) **Spočtení náhradní hodnoty**. Velmi složitý proces, na jehož pozadí stojí dlouhodobý výzkum. Proces spočtení náhrad chybných nebo chybějících dat je znám pod názvem *imputace* (imputation) = náhrada.

3. Normalizace

Normalizace je proces transformace vstupní datové množiny takovým způsobem, aby transformovaná data vyhovovala předem daným statistickým vlastnostem. Typickým jednoduchým příkladem transformace je převod rozsahu hodnot vstupních dat pouze do intervalu [0.0, 1.0]. Jiné formy normalizace konvertují data takovým způsobem, že každá proměnná (dimenze) má střední a standardní odchylku. Normalizace je velmi užitečná operace, protože umožňuje srovnávat zdánlivě neporovnatelné proměnné. Je velmi důležitá i v procesu vizualizace, kdy jednotlivé grafické atributy nabývají pouze určitých možných hodnot, tudíž je nutné vstupní data mapovat na tyto atributy. To vede ke konverzi datového rozsahu na rozsah grafických atributů.

Například vezmeme-li d_{min} a d_{max} minimální a maximální hodnotu v daných proměnných, můžeme všechny hodnoty normalizovat do rozsahu 0.0 až 1.0 použitím vzorce:

$$d_{normalized} = (d_{original} - d_{min})/(d_{max} - d_{min})$$

V některých případech nastavujeme pro zjednodušení interpretace dat scale a offset takovým způsobem, aby odpovídaly intuitivním hodnotám minima a maxima. Příkladem může být datová sada, jejíž hodnoty spadají do procentuálního vyjádření v rozsahu 40 a 90. V takovémto případě je intuitivnější tato data naškálovat do rozsahu 0 – 100%.

Normalizace může rovněž zahrnovat ořezání podle **hraničních hodnot**, kdy hodnoty přesahující daný práh jsou uzavřeny na tento práh (je jim přiřazena hodnota tohoto prahu). To je výhodné například v případě mapování takovýchto dat na grafické atributy při vizualizaci.

4. Segmentace

V mnoha případech mohou být data rozdělena do spojitých oblastí, kdy každá oblast odpovídá určité klasifikaci dat. Například data obdržena magnetickou rezonancí mají původně 256 hodnot, kterých může daný datový bod nabývat a poté může být takovýto bod segmentován do jedné ze specifikovaných kategorií, jako například kost, sval, tuk, kůže apod. Jednoduchou segmentaci lze provést pouhým mapováním nesouvislých rozsahů datových hodnot do těchto daných kategorií. Avšak ve většině případů je takovéto mapování nejednoznačné. V takovýchto případech je nutné se podívat na klasifikaci sousedních bodů, což zlepší kvalitu celé klasifikace. Další možností je tzv.

pravděpodobnostní segmentace (probabilistic segmentation), kde je každému datovému bodu přiřazena pravděpodobnost, s jakou patří k dané kategorii.

Typickým problémem segmentace je tzv. **podsegmentování** (velké oblasti slévající nesouvisející regiony) nebo naopak **přesegmentování** (velké množství malých regionů, které spolu souvisí). Řešením je iterativní opakování *split-and-merge* procesu segmentace.

Split-and-merge

- similarThresh = definuje podobnost dvou regionů s danými charakteristikami
- homogeneousThresh = definuje homogenitu (uniformitu) regionu

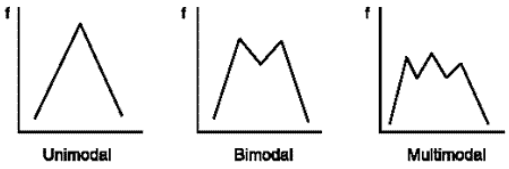
```
do {
   changeCount = 0;
   for each region {
         porovnej region se sousedními regiony a najdi nejpodobnější;
         if nejpodobnější soused leží uvnitř similarThresh aktuálního
regionu {
               spoj tyto dva regiony;
               changeCount++;
         }
         vyhodnoť homogenitu regionu;
         if homogenita regionu je menší nez homogeneousThresh  {
               rozděl region na dvě části;
               changeCount++;
   }
} until changeCount  == 0
```

Iterativní algoritmus split-and-merge funguje následujícím způsobem. Na začátku jsou stanoveny prahové hodnoty *similarThresh* definující podobnost dvou regionů a *homogeneousThresh*, který definuje homogenitu uvnitř jednoho regionu. Poté je spuštěn samotný algoritmus, který projde každý region v dané vstupní datové množině. Takovýto region porovná se sousedními regiony a podle předem definované podobnosti najde ten nejpodobnější. Pokud je tento nalezený soused v toleranci definované prahem *similarThresh*, pak jsou tyto dva regiony spojeny. Tímto způsobem tedy dochází ke slévání regionů. Druhou možností, která může nastat, je rozdělování regionu, který není dostatečně homogenní (definováno prahem *homogeneousThresh*). Při každé změně (slévání či rozdělování) je v algoritmu tato změna zaznamenána pomocí proměnné *changeCount*. Pokud k žádné změně v nové iteraci nedojde, nezmění se ani hodnota této proměnné a algoritmus končí.

