PV251 Vizualizace

Jaro 2017

Výukový materiál

---

4. přednáška: Vizualizace prostorových dat

Vizualizace prostorových dat, do které spadá i celá oblast vizualizace vědeckých dat (scientific visualization), předpokládá, že vstupní datové sady implicitně obsahují prostorové či časoprostorové atributy. Tato skutečnost významně napomáhá vytváření a interpretaci vizualizací takovýchto dat, protože mapování datových atributů na ty grafické je intuitivní a většinou přímočaré. Náš zrakový systém neustále přijímá a interpretuje obrazy fyzikálních jevů, které nás obklopují. Proto je pro člověka poměrně přirozené zpracovávat podobným způsobem obrazy na obrazovce.

Hlavní rozdíly mezi vnímáním okolního světa a prostorových dat na obrazovce jsou:

- Při pozorování skutečného světa nejsme omezeni dvoudimenzionální, diskrétní projekcí s nízkým rozlišením.
- Pomocí obrazovky můžeme vizuálně prozkoumat různé reálné i simulované jevy v různém měřítku.
- Na obrazovce jsme schopni dynamicky měnit kontrast, nasvětlení, rozlišení, hustotu a ostatní parametry zobrazovaných dat.
- Na obrazovce jsme schopni interaktivně prozkoumávat místa, která jsou v reálném světě velmi obtížně dostupná.
- Na obrazovce můžeme interaktivně přidávat a odstraňovat části dat za účelem zlepšení informace o kontextu dat nebo odstranění nesmyslných dat.

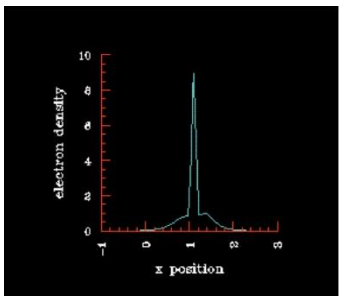
Mapování atributů

Při vizualizaci prostorových dat musíme nejdříve rozhodnout, které prostorové atributy dat budeme mapovat na prostorové atributy (umístění) na obrazovce. Tato fáze v sobě obsahuje různé druhy transformací, jako například scale, rotaci, translaci, zkosení a projekci. Poté je potřeba namapovat i zbývající atributy vstupních dat na další vizualizační komponenty. Mezi tyto atributy patří například barva či textura, ale také velikost a tvar grafických entit.

1D data

Jednodimenzionální data často získáme pomocí vzorkování fyzikálního jevu při pohybu po křivce v prostoru.

Mějme jednodimenzionální sekvenci dat o jedné proměnné. Pak můžeme mapovat prostorová data na jednu z dimenzí (os) obrazovky a samotné hodnoty dat vyneseme buď do druhé dimenze obrazovky (vytvoření čárového grafu) nebo na barvu značky nebo regionu podél první osy (color bar – barevná škála). Data musí být škálována, abychom je mohli zobrazit v rozsahu dimenzí obrazovky (počet pixelů) nebo v rozsahu atributů obrazovky (např. počet podporovaných barev). Části displeje mohou být rezervovány pro podpůrné vizualizační prvky, jako například zobrazení os, značek (labels), atd.



Algoritmus pro vykreslení 1D dat

Předpokládejme, že pro vstupní datovou množinu máme spočteny hodnoty ($data_{min}$, $data_{max}$), které definují minimální a maximální hodnotu v těchto datech a proměnná $data_{count}$ určuje počet bodů (dat), které mají být zobrazeny (mohou to být všechna data nebo jen určitá podmnožina). Dále předpokládejme, že část obrazovky, na které budou data zobrazena, je obdélníková oblast definovaná pomocí (x_{min}, y_{min}, x_{max}, y_{max}). Pro vykreslení čárového grafu z bodů pole označeného jako „data“ lze použít následující kód.

```
DRAW-LINE-GRAPH(data, dataCount, xMin, xMax, yMin, yMax)
1. dataMin <- computeMin(data, dataCount)
2. dataMax <- computeMax(data, dataCount)
3. xFrom <- xMin
4. yFrom <- worldToScreenY(data[0], dataMin, datamax, yMin, yMax)
5. for i<- 1 to dataCount
6.       do xTo <- worldToScreenX(i, dataCount, xMin, xMax)
7.             yTo <- worldToScreenY(data[i], dataMin, dataMax,
                   yMin, yMax)
8.             drawLine(xFrom, yFrom, xTo, yTo)
9.             xFrom <- xTo
10.            yFrom <- yTo
worldToScreenX(index, dataCount, xMin, xMax)
         return (xMin + index * (xMax - xMin)/dataCount)
worldToScreenY(value, dataMin, dataMax, yMin, yMax)
```

```
          return (yMin + (value – dataMin) * (yMax – yMin)/(dataMax
/        dataMin))
```

Funkce worldToScreenX je jednodušší než funkce worldToScreenY, protože předpokládáme, že pole hodnot je indexováno od 0. Pokud bychom tento předpoklad chtěli odstranit a funkci zobecnit, vypadala by následovně:

```
worldToScreenX(index) {
   return (xMin + (index – indexMin) * (xMax – xMin) / (indexMax –
   indexMin));
}
```

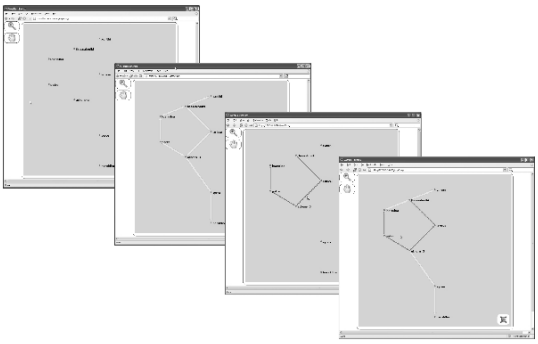
Celá funkce pro vykreslení grafu může být různými způsoby vylepšena. Například, pokud počet vstupních bodů převyšuje počet dostupných pixelů, můžeme vstupní body předzpracovat (např. vzorkovat, průměrovat, …). Můžeme rovněž změnit měřítko zobrazených dat za účelem lepšího zprostředkování dané informace.

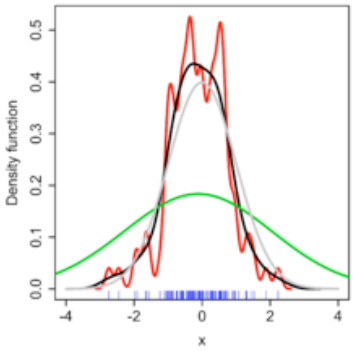
Podobný algoritmus lze aplikovat i na vykreslení dat v podobě barevného sloupce (color bar), kdy jednoduše nahradíme kreslení čar kreslením obdélníků (nejčastěji uniformní velikosti), jejichž barva je úměrná hodnotě zobrazovaných dat a výsledek je transformován do rozsahu dostupných barev poskytovaných výstupním zařízením.

1D multivariate data

Pokud jsou vstupní data označena jako multivariate (obsahují více proměnných nebo více hodnot pro jeden datový vstup), je možné předchozí techniku rozšířit, aby byla schopna takováto data zpracovat. Toho se dá dosáhnout pomocí přikládání k sobě (juxtapositioning) či překládání přes sebe (superimpositioning). Pro čárové grafy to znamená, že vizualizace bude obsahovat sadu izolovaných grafů (zejména v případech, kdy mají proměnné různá měřítka) nebo jediný graf vykreslující data o dvou a více proměnných. V tomto případě je vhodné vykreslit jednotlivé čáry jiným stylem a/nebo jinou barvou, abychom jednotlivá data mohli odlišit.

Podobně lze skládat i sloupcový graf pro barvu. Zde je intuitivnější využití přístupu, kdy vrstvíme jednotlivé sloupce na sebe.





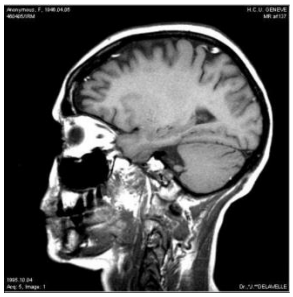
2D data

Data obsahující dvě prostorové dimenze jsou zobrazovány převážně pomocí mapování prostorových atributů dat na prostorové atributy obrazovky. Výsledek může být jeden z následujících typů vizualizací:

1. Obrázek (image)
2. Reliéf (rubber sheet)
3. Cityscape
4. Bodový graf (scatterplot)
5. Mapa
6. Kontura, mapa izobar

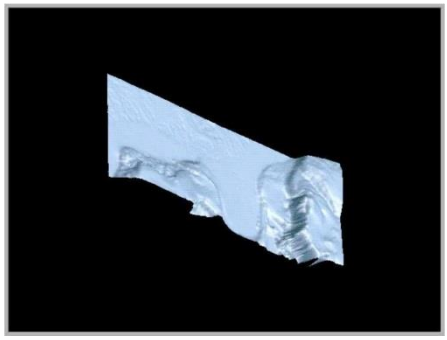
Obrázek

Obrázek jako výsledek vizualizace 2D dat vznikne, pokud jedna datová hodnota v každé pozici je mapována na barvu a všechny mezilehlé pixely jsou obarveny pomocí interpolace. Příkladem je obrázek získaný z tomografie.



Reliéf (rubber sheet)

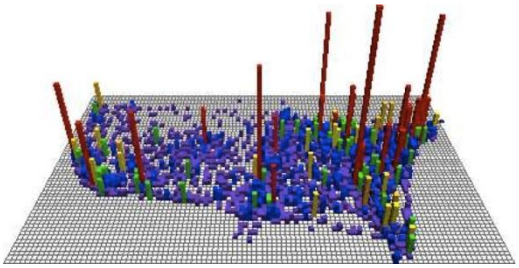
Mějme vstupní data, která mohou být pravidelně nebo i nepravidelně rozmístěna v prostoru. Pokud tato data mapujeme na výšku příslušného bodu ve třídimenzionálním prostoru a navíc jsou body triangulovány takovým způsobem, že vzniká povrch, hovoříme o vizualizaci reliéfu, povrchu (v angličtině rubber sheet).



