Fakulta Informatiky, Masarykova Universita v Brně


Učební text k přednášce

# PV019 – Úvod do geoinformačních systémů

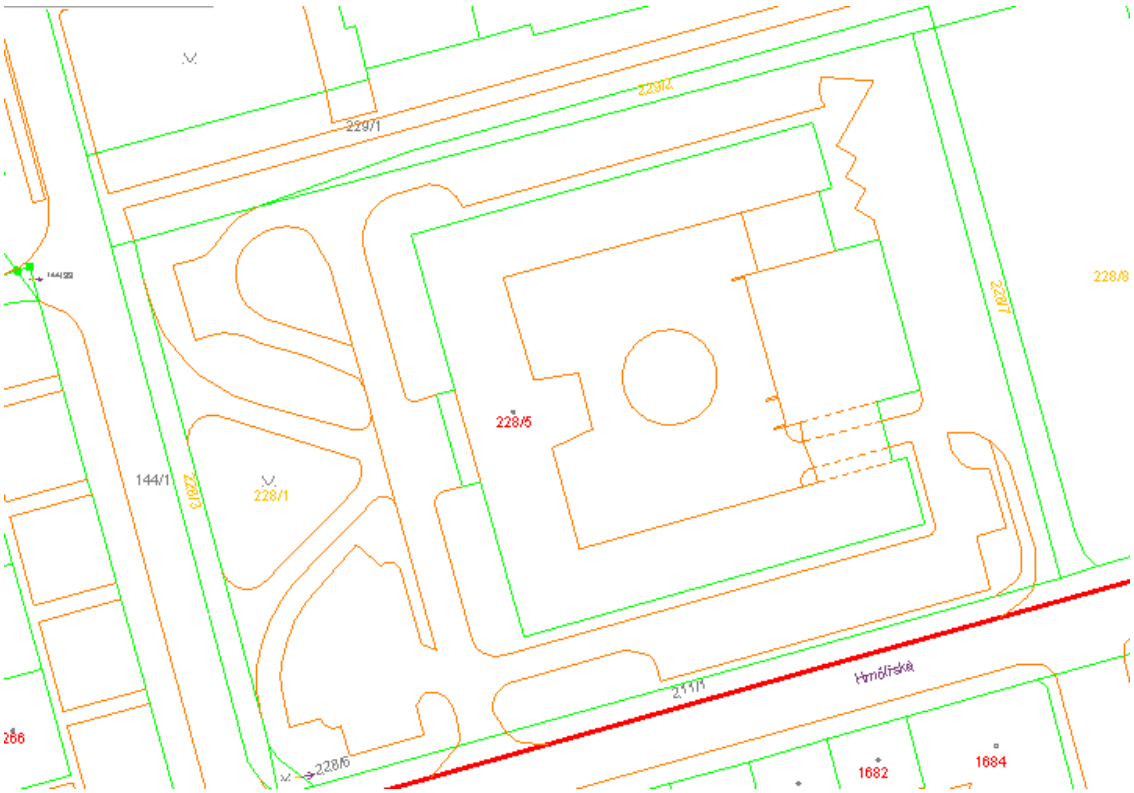
# 1. Geoinformační systém, místo na povrchu Země.

## 1.1. Vymezení pojmu GIS

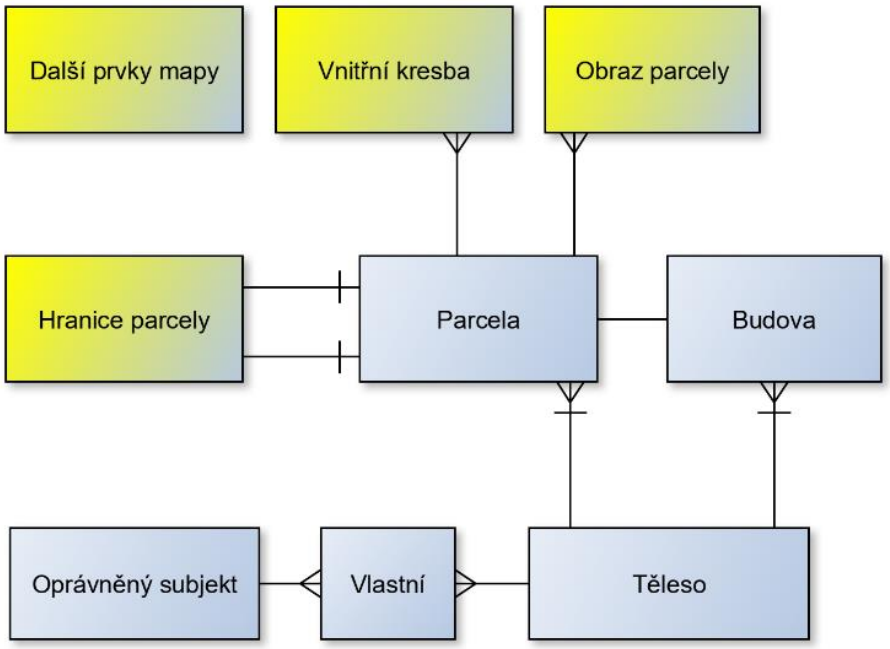
Začneme příkladem z poměrně nedávné historie.

Příklad – Informační systém o nemovitostech

Katastrální úřady evidují vlastnické vztahy fyzických a právnických osob k nemovitostem (budovám a pozemkům). Tato evidence existuje v různých formách v našich zemích již staletí. K prostorovému umístění evidovaných objektů sloužila mapa, k evidenci vlastnických vztahů kartotéka (zemské desky, pozemkové knihy, listy vlastnictví).



Uvažme informační systém o nemovitostech s datovým modelem v relačním databázovém systému, entity systému jsou navrženy v E-R diagramu:




„Klasický“ IS je schopen reagovat na dotazy typu:

- Kdo vlastní parcelu?
- Jaké nemovitosti vlastní osoba? Jakou cenu mají parcely, které vlastní osoba?

Lidsky čitelná odpověď na takové dotazy je převoditelná do textu/ů.

GIS jsou navrhovány tak, aby byly navíc schopny reagovat na dotazy typu:

- Kde se nalézá parcela?

V tomto případě už lidsky čitelná (rozumná) odpověď smysluplně převoditelná do textového tvaru není. Smysluplnou odpovědí je mapa.

Představme si, že si ve webovém prohlížeči zadáme do vyhledání „**`Restaurace Na Růžku, Brno`**“ a systém nám odpoví jen textovým polem **`49.2104877N 16.6002348E`**. Pokud nemáme opravdu výjimečné schopnosti, pravděpodobně důležitou akci v tomto objektu zmeškáme.

Půjdeme-li dále do historie, najdeme bezpočet příkladů používání map – geoinformací. Od navigace až po politiku a vojenství. Vždy se však jednalo o činnosti spojené s prostorovým umístěním jistých objektů na Zemi. Proto si můžeme v této chvíli intuitivně vymezit pojem GIS následujícím způsobem:

GIS je jakýkoli soubor postupů užívaných k ukládání a manipulování geograficky vztažených dat. Za geograficky vztažená data budeme považovat jakákoli data, která obsahují lokalizační složku, tedy taková data, která mají vazbu na místo na zemském povrchu.

Typy GIS – tradiční dělení

- Land Information System (LIS), Land Related Information System (LRIS), územně orientovaný informační systém, který zahrnuje vlastnické vztahy (nemovitosti, vlastníci).

- Geoinformační systém – je systém pracující s jevy, která lze lokalizovat v území, ale ne vždy je lze považovat za geografická (umístění vodovodního šoupátka, dopravní značky).

- Grafický informační systém – je systém pracující s geometrickými daty, která nemá smysl lokalizovat v nějakém (jednotném) prostoru, například schéma rozvodné sítě.

Toto dělení ztratilo postupem doby smysl, po geoinformačních systémech je požadována komplexní funkcionalita.

V tomto kurzu budeme termín:

Geoinformační systém

používat obecně – nebudeme polemizovat, zda mají objekty „geografický“ charakter (řeky, pohoří, etnicita obyvatelstva, kriminalita v obcích atd.) bude nám stačit, že „jev“ je jednoznačně vztažený k povrchu Země.

## 1.2. MÍSTO NA ZEMSKÉM POVRCHU

Uveďme si pro začátek základní pojmy, které budeme potřebovat pro vymezení místa na Zemi.

Geoid (1871 Listing) – Matematické těleso, model povrchu Země. Ekvipotenciální plocha vůči zemské gravitaci, která se nejvíce přimyká ke střední klidové hladině oceánu. Geometrické místo bodů se stejnou úrovní tíhového potenciálu způsobeného gravitací Země.

Rovina místního poledníku – je rovina určená osou rotace Země a určovaným bodem.

Zeměpisná délka ($\lambda$) - úhel, který svírá rovina místního poledníku procházejícího určovaným bodem a rovina nultého poledníku. Udává se většinou v úhlových jednotkách stupňů $[-180°, 180°]$.

Zeměpisná šířka ($\varphi$) – úhel, který svírá rovina rovníku a přímka procházející středem Země a určovaným bodem. Udává se většinou v úhlových jednotkách $[-90°, 90°]$.

**Souřadnice $[\varphi, \lambda]$ jednoznačně určují místo na zemském povrchu.**

Loxodroma – křivka, která protíná poledníky pod stejným úhlem.

Ortodroma – nejkratší spojnice dvou bodů, v případě kulového modelu Země kratší oblouk hlavní kružnice.

## 1.3. KDE JSEM? STRUČNÝ POHLED DO HISTORIE

**Eratosthenes** (cca. 240 před n. l.) Pravděpodobně první dochovaný pokus o změření zemského obvodu s popisem metody pochází od Eratosthena z Kyrény. Použil úhlovou metodu, která spočívá ve změření maximální výšky slunce na dvou místech na stejném poledníku ve stejný čas. Vybral města Alexandrie a Syéné (dnešní Asuán). Eratosthenes totiž pobýval v Alexadrijské knihovně/univerzitě a stačily mu „dostupné“ prameny.

Alexandrie a Asuán – (maps.google.com)

Poměr rozdílu úhlů dopadajícího slunečního paprsku (v poledne) a velikosti kruhového oblouku mezi dvěma místy na stejném poledníku

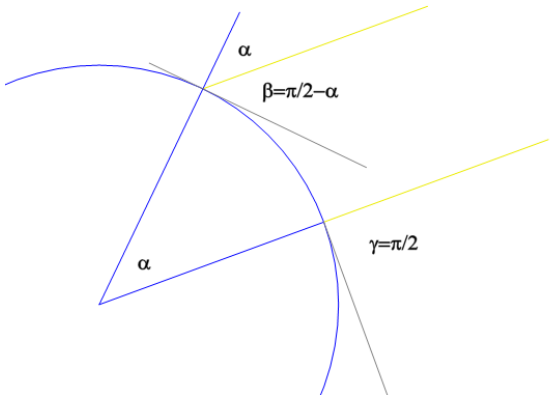
**=**

Poměr $2\pi$ a obvodu Země

Vzdálenost Alexandrie a Syéné byla odhadnuta na 5000 stádií (záznamy obchodníků, pochodující legie kolem řeky Nil), úhlový rozdíl slunečních paprsků by změřen na cca. $2\pi/50$ Úhel byl měřen za slunovratu v Alexandrii, úhel slunce v poledne za slunovratu v Syéné byl znám jako $\pi/2$. Sloupy kolmé k zemi nevrhaly stín – místo leží přibližně na obratníku Raka.

**Obvod Země = 50 x 5000 = 250 000 stádií**

1 stadion = 177.6 metrů, obvod Země je tedy **44 400 000** m.




*Geometrický princip Eratosthenovy metody*

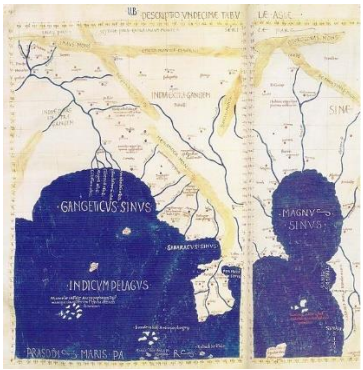
$\beta$ - maximální výška slunce za slunovratu v Alexandrii
$\gamma$ - maximální výška slunce za slunovratu v Syéné (= $\pi/2$)
$\alpha$ - středový úhel mezi Syéné a Alexandrií

$$\alpha = \gamma - \beta$$

Jsou známy různé rozměry antického stadionu, od 157 do 185 m a není jasné se kterým údajem Eratosthenes pracoval. I přes cca chybu 10% se jedná o jeden z nejvýznamnějších výsledků antické vědy.

Obvod Země byl potom v 9. století zpřesněn z příkazu chalífy Al-Mámuna (arabské civilizaci té doby do značné míry vděčíme za kontinuitu antické vědy). Pro určení míst bylo použito pozorování hvězd a výsledek byl neobyčejně přesný (chyba do 5%).

**Ptolemaios (90–160 n. l.)** – první mapové dílo, zavedl do všeobecného používání úhlové jednotky stupně/minuty/vteřiny. Tabulkově zapsal zaměřené a odhadnuté pozice asi 8000 míst na Zemi (města, ústí řek, mysy, hory). Tato místa zobrazil na souboru map, zobrazujících téměř celý, do té doby poznaný svět. Geografie je v Ptolemaiově pojetí „rovinným zobrazením známé Země, se vším, co se na ní nachází“.



**Gerardus Mercator – Geert de Kremer (1512-1594)** – Vlámský matematik a kartograf. Je autorem, v té době revolučního, Mercatorova zobrazení:

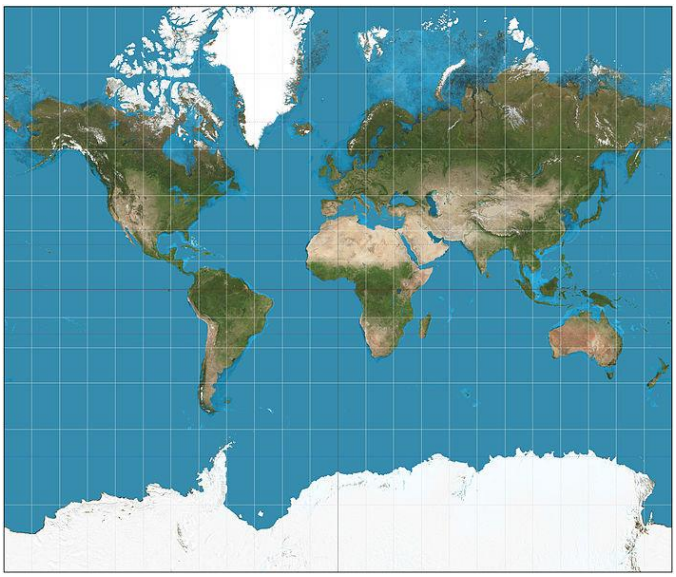
1) Poledníky jsou přímky rovnoběžné s osou y, x-ová souřadnice je přímo úměrná zeměpisné délce.

2) Loxodromy zachovávají azimut, úhel přímky na mapě vůči ose $Y$ se rovná azimutu.

Z 1) okamžitě vyplývá:

3) Loxodromy jsou přímky.

Zobrazení je velmi užitečné pro navigaci z místa A do místa B. Stačí je spojit na mapě přímkou a odečíst azimut úhloměrem. Mercator autorem kolekce map (první použil termín atlas). První mapy byly zpřesněné Ptolemaiovy mapy, postupně přidával i vlastní kartografická díla. V jeho díle pokračoval i jeho syn Rumold.