Nejobtížnější části algoritmu jsou následující:

1) Určení podobnosti dvou regionů. Nejjednodušší metodou je porovnání středních (průměrných) hodnot v každém regionu.

2) Vyhodnocení homogenity regionu. Řešením může být vyhodnocení histogramu hodnot uvnitř tohoto regionu a určení, zda je **jednovrcholový** (unimodal) nebo **vícevrcholový** (multimodal).

3) 

4) Rodělení regionu. Typický algoritmus vytvoří dva (nebo více) podregionů v místech (bodech), kde jsou zjištěny nejvíce rozdílné hodnoty. Tyto body jsou pak označeny jako semínka (*seeds*) a je spuštěn proces vyplňování, kdy regiony „rostou“ až do chvíle, kdy jsou všechny body vstupní množiny někam přiděleny.

Algoritmus může být náchylný k problému vytvoření nekonečné smyčky, kdy neustále rozdělujeme a spojujeme ten samý region. Jednoduchým řešením je změna hodnoty prahu podobnosti nebo homogenity. Sofistikovanější algoritmy zahrnují do výpočtu i jiné vlastnosti regionů, jako například vyhlazení (smoothness) hranic nebo velikost a tvar regionů.

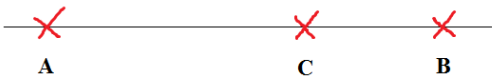
5. Vzorkování a interpolace

Často je nutné datovou množinu definovanou s jistým prostorovým rozložením transformovat do jiné datové množiny s odlišným rozlišením. Jako příklad uveďme obrázek, který chceme zmenšit či zvětšit. Jiným příkladem může být obraz, kde známe pouze několik vzorků vstupních bodů a chceme doplnit hodnoty pro body umístěné mezi těmito vzorky. V obou případech předpokládáme, že data, která máme na vstupu, reprezentují diskrétní vzorky spojitého prostoru a můžeme tedy předpovídat hodnoty v ostatních bodech pomocí prozkoumání nejbližších určených bodů. Doplnění takovýchto dat se označuje jako **interpolace**. Jde o běžně používanou metodu, která je používána v mnoha oblastech, včetně vizualizace. Nejběžnější techniky interpolace jsou následující:

- lineární
- bilineární
- nelineární

Lineární interpolace

Předpokládejme, že máme hodnotu proměnné d ve dvou místech, A a B. Pomocí lineární interpolace odhadneme hodnotu této proměnné v místě C, které se nachází mezi místy A a B.



Nejdříve spočteme procentuální umístění C mezi A a B. To je následně použito ve spojení s velikostí změny hodnoty proměnné mezi A a B a je určena hodnota d v C. Předpokládáme-li, že všechny tři body leží v ose x, platí následující rovnice:

$$(x_C - x_A)/(x_B - x_A) = (d_C - d_A)/(d_B - d_A) \text{ nebo}$$

$$d_C = d_A + (d_B - d_A)*(x_C - x_A)/(x_B - x_A)$$

To je podobné normalizaci, se kterou jsme se setkali dříve. Pro odstranění závislosti na ose x můžeme použít parametrické rovnice, které definují změnu pozice a změnu hodnoty mezi A a B. Z nich spočteme hodnotu parametru v rovnici, která definuje bod C a toto číslo se použije k výpočtu hodnoty v bodě C. Parametrický tvar rovnic je následující:

$$P(t) = P_A + Vt, \text{ kde } V = P_B - P_A$$

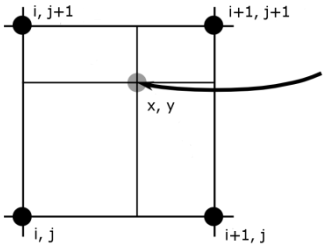
Všimněte si, že jsme nikde nedefinovali počet dimenzí prostoru, ve kterém se pohybujeme. To je díky tomu, že tyto výpočty fungují v prostoru o libovolné dimenzi.