Příkladem je daný obrázek, který znázorňuje mořskou hladinu a její přechod na pevninu na Floridě.

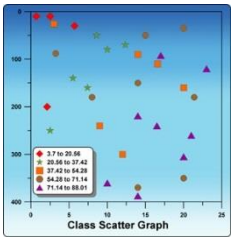
Cityscape

Tento typ vizualizace vzniká tak, že vykreslujeme 3D objekty (obecně kvádry) na různá místa v rovině. Jednotlivá data pak řídí atributy těchto grafických objektů (např. výšku, barvu). Obrázek znázorňuje příklad takovéto vizualizace, kdy vidíme hustotu leteckého provozu nad Spojenými Státy v určitém časovém okamžiku.



Bodový graf (scatterplot)

Klasické zobrazení, kdy při každém umístění dat do grafu jednotlivé datové hodnoty ovlivňují barvu, tvar či velikost jednotlivých značek. Je dobré si uvědomit, že na rozdíl od obrázku zde nedochází k žádné interpolaci.



Mapa

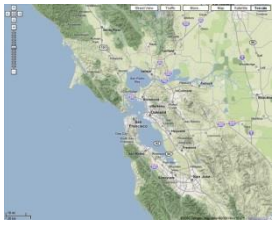
Výsledkem vizualizace je mapa, pokud vstupní data obsahují lineární a plošné vlastnosti, ale i bodové objekty.

Za lineární vlastnost můžeme považovat cestu či řeku a je reprezentována jako sekvence spojených souřadnic, které jsou vykresleny jako úsečkové segmenty.

Plošné vlastnosti, jako například jezero nebo plocha státu definovaná jeho hranicí, je reprezentována pomocí uzavřené křivky – sada souřadnic, kde souřadnice prvního a posledního bodu jsou shodné. Jsou většinou zobrazeny jako polygon, který může být vyplněn barvou, texturou či opakujícím se symbolem.

Bodové objekty (například stožáry vysokého napětí, škola) jsou obecně reprezentovány určitým symbolem umístěným do dané lokace. Vzhledem k možnosti překrytí některých symbolů se často volí technika zobrazení, kdy symbol neleží přesně na odpovídajícím místě v mapě, ale je mírně posunut.

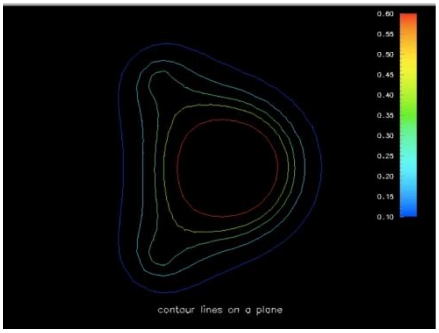
Jednotlivé objekty většinou mají svá označení (labels), což opět komplikuje celý proces zobrazení mapy. Více se o tzv. geovizualizaci dovíme na příští přednášce.



Kontury, izobary

Tento typ vizualizace zobrazuje hraniční informaci, která je odvozena z obrázku a představuje určitý spojitý jev, jako například nadmořskou výšku nebo teplotu. Anglicky se rovněž označuje termínem isovalue (např. izobara), což znamená „jedna hodnota“, proto kontura v mapě určuje hranici mezi body nad touto hodnotu a pod ní. Každá „isovalue“ může generovat několik uzavřených kontur. Násobné izobary mohou být vykresleny najednou, přičemž k jejich rozlišení se používá barva, tloušťka čáry či označení.

Obrázek ukazuje rozložení několika kontur získaných z 2D obrázku molekuly vody. Hodnoty izobar jsou rozlišeny barvou.



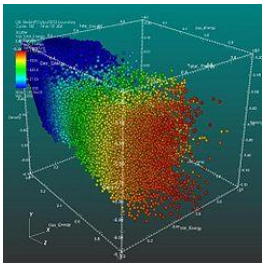
2D multivariate data

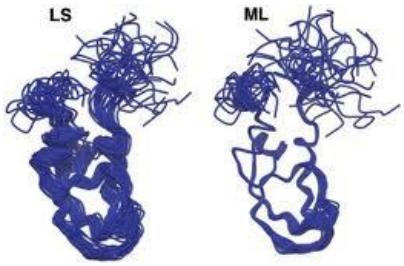
Podobně jako u 1D dat, můžeme tyto techniky rozšířit na data s větším množstvím proměnných (multivariate data) pomocí juxtapositioning a superimpositioning.

Juxtapositioning – jednoduché naskládání několika 2D vizualizací jedné proměnné do 3D vizualizace. Tento přístup umožňuje zobrazení náhledu na veškerá data najednou, nicméně díky překrytí může být obtížné detekovat vztahy mezi daty. Navíc je tato technika limitována počtem proměnných, které mohou být ještě srozumitelně zobrazeny, zvláště u technik, které již třetí dimenzi využívají (například reliéf nebo cityscape).

Superimpositioning – podobně limitován počtem proměnných, které lze takto zobrazit. Například pro reliéfy vykreslujeme jednotlivá data pomocí průsvitných povrchů. Pro mapy rovněž není příliš vhodné nadměrné překrývání, protože znesnadňuje extrahování jednotlivých komponent.

Alternativní přístup k zobrazení multivariate 2D dat je využití „neprostorové multivariate“ vizualizace, kterou se budeme zabývat v pozdějších přednáškách.



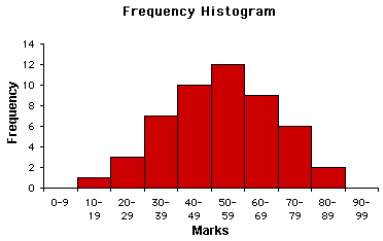


Kromě vizualizačních technik, které zobrazují celé datové množiny, můžeme vizualizovat pouze jejich jednodimenzionální podmnožiny, projekce nebo různé sumarizace dat. Poté již můžeme použít libovolnou z předchozích technik. Nyní se tedy zaměříme na některé z možností projekce:

- Frekvenční histogramy
- Slučování řádků a sloupců
- Lineární „sondy“ (probes)

Frekvenční histogramy

Sumarizaci datových množin o libovolné dimenzionalitě je možné provést výpočtem frekvence, ve které se dané hodnoty nebo intervaly hodnot objevují v datech. Výsledek je pak zobrazen v podobě sloupcového grafu. Pokud pracujeme s podmnožinami vstupních hodnot, musíme velmi pečlivě volit počet jednotlivých podmnožin a jakým způsobem jsou od sebe odlišeny. Při špatné volbě je totiž možná ztráta důležitých vlastností dat. V tomto případě navrhne nejlepší rozdělení dat osoba, která je s obsahem dat důvěrně seznámena. Na druhou stranu často používané jednoduché dělení na fixní počet stejně velkých podmnožin není v tomto případě velmi efektivní.



Slučování řádků a sloupců

Užitečný mechanismus pro lokalizaci hranic oblastí zájmu a regionů s nízkou nebo vysokou variabilitou je technika zobrazující sumarizace řádků a/nebo sloupců obrázku. Pro tento účel se využívají různé techniky, jako například součet, průměr, medián, standardní odchylka, maximum nebo minimum z hodnot. Výsledná 1D vizualizace pak může být zobrazena samostatně nebo častěji zobrazena v kombinaci s 2D vizualizací jako doplňková informace.

Pro tuto techniku se nejčastěji využívají barevné sloupce, čárové grafy a sloupcové grafy.

Lineární „sondy“

Jednodimenzionální sonda (probe) pro 2D datové množiny může být přirovnána k vyvrtané díře v nerostu. Skrz vstupní data prochází přímka a data, která tato přímka protne, jsou zobrazena pomocí některé z předchozích technik pro zobrazení 1D dat. Podobně jako vrták vrtající díru do nerostu. Abychom tohoto mohli dosáhnout, využijeme dva matematické nástroje: parametrické rovnice a bilineární interpolaci.

Začneme definicí sondy pomocí parametrické rovnice. Definice vychází ze vstupních údajů uživatele (buď dvojice bodů nebo bod a směrový vektor). Předpokládáme-li, že přímka je definována dvěma body P_1 a P_2 (jejich souřadnicemi), pak parametrická rovnice přímky je definována jako

$$P(t) = P_1 + t(P_2 - P_1), \text{ kde } 0.0 <= t <= 1.0$$

Nyní můžeme získat souřadnice libovolného bodu na této přímce zvolením hodnoty t. Obecně jsou tyto body na přímce rovnoměrně rozloženy a počet bodů je určen úměrně k délce přímky. Pokud již máme souřadnice navzorkovaných bodů, můžeme pomocí interpolace spočítat jejich hodnoty, které poté mohou být vizualizovány jako 1D datová množina.

3D data

Podobně jako u 2D dat, 3D prostorová data mohou být ve formě diskrétních vzorků spojitého jevu nebo jako struktura popsána vrcholy, hranami a polygony. Ve skutečnosti většina

vědeckých a technických vizualizací obsahují kombinaci těchto dvou reprezentací, jako například proudění vzduchu kolem křídla letadla. Nejdříve se zaměříme na prozkoumání základních technik pro vizualizaci těchto typů dat a poté probereme metody, které tyto techniky kombinují.