*Zobrazení Země v Mercatorově projekci*
*(zdroj: http://en.wikipedia.org/wiki/Mercator_projection)*

Mercatorova projekce je dána rovnicemi:

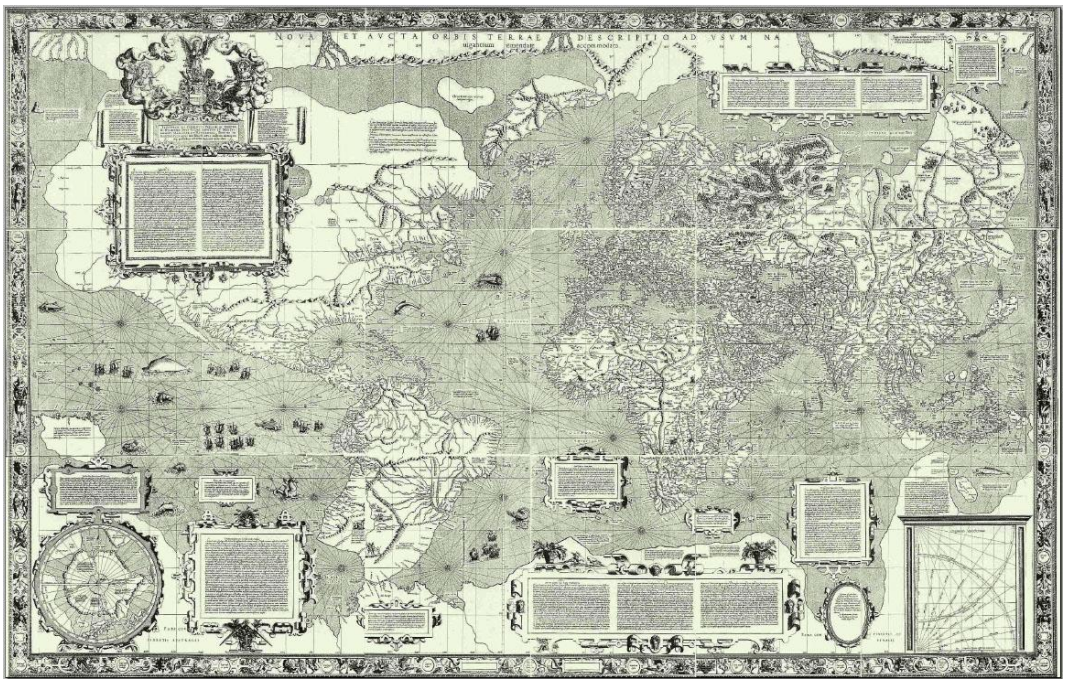
$$x = \lambda R$$

$$y = y(\phi)R$$

Kde $R$ je poloměr Země (glóbu, záleží na měřítku...)

Mercator druhou rovnici (pro $y$) v 16. století neznal, chyběl mu mocný prostředek, infinitezimální počet, který nám o století později poskytl **Isaac Newton** a **Gottfried Wilhelm Leibniz**. Podle pravidel 1), 2) a znalosti elementární trigonometrie byl však

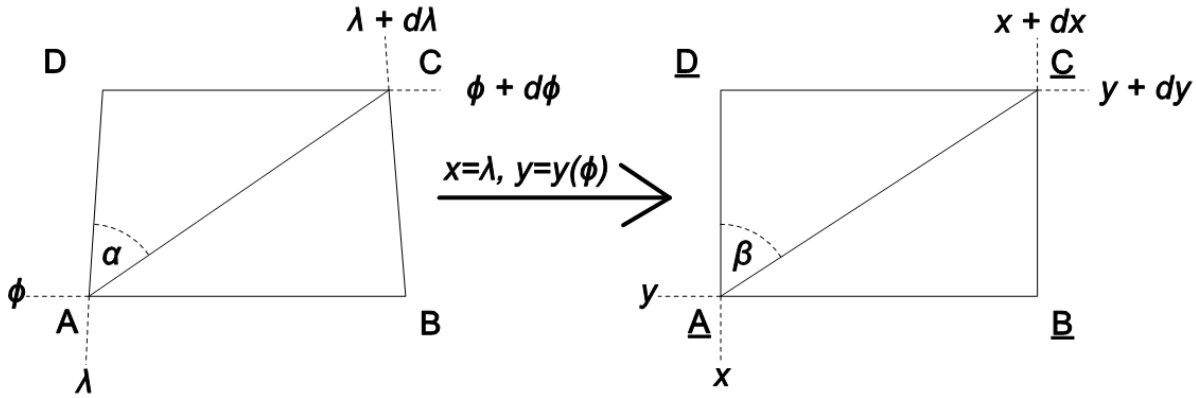
schopen graficky vynést rozumnou aproximaci pravoúhlé sítě poledníků a rovnoběžek, do které vynášel polární souřadnice objektů.



*Mercator 1569 - Nova et Aucta Orbis Terrae Descriptio ad Usum Navigantium Emendate Accommodata*

Odvoďme tedy rovnici $y = y(\phi)$ pro $R = 1$ (měřítkový faktor můžeme doplnit kdykoli).

Pomůže nám následující obrázek. Body A, B, C, D z polární projekce budou transformovány do bodů (roviny) A, B, C, D. Předpokládejme navíc, že zobrazované body jsou od sebe velmi málo („infinitezimálně“) vzdáleny, lokální kulovou plochu můžeme považovat za rovinu.




- Délky oblouku $d\phi$ na poledníku se v závislosti na poloze $[\phi, \lambda]$ na kouli nemění, pro malé hodnoty můžeme jejich průmět do tečné roviny aproximovat přímo hodnotou $d\phi$.

- Délky oblouku $d\lambda$ rovnoběžce jsou závislé na šířce $\phi$. Pro malé hodnoty je můžeme aproximovat hodnotou $cos(\phi)\, d\lambda$ ($cos(\phi)$ je poloměr rovnoběžky a šířce $\phi$).

- Můžeme tedy položit:

$$tan(\alpha) = \frac{d\lambda cos(\phi)}{d\phi}$$

$$tan(\beta) = \frac{dx}{dy}$$

- První rovnici upravíme do tvaru:

$$d\lambda = \frac{tan(\alpha) d\phi}{cos(\phi)}$$

- Z definice projekce $x = \lambda$ okamžitě plyne $dx = d\lambda$, dosazením dostáváme:

$$tan(\beta) = \frac{tan(\alpha) d\phi}{cos(\phi) dy}$$

- Požadavek "Loxodromy zachovávají azimut" lze vyjádřit rovností $\alpha = \beta$, dostaneme tedy jednoduchou diferenciální rovnici:

$$y' = \frac{dy}{d\phi} = \frac{1}{cos(\phi)}$$

Tato rovnice není překvapivá. Uvědomíme-li si, že $2\pi cos(\phi)$ je délka rovnoběžky na šířce $\phi$, můžeme rovnici interpretovat:

„*Změna vertikálních vzdáleností rovnoběžek v Mercatorově projekci je nepřímo úměrná jejich délce*".

Rovnice má bohatou historii. V roce 1599 Edward Wright sestavil tabulky pro hodnoty hledané funkce pomocí tehdejších numerických metod. Učitel navigace **Henry Bond**, ve 40-tých letech 17. století, na základě této tabulky, zformuloval domněnku:

$$y = ln(tan\left(\frac{\pi}{4} + \frac{\phi}{2}\right))$$

Problémem se zabýval v roce 1665 i Isaac Newton. Zdá se, že v polovině 17. století byla tato domněnka, pro její roli v mapování a navigaci, skutečným „matematickým" hitem. Její důkaz, tj. integraci $\int sec(x) dx$, podal v roce 1668 skotský matematik a astronom **James Gregory**.

pramen: https://en.wikipedia.org/wiki/Mercator_projection a https://en.wikipedia.org/wiki/Integral_of_the_secant_function

## 1.4. VÝVOJ NAVIGAČNÍCH POMŮCEK

### Základní pojmy

Substelární bod – je bod na zemském povrchu, ze kterého se objekt (hvězda) jeví v zenitu. Pro Slunce resp. Měsíc se také používají termíny subsolární bod resp. sublunární bod. Substelární body se pro nebeské objekty mění v čase.

Greenwich Hour Angle, $GHA$ - je úhlová vzdálenost nebeského tělesa od nultého poledníku podél nebeského rovníku vztažená k času (GMT).

Deklinace, $Dec$ - úhlová vzdálenost objektu od roviny světového rovníku.

Dvě souřadnice $[Dec, 360° - GHA]$ jsou polární souřadnice $[\phi, \lambda]$ substelárního bodu nebeského tělesa v daném čase.

Z pohledu těchto lze Eratosthenovu metodu změření velikosti Země popsat takto:

- Za slunovratu je v poledne subsolární bod v Syéné.
- Ze znalosti vzdálenosti Alexandrie – Syéné a úhlu Slunce nad obzorem mohu tedy dopočítat obvod Země.

Poloha bodového objektu na Zemi je určena úhlovými souřadnicemi $[\phi, \lambda]$. Proto není překvapivé, že historické navigační pomůcky jsou úhloměrná zařízení.

### Úhloměry

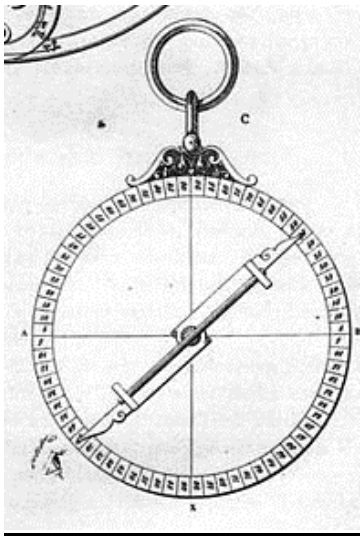
Kamal – dřevěná destička, v jejímž středu je upevněn provázek. Na provázku jsou uzly, které reprezentují zeměpisnou šířku. Provázek se chytne do zubů v místě uzlu s cílovou šířkou. Spodní hrana destičky se ztotožní s horizontem. Je-li Polárka pod horní hranou, jsme jižněji, naopak severněji. Splývá-li Polárka s horní hranou, jsme na správné šířce.

*Kamal (obrázek z http://www.lode.cz/re.php)*

Jakubova hůl – Posuvná destička na tyči, na které je vyznačena stupnice. Její dolní okraj ztotožníme posouváním s horizontem a horní s měřeným objektem, oko je na konci hole. Na tyči potom odečteme údaj. Byla používána od starověku až do 19. století.

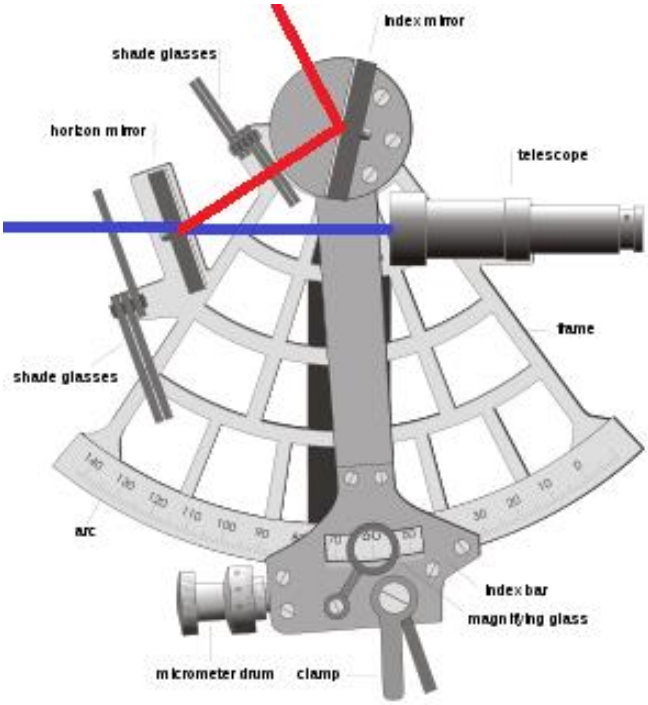

*Jakubova hůl, John Sellers - Practical Navigation (1672)*

Námořní astroláb – závěsná kruhová deska s otočným ramenem pro odměření úhlu proti horizontu.




*Námořní astroláb (obrázek z http://cs.wikipedia.org)*

Sextant – optický přístroj využívající polopropustné zrcadlo ke ztotožnění měřeného objektu s horizontem. Princip byl objeven nezávisle Isaacem Newtonem a Johnem Hadleym. Poskytoval řádově přesnější výsledky (jednotky úhlových minut), než všechny dosavadní přístroje. Poznamenejme, že sextantem lze smysluplně měřit úhel vzdálených objektů, tj. u kterých je zanedbatelná paralaxa způsobená rozdílnou polohou zrcátek.




*Sextant (zdroj http://en.wikipedia.org/wiki/sextant)*

Určování šířky

Historicky se používaly nebeské objety, u kterých se substelární bod neměnil v čase (resp. měnil málo), například Polárka. Například, změřený úhel polárky proti horizontu je (+/-) přímo šířka.

Určování délky

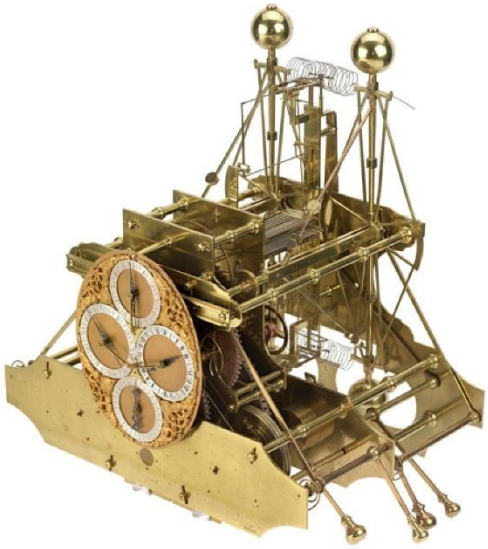
Je problém výrazně komplikovanější, metody vychází z měření místního času, resp. místního poledne. K tomu však potřebujeme „univerzální hodiny“. V historii řešení tohoto problému najdeme skutečně významné osobnosti – **Gallileo Galilei**, **Edmont Halley**, **Leonhard Euler**…

Za zmínku také stojí, že odměna za uspokojivé vyřešení byla uzákoněna parlamentem Spojeného království v roce 1714 (Longitude Act).

První zaznamenaná metoda pochází od Ptolemaia a je založena na skutečnosti, že Měsíc se pohybuje vůči hvězdám (zhruba rychlostí svého poloměru za hodinu). Stačí tedy tabulkově zaznamenat jeho pozici vůči jasným hvězdám v časech nultého poledníku a můžeme (opět úhloměrnými pomůckami), kdekoli stanovit čas nultého poledníku.
Místní čas potom stanovíme okamžikem místního poledne (slunce je nejvýše nad obzorem) a zeměpisnou délku podle rozdílu místního času a času nultého poledníku (24 h = +/- 360°).

Tato metoda byla značně nepřesná, proto i značné chyby v délkách v Ptolemaiových mapách.

Až do vynálezu „přesných“ hodin nebyl však problém určení uspokojivě vyřešen. Přesné hodiny (chyba v minutách za týdny provozu) byly sestrojeny **Johnem Harissonem** (1693–1776).



*Harrisonovy lodní hodiny*
*zdroj:* https://www.rmg.co.uk/discover/explore/longitude-found-john-Harrison

K vyřešení problému určení zeměpisné délky výraznou měrou také přispěly výsledky **Johannese Keplera** a Isaaca Newtona vysvětlující pohyb Země kolem slunce.

Metody určení pozice s úhloměrem a hodinami

K určení pozice na Zemi tedy použijeme:

1) Hodiny ukazující přesný čas (nultého poledníku).