Pokud do levé strany rovnice dosadíme PC, můžeme spočítat hodnotu t a následně ji použít ve výpočtu změny hodnoty:

$$d(t) = d_A + Ut, \text{ kde } U = d_B - d_A.$$

Bilineární interpolace

Předchozí koncept můžeme rozšířit do dvou či více dimenzí opakováním uvedené procedury pro každou dimenzi. Například, běžným úkolem v 2D prostoru je spočtení hodnoty d na pozici (x, y) v uniformní mřížce – gridu (rozložení bodů je uniformní v obou osách). To je případ 2D obrázků. Pokud pozice (x, y), pro kterou hledáme hodnotu, odpovídá nějakému bodu v gridu, pak hledanou hodnotou je přesně hodnota daného bodu mřížky. Pokud ale hledaná pozice leží mezi body v mřížce, je nutné hodnotu spočítat. K tomu je potřeba nalézt čtyři sousední body mřížky, které obklopují bod (x, y). Předpokládáme-li, že pozice bodů mřížky jsou reprezentovány celočíselnými hodnotami a vzdálenost mezi jednotlivými body je 1.0 a hledaná hodnota (x, y) má obě složky ve tvaru zlomku, pak bounding box tvořený body gridu obsahující hledaný bod (x, y) je ve tvaru (i, j), (i+1, j), (i, j+1) a (i+1, j+1), kde i je největší celé číslo menší než x a j je největší celé číslo menší než y.



Nyní budeme v tomto nalezeném prostoru interpolovat. Nejdříve provedeme horizontální interpolaci, poté vertikální. S využitím již popsané lineární interpolace spočteme procentuální vzdálenost bodu x mezi body i a i+1. Tuto hodnotu si označíme jako s. Nyní můžeme spočítat hodnotu d v pozicích (x, j) a (x, j+1) za použití hodnot ve čtyřech hraničních bodech. Podobně spočteme procentuální vzdálenost bodu y mezi body j a j+1, tu označíme jako t. Nakonec spočteme hodnotu v pozici (x, y) interpolací výše spočtených hodnot získaných z horizontální interpolace a hodnoty t, tedy:

$$d_{x,y} = d_{x,j} + t * (d_{x,j+1} - d_{x,j})$$

$d_{x,y}$ je tedy vážený průměr čtyř hraničních hodnot s ohledem na pozici bodu (x,y).

Nelineární interpolace

Jedním z hlavních problémů lineární interpolace je skutečnost, že zatímco lokální změny hodnot mají hladký přechod, změny na opačné straně gridového bodu mohou být výrazně odlišné. Ve skutečnosti je spojitost v bodu mřížky rovna hodnotě 0. Toto lze vylepšit použitím jiného typu interpolace – použitím polynomů vyššího řádu. Tím do výpočtu zahrnujeme kvadratické a kubické křivky – splajny. Jejich hlavním účelem je hladká interpolace v bodech, přičemž máme dány kontrolní body křivky a funkce pro míchání (blending). Na obrázku můžete vidět příklad hladkého napojení kvadratických a kubických Bézierových křivek.

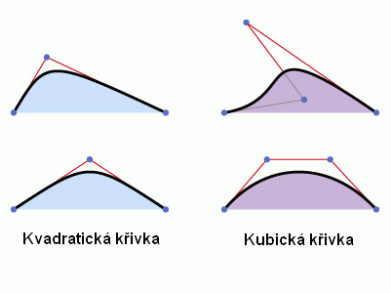
Pro připomenutí:

Spojitost C0 = koncový bod prvního segmentu je počátečním bodem segmentu druhého.

Spojitost C1 = tečný vektor v koncovém bodě segmentu je roven tečnému vektoru v jeho počátečním bodě.

Spojitost C2 = je požadována rovnost vektoru první a druhé derivace.

atd.