Mezi základní techniky pro vizualizaci 3D dat patří:

- Vizualizace explicitních povrchů
- Vizualizace objemových dat
- Implicitní povrchy

Vizualizace explicitních povrchů

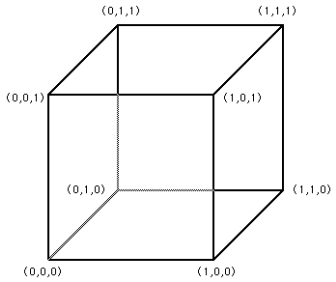
Explicitní povrch je takový povrch, který je definován jedním z následujících způsobů:

1. Seznam 3D vrcholů, seznam hran mezi nimi (specifikovány jako dvojice indexů do seznamu vrcholů), seznam planárních polygonů (obvykle specifikovány jako fixní nebo variabilní seznam indexů do seznamu hran).
2. Sada parametrických rovnic definujících x, y, z souřadnice bodů na povrchu společně se strategií jejich spojování (například triangulace), pomocí které vznikají hrany a polygony. Pomocí určení kroku parametru v rovnicích můžeme ovlivňovat hladkost povrchu.

Příklad 1:

Uveďme si příklad jednotkové kostky, která může být reprezentována následujícím způsobem. Všimněme si, že každá hrana je sdílena právě dvěma hranami a každý vrchol je součástí tří hran. Každá stěna má navíc definovánu přední a zadní stěnu pomocí orientace hran (po směru nebo proti směru hodinových ručiček). Tato podmínka je významná zejména pro korektní výpočet povrchových normál.

```
vertex[0] = (0., 0., 0.)
vertex[1] = (0., 0., 1.)
vertex[2] = (0., 1., 1.)
vertex[3] = (0., 1., 0.)
vertex[4] = (1., 0., 0.)
vertex[5] = (1., 0., 1.)
vertex[6] = (1., 1., 1.)
vertex[7] = (1., 1., 0.)
edge[0] = (0, 1)
edge[1] = (1, 2)
edge[2] = (2, 3)
edge[3] = (3, 0)
edge[4] = (0, 4)
edge[5] = (1, 5)
edge[6] = (2, 6)
edge[7] = (3, 7)
edge[8] = (4, 5)
```



edge[9] = (5, 6)
edge[10] = (6, 7)
edge[11] = (7, 4)
face[0] = (0, 1, 2, 3)
face[1] = (8, 9, 10, 11)
face[2] = (0, 5, 8, 4)
face[3] = (1, 6, 9, 5)
face[4] = (2, 7, 10, 6)
face[5] = (3, 4, 11, 7)

Příklad 2:

Dalším příkladem je parametrická definice jednotkového válce umístěného v ose y.

$y = 1.0, \quad x = \cos \Theta, \quad z = \sin \Theta,$

$0.0 \leq \Theta \leq 2\pi$ (horní podstava)

$y = 0.0, \quad x = \cos \Theta, \quad z = \sin \Theta,$

$0.0 \leq \Theta \leq 2\pi$ (dolní podstava)

$y = h, \quad x = \cos \Theta, \quad z = \sin \Theta,$

$0.0 \leq \Theta \leq 2\pi, \; 0.0 \leq h \leq 1.0$ (plášť)

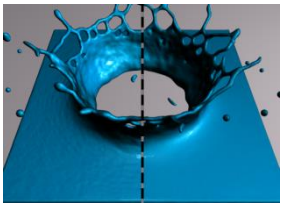
Manipulací s parametrem Θ můžeme měnit hladkost povrchu válce. Pokud nastavíme hodnotu h na 1 a hodnotu Θ na π/2, pak získáme kostku o výšce 1.

Dalšími příklady parametrických povrchů, se kterými se běžně setkáváme v počítačové grafice, jsou Bézierovy křivky a B-splajny.

Obecně, vizualizace prostorových dat pomocí explicitních povrchů závisí na tom, zda jsou vstupní data asociována s vrcholy, hranami nebo stěnami. Příklady pro každý z těchto případů jsou:

- Teplota nebo váha uzlu (informace o vrcholech)
- Síla chemické vazby (informace o hranách)
- Pokrytí oblasti mapou (informace o stěnách)

Zobrazovaná informace může být mapována na některý z neprostorových grafických atributů – barva, průhlednost, textura atd.



Obrázek pochází z článku Explicit Mesh Surfaces of Particle Based Fluids publikovaný na konferenci Eurographics 2012.

Vizualizace objemových dat

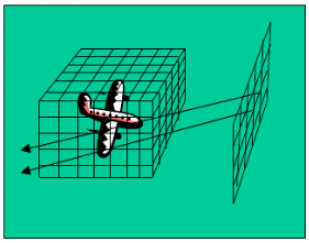
Podobně jako jsou pro 2D vizualizaci využívány pixely, pro 3D vizualizace používáme voxely (volume elements). Objemová (volumetrická) data obecně vznikají navzorkováním spojitého jevu a mohou být nasnímána pomocí senzorů (např. tomografie) nebo generovány různými simulacemi (např. CFD – computational fluid dynamics). V obou případech dostáváme několikadimenzionální data s pravidelně či nepravidelně rozloženými pozicemi a cílem je uživateli zprostředkovat informaci o struktuře, opakujících se vzorcích a anomáliích v těchto datech.

Většina přístupů k vizualizaci objemových dat spadá do jedné z následujících kategorií:

- „Plátování“ (slicing) – využití ořezávací roviny, která může být osově zarovnána nebo může mít libovolnou orientaci. Výsledkem je 2D plát vystřižený z dat, který můžeme následně zobrazit pomocí některé z 1D nebo 2D vizualizačních technik.

- Izopovrchy (isosurfaces) – uživatel specifikuje vstupní parametry, podle kterých je generován popis povrchu datové sady a je vizualizován pomocí jedné z technik vizualizace explicitního povrchu.

- Přímé renderování objemu (direct volume rendering) – dvě metody: první je založena na vrhání paprsku do scény obsahující objemové těleso a výpočet hodnot v jednotlivých pixelech na zákaděvoxelů, které paprsek protnul. Druhá metoda promítne každý voxel na projekční rovinu za použití některé z metod akumulace efektů voxelů na pixely.



Převzorkování

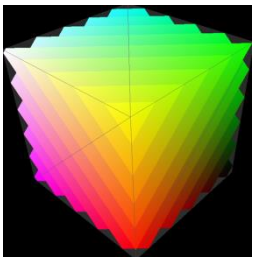
Pro všechny výše zmíněné techniky vizualizace objemových dat je základním prvkem iniciální převzorkování.

Například pro izopovrchy musíme nalézt místa, kde data odpovídají zvolené „izohodnotě“, což je ve většině případů někde mezi body původní datové množiny – proces velmi podobný vytváření kontur ve 2D.

Při plátování, zvláště pokud není ořezávací rovina osově zarovnána se souřadnou soustavou, je nutné vstupní data převzorkovat, abychom získali rovnoměrně rozloženou sadu pixelů. Pokud jsou navíc vstupní data rozložena nerovnoměrně, je nutné navíc využít interpolaci.

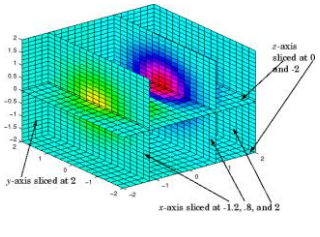
U techniky přímého renderování objemu musíme navzorkovat data podél paprsků, což opět implikuje převzorkování. Pouze při využití paralelní projekce podél hlavních os není převzorkování potřeba.
Je tedy zřejmé, že proces převzorkování hraje v drtivé většině technik pro vizualizaci objemových dat velmi významnou roli.

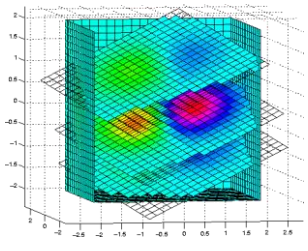
Plátování objemových dat ořezávacími rovinami
Podobně jako u vizualizace 2D dat můžeme využít techniku využití sondy pro vytvoření podmnožiny vstupních dat o nižší dimenzi. Populární technika pro objemová data založená na tomto principu je použití ořezávacích rovin, kdy je datový blok „nařezán“ rovinou dané pozice a orientace a data, která rovina protíná, jsou mapována na obrazovku.



Nejjednodušší implementace této techniky omezuje orientaci ořezávací roviny takovým způsobem, že normála roviny se shoduje s jednou z hlavních os. Uživatel poté specifikuje řádek, sloupec či hloubku v datovém bloku a odpovídající plát je následně zobrazen použitím jedné z technik pro vizualizaci 2D prostorových dat, se kterými jsme se seznámili dříve.
Pokud odstraníme omezení na orientaci ořezávací roviny, pak každý voxel, který je protnut ořezávací rovinou, může ovlivnit hodnotu v jednom nebo i více pixelech. V tomto případě se můžeme rozhodnout převzorkovat vstupní datovou množinu na místech, kde se protíná s ořezávací rovinou a to tak, že na ořezávací rovinu umístíme pravidelnou mřížku. Alternativně můžeme vybrat voxel nejbližší pixelu ořezávací roviny, který pak reprezentuje objem v tomto místě roviny. Dalším přístupem je kombinace hodnot v nejbližších voxelech k danému pixelu na ořezávací rovině, přičemž jejich příspěvek se počítá jako váha přiřazená na základě vzdálenosti středu voxelu od ořezávací roviny.
Pro specifikaci ořezávací roviny musíme nastavit šest parametrů – tři poziční a tři pro definici normály roviny.
Existují mnohé variace této techniky, přičemž každá z nich se soustředí na specifické detaily uvnitř objemových dat. Příkladem může být:

- Nerovinné pláty
- Sekvence plátů s různými orientacemi
- Pláty naskládáné „na sobě“, zobrazeny souběžně
- Pláty umístěné ortogonálně, zobrazeny souběžně

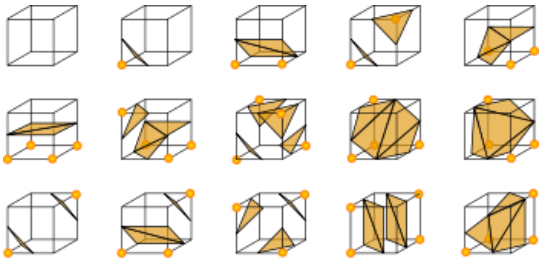




Získání izopovrchu pomocí Marching Cubes

Algoritmus Marching Cubes (algoritmus pochodujících kostek) vznikl v roce 1987 a jeho autory jsou Lorensen a Cline a je zaměřen na renderování izopovrchů objemových dat. Podobná technika vznikla již o rok dříve (Wyvill a spol.), nicméně pochodující kostky se staly nejpopulárnějším algoritmem pro tento problém.