2) Úhloměrné zařízení (sextant).

3) Metodu pro určení substelárního bodu.

Metoda místního poledne.

Provedeme měření úhlu $\alpha$ slunce nad horizontem a zaznamenáme čas $t$ (GMT) v kulminaci. Čas kulminace je roven času místního poledne. Substelární bod $[dec, 360° - GHA]$ leží na našem poledníku. Úhlová vzdálenost k substelárnímu bodu je $\frac{\pi}{2} - \alpha$ a tedy naše šířka je:

$$\phi = \frac{\pi}{2} - \alpha + dec$$

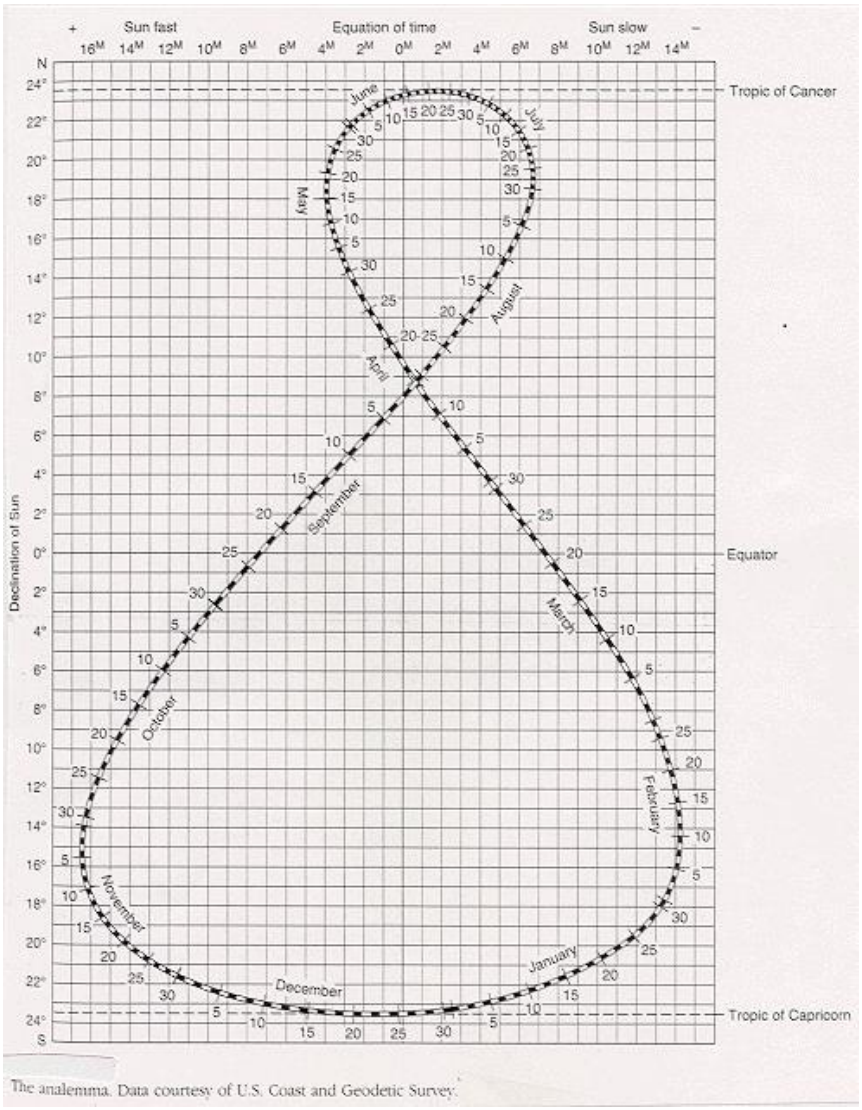
Nabízí se použít přímo čas našeho místního poledne $t$ pro výpočet naší délky. Vlivem sklonu zemské osy vůči rovině ekliptiky a 2. Keplerova zákona, se však čas

místního poledne (čas kulminace) různí od středního poledne (čas na hodinkách) až o cca. 16 minut.

Rozdíl času oproti nultému poledníku tedy musíme korigovat.
Provedeme tedy korekci času $t$. Označíme-li $\Delta t$ rozdíl, mezi korigovaným časem a polednem v hodinách potom je naše délka ve stupních (za hodinu 15 stupňů):

$$\lambda = 15(t + \Delta t)$$

Korekční tabulku může velmi zhruba nahradit křivka $analema$, kde na jedné ose je zobrazena deklinace ($dec$), na druhé časová diference místního poledne proti střednímu poledni. Na křivce je potom vyznačený průběh roku.




*Analema – zdroj U.S. Coast and Geodetic Survey*

Na první pohled jsou korekce podle analemy velmi hrubé. Nezohledňují hodiny, resp. přestupné roky a vzhledem k tomu, že korekce se mění poměrně dramaticky, v určitém období i o 30 úhlových minut za den, se pro praktickou navigaci nehodila. Používaly přesnější tabulky vydávané pro každý rok v tzv. Námořním almanachu. V

tomto jsou pro jednotlivé dny rozepsány potřebné údaje po hodinách. To neznamená, že můžeme měřit jenom v celé hodiny – lineární aproximace mezi dvěma řádky v almanachu je dostačující, chyba aproximace je zanedbatelná.

**2019 August 11 to Aug. 13**

| h | Sun | | Moon | | | | |
|---|---|---|---|---|---|---|---|
| Sun | GHA | Dec | GHA | ν | Dec | d | HP |
| 0 | 178°39.9 | N15°24.8 | 51°52.7 | 9.8′ | S21°37.6 | -3.3′ | 55.8′ |
| 1 | 193°40.0 | 24.1 | 66°21.6 | 9.8′ | 21°40.8 | -3.2′ | 55.8′ |
| 2 | 208°40.1 | 23.3 | 80°50.4 | 9.8′ | 21°43.9 | -3.0′ | 55.7′ |
| 3 | 223°40.2 | ·· 22.6 | 95°19.3 | 9.8′ | 21°46.9 | -2.9′ | 55.7′ |
| 4 | 238°40.3 | 21.9 | 109°48.1 | 9.8′ | 21°49.8 | -2.8′ | 55.7′ |
| 5 | 253°40.4 | 21.1 | 124°16.9 | 9.8′ | 21°52.6 | -2.7′ | 55.7′ |
| 6 | 268°40.5 | N15°20.4 | 138°45.8 | 9.8′ | S21°55.2 | -2.6′ | 55.6′ |
| 7 | 283°40.6 | 19.7 | 153°14.6 | 9.9′ | 21°57.7 | -2.5′ | 55.6′ |
| 8 | 298°40.7 | 18.9 | 167°43.5 | 9.9′ | 22°00.2 | -2.4′ | 55.6′ |
| 9 | 313°40.8 | ·· 18.2 | 182°12.4 | 9.9′ | 22°02.5 | -2.2′ | 55.6′ |
| 10 | 328°40.9 | 17.4 | 196°41.2 | 9.9′ | 22°04.6 | -2.1′ | 55.6′ |
| 11 | 343°41.0 | 16.7 | 211°10.1 | 9.9′ | 22°06.7 | -2.0′ | 55.5′ |
| 12 | 358°41.1 | N15°16.0 | 225°39.0 | 9.9′ | S22°08.6 | -1.9′ | 55.5′ |
| 13 | 13°41.2 | 15.2 | 240°07.9 | 9.9′ | 22°10.5 | -1.8′ | 55.5′ |
| 14 | 28°41.3 | 14.5 | 254°36.8 | 9.9′ | 22°12.2 | -1.7′ | 55.5′ |
| 15 | 43°41.4 | ·· 13.7 | 269°05.7 | 9.9′ | 22°13.8 | -1.5′ | 55.4′ |
| 16 | 58°41.5 | 13.0 | 283°34.6 | 9.9′ | 22°15.3 | -1.4′ | 55.4′ |
| 17 | 73°41.6 | 12.2 | 298°03.5 | 9.9′ | 22°16.6 | -1.3′ | 55.4′ |
| 18 | 88°41.7 | N15°11.5 | 312°32.4 | 9.9′ | S22°17.9 | -1.2′ | 55.4′ |
| 19 | 103°41.8 | 10.8 | 327°01.4 | 10.0′ | 22°19.0 | -1.1′ | 55.4′ |
| 20 | 118°41.9 | 10.0 | 341°30.3 | 10.0′ | 22°20.1 | -1.0′ | 55.3′ |
| 21 | 133°42.0 | ·· 09.3 | 355°59.3 | 10.0′ | 22°21.0 | -0.8′ | 55.3′ |
| 22 | 148°42.1 | 08.5 | 10°28.3 | 10.0′ | 22°21.8 | -0.7′ | 55.3′ |
| 23 | 163°42.2 | 07.8 | 24°57.3 | 10.0′ | 22°22.4 | -0.6′ | 55.3′ |
| | SD.=15.8 | d=-0.7 | S.D.=15.2 | | | | |
| Mon | GHA | Dec | GHA | ν | Dec | d | HP |
| 0 | 178°42.3 | N15°07.0 | 39°26.3 | 10.0′ | S22°23.0 | -0.5′ | 55.3′ |
| 1 | 193°42.4 | 06.3 | 53°55.4 | 10.0′ | 22°23.4 | -0.4′ | 55.2′ |
| 2 | 208°42.6 | 05.5 | 68°24.4 | 10.1′ | 22°23.8 | -0.3′ | 55.2′ |
| 3 | 223°42.7 | ·· 04.8 | 82°53.5 | 10.1′ | 22°24.0 | -0.2′ | 55.2′ |
| 4 | 238°42.8 | 04.0 | 97°22.6 | 10.1′ | 22°24.1 | -0.0′ | 55.2′ |
| 5 | 253°42.9 | 03.3 | 111°51.7 | 10.1′ | 22°24.1 | 0.1′ | 55.2′ |

| Lat. | Twilight Naut. | Twilight Civil | Sunrise | Sunset | Twilight Civil | Twilight Naut. |
|---|---|---|---|---|---|---|
| N 72° | ▭ | ▭ | 01:59 | 22:04 | ▭ | ▭ |
| N 70° | ▭ | ▭ | 02:39 | 21:27 | ▭ | ▭ |
| 68° | ▭ | 01:13 | 03:05 | 21:02 | 22:48 | ▭ |
| 66° | ▭ | 02:03 | 03:25 | 20:42 | 22:03 | ▭ |
| 64° | ▭ | 02:33 | 03:42 | 20:26 | 21:34 | ▭ |
| 62° | 01:05 | 02:55 | 03:55 | 20:13 | 21:12 | 22:57 |
| 60° | 01:50 | 03:13 | 04:06 | 20:02 | 20:55 | 22:16 |
| N 58° | 02:18 | 03:27 | 04:16 | 19:53 | 20:41 | 21:49 |
| 56° | 02:38 | 03:40 | 04:24 | 19:44 | 20:29 | 21:29 |
| 54° | 02:55 | 03:50 | 04:32 | 19:37 | 20:18 | 21:13 |
| 52° | 03:09 | 04:00 | 04:39 | 19:30 | 20:09 | 20:59 |
| 50° | 03:21 | 04:08 | 04:45 | 19:24 | 20:01 | 20:47 |
| 45° | 03:45 | 04:25 | 04:58 | 19:12 | 19:44 | 20:24 |
| N 40° | 04:03 | 04:39 | 05:08 | 19:01 | 19:30 | 20:06 |
| 35° | 04:18 | 04:50 | 05:17 | 18:52 | 19:19 | 19:52 |
| 30° | 04:30 | 05:00 | 05:25 | 18:44 | 19:10 | 19:40 |
| 20° | 04:49 | 05:16 | 05:39 | 18:31 | 18:54 | 19:21 |
| N 10° | 05:03 | 05:29 | 05:51 | 18:19 | 18:41 | 19:07 |
| 0° | 05:15 | 05:40 | 06:02 | 18:08 | 18:30 | 18:55 |
| S 10° | 05:26 | 05:51 | 06:13 | 17:58 | 18:19 | 18:45 |
| 20° | 05:35 | 06:01 | 06:24 | 17:46 | 18:09 | 18:35 |
| 30° | 05:44 | 06:12 | 06:37 | 17:34 | 17:58 | 18:27 |
| 35° | 05:48 | 06:18 | 06:44 | 17:26 | 17:53 | 18:23 |
| 40° | 05:52 | 06:24 | 06:53 | 17:18 | 17:46 | 18:18 |
| 45° | 05:57 | 06:32 | 07:02 | 17:08 | 17:39 | 18:14 |
| S 50° | 06:02 | 06:40 | 07:14 | 16:57 | 17:31 | 18:09 |
| 52° | 06:04 | 06:44 | 07:20 | 16:51 | 17:27 | 18:07 |
| 54° | 06:06 | 06:48 | 07:26 | 16:45 | 17:23 | 18:05 |
| 56° | 06:08 | 06:52 | 07:32 | 16:39 | 17:19 | 18:03 |
| 58° | 06:11 | 06:57 | 07:40 | 16:31 | 17:14 | 18:01 |
| S 60° | 06:13 | 07:03 | 07:48 | 16:23 | 17:08 | 17:58 |

*Námořní almanach – zdroj [http://thenauticalalmanac.com](http://thenauticalalmanac.com)*

Tabulky v Námořním almanachu nám zejména poskytují souřadnice substelárního (subsolárního, sublunárního) bodu planet, navigačních hvězd, Slunce a Měsíce vzhledem k času GMT a časové korekce.

Zeměpisnou délku v metodě místního poledne můžeme získat také přímo ze sloupce $GHA$ – substelární bod leží na našem místním poledníku. Čas kulminace však není úplně jednoduché odhadnout. Chyba v odhadu poledne v řádu 1 minuty nám způsobí v naší zeměpisné šířce chybu polohy cca 18 km.

Metoda pozičních kružnic.

Opět použijeme úhloměrné zařízení, Námořní almanach a přesné hodinky. Změříme úhel navigačního objektu (hvězda, planeta, slunce) proti horizontu. Geometrické místo na zemském povrchu, ze kterého je vidět objekt pod změřeným úhlem je kružnice se středem v příslušném substelárním bodě. Její (úhlový) poloměr je $\pi/2$ mínus změřený úhel. Takové kružnice získáme dvě, tři a více. Společný průsečík je naše poloha.

Rovnice pro výpočet průsečíku pozičních kružnic:

Vstup:
$[\phi_1, \lambda_1]$ – pozice substelárního bodu 1. objektu (z almanachu)
$[\phi_2, \lambda_2]$ – pozice substelárního bodu 2. objektu (z almanachu)
$\alpha_1$ – naměřený úhel 1. objektu proti horizontu
$\alpha_2$ – naměřený úhel 2. objektu proti horizontu

$\delta_1 = \frac{\pi}{2} - \alpha_1$ – úhlová vzdálenost k 1. bodu
$\delta_2 = \frac{\pi}{2} - \alpha_2$ – úhlová vzdálenost k 2. bodu

Převod do kartézské soustavy:
Převedeme substelární body do kartézské soustavy na jednotkovou kouli se středem v nule (skutečný poloměr Země nemusíme zahrnovat, jde nám o jednotkové směrové vektory). Rovnice pro tento převod mají následující tvar:

$$F(\phi, \lambda) = [x, y, z] \qquad (0)$$

kde:

$$x = cos(\lambda)\, cos(\phi)$$
$$y = sin(\lambda)\, cos(\phi)$$
$$z = sin(\phi)$$

(pro zvídavé čtenáře: zkuste si tento převod ověřit, stačí tužka a papír, začněte od "$z$")
Pomocí těchto rovnic převedeme oba substelární body do jednotkových vektorů $[a_1, b_1, c_1]$ resp. $[a_2, b_2, c_2]$:

$$[a_1, b_1, c_1] = F(\phi_1, \lambda_1)$$
$$[a_2, b_2, c_2] = F(\phi_2, \lambda_2)$$

Sestavení rovnic:
Nyní můžeme sestavit rovnice, jejichž řešení nám přinese kartézskou reprezentaci hledaných průsečíků na jednotkové kouli. Položme nejprve:

$$d_1 = cos\ (\delta_1)$$
$$d_2 = cos(\delta_2)$$

a sestavme soustavu rovnic. Použijeme formule, která nám říká, že skalární součin dvou jednotkových vektorů se rovná kosinu úhlu jimi sevřených:

$$x^2 + y^2 + z^2 = 1 \qquad (1)$$
$$a_1 x + b_1 y + c_1 z = d_1 \qquad (2)$$
$$a_2 x + b_2 y + c_2 z = d_2 \qquad (3)$$

(1) Bod leží na jednotkové kouli.
(2) Úhel k 1. substelárnímu bodu je $\delta_1$.
(3) Úhel k 2. substelárnímu bodu je $\delta_2$.