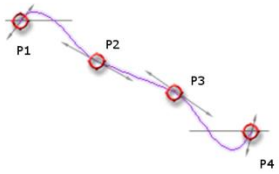


Catmull-Rom spline

Mějme 4 kontrolní body (p_0, p_1, p_2, p_3). Pak kubická Catmull-Rom křivka, která prochází těmito body, je definována viz slide. Další možná definice je:

$q(t) = 0.5 * ((2*p_1) + (-p_0 + p_2) * t + (2*p_0 - 5*p_1 + 4*p_2 - p_3) * t^2 + (-p_0 + 3*p_1 - 3*p_2 + p_3) * t^3)$

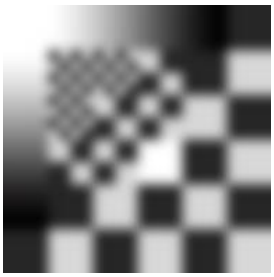
$$q(t) = 0.5 * (1.0\ t\ t^2\ t^3) * \begin{pmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{pmatrix} * \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix}$$



Výsledek aplikace interpolace pomocí Catmull-Rom křivek lze ukázat na následujícím příkladu. Úkolem je zvětšit původní obrázek o velikosti 24 x 24 pixelů. Na slajdu je porovnání výsledků při použití kubického B-spline filtru a Catmull-Rom.

Původní obrázek (24x24 pixelů)

kubický B-spline filter



Catmull-Rom

Metody převzorkování

V závislosti na hustotě vstupních dat jsou možné dvě operace: redukce velikosti vstupních dat (např. při zmenšování vstupního obrázku) nebo replikace dat za účelem zvětšení vstupní množiny dat (jako v případě zobrazeném na slajdu). **Subsampling** – první případ – může mít

velmi jednoduché řešení. Například vybereme pouze každý n-tý záznam ze vstupní datové množiny. Je ale zřejmé, že takto může lehce dojít ke ztrátě některých důležitých aspektů vstupních dat. Jako příklad uveďme výběr každého čtvrtého bodu na mapě. Nemáme zaručeno, že důležitá informace (např. cesta) není obsažena právě v odstraněných třech bodech. Kvalitnější přístupy zahrnují průměrování sousedních hodnot, výběr mediánu nebo výběr náhodných dat v subregionu definovaném kolem zkoumané hodnoty.

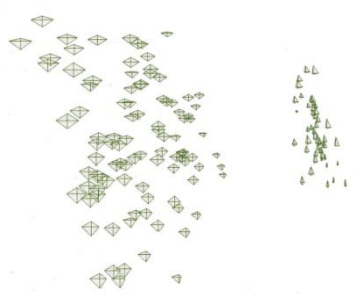
Další často používanou metodou převzorkování je „data subsetting“. Tato metoda je využívána zejména pro zpracování velkých datových množin, kdy vizualizace celé takovéto množiny může vést k nepříjemným vizuálním artefaktům (např. při přeplnění paměti apod.). Uživatel může specifikovat sadu omezení (query), která se použijí na filtraci vstupní množiny. Příkladem může být filtrace, která vrátí data pořízená pouze v určitém časovém úseku. Častým omezením je tedy nastavení prahu určitému atributu. Subsetting může být použit nejen ve fázi preprocessingu, ale i při samotné vizualizaci, kdy uživatel interaktivně pracuje se zobrazenými daty (zvýraznění, kreslení, výběr apod.). Interaktivní subsetting je obecně efektivnější než query-based subsetting, protože uživatel může přímo kontrolovat a rozhodovat o filtraci vstupních dat. Naopak výhodou query-based přístupu je skutečnost, že nemusíme celou vstupní množinu natahovat do programu.

6. Snižování dimenze

V případě, kdy rozměrnost (dimensionality) vstupních dat překročí možnosti či schopnosti dané vizualizační techniky, je nutné najít cestu, jak tuto rozměrnost zredukovat. Samozřejmě je nutné zachovat co největší množství informací obsažených ve vstupní množině. Přístupy k řešení tohoto problému lze rozdělit do dvou skupin:

1) Manuální přístup, kdy uživatel musí zadat ty dimenze, které jsou pro něj v dané situaci nepostradatelné.

2) Techniky automatického výpočtu, jako například **PCA** (principal component analysis), **MDS** (multidimensional scaling) a **SOMs** (Kohonen self-organizing maps). Všechny tyto metody jsou schopny zachovat většinu podstatných vlastností vstupní množiny, jako například klastry, vzory či kontury. Avšak výstup těchto metod je silně závislý na vstupní konfiguraci a parametrech výpočtu, což vede k jiným výsledkům.



Obrázek ukazuje výsledek redukce čtyřdimenzionální datové množiny do 2D prostoru za použití metody PCA. Jednotlivé záznamy jsou zde znázorněny pomocí piktogramu reprezentujícího 4 proměnné (jedna pro každou dimenzi). Velikost hodnoty v proměnných je dána velikostí úsečky se začátkem ve středu piktogramu a koncem v příslušném vrcholu.