Základem je definice voxelu jako krychle (kostky) s hodnotami obsaženými v jejích osmi rozích. Jestliže jeden nebo více rohů kostky obsahuje hodnoty nižší než je uživatelem specifikovaná „izohodnota“ a jeden nebo více rohů zároveň obsahuje hodnoty vyšší než je izohodnota, je zřejmé, že takový voxel bude nějakým způsobem přispívat k izopovrchu. Pomocí určení, které hrany kostky jsou protnuty izopovrchem, vytváříme trojúhelníkové vzorky, které uvnitř kostky oddělují region ležící „uvnitř“ izopovrchu od regionu ležícího vně. Výsledným spojením vzorků ze všech kostek na hranici izopovrchu získáme finální povrchovou reprezentaci.



Nyní si tento algoritmus popíšeme detailněji. Algoritmus se skládá ze dvou hlavních komponent. První má na starosti definici sekce nebo sekcí povrchu, které „rozsekávají“ jednotlivé kostky. Pokud každý roh kostky klasifikujeme jako spadající pod nebo nad definovanou izohodnotu, získáváme 256 možných konfigurací. Dvě z nich jsou triviální – všechny rohy kostky spadají dovnitř nebo vně. V takovém případě kostka nepřispívá do izopovrchu. U všech ostatních konfigurací musíme určit, které hrany kostky izopovrch protíná a tyto průniky se následně využijí pro vytvoření jednoho nebo více trojúhelníkových vzorků tvořících izopovrch.

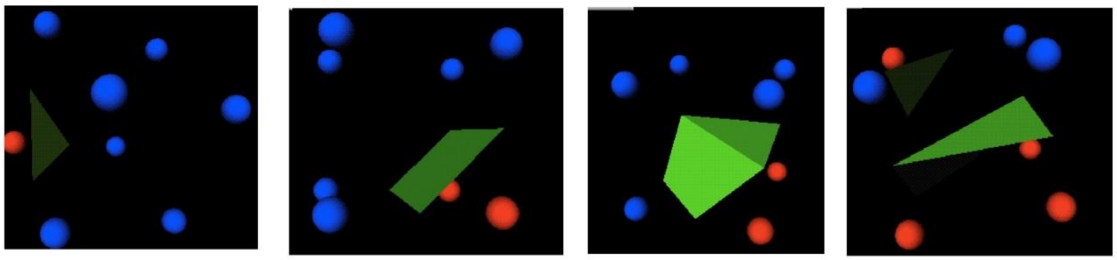
Pokud bereme v úvahu symetrie, pak ve zbývajících 254 vzorcích existuje pouze 14 jedinečných konfigurací. Pokud pouze jeden roh obsahuje hodnotu menší než je izohodnota, vytvoří se jediný trojúhelník, jehož vrcholy leží na hranách přiléhajících k tomuto rohu. Normála trojúhelníku směřuje od rohu. Tento případ generuje díky symetriím celkem 8 konfigurací. Jedna taková konfigurace je znázorněna na obrázku vlevo.

V případě, že dva rohy mají hodnotu menší než izohodnotu, jsou generovány 3 unikátní konfigurace (viz obrázek druhý zleva). Ty závisí na tom, zda rohy patří stejné hraně nebo stejné stěně nebo jsou „zasaženy“ diagonálně.

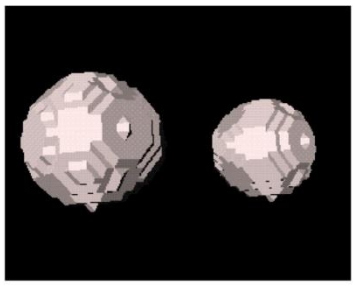
Pro tři rohy s menší hodnotou než je izohodnota máme opět tři unikátní konfigurace (obrázek druhý zprava). Ty závisí na tom, zda mají 0, 1, nebo 2 sdíleńé hrany.

Pokud máme „zasaženy“ čtyři rohy, máme k dispozici 7 unikátních konfigurací (obrázek vpravo). Výsledek závisí na tom, zda rohy sdílejí 0, 2, 3 nebo 4 hrany.

Každá z netriviálních konfigurací přidá do izopovrchu 1 až 4 nové trojúhelníky. Vrcholy trojúhelníků lze spočítat pomocí interpolace podél hran nebo se pro zjednodušení bere střed hrany. Je zřejmé, že interpolované vrcholy nám poskytnou hladší povrch.



Obrázek ukazuje atom vodíku renderovaný pomocí jednoduché verze Marching Cubes, kdy jsou jako vrcholy trojúhelníků brány středy odpovídajících hran kostek. Výsledek není uspokojující rovněž díky malé vstupní datové množině.
Když nyní umíme sestrojit část izopovrchu pro jeden voxel, můžeme tento postup aplikovat na celý objem. Můžeme buď každou kostku zpracovávat zvlášť nebo můžeme postupovat tak, že bereme postupně kostky, které sdílí hrany.



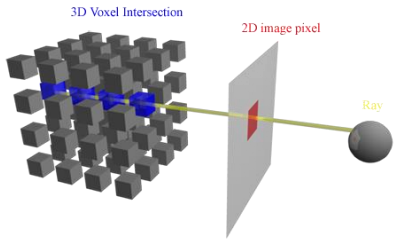
Problémy Marching Cubes
Jeden ze zřejmých problémů je paměťová náročnost při ukládání výsledného povrchu. Každá hraniční kostka totiž může generovat až 4 plošky. Velikost ukládaných dat se dá redukovat pomocí sdílení vrcholů a hran nebo dokonce slučování koplanárních vzorků do větších plošek. Dalším řešením může být „napasování“ parametrických povrchů na skupiny hraničních bodů. Tento přístup je však obtížný pro složité povrchy.
Další problém nastává, pokud nemáme objem zcela vyplněný voxely. K tomu může dojít, pokud jsou data nasnímána nekvalitně. Výsledkem jsou díry v datech, kterým musí být přiřazeny nějaké hodnoty, nejčastěji pomocí interpolace. Je zřejmé, že nově přidané hodnoty mohou snižovat validitu výsledného povrchu.

Vizualizace explicitních povrchů

Direct volume rendering (přímá vizualizace objemu) techniky nevyužívají pro vykreslení žádné 3D polygony. Místo toho jsou pixely výsledného obrazu spočteny individuálně, buď

pomocí vrhání paprsku do scény pro každý pixel nebo pomocí projekce voxelů na rovinu projekce.
Základní proces renderování volumetrických dat začíná transformací pozic jednotlivých voxelů do pohledového souřadného systému, přičemž je využita standardní 3D grafická pipeline. Uživatel musí specifikovat pohledový referenční bod a směrový vektor pohledu, šířku a výšku obrázku, který má být promítnut na projekční rovinu a při perspektivní projekci je nutné rovněž specifikovat vzdálenost kamery od projekční roviny.
Následně máme možnost si zvolit jednu z následujících metod:

- **Dopředné mapování** – promítnutí každého voxelu na projekční rovinu a určení, které pixely budou ovlivněny a jak.
- **Zpětné (inverzní) mapování** – rovněž nazýváno „vrhání paprsku“. Jde o vyslání paprsku z každého pixelu projekční roviny skrz objemová data, kdy dochází ke vzorkování hodnot dat, které paprsek protne a tím určí výslednou hodnotu pro každý pixel.



Dopředné mapování – problémy:

F1. Jak zacházet s pixely, které jsou ovlivněny více voxely

F2. Jak zacházet s pixely, na něž se nemapuje žádný voxel

F3. Jak se vypořádat s faktem, že voxely se obvykle promítají na pozice mezi pixely.

Zpětné (inverzní) mapování – problémy:

I1. Jak zvolit počet bodů, které budou vzorkovány podél paprsku.

I2. Jak spočítat hodnotu v těchto bodech, které často padnou mezi voxely.

I3. Jak zkombinovat body, které paprsek na své cestě protnul.

Řešení:

Problémy F2 a F3 se dají řešit pomocí mapování každého voxelu na region projekční roviny, přičemž dovolíme, aby částečně ovlivnil hodnotu několika pixelů sousedících s pozicí, na kterou se voxel promítl. Metoda využívající tento princip je založena na nastavení vah jednotlivých voxelů pro daný pixel, přičemž váha je odvozena ze vzdálenosti mezi pixelem a promítnutým umístěním daného voxelu. Obecně jsou takto ovlivněny nejvýše čtyři pixely.

Další metodou využívající tento princip je tzv. „splatting“ (rozplácnutí), kdy s každým voxelem asociujeme malý region textury a tento region se promítá na projekční rovinu.

Problém I1 lze snadno vyřešit určením rozestupu mezi voxely a nastavení vzorkovací frekvence na menší hodnotu, než jsou tyto rozestupy. Takto není možné pominout některé vlastnosti a charakteristiky zobrazovaného objemu. Samotné vzorkování může být doprovázeno převzorkováním či interpolací diskutovanými dříve.