Výpočet:
Rovnice (2) a (3) jsou lineární, lze z nich tedy snadno (např. Cramerovým pravidlem) odvodit $x$, resp. $y$ v závislosti na $z$.

$$x = e_1 z + f_1 \quad (4)$$
$$y = e_2 z + f_2 \quad (5)$$

kde:

$$e_1 = \frac{b_1 c_2 - b_2 c_1}{|S|},\ f_1 = \frac{b_2 d_1 - b_1 d_2}{|S|}$$

$$e_2 = \frac{a_2 c_1 - a_1 c_2}{|S|},\ f_2 = \frac{a_1 d_2 - a_2 d_1}{|S|}$$

$$|S| = a_1 b_2 - a_2 b_1$$

Substitucí (4) a (5) do (1) dostaneme kvadratickou rovnici

$$Az^2 + Bz + C = 0 \quad (6)$$

kde:

$$A = e_1^2 + e_2^2 + 1,\ B = 2(e_1 f_1 + e_2 f_2), C = f_1^2 + f_2^2 - 1$$

Řešení:

Vyřešíme kvadratickou rovnici (6) a dostaneme dvě hodnoty

$$z_{1,2} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

Z rovnic (4) a (5) dostaneme kartézské souřadnice pro průsečíky $[x_1, y_1, z_1]$ a $[x_2, y_2, z_2]$. Pro zpětný převod do polárních souřadnic použijeme triviálně odvoditelnou inverzní funkci:

$$F^{-1}(x, y, z) = [\phi, \lambda]$$

$$\phi = sin^{-1}(z)$$

$$\lambda = cos^{-1}\left(\frac{x}{cos\ (\phi)}\right)$$

(ponechávám čtenáři vyřešení případu, kdy se nacházíme na pólech a tedy $cos(\phi) = 0$).

Přímý výpočet je pro praktickou navigaci příliš komplikovaný, představme si, že nemáme po ruce žádný „počítač“ nebo kalkulačku, který/á za nás uvedený postup provede, máme jen tabulky, tužku a papír.

Dále, není možné vynášet do map kružnice „graficky“. Jednak jsou jejich poloměry hodně velké (tisíce km, tedy i v měřítku 1:1000 000 bychom museli mít několikametrové kružítko a mapu), jednak obrazem kružnice na zemském povrchu ve 2D mapě není kružnice.

Proto se používala metoda pozičních linií (interceptu), ve které jsou kružnice nahrazeny tečnami – ty už lze do mapy vynášet pravítkem. Postup je následující:

- Změříme úhel nebeského objektu a zaznamenáme čas měření a z almanachu určíme substelární bod SP.

- Odhadneme naši polohu bodem AP (Assumed Position).

- Z almanachu odečteme souřadnice substelárního bodu SP v zaznamenaném čase.

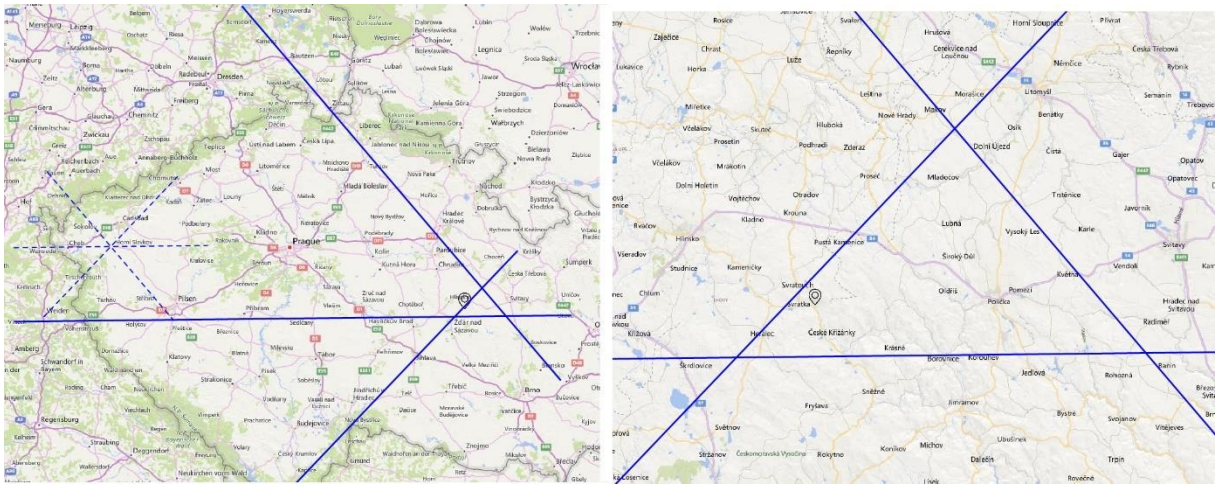
- Určíme azimut AP-SP, kolmice němu je (+/-) rovnoběžná s tečnou poziční kružnice.

- Úhlová délka naší skutečné pozice k SP je doplněk do 90° námi změřeného úhlu, úhlovou délku AP-SP lze spočítat. Tečnu z předešlého bodu posuneme ve směru azimutu podle diference úhlových délek.

- Postup opakujeme pro jiný objekt a tím dostaneme více tečen. Jejich „průsečík“ je naše pozice.

- Azimut, resp. úhlové vzdálenosti počítáme pomocí vzorců sférické geometrie.

Metoda má tu nespornou výhodu, že je použitelná v libovolném čase, ve dne můžeme měřit Slunce v různých časech.



*Metoda interceptu ze tří měření*

Sférická geometrie – úhlová vzdálenost a azimut

Úhlová vzdálenost bodů na kouli:

$[\phi_1, \lambda_1]$ – pozice 1. bodu
$[\phi_2, \lambda_2]$ – pozice 2. bodu
$\delta$ – úhlová vzdálenost

Metoda – spočteme kartézské souřadnice 1. a 2. bodu podle (0). Skalární součin těchto bodů (vektorů) je kosinus hledané úhlové vzdálenosti, tedy:

$$cos(\delta) =$$

$$cos(\lambda_1)cos(\phi_1)cos(\lambda_2)cos(\phi_2) + sin(\lambda_1)cos(\phi_1)sin(\lambda_2)cos(\phi_2) + sin(\phi_1)sin(\phi_2) =$$

$$cos(\phi_1)cos(\phi_2)(cos(\lambda_1)cos(\lambda_2) + sin(\lambda_1)sin(\lambda_2)) + sin(\phi_1)sin(\phi_2)$$

$$\delta = cos^{-1}(cos(\phi_1)cos(\phi_2)cos(\lambda_2 - \lambda_1) + sin(\phi_1)sin(\phi_2)) \quad (7)$$

Výchozí azimut z bodu do bodu (výchozí úhel ortodomy):

Použijeme základní formuli sférické trigonometrie, větu kosinovou.

Věta kosinová: Pro libovolný trojúhelník ABC na kulové ploše (tj. strany jsou ortodromy) platí:

$$\frac{sin(a)}{sin(\alpha)} = \frac{sin(b)}{sin(\beta)} = \frac{sin(c)}{sin(\gamma)}$$

Pozn: strany $a, b, c$ jsou na kouli také úhly.

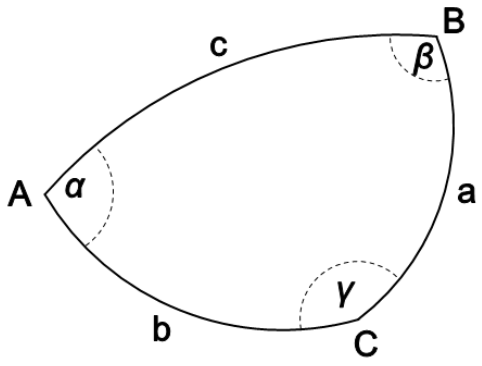
Předpokládejme nyní, že:

A – naše pozice $[\phi_0, \lambda_0]$

B – je severní pól

C – pozice substelárního bodu $[\phi_1, \lambda_1]$

$\alpha$ – výchozí azimut



Zřejmě:

$\beta = \lambda_1 - \lambda_0$

$b = \delta$ (úhlová vzdálenost AC z předešlého výpočtu)

$a = \frac{\pi}{2} - \phi_1$

$$\frac{sin(a)}{sin(\alpha)} = \frac{sin(b)}{sin(\beta)}$$

tedy,

$$sin(\alpha) = \frac{sin(a)\, sin(\beta)}{sin(b)}$$

$$\alpha = sin^{-1}(\frac{cos(\phi_1)\, sin(\lambda_1 - \lambda_0)}{sin(\delta)}) \quad (8)$$

Majáky

Metody navigace byly v průběhu vývoje doplňovány pozemními navigačními body.

Optické majáky pro lokální navigaci. Byly budovány od Říše římské, dosud hojně používané na pobřežích všech moří.

Radiomajáky. Síť budována od druhé poloviny 20. století, hlavní navigační metoda pro leteckou dopravu. Původní nesměrový systém NDB, který vyžadoval směrovou anténu přijímače, je postupně nahrazován systémem VOR, u kterého směrová anténa nutná není. Pomocí statické a rotační antény, VOR systém vysílá směrově závislý signál, přijímač je schopen určit azimut majáku nezávisle na síle signálu.

Global Positioning System **GPS**

(Viz https://en.wikipedia.org/wiki/Global_Positioning_System)
V současné době je nejmasovějším prostředkem pro určení polohy systém GPS. Je provozován Ministerstvem obrany USA, vývoj se datuje od první poloviny 70. let 20. století.
Jeho využívání pro civilní účely je bohužel spjato s tragickým osudem 269 pasažérů letu KAL 007. Civilní let KAL 007, společnosti Korean Air se vlivem navigační chyby v roce 1983 odchýlil o běžné trasy a dostal se tak do sovětského vzdušného prostoru. Co způsobilo chybu není jasné, snad chybně zadaná trasa do palubního počítače. V době studené války se tato chyba stala osudnou. Civilní let byl sovětskou stíhačkou sestřelen. Po tomto hrůzném incidentu oznámil americký prezident Ronald Reagan, že po dokončení bude systém GPS bude k dispozici i pro civilní účely.

GPS je založen na 32 satelitech schopných vysílat svoji přesnou polohu a přesný čas. Čas je pro všechny satelity velmi přesně synchronizován. Přijímače mají hodiny, které nejsou synchronizovány se satelity, ale jsou schopny měřit velmi přesně časový interval.

Úsměvný se z dnešního pohledu jeví počet satelitů – pro jejich identifikaci bylo rezervováno 5 bitů, přidání dalšího by znamenalo změnit komunikační protokol. Pro základní představu uvedeme analogii v 1D prostoru.

Příklad – 1D GPS:
Bedřich, Jan a Ctirad mají jazzovou kapelu a shodou okolností bydlí po řadě v Bedřichově Janově a Ctiradově. Tato sídla jsou na jedné rovné linii železniční trati Bedřichov–Janov–Ctiradov. Janov je třeba na půli cesty, vzdálenosti všichni dobře znají.
Bedřich a Ctirad jsou dříči a pedanti, každou celou hodinu (na milisekundy!) si přehrají nutné etudy. Bedřich vždy začíná tónem B. Není divu, jeho nástroj je soprán saxofon. Ctirad hraje C-kornet, snad proto jeho cvičení začíná vždy tónem C.
Zato Jan, to je bohém. Vůbec necvičí, ale má absolutní hudební sluch a geniální cit pro rytmus. A ta jeho životospráva... Jednou po zkoušce trochu přebral a ujel mu vlak. I vydal se podél trati a po cestě usnul. Po probuzení byl zoufalý, nevěděl, odkud vyšel a kterým směrem se vydat. Až najednou, Bedřich a Ctirad začali cvičit – a Jan najednou ví, kde je… Zaznamenal časový rozdíl, kdy zaslechl Bedřichův a Ctiradův nástroj.

Poznámka: Uvedený příklad lze vyřešit triviální lineární rovnicí, ve více dimenzích je to obtížnější. Řešení vede na soustavu nelineárních rovnic. Zkusme si nyní tyto rovnice sestavit. Pro jednoduchost zanedbáme některé faktory, které se musí do reálného výpočtu zahrnout:

- čas vyslání signálu satelitu je mírně zpožděn oproti poloze
- ionosférická a troposférická refrakce elektromagnetických vln
- vliv ionosféry a troposféry na rychlost světla
- relativistické vlivy, satelity se pohybují nezanedbatelnou rychlostí v gravitačním poli Země

Mějme tedy 4 satelity, které vyslaly svoji polohu a čas:

$$[x_i, y_i, z_i, t_i], i = 1,..,4$$

Tyto signály zachytil přijímač v časech svých hodin:

$$\tau_i, i = 1,..,4$$

Hodiny přijímače nejsou přesně synchronizovány s hodinami satelitů. Předpokládejme ale, že se jejich rozdíl během měření nezměnil a označme ho symbolem $\tau$.

Označme hledanou pozici:

$$\boldsymbol{X} = [x, y, z]$$

a

$r_i$ – vzdálenost hledané pozice od satelitu $i$.

Je

$$r_i = r_i(\boldsymbol{X}) = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}$$

a

$$r_i = (\tau_i - t_i)c - \tau c$$

kde $c$ je rychlost světla.