PCA (Principal Component Analysis)

PCA vytváří doplňkové atributy, které jsou lineární kombinací původních datových proměnných. Tyto nové atributy definují podprostor proměnných, který minimalizuje průměrnou chybu ztracené informace. PCA má následující kroky:

1) Předpokládejme, že vstupní data mají *m* dimenzí/atributů. Od každé položky jednotlivých záznamů se odečte střední hodnota příslušné dimenze. Výsledkem je datová množina se střední hodnotou rovnou nule.

2) Spočteme kovarianční matici (čtvercová matice popisující závislost souboru náhodných veličin).

3) Spočteme vlastní vektory a vlastní hodnoty této kovarianční matice.

4) Setřídíme vlastní vektory na základě jejich vlastních hodnot – od největší po nejmenší.

5) Vybereme prvních m_r vlastních vektorů, kde m_r je počet dimenzí, na které chceme vstupní data redukovat.

6) Vytvoříme matici těchto vlastních vektorů, kdy vektory tvoří řádky matice a první vlastní vektor představuje první řádek této matice.

7) Pro každý datový záznam vytvoříme vektor jeho hodnot, transponujeme jej a přenásobíme s předchozí maticí. Tím dosáhneme výsledné transformace záznamu – každý záznam je nyní převeden do redukovaného prostoru.

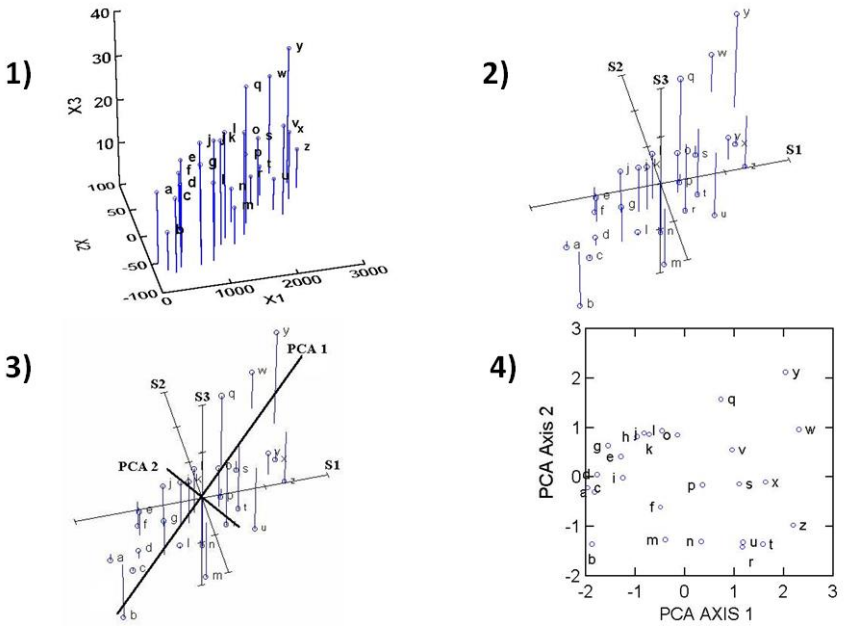
Více intuitivní popis metody PCA:

1) Vybereme přímku v prostoru, který zobrazuje n-dimenzionální data. Tato přímka pokrývá co nejvíce těchto vstupních dat. Přímka pak reprezentuje první hlavní komponentu (PC).

2) Vybereme přímku kolmou k první přímce – ta tvoří druhou PCA.

3) Opakujeme tento postup, dokud nejsou spočteny všechny PC dimenze nebo dokud není dosažen požadovaný počet hlavních komponent (PCAs).

Uveďme si příklad využití PCA. Na obrázku 1 vidíte hypotetický příklad výsledku nějakého měření tří odlišných živočišných druhů X1, X2 a X3. Na tomto příkladu se dá s menšími obtížemi vypozorovat, že druhy X1 a X2 jsou v nějakém vztahu. Nicméně už je téměř nemožné z této reprezentace vyhodnotit jakýkoliv vztah mezi X3 a zbývajícími dvěma druhy.
V první fázi (obrázek 2) provedeme rotaci datové množiny pomocí odečtení střední odchylky a podělením standardní odchylkou. To nazýváme standardizací. Tím je těžiště celé datové množiny nastaveno na nulu. Standardizované osy jsou označeny jako S1, S2 a S3. Relativní pozice jednotlivých dat zůstává stejná. Z této reprezentace už je možné odpozorovat gradient jdoucí ze spodního předního rohu do horního zadního. Je vidět, že podél tohoto gradientu se zvyšují hodnoty u druhů X1 a X2.