Problémy F1 a I3 jsou běžně řešeny pomocí techniky známé jako „compositing“ (skládání). Nejjednodušší formy této techniky, jako je například spočtení maximální hodnoty objemu nebo průměrování všech objemových hodnot asociovaných s pixelem/paprskem. Běžnější přístupy předpokládají, že každý voxel má v sobě asociovanou hodnotu průhlednosti a tu využívají při integraci všech voxelů mapovaných na daný pixel. Pokud předpokládáme, že voxel *i* má barvu c_i a průhlednost o_i, pak jeho příspěvek k výsledné hodnotě pixelu je

$$c_i * o_i * \prod_{j=0}^{i-1}(1-o_j)$$

Jinými slovy musíme určit akumulovanou průhlednost mezi projekční rovinou a voxelem a použít ji k přiřazení intenzity (c_i * o_i) voxelu. Výsledná hodnota pixelu je pak dána výpočtem

$$I(x,y) = \sum_{i=0}^{n} c_i * o_i * \prod_{j=0}^{i-1}(1-o_j)$$

Klasifikace

Důležitým aspektem při renderování objemových dat je určení průhlednosti a barvy přiřazené příslušným datovým hodnotám. Tento proces je označován jako klasifikace a jeho výsledkem je sada funkcí definujících, jakým způsobem bude jednotlivých voxelům přiřazena jejich průhlednost a RGB kanály (nebo HSV kanály). Tyto funkce jsou označovány jako transferové (transfer functions). Iniciálně mohou být nastaveny při analýze dat. Alternativně může uživatel chtít interaktivně ovlivňovat tyto funkce. Takto je možné nastavit tzv. oblasti zájmu (regions of interest), kde je možné nastavit jinou barvu a průhlednost než v jiných oblastech a tím tuto oblast zvýraznit či upřednostnit.

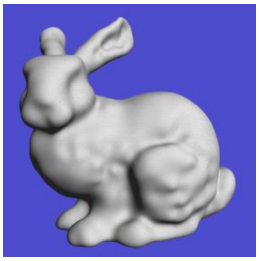
Výpočet osvětlení a stínování

Další důležitý problém, který je nutné řešit při přímém zobrazování objemu, je výpočet osvětlení a stínování. Vzhledem k tomu, že nemáme k dispozici žádný explicitní povrch, který obsahuje normály, je nutné provést aproximaci založenou na výpočtu *gradientu* (poměr změny) v každém směru. Pro daný voxel (v_x, v_y, v_z) můžeme odhadnout, jak rychle se mění jeho hodnota například ve směru osy x pomocí prozkoumání sousedních voxelů v tomto

směru, konkrétně (v_{x-1}, v_y, v_z) a (v_{x+1}, v_y, v_z). Nejjednodušší gradient je určen pouze rozdílem voxelu a jednoho z jeho nejbližších sousedů. Takže g_x, což je x-ová komponenta gradientu, má hodnotu $v_x - v_{x-1}$. Nazývá se **střední diferenční operátor** (intermediate difference operator). Spočtením tří rozdílů – mezi voxelem a jeho sousedy ve třech směrech – dostaneme vektor, který představuje velmi hrubý odhad směru maximální změny v hodnotě voxelu. Tento vektor pak může být použit namísto normály při výpočtu stínování.
Mnohem přesnější odhad tohoto vektoru můžeme získat prozkoumáním většího okolí daného voxelu. Nazývá se **centrální diferenční odhad gradientu** (central difference gradient estimator) a využívá body po obou stranách voxelu (ne však hodnotu voxelu samotného).
V tomto případě je g_x spočteno jako $v_{x+1} - v_{x-1}$. Ignorováním hodnoty v samotném voxelu (v_x, v_y, v_z) můžeme ztratit důležitou informaci.
Větší operátory gradientu, jako například 3D Sobelův operátor, pracují se všemi 26-ti nejbližšími sousedními voxely a lze použít dokonce větší okolí. Přesný odhad gradientu má významný vliv na výsledný vizuální vjem objemových dat.

Implicitní povrchy

Typickou metodou modelování povrchů v počítačové grafice je využití parametrických rovnic pro definici bodů na povrchu. Tyto body jsou následně spojovány do polygonálních sítí (meshí). Tato reprezentace je výhodná zejména pro aplikaci transformací a výpočet normál povrchu. Alternativní metoda je využití takzvané **implicitní reprezentace**, kdy je povrch definován jako tzv. zero contour (počáteční kontura) funkce o dvou nebo třech proměnných. Implicitní reprezentace má své silné stránky při aplikování operací jako je blending (míchání) nebo přeměna (metamorphosis).

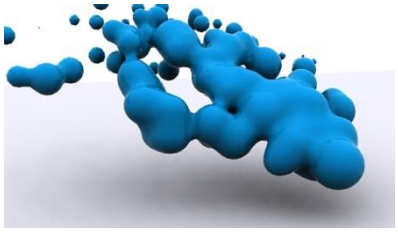


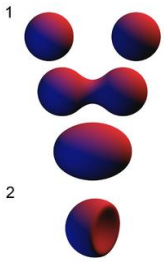
Na obrázku vidíme rozdíl mezi polygonální reprezentací modelu a tentýž model zobrazený pomocí implicitního povrchu, který je definován 800 vrcholy a jejich normálami.

Metaballs

Příkladem implicitní modelovací techniky jsou takzvané metaballs (nebo jinak blobby objects). Na metaball se můžeme dívat jako na částici obklopenou polem hustot, kdy se hustota snižuje se vzdáleností od částice samotné. Povrch metaballu je určen pomocí izopovrchu z tohoto pole hustot – čím větší hodnota izopovrchu, tím blíže je tento povrch k částici. Jedním z důležitých aspektů metaballů je jejich možnost kombinování. Jednoduchým součtem vlivů každého metaballu na daný bod získáme velmi hladké smíchání sférických polí vlivu.

Klíčovým prvkem při použití metaballů je definice výpočtu použitého pro specifikaci vlivu libovolné částice na libovolný bod. Blinn použil pro každou částici exponenciálně klesající pole v kombinaci s Gaussovou hrbolatostí (bump) o výšce *b* a standardní odchylce *a*. Pokud *r* je vzdálenost částice od daného místa v poli, pak vliv částice je spočten jako b^{-ar}. Z důvodu efektivity byl tento výpočet změněn na čtverec vzdálenosti. Výsledná hustota v libovolném místě je dána jednoduchým součtem příspěvků všech částic.

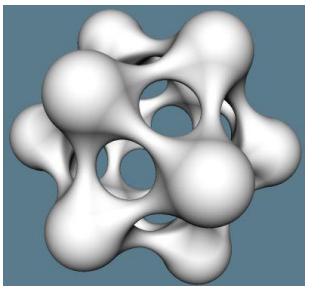




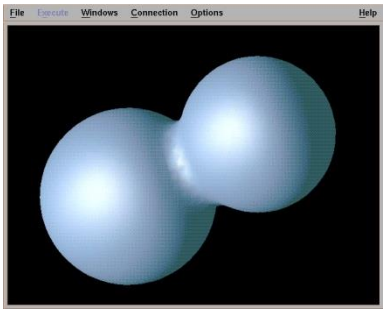
Wyvill a spol. zjednodušili výpočet zavedením kubického polynomu založeného na poloměru vlivu pro danou částici a vzdálenosti od středu částice do dotazovaného místa v poli. Klíčový poznatek je, že hodnota vlivu musí být 1.0 pro vzdálenost r rovné 0.0 a vliv má hodnotu 0.0, pokud je vzdálenost rovna poloměru R vlivu. Funkce vyhovující této podmínce je definována jako:

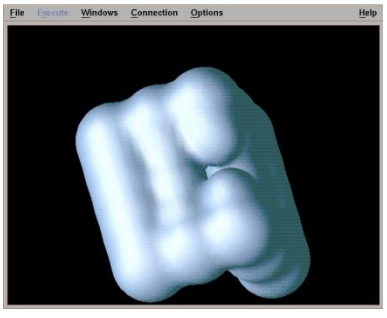
$$C(r) = 2r^3/R^3 - 3r^2/R^2 + 1$$

Díky výpočtu druhé odmociny je tato rovnice pomalá, proto byla opět přetvořena jako funkce parametrů r^2 a R^2.



Pro vstupní datovou množinu definujeme grid a spočteme hustotu v každém jeho bodě na základě hodnot v sousedech. Jakmile máme tento grid vytvořen, můžeme využít libovolnou techniku pro renderování, jako například dříve zmíněné Marching Cubes. Obrázek ukazuje různé příklady typů renderování, které mohou být aplikovány na implicitní povrchy. Obrázek vlevo byl generován pomocí dvou koulí s izohodnotou 0.11. Protínající se části byly sečteny. Všimněte si hladkého přechodu mezi těmito dvěma objekty. Obrázek vpravo obsahuje 24 koulí formujících tvar písmene L, uprostřed je díra. Tyto koule mají nastaveny různé hodnoty rozsahu vlivu. Je vidět, že již poměrně složitý povrch byl definován pomocí 24 pozic, poloměrů a jednou izohodnotou.





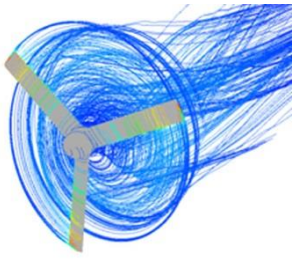
Dynamická data

Vizualizace toku (flow visualization) představuje metody zobrazení dynamického chování tekutin a plynů. Základy této disciplíny sahají až do 15. století, kdy Leonardo da Vinci nakreslil skicy částeček písku a dřevěných hoblin, které byly vhozeny do tekutiny. Od té doby se díky laboratorním testům stala vizualizace toku čím dál více přesná. Pokroky ve fotografii rovněž přispěly k pochopení, jak tekutiny a plyny proudí za různých podmínek.