Dostáváme soustavu nelineárních rovnic, $i = 1,..,4$:

$$\sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} + \tau c = (\tau_i - t_i)c \quad (1)$$

Označme levou stranu rovnic:

$$F_i(x, y, x, \tau) = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} + \tau c$$

Soustavu rovnic:

$$F_i(x, y, x, \tau) = (\tau_i - t_i)c$$

nyní linearizujeme (abychom ji vůbec mohli nějak řešit). To provedeme tak, že funkci $F_i(x, y, x, \tau)$ aproximujeme lineárním členem Taylorova rozvoje v nějakém bodě $\boldsymbol{X}_0 = [x_0, y_0, z_0]$ a časovým posunem hodin přijímače od satelitů $\tau_0$. Poznamenejme, že tento postup není výlučný pro výpočet polohy přijímače GPS, jedná se o aplikaci Newton-Raphsonovy metody řešení nelineárních rovnic, kdy jsou známé první parciální derivace levých stran, viz Příloha 9.1.

$$F_i(\boldsymbol{X}, \tau) \approx$$

$$F_i(\boldsymbol{X}_0, \tau_0) + \frac{\partial F_i}{\partial x}(\boldsymbol{X}_0, \tau_0)(x - x_0) + \frac{\partial F_i}{\partial y}(\boldsymbol{X}_0, \tau_0)(y - y_0) +$$

$$\frac{\partial F_i}{\partial z}(\boldsymbol{X}_0, \tau_0)(z - z_0) + \frac{\partial F_i}{\partial \tau}(\boldsymbol{X}_0, \tau_0)(\tau - \tau_0)$$

Zřejmě (prostou parciální derivací podle $x, y, z, \tau$) je:

$$\frac{\partial F_i}{\partial x}(\boldsymbol{X}_0, \tau_0) = \frac{x_0 - x_i}{r_i(\boldsymbol{X}_0)}, \qquad \frac{\partial F_i}{\partial y}(\boldsymbol{X}_0, \tau_0) = \frac{y_0 - y_i}{r_i(\boldsymbol{X}_0)}$$

$$\frac{\partial F_i}{\partial z}(\boldsymbol{X}_0, \tau_0) = \frac{z_0 - z_i}{r_i(\boldsymbol{X}_0)}, \qquad \frac{\partial F_i}{\partial \tau}(\boldsymbol{X}_0, \tau_0) = c$$

Označíme-li $\Delta x = x - x_0, \Delta y = y - y_0, \Delta z = z - z_0, \Delta\tau = \tau - \tau_0$ dostáváme soustavu lineárních rovnic:

$$\begin{bmatrix} \frac{x_0 - x_1}{r_1(\boldsymbol{X}_0)} & \frac{y_0 - y_1}{r_1(\boldsymbol{X}_0)} & \frac{z_0 - z_1}{r_1(\boldsymbol{X}_0)} & c \\ \frac{x_0 - x_2}{r_2(\boldsymbol{X}_0)} & \frac{y_0 - y_2}{r_2(\boldsymbol{X}_0)} & \frac{z_0 - z_2}{r_2(\boldsymbol{X}_0)} & c \\ \frac{x_0 - x_3}{r_3(\boldsymbol{X}_0)} & \frac{y_0 - y_3}{r_3(\boldsymbol{X}_0)} & \frac{z_0 - z_3}{r_3(\boldsymbol{X}_0)} & c \\ \frac{x_0 - x_4}{r_4(\boldsymbol{X}_0)} & \frac{y_0 - y_4}{r_4(\boldsymbol{X}_0)} & \frac{z_0 - z_4}{r_4(\boldsymbol{X}_0)} & c \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta\tau \end{bmatrix} = \begin{bmatrix} (\tau_1 - t_1)c - F_1(\boldsymbol{X}_0, \tau_0) \\ (\tau_2 - t_2)c - F_2(\boldsymbol{X}_0, \tau_0) \\ (\tau_3 - t_3)c - F_3(\boldsymbol{X}_0, \tau_0) \\ (\tau_4 - t_4)c - F_4(\boldsymbol{X}_0, \tau_0) \end{bmatrix} \tag{2}$$

Řešení soustavy (1) dostaneme iteračním procesem. Odhadneme počáteční hodnoty $(\boldsymbol{X}_0, \tau_0)$ a řešíme systém (2). Dostaneme „opravu“ $\Delta(\boldsymbol{X}, \tau) = \Delta x, \Delta y, \Delta z, \Delta\tau$ pro následující krok iterace použijeme odhad $(\boldsymbol{X}_0, \tau_0) := (\boldsymbol{X}_0, \tau_0) + \Delta(\boldsymbol{X}, \tau)$. Proces opakujeme, dokud $|\Delta(\boldsymbol{X}, \tau)| \approx 0$. Pro počáteční odhad použijeme jakékoli místo na Zemi, nebo $[0,0,0,0]$ metoda konverguje velmi rychle, stačí jednotky iterací a $|\Delta\boldsymbol{X}| \leq 1\,m$.
V případě, že máme k dispozici více satelitů, je do výpočtu ještě zapojena lineární regrese (metoda nejmenších čtverců, viz Přílohy). Rovnice (2) mají více řádků, než 4 (počet viditelných satelitů), maticově:

$$\boldsymbol{A}\Delta x = b$$

v každém kroku řešíme soustavu rovnic:

$$\boldsymbol{A}^T\boldsymbol{A}\Delta x = \boldsymbol{A}^T b \tag{3}$$

Funkce $F_i$ jsou nazývány „$pseudo\ range$“ – pseudovzdálenost.

Náš zjednodušený příklad výpočtu polohy přijímače GPS jasně ukazuje, že bez výpočetní techniky se neobejdeme. GPS je mimo jiné i počítač. Ruční počítání rovnic (2) nebo (3) je (dnes) nepředstavitelné.

Pro odvození rovnic viz např.:
http://www.nbmg.unr.edu/staff/pdfs/Blewitt%20Basics%20of%20gps.pdf

Měření polohy pomocí GPS dosahuje přesnost 10-20 metrů.

GLONAS
Ruskou variantou je systém GLONAS, původně vyvinut také pro vojenské účely, do operačního provozu byl uveden 2011.

BeiDou
Čínská varianta, očekávaná plná operační schopnost v roce 2020.

Galileo
Projekt EU, očekávaná plná operační schopnost také v roce 2020.

Pomocí diferenciálních GPS stanic a lokálních korekcí lze dosáhnout přesnost měření až 0.1 m. Princip je ten, že máme k dispozici systém stacionárních GPS stanic, u kterých je přesně zaměřená poloha. Tato GPS vysílá odchylky vypočtené polohy od své skutečné polohy a diferenciální přijímače GPS v okolí podle toho korigují svou polohu. Předpokladem je to, že v relativně malém okolí mají všechny přijímače GPS stejnou chybu.

Navigátoři by však měli být schopni určit svou polohu, pro případ kolapsu technologie, čistě pomocí mechanických pomůcek a tabulek.

## 1.5. GEODÉZIE

https://cs.wikipedia.org/wiki/Geodézie

Doposud jsme se zabývali určením místa na zemském povrchu v malých měřítkách. S výjimkou diferenciálních GPS jsme nedosahovali přesnosti, která je vyžadována pro evidenci a analýzu jevů, jako jsou nemovitosti, inženýrské sítě a podobně. Určením vzájemné polohy bodů na zemském povrchu nebo v prostoru ve zvoleném souřadnicovém systému se zabývá obor geodézie. V základním dělení se geodézie dělí na vyšší a nižší.

Vyšší geodézie **–** se zabývá územím velkého rozsahu, pro které je nutné uvažovat zakřivení zemského povrchu. Výsledkem její činnosti jsou hlavně zaměření a výpočty geodetických sítí, které tvoří základ pro podrobná měření polohopisná a výškopisná. Vyšší geodézie už nepovažuje tvar Zemského tělesa za kouli, musí počítat s přesnějším modelem – elipsoidem, viz další kapitola.

Nižší geodézie – se zabývá územím, na kterém lze zakřivení zemského povrchu zanedbat. Výsledkem její činnosti je zejména podrobné měření polohopisné a výškopisné.

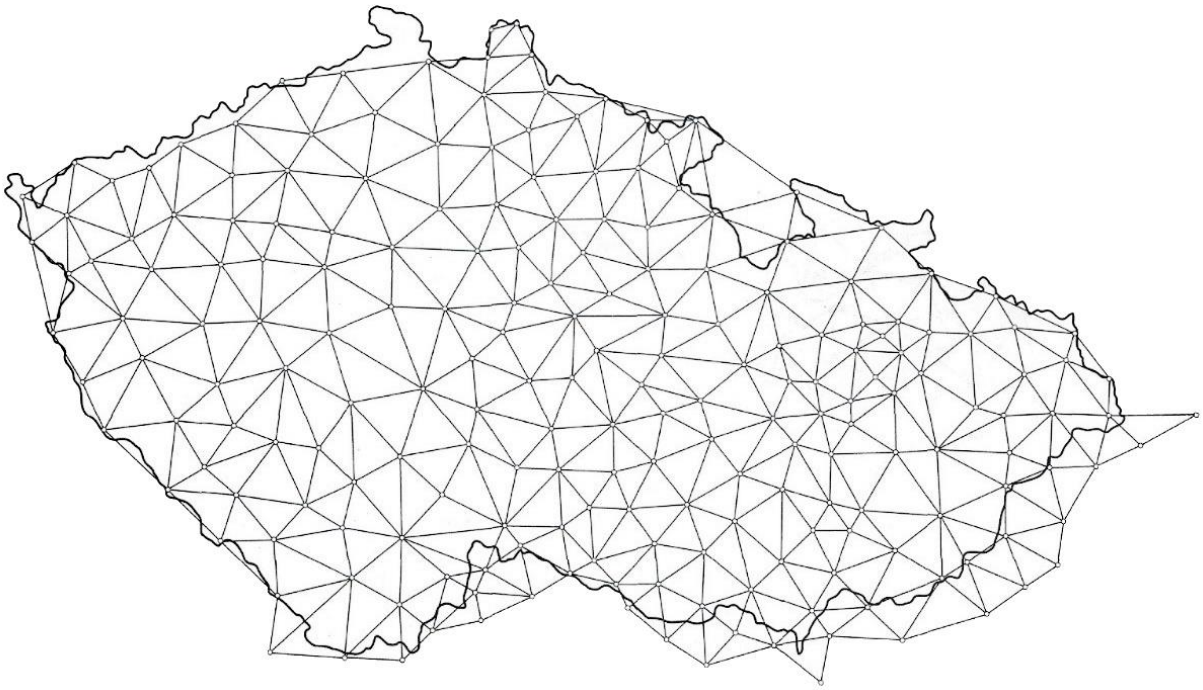
Česká státní trigonometrická síť
https://bodovapole.cuzk.cz/vyznamneTB.aspx

Je síť stabilizovaných bodů, tvoří základ pro podrobná měření na území ČR. Trigonometrický bod se podle metody zaměření/výpočtu dělí na 5 kategorií

Poloha bodů I. Řádu byla přesně geometricky zaměřena v rámci trigonometrické (trojúhelníkové) sítě metodou triangulace. Pomocí geodetického přístroje pro měření úhlů – teodolitu byly měřeny vodorovné úhly mezi směry na sousední body trigonometrické sítě a následně byly výpočtem určeny přesné souřadnice, které se využívají pro další geodetická měření a mapování.

Po vzniku Československa bylo rozhodnuto vybudovat jednotné geodetické základy na celém území státu, proto byla vybudována Jednotná trigonometrická síť katastrální.

Na území České republiky bylo zřízeno 181 trigonometrických bodů I. řádu a přibližně 75 tisíc bodů II. – V. řádu.



Česká státní trigonometrická síť I. Řádu

https://bodovapole.cuzk.cz/vyznamneTB.aspx

Pro podrobné měření bodů geodeti používají zejména tyto přístroje:

Teodolit – Optický přístroj pro měření vodorovných a vertikálních úhlů objektů s přesností až na úhlové vteřiny.

Dálkoměr – Přístroj pro měření délek.

Totální stanice – Měření úhlů i délek, většinou je vybavena počítačem pro ukládání bodů a provádění výpočetních úloh.

Nivelační přístroj – přístroj pro určování výšek, vytváří vodorovnou rovinu pomocí vodorovného dalekohledu.

# 2. Kartografická zobrazení a mapy

## 2.1. ZÁKLADNÍ TYPY ZOBRAZENÍ

Zemský povrch, geoid, je geometricky poměrně složitý útvar, proto je modelován rotačním elipsoidem, který je určen hlavní a vedlejší poloosou. Pro různá kartografická zobrazení jsou používány různé elipsoidy.

| | Bessel | Hayford | Krasovskij | IAG 1967 | WGS-84 |
|---|---|---|---|---|---|
| **rok** | 1841 | 1909 | 1940 | 1967 | 1984 |
| **a[m]** | 6377397.16 | 6378388 | 6378245 | 6378160 | 6378137 |
| **b[m]** | 6356078.96 | 6356911.95 | 6356863.02 | 6356774.52 | 6356752.31 |

Geodetické datum: model zemského tělesa (koule, elipsoid ..), jeho umístění (orientace) vůči zemskému tělesu a datum určení.

Matematická kartografie – disciplína zabývající se převodem zemského povrchu do roviny.

Konformní zobrazení – ponechává nezkreslené úhly, značně jsou však zde zkreslovány plochy.

Ekvidistantní zobrazení – nezkresluje délky určité soustavy. Zpravidla touto soustavou bývají zeměpisné poledníky nebo rovnoběžky. Nelze definovat ekvidistantní zobrazení, které by nezkreslovalo žádné délky.

Ekvivalentní zobrazení – nezkresluje plochy, zkreslení úhlů je však zde poměrně značné, což se projevuje zejména ve tvarech ploch.

Geografické souřadnice – určení polohy bodu na ploše elipsoidu pomocí zeměpisné šířky $\varphi$ a zeměpisné délky $\lambda$.

Geocentrické souřadnice $[X, Y, Z]$ - prostorový souřadný systém s počátkem ve středu elipsoidu, osa $X$ je vložena do průsečíku rovníku a roviny základního (nultého) poledníku, osa $Z$ spojuje střed elipsoidu a severní pól a osa $Y$ leží v rovině rovníku otočena o $90^0$ proti směru hodinových ručiček od osy $X$.

Rovinné souřadnice – určení polohy v rovině pomocí dvojice rovinných souřadnic $[X, Y]$ v ortogonálním souřadném systému. V některých zobrazeních (zobrazení UTM) se používá symbolika *E, N* (Easting, Northing), aby bylo zřejmé, v jakém směru rostou rovinné souřadnice.

Základní typy převodu geografických do rovinných souřadnic

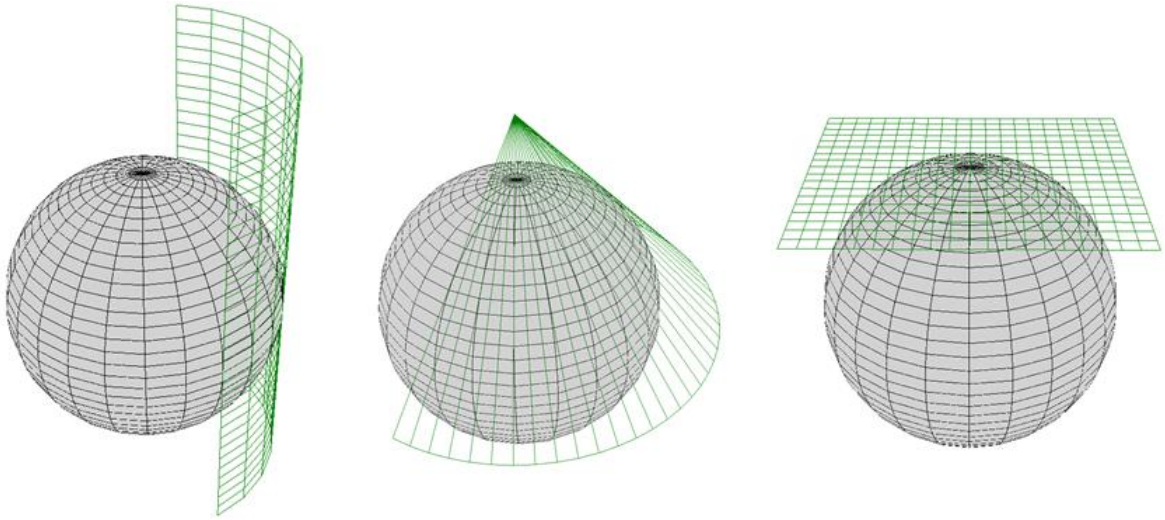
Základní typy převodu jsou dány 3D geometrickými tělesy, které jsou jednoduše převoditelné do roviny. Takovými tělesy jsou:

- rovina
- válec
- kužel

Princip je „geometricky jednoduchý“. Těleso umístíme do polohy vzhledem k modelu Země (koule, elipsoid) tak, aby projekce bodů na něj splňovala co nejvíce požadavky kladené na projekci. V praxi tento „jednoduchý“ požadavek vede k velmi komplikovaným konstrukcím, problémem se zabývá celý obor, *Matematická kartografie* (zkuste si třeba znovu sestavit triviální diferenciální rovnici pro Mercatorovo zobrazení, aniž byste listovali zpět).