PCA metoda vybere první PCA přímku (osu) jako přímku procházející těžištěm, která zároveň minimalizuje čtverec vzdáleností každého z bodů k této přímce. Jinak řečeno, tato přímka je ke všem datům nejblíže, jak to jde.

Druhá PCA osa musí rovněž procházet těžištěm a musí splňovat stejné podmínky, jenom má nastaveno omezení vzhledem k její pozici vůči první PCA ose (např. musí být kolmá).

Pokud nyní orotujeme souřadný systém definovaný pomocí PCA1 a PCA2, kdy definujeme, že PCA1 odpovídá ose x a PCA2 ose y, získáme diagram na obrázku 4.

MDS (Multidimensional Scaling)

Jinou metodou pro redukci dimenze vstupních dat je multidimensional scaling (MDS), známá též jako *gradient descent approach*. MDS se snaží nalézt takovou reprezentaci vstupních dat v nižší dimenzi, která nejlépe zachovává vzdálenosti mezi jednotlivými body vstupních dat původní množiny. Jinak řečeno, cílem je pro každou dvojici bodů (i, j) redukovat rozdíl mezi $d_{i,j}$ (vzdálenost mezi i a j v původním n-dimenzionálním prostoru) a $delta_{i,j}$ (vzdálenost mezi těmito body v redukovaném prostoru). Tento rozdíl vzdáleností se označuje jako *stress*.

Algoritmus funguje následovně:

1) Spočítáme vzdálenosti všech dvojic bodů v původním prostoru. Pokud máme na vstupu n bodů, tento výpočet vyžaduje n(n – 1)/2 operací.

2) Převedeme všechny body vstupní množiny do prostoru o požadované nižší dimenzi (často náhodně).

3) Spočteme *stress*, tedy rozdíl mezi vzdálenostmi bodů v původním a novém prostoru. Tento výpočet může být proveden různými způsoby.

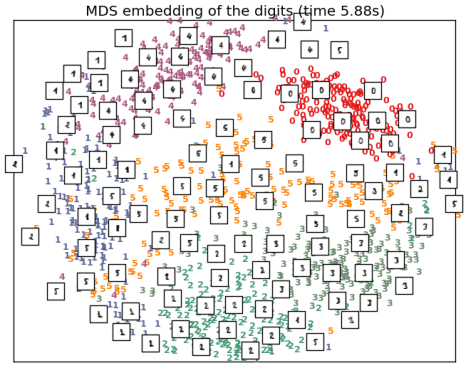
4) Jestliže průměrný nebo akumulovaný *stress* je menší než uživatelem definovaný práh, ukončíme celý algoritmus a vrátíme výsledek.

5) Jestliže je práh překročen, pak každému datovému bodu spočteme směrový vektor označující směr, ve kterém by se bod měl posunout, abychom redukovali *stress* mezi ním a všemi ostatními body. Ten je určen jako vážený průměr vektorů mezi tímto bodem a všemi jeho sousedy, jeho směr může být k nebo od souseda a váha je odvozena ze *stressu* spočteného mezi jednotlivými páry. Kladná hodnota *stress* body vzájemně oddaluje, záporná přibližuje a čím větší je absolutní hodnota *stressu*, tím větší je pohyb.

6) Na základě těchto výpočtů transformujeme datové body do cílové nižší dimenze – s ohledem na spočtené vektory. Vrátíme se ke kroku 3 algoritmu.

Potenciální problémy tohoto algoritmu jsou vytvoření nekonečné smyčky či uvíznutí v nelokálním optimu. Tomu se lze vyhnout opakováním algoritmu s různými vstupními parametry nebo povolení „nestandardního“ pohybu v jiném směru, než byl spočtený směrový vektor.

Mnoho vizualizačních a statistických grafických balíků využívá obou metod – PCA i MDS. Některé dokonce využívají kombinaci obou, kdy je nejdříve použit PCA pro spočtení iniciálních pozic bodů, na které je pak aplikována metoda MDS. To vede k velké redukci počtu iterací potřebných pro dosažení konfigurace se nejnižší hodnotou *stress.*



Praktickým příkladem použití algoritmu MDS může být reorganizace fotografického alba podle příbuznosti – v tomto případě je příbuznost definována barvou. Obrázek je výstupem aplikace Yorg (http://lear.inrialpes.fr/src/yorg/doc/index.html).