Při vizualizaci toku je hlavním cílem vizualizovat změnu informace. Typicky vstupní data pro takovéto množiny obsahují více než 3 dimenze. Vizualizační techniky si v tomto případě kladou za cíl poskytnout uživateli náhled na vstupní data i detailní zobrazení. Vstupní data jsou nejčastěji získávána pomocí simulací (například v leteckém, lodním či automobilovém průmyslu, v počasí, medicíně – tok krve, atd.), pomocí měření (například ve větrném tunelu) nebo modelováním (výpočtem obyčejných diferenciálních rovnic).

Oblast CFD (computational fluid dynamics) rozšířila možnosti a schopnosti vědců studovat proudy pomocí vytváření simulací dynamického chování tekutin pod širokou škálou podmínek. Výsledkem této analýzy je obvykle 2D nebo 3D mřížka vektorů rychlostí, které mohou být uniformně nebo neuniformně rozloženy. Cílem je následná analýza tohoto vektorového pole za účelem identifikace různých vlastností, jako například turbulence, vírů, sedlových bodů atd.
Rozlišujeme několik variant struktur uvnitř dat generovaných těmito experimenty:

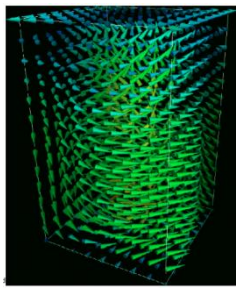
- Statické pole – obsahuje pouze jedno neměnné pole rychlostí
- V čase se měnící pole – mohou mít fixní pozici s měnícími se hodnotami vektorů nebo mohou měnit obojí – pozici i vektory. Jsou též nazývány nestabilními.

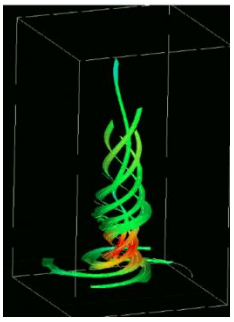


Během let byla vyvinuta řada technik pro vizualizaci proudu (toku). Nejjednodušší metodou je zobrazení samotných dat pole průtokových rychlostí buď v podobě vektorů využívajících směrové značky nebo jako skalární hodnoty využívající obrazové, povrchové nebo prostorové vizualizační techniky. Počet a rozmístění zobrazovaných komponent hraje zásadní roli při zprostředkování důležitých znaků vstupních dat – příliš mnoho komponent může vést k

přílišnému překrývání, zatímco málo komponent může opomenout důležité informace. Nejjednodušším řešením je nechat uživatele interaktivně manipulovat s hustotou zobrazených prvků, nicméně poslední výzkumy se zaměřují na automatické rozmístění zobrazovaných komponent pomocí analýzy vstupních dat a identifikace regionů s potenciálně zajímavým tokem.

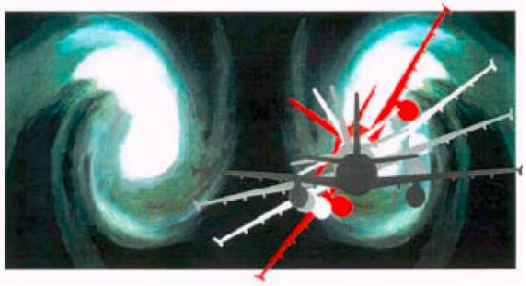
Vizualizaci toku dělíme na přímou a nepřímou. Přímá vizualizace poskytuje náhled na aktuální stav toku (obrázek vlevo) a je často vizualizována pomocí sady vektorů. Nepřímá vizualizace dále tento náhled obohacuje o zobrazení vývoje toku v čase (obrázek vpravo). K tomu se využívají tzv. streamlines a streamsurfaces (viz dále).

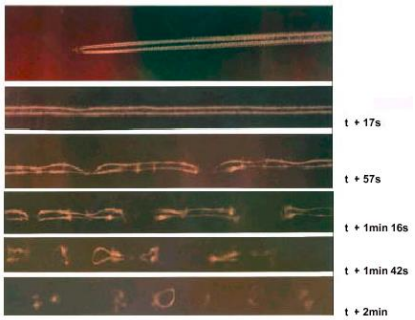




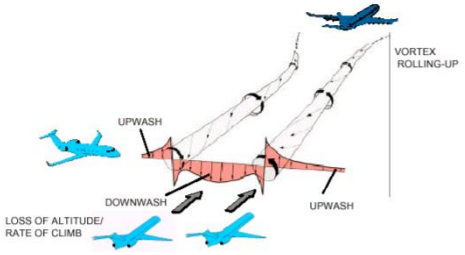
Příklad

Uveďme si příklad využití vizualizace toku, která má významný praktický dopad. V letectví je významným problémem vznikající turbulence za letadlem.

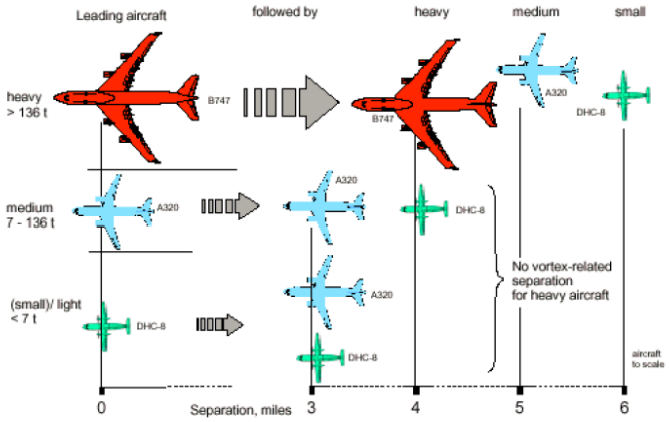




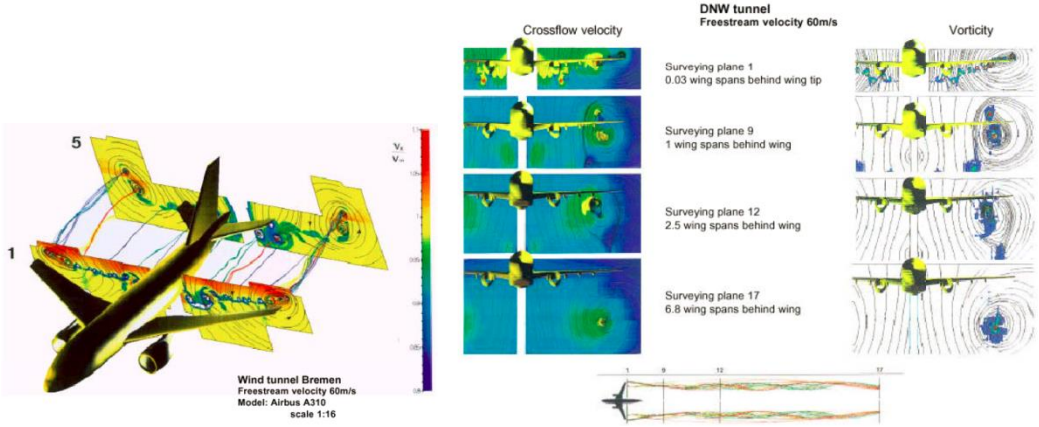
Tyto víry jsou zejména po vzletu letadla velmi nebezpečné, protože další letadla na vzletové dráze mohou být těmito víry významně vychýlena při startu.



Proto je při vzletu dalšího letadla nutné dodržovat určité vzdálenosti.

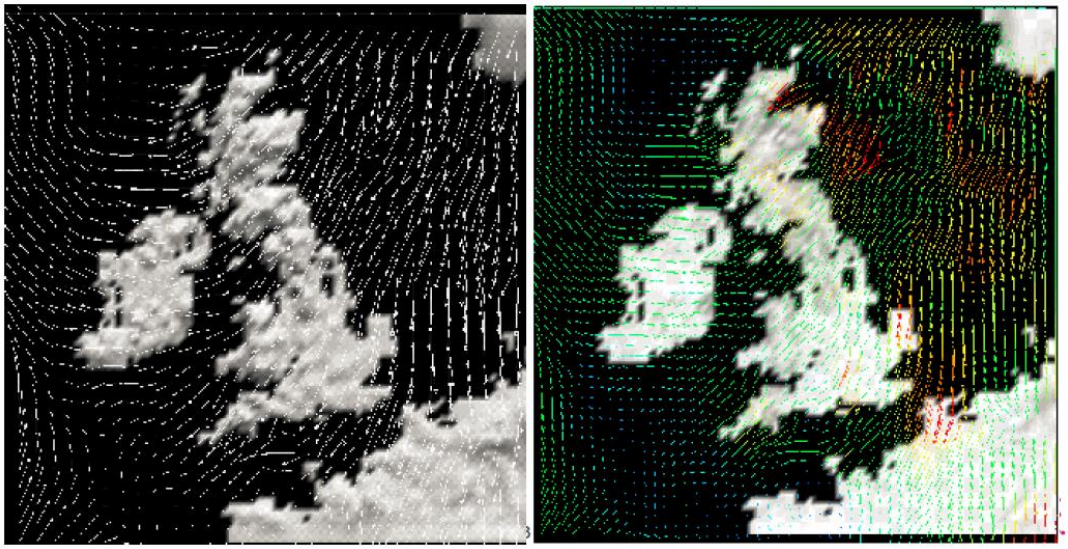


V praxi je testování vírů a jejich vlivu prováděno ve větrném tunelu a poté jsou získaná data vizualizována.