Například, umístění kuželu pro projekci S-JTSK je takové, aby délkové zkreslení na území bývalé ČSR bylo minimální.

Jednotlivé body potom promítáme na těleso od středu Země.



Kartografický souřadný systém **–** je soubor těchto údajů:

- Geodetické datum (elipsoid, referenční bod, datum určení).
- Souřadný systém geografických souřadnic $\varphi, \lambda$ (volba základního poledníku).
- Zobrazovací rovnice z geografických do rovinných souřadnic $[\phi, \lambda] \rightarrow [x, y]$.

Příklad: Souřadný systém „Popular Mercator“ je definován:

| | |
|---|---|
| Revision date: | 2008-03-13 |
| Ellipsoid: | Popular Visualisation Sphere ($R = 6378137$). |
| Prime meridian: | Greenwich |
| Remarks: | Used only for Web approximate mapping and visualisation. Not recognised by geodetic authorities. |

$$y = ln\left(tan\left(\frac{\pi}{4} + \frac{\phi}{2}\right)\right)R$$

$$x = \lambda R$$

Viz https://epsg.io/6055-datum

## 2.2. NEJPOUŽÍVANĚJŠÍ SOUŘADNÉ SYSTÉMY V ČR

Civilní souřadnicový systém S-JTSK je určen – Besselovým elipsoidem z roku *1841* s referenčním bodem Herrmanskogel, zeměpisné délky se určují od ferrského poledníku, zobrazovací rovnice dvojitého konformního kuželového zobrazení v obecné poloze (Křovákovo zobrazení) s volbou délkového faktoru *0.9999* pro snížení vlivu délkového zkreslení.

Vojenský souřadnicový systém S-42 je určen Krasovského elipsoidem z roku *1942* s referenčním bodem Pulkovo, zeměpisné délky se měří od Greenwiche, zobrazovací rovnice Gaussova–Krügerova zobrazení s opakovatelností vždy pro šestistupňové poledníkové pásy. Od r. 2005 je nahrazen WGS84 – zobrazení UTM (Universal Transversal Mercator)

Souřadný systém WGS 84 - Word Geodetic System *1984* Systém byl původně definován Ministerstvem obrany USA pro obranné účely, dnes je celosvětově používanou technologií prostorové lokalizace.

UTM (Universal Transversal Mercator) – Systém transverzálního válcového zobrazení používající elipsoid WGS 84. Jsou použity šestistupňové pásy.

Cassini-Soldner – Válcové zobrazení na Zachově elipsoidu s referenčními body Sv. Štěpán, resp. Gusterberg. Definováno v Rakousko Uherské monarchii pro stabilní katastr v měřítkách 1:2800 a 1:1440 (dodnes používaná).

ETRS-LAEA: ETRS89 (Lambert Azimuthal Equal Area) – Elipsoid ETRS89 (téměř shodný s WGS 84), azimutální zobrazení používané pro střední a malá měřítka.

## 2.3. TRANSFORMACE SOUŘADNÝCH SYSTÉMŮ

Problém: Mapy různých kartografických zobrazení transformovat do zobrazení cílového. Základem je znalost, jak transformovat bod.

Příklad: V systému, který je provozován v kartografické projekci S-JTSK (národní projekce ČR) požadujeme, aby lokalizoval objekt o souřadnicích zadaných v projekci WGS84 (např. GPS).

Přímá transformace:

1. Zdrojové souřadnice $[x, y]$ převedeme na geografické souřadnice zdrojového systému $[\varphi, \lambda]$.

2. $[\varphi, \lambda]$ transformujeme do cílových geografických souřadnic $[\varphi', \lambda']$

3. Geografické souřadnice $[\varphi', \lambda']$ převedeme do cílového rovinného zobrazení $[x', y']$

Transformace geografických souřadnic:

1. Geografické souřadnice $[\varphi, \lambda]$ převedeme na geocentrické souřadnice$[x_S, y_S, z_S]$

2. Pro převod mezi dvěma systémy geocentrických souřadnic použijeme tzv. Helmertovu prostorovou transformaci:

$$x = (1 + m)(\ \ x_S + \gamma y_S - \beta z_S) + \Delta x$$

$$y = (1 + m)(-\gamma x_S + \ \ y_S + \alpha z_S) + \Delta y$$

$$z = (1 + m)(\ \beta x_S - \alpha y_S + \ \ z_S) + \Delta z$$

3. Geocentrické souřadnice z 1. převedeme na geografické.Poznámka:

Parametry Helmertovy prostorové transformace jsou získávány ze sad identických bodů, které se mohou lišit vlivem nehomogenity kartografické projekce pro různá území. Lze je interpretovat jako:

- vektor posunu – $[\Delta x, \Delta y, \Delta z]$
- úhel otočení pro jednotlivé osy - $[\alpha, \beta, \gamma]$
- změna měřítka - $m$

Tím dostáváme velmi snadno transformaci inverzní, jednoduše změníme znaménka všech parametrů.

Poznámka: Pomocí Helmertovy transformace jednoduše odvodíme transformace geocentrických souřadnic pro „neznámé“ elipsoidy. Stačí například znát pro všechny používané elipsoidy transformační klíč pro jeden pevný elipsoid, například WGS84. Pro transformaci geocentrických souřadnic z elipsoidů:

$$EL1 \rightarrow EL2$$

použijeme postup:

$$EL1 \rightarrow WGS84 \rightarrow EL2$$

Polynomiální transformace

Ze znalosti identických bodů ve zdrojovém a cílovém rovinném zobrazení $[X_i, Y_i] \rightarrow [X'_i, Y'_i]$ určíme koeficienty polynomiální transformace a tuto potom použijeme pro jednotlivé body. Používáme polynomy do 3. stupně, vyšší stupeň vede k nestabilitě řešení (omezený počet platných cifer).

Používá se v případech, kdy:

- Není známá zdrojová kartografická projekce prostorových dat.
- Zdrojová projekce je sice známá, avšak je zatížena takovou chybou, že obecná polynomiální transformace dává lepší výsledky.
- Zdrojová projekce je známá, ale její výpočet je příliš náročný vzhledem k počtu geometrických elementů a polynomiální transformace výsledek významně nezkreslí. V tomto případě použijeme kartografickou transformaci pro generování sítě odpovídajících si bodů (např. rohové body dotazového okna) a tyto body potom použijeme pro výpočet transformačního klíče.

Základní typy polynomiálních transformací:

Lineárni:

$$u = f(x,y) = a_1x + b_1y + c_1$$
$$v = g(x,y) = a_2x + b_2y + c_2$$

Bilineární:

$$f(x,y) = a_1x + b_1y + c_1xy + d_1$$
$$g(x,y) = a_2x + b_2y + c_2xy + d_2$$

Obecně - Kvadratická, kubická, obecná polynomiální …

Nalezení koeficientů polynomiální transformace:

Vstup: Dva seznamy „odpovídajících“ si bodů:

$$[x_1, y_1], .., [x_n, y_n] \text{ a } [u_1, v_1], .., [u_n, v_n]$$

Výstup: Seznam koeficientů polynomiální transformace zvoleného stupně.

Obecně, položíme-li:

$$\boldsymbol{A} = \begin{bmatrix} x_1 & y_1 & x_1^2 & x_1y_1 & y_1^2 & \dots & 1 \\ x_2 & y_2 & x_2^2 & x_2y_2 & y_2^2 & \dots & 1 \\ \vdots & & & & \vdots & & \vdots \\ x_n & y_n & x_n^2 & x_ny_n & y_n^2 & \dots & 1 \end{bmatrix}$$

a označíme-li hledané koeficienty:

$$k = \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_m \end{bmatrix}$$

hledáme co nejlepší řešení soustavy rovnic:

$$\begin{bmatrix} x_1 & y_1 & x_1^2 & x_1y_1 & y_1^2 & \dots & 1 \\ x_2 & y_2 & x_2^2 & x_2y_2 & y_2^2 & \dots & 1 \\ \vdots & & & & \vdots & & \vdots \\ x_n & y_n & x_n^2 & x_ny_n & y_n^2 & \dots & 1 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_m \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$$

tj. maticově:

$$\boldsymbol{A}k = u \text{ resp. } \boldsymbol{A}k = v$$

A tuto úlohu vyřešit umíme pomocí soustavy normálních rovnic (příloha – Metoda nejmenších čtverců):

$$\boldsymbol{A}^T\boldsymbol{A}k = \boldsymbol{A}^T u$$

Příklad

Pro lineární transformaci $u = f(x, y) = a_1x + b_1y + c_1$ nalezneme hledané koeficienty soustavou rovnic:

$$\begin{bmatrix} \Sigma x_i^2 & \Sigma x_iy_i & \Sigma x_i \\ \Sigma x_iy_i & \Sigma y_i^2 & \Sigma y_i \\ \Sigma x_i & \Sigma y_i & n \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} = \begin{bmatrix} \Sigma x_iu_i \\ \Sigma y_iu_i \\ \Sigma u_i \end{bmatrix}$$

Příklad: Ukázka zdrojového kódu pro převod geografických souřadnic do systému S-JTSK. Je zřejmé, že použití např. bilineární transformace je mnohem méně výpočtově náročné, v některých případech (např. pro rastrový obrázek 1024*1024 pixelů) hodně významně.

```
/* konstanty Besselova elipsoidu */
Double consta = 6377397.15508; Double constb = 6356078.96290;
/* konstanty */
Double constalfa = 1.00059749837159;Double constr = 6380703.610617;
Double constk = 0.996592486879232;

/* uhel v radianech odpovidajici 45 */
Double constu45 = 0.785398163397448;
/* uhel v radianech odpovidajici 78:30 */
Double constu7830 = 1.370083462815548;
/* uhel v radianech odpovidajici 59:42:42.6969 */
Double constuk = 1.042168563853224;
/* uhel v radianech odpovidajici 42:31:31.41725 */
Double constvk = 0.742208135432484;
/* uhel v radianech odpovidajici 17:39:59.7354 */
Double constferra = 0.308340218368665;

public override bool FromEllipsoid(
    double lat, double lon, double alt, ref double x, ref double y, ref double h)
{
    try
    {
        Double dpom;
        Double de, dsfi;
        Double du, dv, ddeltav;
        Double ds, dd;
        Double dro, depsilon;
        Double xx, yy;

        /* prevod vzhledem k ferra */
        lon += constferra;

        /* prevod lat,lon (Bessel) -> u,v (Gaussova koule) */
        dv = constalfa * lon;
        de = Math.Sqrt(consta * consta - constb * constb) / consta;
        dsfi = Math.Sin(lat);
        dpom = (1 - de * dsfi) / (1 + de * dsfi);
        dpom = Math.Pow(dpom, de / 2);
        dpom = dpom * Math.Tan(lat / 2 + constu45);
        dpom = Math.Pow(dpom, constalfa) / constk;
        du = 2 * (Math.Atan(dpom) - constu45);

        /* prevod u,v (Gaussova koule) -> s,d (sirka,delka) */
        ddeltav = constvk - dv;
        dpom = Math.Sin(constuk) * Math.Sin(du) +
               Math.Cos(constuk) * Math.Cos(du) * Math.Cos(ddeltav);
        ds = Math.Asin(dpom);
        dd = Math.Asin(Math.Sin(ddeltav) * Math.Cos(du) / (ds));

        /* prevod s,d -> x,y (rovinne, JTSK) */
        dpom = Math.Tan(constu7830 / 2 +
               constu45) / Math.Tan(ds / tu45);
        dpom = Math.Pow(dpom, Math.Sin(constu7830));
        dro = dpom * 0.9999 * constr / Math.Tan(constu7830);
        depsilon = Math.Sin(constu7830) * dd;
        x = dro * Math.Cos(depsilon);
        y = dro * Math.Sin(depsilon);

        h = alt;

        return true;
    }
    catch
    {
        return false;
    }
   }
```

Katalog kódů projekcí a jejich hlavních vlastností lze nalézt například na:

http://www.epsg-registry.org/

Poznámka:

Při vytváření normálních rovnic nám hodně pomůže jejich maticová reprezentace a obecně odvozená metoda nejmenších čtverců, viz příloha 9.2

## 2.4. TRADIČNÍ MAPY A POJMY SOUVISEJÍCÍ S GIS

- *Mapování* – vytváření map měřením nebo fotogrammetrickým mapováním pomocí geodetických základů – bodů geodetických sítí. Mapa je 2D obraz zemského povrchu.

- *Měřítko mapy* – v GIS kontextu neznamená doslova poměr skutečné a zobrazované velikosti objektů. Měřítkem mapy se spíše rozumí úroveň územní podrobnosti obsahu geografického informačního systému.

- *Dálkový průzkum Země (DPZ)* – získávání informací o zemském povrchu a jeho blízkém okolí pomocí snímacích zařízení (kamery, skenery) umístěných v letadlech nebo satelitech Země. Využitelné a využívané jsou také drony.

- *Topografická mapa* – je grafická prezentace (zobrazení) části zemského povrchu se standardizovaným obecným obsahem (voda, lesy, komunikace, objekty viditelné na zemském povrchu...)

- *Tematická mapa* – zobrazení geografických dat a jevů v topografickém podkladu pomocí grafické reprezentace prostorových dat: bodů, linií a areálů. Metody reprezentace: bodové značky, lokalizované kartodiagramy, kartodiagramy, symbolika čar, kartogramy.

## 2.5. MAPOVÉ DÍLO V ČR

*Mapy velkých měřítek do 1:5000*

- Katastrální mapy (mapy stabilního katastru) v systému Cassini–Soldner (počátek Gusterberg v Čechách, Sv. Štěpán na Moravě) v sáhových měřítkách 1:2880, 1:1440, 1:720 (měřítko je odvozeno ze vztahu 1 jitro – 1600 čtverečních sáhů – je zobrazeno jako čtvereční palec), ale v dekadických měřítkách.

- Katastrální mapy v S-JTSK (systém jednotné trigonometrické sítě katastrální – Křovák), měřítka 1:1000 ve městech (intravilán), 1:2000 v extravilánu vznikaly po roce 1928.

- Státní mapa odvozená v měřítku 1:5000, systém JTSK, obsah: vlastnické hranice, polohopis (vnitřní kresba).

- Digitální katastrální mapa – mapa vedená digitálně, udržovaná katastrálními úřady.

Státní mapové dílo velkých měřítek v České republice vznikalo v průběhu dvou století. Mapové dílo je charakteristické svou rozmanitostí a rozdílnou kvalitou (především vzhledem k přesnosti a aktuálnosti mapy).