Mapování nominálních hodnot na ordinální

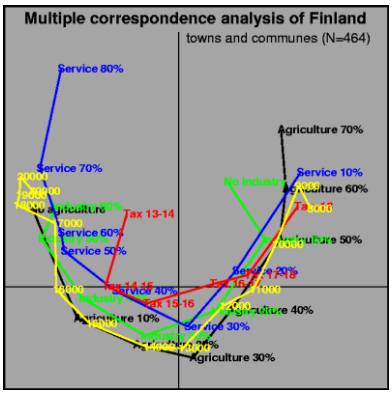
Je mnoho oblastí, kde jedna nebo více dimenzí datové množiny je obsahuje nominální (jmenné) hodnoty. Pro zpracování takovýchto dat může být zvoleno z několika strategií, záleží na vlastnostech těchto dat – počet takovýchto dimenzí, jak široké škály hodnot mohou hodnoty nabývat nebo zda je možné hodnoty nějak třídit nebo definovat jejich vzdálenost. Klíčem je najít způsob mapování dat na grafické entity nebo atributy, které do dat nevnáší vztahy, které se v původních datech vůbec nevyskytují.

Jako příklad si vezměme databázi aut. Datová množina obsahující informace o jednotlivých autech obsahuje nominální hodnoty „výrobce“ a „model auta“. Jak bychom tyto hodnoty měli namapovat do grafu? Možným způsobem je přiřazení celočíselné hodnoty každé z nominálních hodnot – třeba podle abecedního uspořádání. To ale může zavést mylné vztahy, kde například Honda bude v grafu blíže k Fordu než k Toyotě. Přitom žádný takový vztah ve skutečnosti neexistuje.

Pokud se ve vstupní datové množině vyskytuje jedna nominální proměnná, existuje několik možných technik, které můžeme využít. Nejjednodušší je využití této proměnné jako značky

(label) daného grafického elementu, který zobrazujeme. Tato metoda je vhodná pro malé datové množiny, při jejím nárůstu se rychle stává nepoužitelnou. Moderní techniky jsou schopny tuto metodu lokalizovat, což znamená například zobrazení labelu pouze u proměnných v blízkosti kurzoru a podobně.

Při hledání metody pro mapování nominální proměnné na číslo se můžeme řídit podobností mezi numerickými proměnnými, které jsou s těmito nominálními spojeny do jednoho datového záznamu. Pak je zřejmé, že záznamy s podobnými vlastnostmi by měly být nominální hodnoty těchto záznamů mapovány na podobnou číselnou hodnotu. Pokud máme vyhodnocenu podobnost mezi všemi dvojicemi datových záznamů, můžeme použít například techniku MDS a namapovat nominální hodnoty na pozice v jedné dimenzi. Toto je zjednodušený případ techniky zvané **korespondenční analýza** (correspondence analysis), která je používána ve statistice. Tato technika může být použita i v případech, kdy jsou všechny dimenze vstupní datové množiny nominální. Pak se tato technika nazývá **násobná korespondenční analýza** (multiple correspondence analysis).

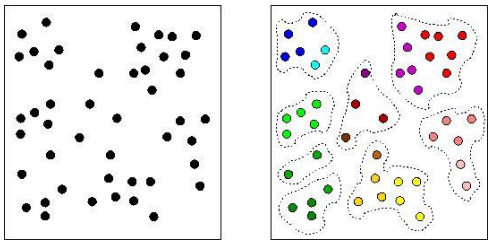


7. Agregace dat

V případě, kdy zpracováváme velké množiny vstupních dat, je velmi užitečné tato data shlukovat do klastrů podle podobnosti jejich hodnot a/nebo pozice. Tyto skupiny jsou pak reprezentovány mnohem menším množstvím dat. Toho lze dosáhnout obyčejným zprůměrováním hodnot, nové záznamy však mohou obsahovat i více popisných dat, jako například počet členů jednotlivých skupin nebo rozsah jejich pozic nebo hodnot. Tomuto postupu se říká **agregace** (**shlukování**).

Metoda agregace má tedy dvě komponenty: metodu shlukování bodů a metodu zobrazení výsledných skupin. Shlukování lze provést mnoha způsoby, jako například slučování sousedních bodů, dělení prostoru nebo iterativní split-and-merge metody. Důležitým aspektem všech metod je výpočet vzdálenosti mezi datovými body a kvalita klastrování včetně definice oddělení jednotlivých klastrů.

Při vizualizaci klastrů je nutné uživateli předložit dostatek informací k tomu, aby se mohl rozhodnout, zda potřebuje jednotlivé klastry blíže zkoumat či nikoliv. Pouhé zobrazení jednoho reprezentanta daného klastru většinou nestačí k pochopení variability dat uvnitř klastru.