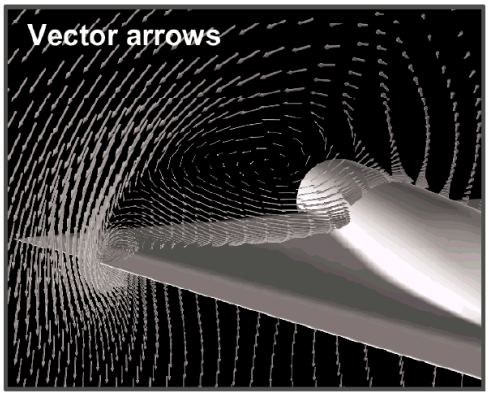


(konec příkladu)

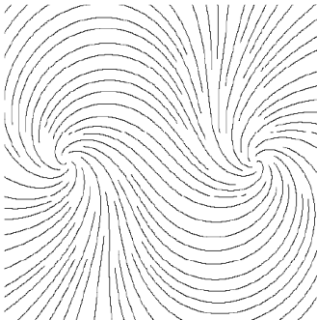
Běžným způsobem vizualizace toku je jeho zobrazení pomocí šipek. Ty jsou buď příslušným způsobem škálovány či obarveny, aby odpovídaly hodnotám v datech. Na obrázku je uveden příklad obou těchto metod. Vlevo je tok mapován na velikost šipek, vpravo pak na barvu.



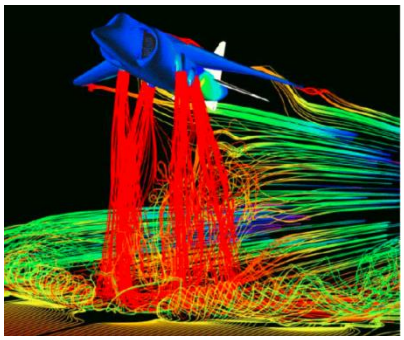
Ve 3D se pro zvýšení přehlednosti používá zobrazení šipek pouze v určitých vrstvách.



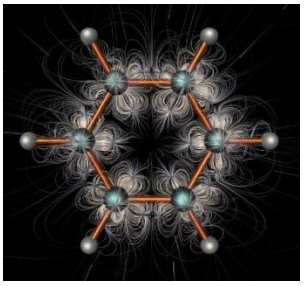
Druhou nejběžnější technikou je generování streamlines (tečné přímky procházející rychlostním polem) založených na statickém rychlostním poli. Uživatel vybere seed lokace (často umístěné na úsečce nebo ve 2D mřížce) a spočte cestu každého semínkového bodu skrz pole.



Streamlines 3D:



Pro lepší vnímání 3D prostoru se využívají rovněž tzv. osvětlené (illuminated) streamlines.



Algoritmus pro umístění jednotlivých streamlines je založen na předpokladu, že generované streamlines by neměly být umístěny příliš blízko sebe.

Algoritmus používá pro základní nastavení dva parametry:

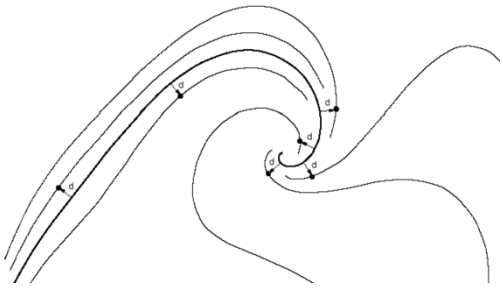
- d_{sep} startovní vzdálenost při vyhledávání počátku další streamline
- d_{test} minimální vzdálenost mezi jednotlivými generovanými streamlines

Algoritmus má tyto základní kroky:

```
Spočítej počáteční streamline, vlož do fronty
Nastav počáteční streamline jako aktivní
WHILE není dokončeno DO
TRY získej nový bod ve vzdálenosti dsep od aktivní streamline
IF nalezeno THEN spočti novou streamline a vlož do fronty
ELSE IF fronta je prázdná
     THEN ukonči smyčku
     ELSE další streamline ve frontě se stane aktivní
```

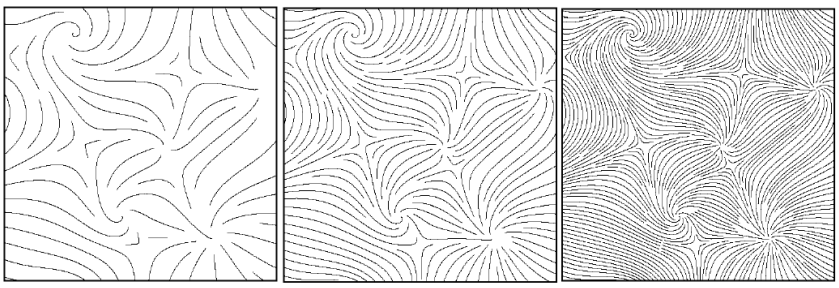
Tvorba streamline je ukončena, pokud dojde ke splnění jedné z následujících podmínek:

- Když je vzdálenost k sousední streamline ≤ d_{test}
- Když streamline opustí definovanou doménu
- Když se streamline příliš přiblíží sama k sobě
- Po provedení předem definovaného počtu kroků

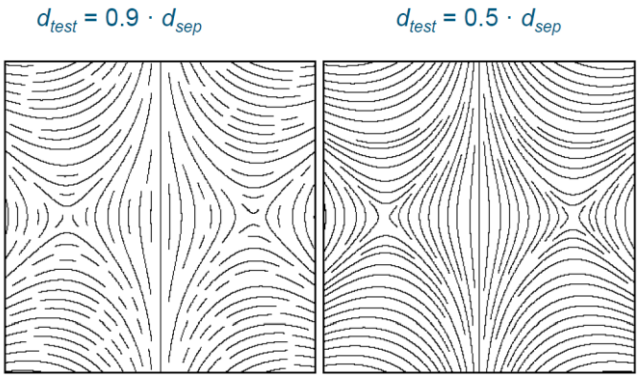


Následující obrázek znázorňuje vliv parametru d_{sep} na hustotu generovaných streamlines. Číslo vyjadřuje procentuální poměr vzhledem k šířce obrázku.

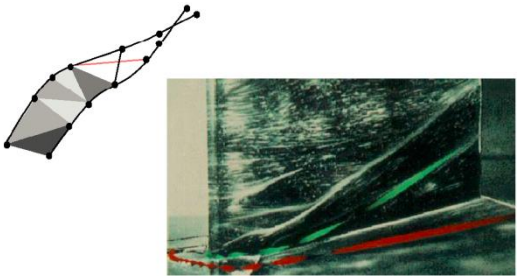
6% 3% 1.5%



Na dalším obrázku je uveden vztah mezi dvěma uvedenými parametry:

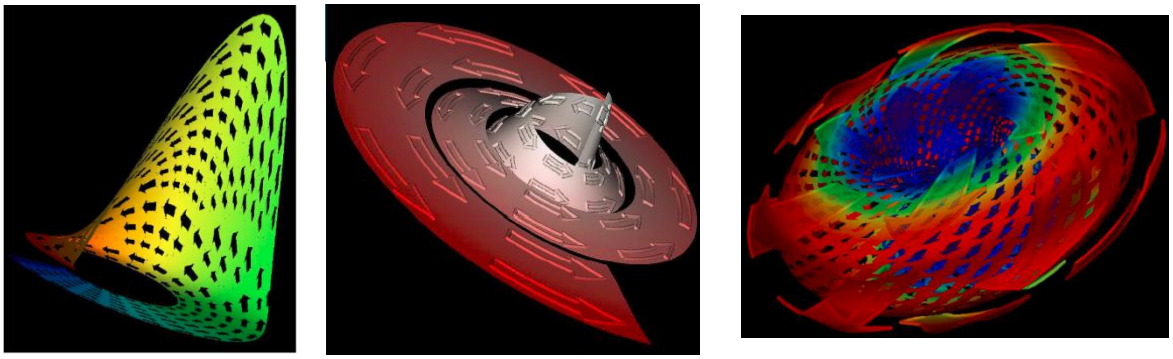


Kromě čar označujících proudy můžeme použít rovinné nebo pevné objekty, jako například stuhy (ribbons) či hadice (tubes). Výhodou je, že nyní můžeme mapovat i další atributy pole, jako například velikost či vířivost, na atributy těchto zvolených objektů (například na barvu, velikost či zkroucení).



Další možnou technikou, kterou lze pro flow data použít, jsou tzv. **streaklines**. Streaklines jsou většinou zobrazeny jako spojitý proud částic vyzařujících z diskrétní sady bodů a proudící skrz pole (viz obrázek). Jednotlivé body dané stopy (všechny částice vycházející s daného místa patří do jedné stopy) mohou být barevně odlišeny, což je výhodné zejména v místech s vysokou průtokovou rychlostí či turbulencí.
Další z novějších technik se nazývá „streamball“ a je založena na použití metaballů.

Dalším rozšířením streamlines jsou tzv. **streamsurfaces**.

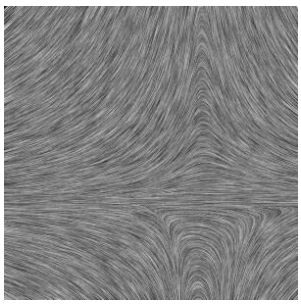


Line integral convolution

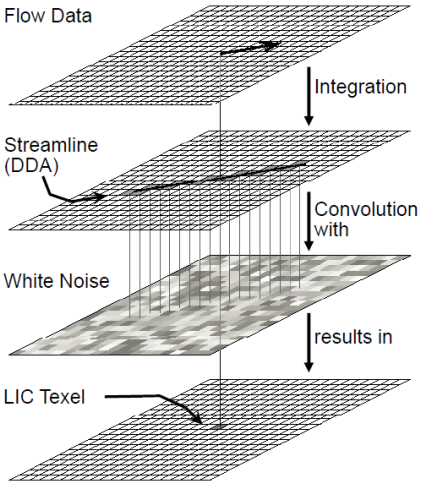
Zajímavý přístup k vizualizaci vektorových polí byl navržen Cabralem a Leedomem v roce 1993. Jejich metoda se nazývá Line integral convolution (LIC) využívá náhodné pole (např. texturu) a vektorové pole stejné výšky a šířky pro generování hustého zobrazení průtokové informace (viz obrázek).