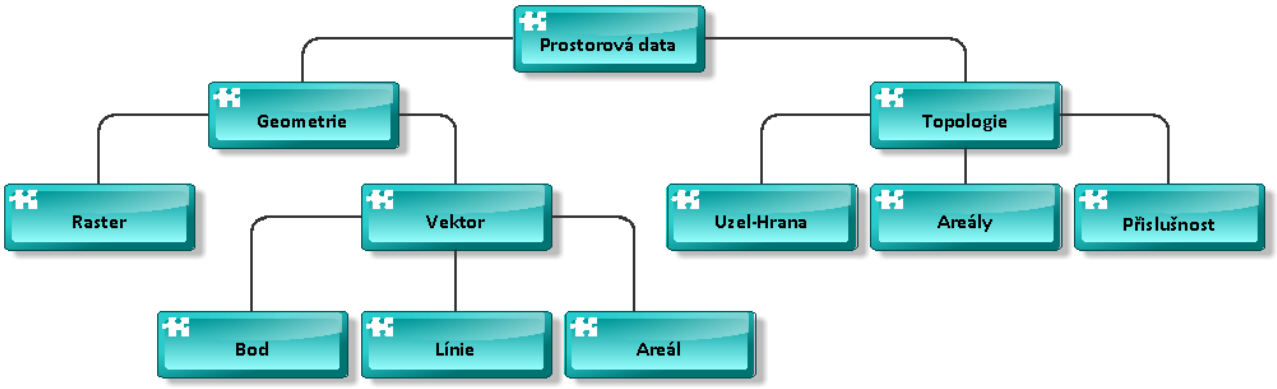
*Mapy středních měřítek 1 : 10000 až 1 : 200 000*

- Základní mapa středního měřítka – v měřítkách 1:10000, 1:25000, 1:50000, 1:100000, 1:200000 s obsahem topografické mapy, v digitální formě mapové dílo ZABAGED.

- Topografická mapa Generálního štábu armády ČR, měřítko 1:25000 (v některém území i 1:10000)

# 3. Datové sklady geoinformačních systémů

## 3.1. TYPY PROSTOROVÝCH DAT




Vektorová data – reprezentují objekty pomocí datových struktur, jejichž základní položkou je bod 2–rozměrného spojitého (euklidovského) prostoru. Termínem "spojitý" myslíme spojitý, až na technické omezení použité počítačové aritmetiky.

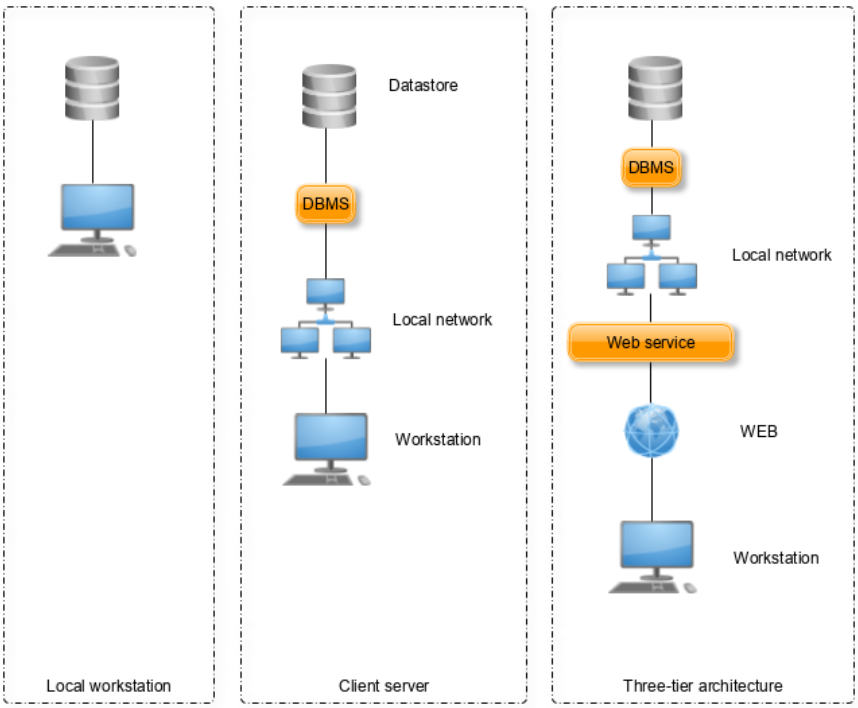
Rastrová data – podmnožina 2D prostoru je pravidelně rozdělena (většinou čtvercovou) sítí, každý element této sítě je nositelem tematické části (geografické) informace. Prostorová lokalizace je určena indexem elementární složky sítě, popřípadě jeho zobrazením do cílového (kartografického) souřadného systému.

Grid data – Základem této reprezentace je opět pravidelná síť položená na 2D prostor. Rozdíl oproti rastrovým datům spočívá v tom, že tematická část informace je získávána na základě předem definované sítě, kterou je rozděleno zájmové území.

Topologie – vymezuje vztahy mezi entitami (objekty) systému, aniž by musela obsahovat umístění objektu v prostoru. Například informační systémy o spojení míst silniční sítí nevyžadují přesné umístění uzlů v prostoru, pracují pouze s relací na množině všech uzlů.

## 3.2. ARCHITEKTURA POSKYTOVÁNÍ DAT V GIS

V tomto se geoinformační systémy v ničem neliší od systémů jiných. Tradičně se používají architektury lokálních stanic, architektura klient/server a konečně i více-úrovňové architektury, kdy k datům přistupujeme prostřednictvím aplikačních serverů (webových služeb).




## 3.3. RASTROVÉ DATOVÉ SKLADY

Využívají běžné formáty obrazových dat (bmp, png, jpeg tif, gif,ecw..). Některé z nich mohou mít informace o vztahu k souřadnicím kartografické projekce přímo ve své hlavičce (ecw, tif) u ostatních se používá „doplňující“ textový soubor (*.**tfw**), který obsahuje parametry lineárního převodu z/do pixelových do/z kartografických souřadnic.
Nechť $[i, j]$jsou pixelové souřadnice, $[x, y]$ kartografické souřadnice, potom převod z pixelových do kartografických je reprezentován lineárními rovnicemi:

$$x = a_x i + b_x j + c_x$$
$$y = a_y i + b_y j + c_y$$

**TFW** soubor je potom textový soubor obsahující parametry $a_x, b_x, a_y, b_y, c_x, c_y$

Příklad `TFW` souboru:

```
12.7011007
0
0
-12.7011007
-775450
-965225
```

Poznámka: V naprosté většině případů je $b_x = 0$ a $a_y = 0$. To znamená, že osy obrázku jsou rovnoběžné s osami kartografického zobrazení.

V případě, že nemáme k dispozici parametry lineární transformace, snadno je spočítáme ze dvou sad identických bodů.

### 3.4. VEKTOROVÉ DATOVÉ SKLADY

Pro fyzickou reprezentaci je možné použít vlastních datových struktur a ukládat je přímo v souborovém systému operačního systému. Přes nesporné výhody tohoto přístupu, jako je optimalizace uložení prostorové složky informace, převažují nevýhody, zejména aplikační závislost. Běžnější přístup je použití robustních databázových strojů, které vektorovou geometrii běžně používají (Oracle, Microsoft SQL-Server, Postgresql, MySQL).

Co je však hlavní problém je, žen ení jednotný standard ukládání vektorové geometrie.

Nejpoužívanější veřejné formáty:

Shape file (systém ARC/INFO fy. ESRI)

Geograficky vztažená informace je obsažena ve trojici souborů
- *.shp prostorová informace
- *.shx prostorový index
- *.dbf popisná informace a topologické vazby

Základní geometrické primitivy:

| | |
|---|---|
| **Point** | bod |
| **MultiPoint** | body |
| **Line** | lomená čára |
| **MultiLine** | lomené čáry |
| **Polygon** | areál |
| **MultiPolygon** | areály |

Vše 2D a 3D varianty. Formát neobsahuje symboliku (barva, síla, styl linií, výplň polygonu). Tu obsluhuje aplikace na základě popisných informací. Norma neobsahuje „heterogenní“ kolekce geometrií a tím ani definici mapových symbolů. Jako mapové symboly jsou použity speciální znakové sady (fonty).

Shape file byl jedním z prvních obecně používaných formátů, dodnes je podporován většinou „velkých“ geoinformačních systémů.

DGN file, norma IGDS (Intergraph, Bentley)

Geometrická informace je obsažena v souboru *.DGN, soubor sám o sobě nenese popisnou informaci, ta je uložena v relační databázi (nebo *.dbf souboru), DGN soubor obsahuje pouze tzv. „link“ = společný klíč v souboru/databázi.

Základní geometrické primitivy:

| | |
|---|---|
| **CELL_HEADER_ELM** | Hlavička buňky |
| **LINE_ELM** | Úsečka |
| **LINE_STRING_ELM** | Lomená čára |
| **SHAPE_ELM** | Polygon |
| **TEXT_ELM** | Text |
| **TEXT_NODE** | Textový uzel |
| **CURVE_ELM** | Křivka |
| **CMPLX_STRING_ELM** | Komplexní linie |
| **CMPLX_SHAPE_ELM** | Komplexní polygon |
| **ELLIPSE_ELM** | Elipsa |
| **ARC_ELM** | Oblouk |
| **POINT_STRING_ELM** | Body |
| **BSPLINE_ELM** | B-spline |
| **DIMENSION_ELM** | Kóta |

- Komplexní linie se mohou skládat z různých segmentů – např. lomených čar a oblouků.

- Geometrie obsahuje symboliku geometrických primitiv.

- Typ **CELL** může opět obsahovat **CELL**.

Podobné vlastnosti mají i ostatní CAD formáty (DXF, DWG...)

Bohatý repertoár geometrických primitiv CAD formátů umožňuje i definici mapových symbolů. Problematické jsou interpretace:

- nelineárních geometrií v různých klientských aplikacích (např. B-spline)

- složitějších zobrazovacích symbolik (např, linie vzorovaná symbolem – směr toku media ve vodovodní síti).

- operace na složitějšími geometriemi (např. průnik polygonů, jejichž hranice se skládají z lomených čar a eliptických oblouků …).

Příklad: Definice složitého mapového symbolu (Sídla Městského úřadu v GIS pro města). Je zřejmé, že v takovém případě je výhodné mít k dispozici robustní CAD formát vektorových dat, u kterého si každý geometrický element nese i výchozí symboliku kresby.

ORACLE 7x (Spatial Data Option):

Geometrie je reprezentována čtyřmi tabulkami:

**_SDOLAYER** – obsahuje meta údaje pro prostorovou indexaci
**_SDODIM** – obsahuje rozsah pro jednotlivé dimenze geometrie
**_SDOGEOM** – obsahuje vlastní geometrii
**_SDOINDEX** – obsahuje prostorové indexy objektů

možné typy geometrie jsou: bod, lomená čára a areál,

Jednalo se o první pokus o standardizaci geometrie – ten se vlivem denormalizace uložení souřadnic ukázal jako slepá ulička, v současné době není rozvíjen.

ORACLE 8x a výše – datový typ SDO_GEOMETRY:

| | |
|---|---|
| **UNKNOWN_GEOMETRY** | neznámá geometrie |
| **POINT** | bod |
| **LINESTRING** | lomená čára |
| **POLYGON** | areál (oblast) |
| **COLLECTION** | kolekce geometrií |
| **MULTIPOINT** | body |
| **MULTILINESTRING** | kolekce linií |
| **MUTIPOLYGON** | kolekce areálů |

- Linie jsou tvořeny sekvencemi bodů a kruhových oblouků.
- Typ COLLECTION nemůže obsahovat typ COLLECTION.
- Definice neobsahuje symboliku geometrických primitiv.

Open GIS Consortium, Inc.

Je sdružení soukromých, veřejných organizací (univerzit, komerčních firem…) se zájmem na vybudování „standardu“ struktur a služeb v prostorových datech. I přes byrokratickou těžkopádnost způsobenou množstvím subjektů, které se účastní vývoje standardů lze konstatovat, že standardy OGC jsou poměrně rozšířeny a podporovány (např. Oracle podporuje konverzi datových typů, prostorová informace MS-SQL a MySQL je přímou implementací OGC).

*Our Vision - is a world in which everyone benefits from geographic information and services made available across any network, application, or platform.*

*Our Mission - is to deliver spatial interface specifications that are openly available for global use.*

OGC Well known binary:

Je norma binárního ukládání vektorové geometrie včetně C-definic struktur:

```
// Basic Type definitions
// byte: 1 byte
// uint32: 32 bit unsigned integer (4 bytes)
// double: double precision number (8 bytes)

// Building Blocks : Point, LinearRing

Point {
 double x;
 double y;
};

LinearRing {
 uint32    numPoints;
 Point     points[numPoints];
}

enum wkbByteOrder {
 wkbXDR = 0,
 // Big Endian
 wkbNDR = 1
 // Little Endian
};

WKBPoint {
   byte               byteOrder;
   uint32             wkbType;     // 1
   Point              point;
};
```

```
WKBLineString {
   byte               byteOrder;
   uint32             wkbType;      // 2
   uint32             numPoints;
   Point              points[numPoints];
};

WKBPolygon {
     byte             byteOrder;
     uint32           wkbType;      // 3
     uint32           numRings;
     LinearRing rings[numRings];
};

WKBMultiPoint {
     byte             byteOrder;
     uint32           wkbType;        // 4
     uint32           num_wkbPoints;
     WKBPoint         WKBPoints[num_wkbPoints];
};

WKBMultiLineString {
 Byte           byteOrder;
 uint32               wkbType;        // 5
 uint32               num_wkbLineStrings;
 WKBLineString  WKBLineStrings[num_wkbLnStrgs];
};

wkbMultiPolygon {
 byte           byteOrder;
 uint32               wkbType;       // 6
 uint32             num_wkbPolygons;
 WKBPolygon           wkbPolygons[num_wkbPolygons];
}

WKBGeometry {
union {
 WKBPoint                 point;
 WKBLineString            linestring;
 WKBPolygon                    polygon;
 WKBGeometryCollection    collection;
 WKBMultiPoint            mpoint;
 WKBMultiLineString       mlinestring;
 WKBMultiPolygon          mpolygon;
   }
};

WKBGeometryCollection {
 Byte           byte_order;
 uint32               wkbType;// 7
 uint32               num_wkbGeometries;
 WKBGeometry          wkbGeometries[num_wkbGeoms];
}
```

Základní datové typy WKB neumožňují vykreslit plnohodnotnou mapu, neobsahují:

- Symboliku (grafická reprezentace – barva, síla, tloušťka) geometrických elementů.

- Reprezentaci bodových prvků, mapové symboly.

- Texty (velikost, rotace, font …).

Definice WKB neobsahuje definici kruhových oblouků, což může působit obtíže při práci v měřítkách, kde platí Euklidovská geometrie a „kružítko“ běžně používáme.

Rekurzivní definice **`WKBGeometryCollection`** umožňuje i definici komplexních mapových symbolů.