8. Vyhlazování a filtrace

Při zpracování signálu je běžným procesem vyhlazování vstupních dat. Cílem je redukce šumu a rozmazání ostrých nespojitostí. Typicky je vyhlazování prováděno pomocí tzv. **konvoluce**. Pro naše účely postačí, pokud se na konvoluci budeme dívat jako na vážený průměr sousedů obklopujících daný bod. V jednodimenzionálním prostoru je možné konvoluci aplikovat pomocí následujícího vzorce:

$$p_i = \frac{p_{i-1}}{4} + \frac{p_i}{2} + \frac{p_{i+1}}{4}$$

kde každé p_i je zpracovávaný bod.

Po aplikování této operace na daný bod je jeho původní hodnota, která se významně lišila od jeho sousedů, nahrazena hodnotou výrazně podobnější sousedním. Změnou vah nebo tvaru a velikosti uvažovaného okolí můžeme měnit výsledný obraz.

9. Konverze rastrových dat do vektorových

V počítačové grafice jsou objekty typicky reprezentovány pomocí sady polygonů tvořených vrcholy a hranami. Úkolem je vytvořit rastrovou reprezentaci takovýchto objektů na úrovni pixelů, nadefinovat vlastnosti jejich povrchu, definovat jejich interakci se světlem a s jinými objekty.

V určitých případech je výhodné extrahovat lineární struktury z rastrové datové množiny (např. z obrázku). Důvody pro tuto konverzi mohou být následující:

- Komprese obsahu, např. pro přenos. Seznam vrcholů a hran je téměř vždy kompaktnější vyjádření než rastrová reprezentace.

- Porovnání obsahu dvou a více obrázků. Je jednodušší porovnávat atributy vyššího řádu než jen jednotlivé pixely.

- Transformace dat. Afinní transformace, jako například rotace a scaling, je jednodušší aplikovat na vektorovou reprezentaci než na rastr.

- Segmentace dat. Izolování regionů zvýrazněním jejich hranice je účinným nástrojem interaktivní evaluace a vytváření modelů.

Zpracování obrazu a počítačové vidění jsou oblasti, ve kterých byla vyvinuta řada technik pro konverzi rastrových obrázků do jejich vektorové podoby. Jmenujme některé z nich:

- **Prahování** (thresholding). Principem je definice jedné nebo několika mezních hodnot, pomocí kterých je následně vstupní množina rozdělena do několika regionů. Po určení jejich hranic mohou být generovány jednotlivé hrany a vrcholy. Hodnoty prahů mohou být definovány uživatelem nebo spočteny na základě analýzy histogramu obrázku. Adaptivní prahování umožňuje přiřadit daný práh pouze dané oblasti obrázku.
- **Narůstání regionu** (region growing). Začínáme od tzv. semínek v obrázku, která jsou definována uživatelem nebo spočtena pomocí skenování dat. Pomocí semínkového vyplňování slučujeme pixely do klastrů podle jejich podobnosti. Hlavním problémem je určení vhodné definice podobnosti pixelů.
- **Detekce hranice** (boundary-detection). Provede se pomocí konvolucí vstupního obrázku pomocí příslušné matice – konvolučního jádra. Každý pixel a jeho sousedé jsou vynásobeni příslušnou hodnotou v konvolučním jádře odpovídajícím jejich poloze vůči zpracovávanému pixelu ve středu konvolučního jádra. Tyto násobky jsou následně sečteny a hodnota je přiřazena zpracovávanému pixelu. Díky různým podobám konvolučního jádra lze provést různé transformace obrazu. Pro detekci hranice používáme konvoluční matici, která zvýrazňuje horizontální, vertikální nebo diagonální hraniční čáry a naopak potlačuje pixely, jejichž hodnota je velmi podobná těm v jejich sousedech.
- **Ztenčování** (thinning). Rovněž je využit konvoluční proces, cílem je redukovat široké lineární čáry, jako například tepny, do čar o šířce jednoho pixelu. Tyto výsledné pixely tvoří střední osu regionů, které byly ztenčeny.

Všechny techniky uvedené v této přednášce slouží ke zvýšení efektivity jejich vizualizace a mohou vést k objevení nových skutečností skrývajících se v těchto datech. Je však třeba si uvědomit, že cílového uživatele je třeba informovat, že data byla tímto způsobem upravena. Pochopení jednotlivých typů transformací, které byly aplikovány na data, mohou významně pomoct při jejich interpretaci.