Každý pixel výsledného obrázku je váženým průměrem sekvence sousedících pixelů v náhodném poli podél lineární cesty procházející daným pixelem a sledující streamline (proudnici), která prochází skrz pixel.

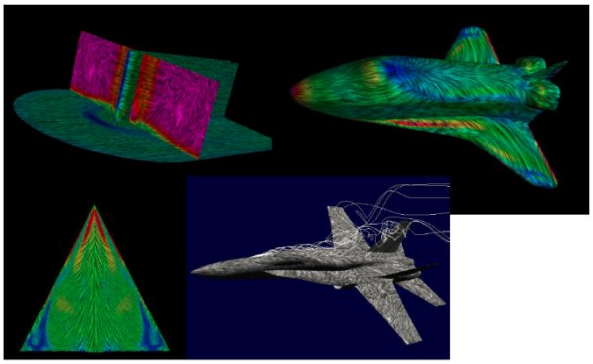
Formálněji, začínáme v místě (i, j) vektorového pole a vytvoříme dva řetězce pixelů vycházející z tohoto místa v opačných směrech a navíc ve směru tečny orientace vektoru. Délka řetězce je nastavena na 2L, kde L je vzdálenost, kterou pixely urazily od počátečního místa v každém směru. Tyto pixely pak tvoří jádro filtru o šířce jednoho pixelu. Odpovídající pixely v náhodném poli (textura, bílý šum, …) jsou sečteny a normalizovány délkou řetězce a výsledná hodnota je uložena na pozici (i, j) ve výsledném obrázku.



LIC využívá texturu pro zobrazení korelace mezi vizualizací a tokem. Výpočet hodnoty se pak spočítá náhledem na streamline z daného bodu a filtrací bílého šumu podél streamline.



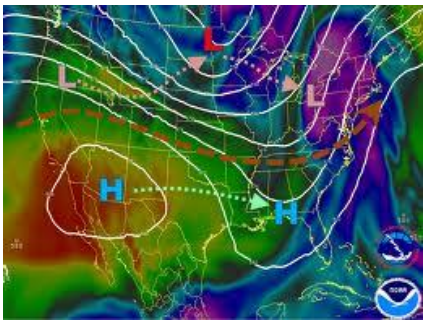
LIC může být poté například mapován na povrch 3D objektu:



Kombinované techniky

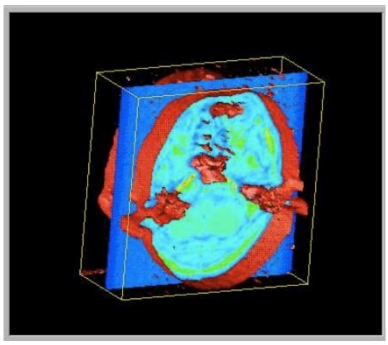
Většina efektivních vizualizací je ve skutečnosti kombinací dvou nebo více výše popsaných technik. Každá z technik má své silné a slabé stránky z pohledu informace, kterou je nebo není schopna efektivně zobrazit. Silné stránky jednotlivých technik můžeme vyzdvihnout jejich kombinací, přičemž musíme samozřejmě dbát na to, aby byl minimalizován překryv. Zároveň je čím dál větší tlak na souběžné zobrazení dat několika různými způsoby, abychom lépe pochopili sdělovanou informaci. Například předpověď počasí v sobě zahrnuje zobrazení teploty, rychlosti větru, relativní vlhkosti a řadu dalších faktorů, které ovlivňují přesnost předpovědi.

V této části se zaměříme na vizualizace, které vznikly kombinací předchozích technik.



Pláty kombinované s izopovrchem

Na obrázku je ilustrována technika, kdy je izopovrch medicínských dat kombinován s ortogonálním plátováním (slicing) stejné datové množiny. Izopovrch je mapován na jednu barvu (červenou) a výsledné hodnoty po plátování jsou zobrazeny modře. Izopovrch je schopen odhalit strukturu povrchu, která by byla obtížně rozpoznatelná pomocí samotného plátování objemu. Plát poskytuje detailní 2D informaci, zvláště pokud je dobře zvoleno přiřazení barev. Může uživateli zprostředkovat informaci o relativně uniformních regionech i o těch, ve kterých dochází k dynamickým změnám. Další výhodou je, že plát zobrazí i vnitřní (vnořené) regiony v příslušném rozsahu hodnot, zatímco obecný izopovrch zobrazuje pouze vnější povrch.



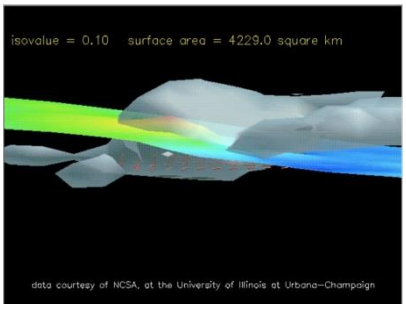
Při návrhu tohoto typu vizualizace je důležité vzít v úvahu následující skutečnosti:

- Není vhodné podporovat rychlé změny hodnot izopovrchu
- Pozice a orientace (podél tří os) plátu by měla být jednoduše ovladatelná uživatelem, včetně možnosti animace pozice plátu
- Pro zprostředkování pohledu na data ze všech stran je nutné umožnit uživateli měnit pozici a orientaci kamery
- Zásadní je přiřazení barvy – některé zájmové oblasti uvnitř plátu mohou být objeveny pouze pečlivou volbou barevného schématu. Zároveň je důležité použít pro zobrazení izopovrchu jinou barvu, která se neobjevuje v barevné mapě plátu, abychom zabránili špatnému výkladu.
- Uživatel by měl mít možnost jednoduše skrýt kteroukoliv z těchto dvou vizualizačních komponent, případně alespoň měnit průhlednost.

Kombinace izopovrchu a piktogramů

Jak již bylo řečeno, izopovrchy jsou vhodné pro zprostředkování detailů 3D povrchů, ale obecně neobsahují žádné jiné aspekty vstupních dat. Piktogramy (glyfy), jako například běžná šipka, jsou využívány pro zobrazení velikosti nebo směru změny v datové množině – buď jako gradient ve statických datech nebo jako tok v datech dynamických. Glyfy mohou být umístěny v těsné blízkosti izopovrchu.

Obrázek ukazuje bouřkový mrak, kde izopovrch znázorňuje hustotu vody uvnitř mraku a šipky ukazují směr a sílu větru. Navíc je použita ořezávací rovina pro zobrazení detailů hustoty vody. Pohybem se startovními pozicemi šipek (jsou spojeny s pohyblivou rovinou) můžeme odhalit „klidné“ regiony a naopak regiony s turbulencí.



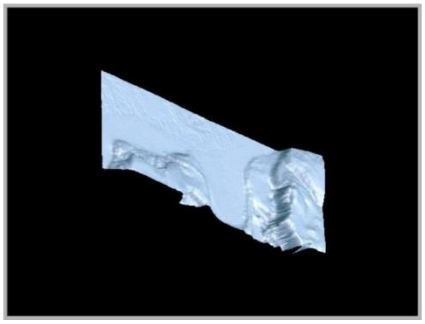
Při návrhu tohoto typu kombinované vizualizace je vhodné zachovávat podobná pravidla jako v přechozím případě. Interaktivní kontrola parametrů vizualizace – hodnoty izopovrchu, základní pozice glyfu, úhel pohledu – přispívají ke zefektivnění dané vizualizace. Další aspekty jsou nutné pro zlepšení informačního obsahu glyfů, jako například:

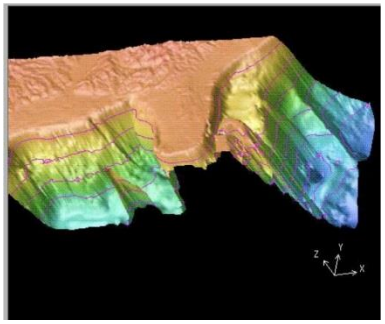
- Měnící se hustota glyfů
- Škálování velikosti glyfů
- Přiřazování barvy glyfům
- Výpočet základní pozice glyfů na základě oblastí zájmu ve vektorovém poli

Další potenciální vylepšení této techniky může být využití ořezávací roviny nebo reliéfu (rubber sheet).

Reliéf + kontura + barva

Již jsme zmínili, že je výhodné zobrazovat data „nadbytečně“, čímž je myšleno jejich mapování na různé vizuální atributy. V tomto případě začínáme se zobrazením reliéfu, které znázorňuje hodnoty 2D pole v podobě výškové mapy. Takto jsme schopni odhalit „výčnělky a údolí“ v datech. Tuto vizualizaci můžeme dále rozšířit o mapování barvy na vyvýšeniny, čímž umožníme jednodušší identifikaci oblastí s podobnou nadmořskou výškou. Nakonec můžeme reliéf proložit konturami o určitých hladinách, čímž zvýrazníme hodnoty gradientu.
Každá přidaná metoda zlepšila naše pochopení dat.





Shrnutí

Seznámili jsme se s několika běžně používanými technikami pro vizualizaci časoprostorových dat. Díky tomu, že jsme procházeli techniky podle dimenzionality dat, kterou jsme postupně zvyšovali, poznali jsme základy, na nichž jsou jednotlivé techniky postaveny. Vzhledem k tomu, že pro vizualizaci určité datové množiny je možné použít několik různých přístupů, je důležité porozumět kladům a záporům jednotlivých technik – zejména které vlastnosti dat techniky jasně prezentují. Často je výhodné jednotlivé techniky kombinovat.