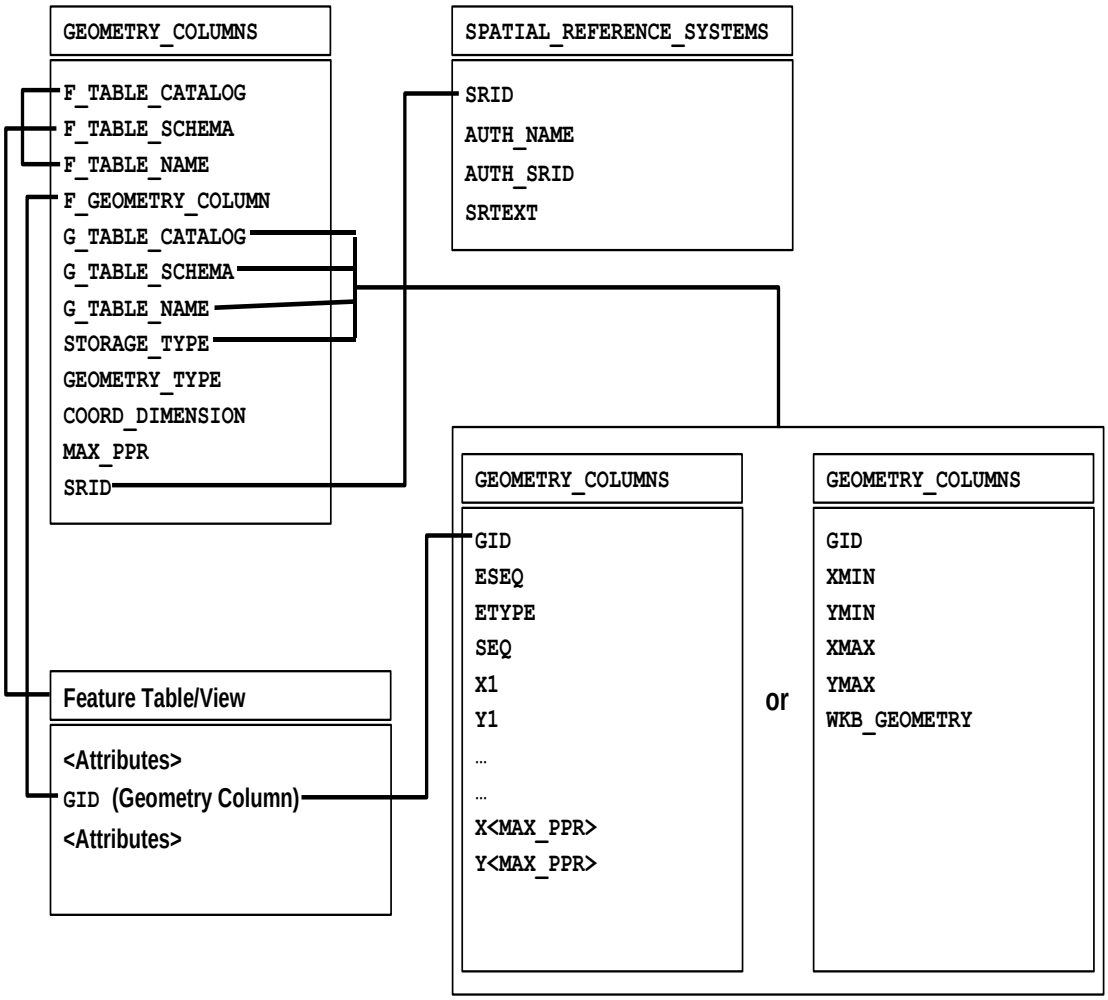
GML – Geographic Markup Language:

Původně formát WKB v XML formátu, od verze 3.2 značně rozšířená (křivky…). Geometrie mohou obsahovat i kód kartografické projekce, a to teoreticky umožňuje v rámci jedné datové sady používat projekcí více. V EU (projekt INSPIRE) byla specifikace GML převzata jako norma pro výměnu vektorových prostorových dat. Formát GML definuje XML reprezentaci geometrie, popisnou část informace nijak neomezuje. Vzhledem k masivní podpoře XML parserů mnoha vývojových prostředí se zřejmě jedná nejuniverzálnější výměnný formát. Pro ukládání prostorových informací je však nevhodný, jeho velikost je oproti binárnímu formátu až desetinásobná. Příklad GML:

```
<CP:geometry>
 <gml:Polygon srsName="urn:ogc:def:crs:EPSG::2065"
  gml:id="Polygon_1828023805">¨
  <gml:exterior>
   <gml:LinearRing>
    <gml:posList srsDimension="2">
     541484.77  1113662.92  541484.36  1113666.36
     541476.26  1113665.3   541469.89  1113664.51
     541469.62  1113662.23  541453.022  1113662.019
     541452.31  1113662.01  541451.77  1113664.1
     541446.27  1113662.85  541446.96  1113668.85
     541438.25  1113668.81  541440.74  1113661.7
     541432.057 1113660.152 541427.45  1113659.33
     541426.737 1113661.296 541425.54  1113664.6
     541419.741 1113662.802 541401.42  1113657.12
     541397.74  1113655.81  541391.63  1113653.68
     541394.736 1113649.478 541395.23  1113648.81
     541407.55  1113652.235 541427.96  1113657.91
     541438.177 1113658.825 541438.209 1113658.828
     541440.8   1113659.06  541469.46  1113660.94
     541484.77  1113662.92
    </gml:posList>
   </gml:LinearRing> …
```

OGC specifikace pro ukládání prostorových informací v relačních databázích:

(OpenGIS Simple Features Specification For SQL metamodel)



## 3.5. WEBOVÉ SLUŽBY A JEJICH STANDARDY

### 3.5.1. OGC – WEB MAP SERVICE (WMS):

Je webová služba poskytující mapu v zadaném výřezu a zadané kartografické projekci. Obsahuje (mimo jiné) dva základní dotazy:

**`GetCapabilities`** – vrací metainformace o službě ve formátu XML. Metainformace obsahují zejména výčet mapových vrstev, podporované kartografické projekce a rozsah měřítek, pro které jsou tato data poskytována.

```
<Layer queryable="0" opaque="0" noSubsets="0">
  <Title>WMS KN - CUZK</Title>
  <SRS>EPSG:102067</SRS>
  <SRS>EPSG:32633</SRS>
  <SRS>EPSG:32634</SRS>
  <LatLonBoundingBox
       minx="11.849344737243492"
       miny="48.201283122633363"
       maxx="18.981602263421369"
       maxy="51.4021596360892" />
  <BoundingBox SRS="EPSG:32633"
       minx="197613" miny="5295313"
       maxx="891648" maxy="5784919" />
  <BoundingBox SRS="EPSG:32634"
       minx="-142500" miny="5363000"
       maxx="354000" maxy="5716000" />
  <BoundingBox SRS="EPSG:102067"
       minx="-910000" miny="-1230000"
       maxx="-430000" maxy="-930000" />
 <Layer queryable="0" opaque="0" noSubsets="0">
   <Name>obrazy_parcel</Name>
   <Title>Obrazy parcel</Title>
   <ScaleHint min="0" max="0.997806228061693" /> …
```

**GetMap** – vrátí obrázek s požadovanou mapou.

Příklad:

```
http://wms.cuzk.cz/wms.asp?
request=GetMap&
version=1.1.1&
layers=dalsi_p_mapy_i,hranice_parcel_i,
obrazy_parcel_i,OMP,parcelni_cisla_i&
srs=EPSG:102067&
bbox=-815350,-1096562,-815058,-1096388&
width=1371&
height=815&
bgcolor=0x999999&
Format=image/gif
```

### 3.5.2. OGC – WEB MAP TILE SERVICE (WMTS)

Je služba, která poskytuje „dlaždice map“ v definované kartografické projekci. Je používána tam, kde se obsah mapy příliš nemění v čase a dlaždice je možné předem připravit. Na podobném principu pracují i robustní světové servery (Google maps, Bing maps). Služba obsahuje opět mimo jiné dva základní dotazy:

**`GetCapabilities`** – vrací metadata služby ve formátu XML. Metadata obsahují informace o dlaždicových sadách (**`TileMatrixSet`**, **`TileMatrix`**), jejich měřítkách a umístění v souřadném systému.

Příklad: ***`URL`*`?request=GetCapabilities&version=1.1.1&service=WMTS`**

```
<TileMatrixSet>
  <ows:Abstract>OrtoFoto</ows:Abstract>
  <ows:Identifier>ORTOMAPA</ows:Identifier>
  <ows:SupportedCRS>EPSG:5514</ows:SupportedCRS>
 <TileMatrix>
    <ows:Abstract>Zoom_09</ows:Abstract>
    <ows:Identifier>Zoom_09</ows:Identifier>
 <ScaleDenominator>14285.714285714286</ScaleDenominator>
  <TopLeftCorner>-931496 -819102</TopLeftCorner>
  <TileWidth>256</TileWidth>
  <TileHeight>256</TileHeight>
  <MatrixWidth>512</MatrixWidth>
  <MatrixHeight>512</MatrixHeight>
  </TileMatrix>
 <TileMatrix>
    <ows:Abstract>Zoom_10</ows:Abstract>
    <ows:Identifier>Zoom_10</ows:Identifier>
 <ScaleDenominator>7142.8571428571431</ScaleDenominator>
    <TopLeftCorner>-931496 -819102</TopLeftCorner>
    <TileWidth>256</TileWidth>
    <TileHeight>256</TileHeight>
    <MatrixWidth>1024</MatrixWidth>
    <MatrixHeight>1024</MatrixHeight> </TileMatrix>
```

**GetTile** – vrací mapovou dlaždici.

```
URL&service=WMTS&request=GetTile&
TileMatrixSet=ORTOMAPA&TileMatrix=Zoom_11&
TileRow=1024&TileCol=1024
```

### 3.5.3. OGC Web Feature Service (WFS)

WFS je služba, která poskytuje vektorová data v normě GML a atributy libovolné struktury.

**GetCapabilities** – vrací **XML** soubor popisující služby a seznam poskytovaných datových datové sad (feature).

```
.
.
<FeatureType xmlns:CP="urn:x-
inspire:specification:gmlas:CadastralParcels:3.0">
<Name>CP:CadastralParcel</Name>
<Abstract>Cadastral parcel polygons</Abstract>
<ows:Keywords>
<ows:Keyword>Cadastral parcel polygons</ows:Keyword>
</ows:Keywords>
<DefaultCRS>urn:ogc:def:crs:EPSG::102067</DefaultCRS>
<OtherCRS>urn:ogc:def:crs:EPSG::102066</OtherCRS>
<OtherCRS>urn:ogc:def:crs:EPSG::2065</OtherCRS>
<OutputFormats>
<Format>text/xml; subtype=gml/3.2.1</Format>
</OutputFormats>
<ows:WGS84BoundingBox>
<ows:LowerCorner>10 43</ows:LowerCorner>
<ows:UpperCorner>22 55</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>
.
```

.

**DescribeFeatureType** – vrací **XSD** šablonu pro požadovaný feature (= typu datové sady).

**ListStoredQueries** – vrací **XML** soubor se seznamem uložených dotazů a návratových typů.

```
<StoredQuery id="CadastralBoundary_Envelope_Query">
<ReturnFeatureType>
 CP:CadastralBoundary
</ReturnFeatureType>
</StoredQuery>
<StoredQuery id="CadastralParcel_Envelope_Query">
<ReturnFeatureType>
 CP:CadastralParcel
</ReturnFeatureType>
</StoredQuery>
<StoredQuery id="CadastralZoning_Envelope_Query">
<ReturnFeatureType>
 CP:CadastralZoning
</ReturnFeatureType>
</StoredQuery>
<StoredQuery id="CadastralBoundary_BBOX_Query">
<ReturnFeatureType>
 CP:CadastralBoundary
</ReturnFeatureType>
</StoredQuery>
<StoredQuery id="CadastralParcel_BBOX_Query">
<ReturnFeatureType>
 CP:CadastralParcel
</ReturnFeatureType>
</StoredQuery>
```

**DescribeStoredQueries** – vrací **XML** soubor s popisem parametrů uložených dotazů

```
<StoredQueryDescription id="CadastralBoundary_BBOX_Query">
<Abstract>Cadastral Parcel Boundaries</Abstract>
<Parameter type="xsd:double" name="XMIN"/>
<Parameter type="xsd:double" name="YMIN"/>
<Parameter type="xsd:double" name="XMAX"/>
<Parameter type="xsd:double" name="YMAX"/>
<Parameter type="xsd:string" name="CRSCODE"/>
</StoredQueryDescription>
```

**GetFeature** – Vrací sadu prostorových (geometrických) dat. Požadavek je definován parametrem **Filter**, což je **XML** fragment definující prostorovou a/nebo logickou podmínku na požadovaná data. Je možné definovat velmi složité podmínky.

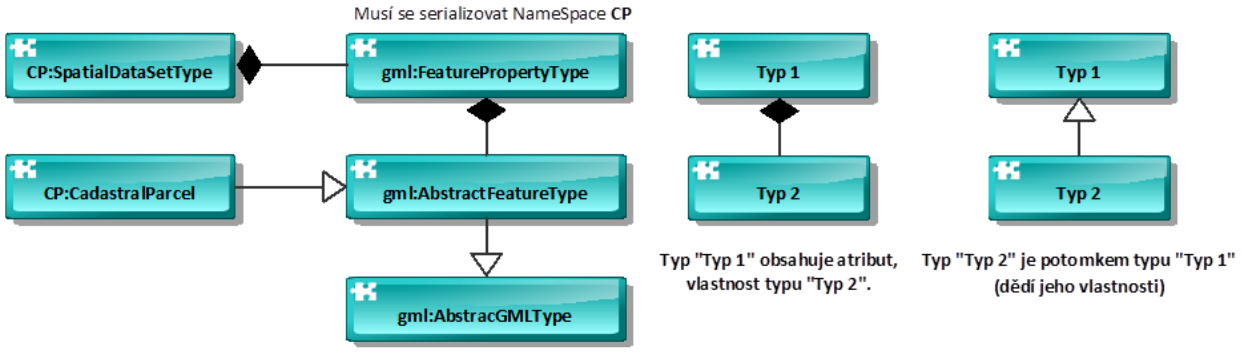
```
<Filter…>
 <BBOX>
  <gml:Envelope
    srsName="urn:ogc:def:crs:EPSG::2065">
     <gml:lowerCorner>557658 106440</gml:lowerCorner>
     <gml:upperCorner>557489 106330</gml:upperCorner>
   </gml:Envelope>
 </BBOX>
</Filter>

<Filter…>
 <And>
  <PropertyIsEqualTo>
   <ValueReference>CP:zoning</ValueReference>
   <Literal>KU601977</Literal>
  </PropertyIsEqualTo>
  <PropertyIsBetween>
   <ValueReference>CP:label</ValueReference>
    <LowerBoundary>
     <Literal type="xs:string">198</Literal>
    </LowerBoundary>
    <UpperBoundary>
      <Literal type="xs:string">215</Literal>
    </UpperBoundary>
  </PropertyIsBetween>
  <PropertyIsGreaterThan>
   <ValueReference>
    CP:beginLifespanVersion
   </ValueReference>
   <Literal type="xs:date">
    2010-03-08T12:46:20
   </Literal>
  </PropertyIsGreaterThan>
  <Not>
     <PropertyIsNull>
       <ValueReference>CP:zoning</ValueReference>
     </PropertyIsNull>
  </Not></And></Filter>
```

Zdrojový kód pro poskytovatele/konzumenta je většinou generován strojově.
Výsledný kód má totiž řádově $10^4$ až $10^5$ řádků. Je téměř nemyslitelné, aby kód, který akceptuje tuto službu byl sestaven ručně a bezchybně programátorem.

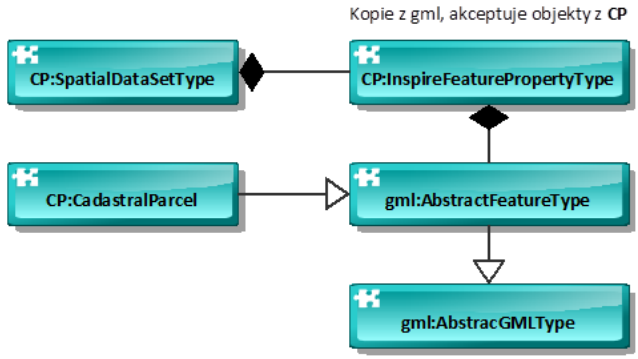
Příklad – implementace tříd směrnice INSPIRE (EU)

Třídy jmenného prostoru (namespace) gml byly vygenerovány dříve a jsou fyzicky v jiném sestavení (DLL), než objekty jmenného prostoru CP v konkrétní implementaci WFS. Má to logiku, nebudeme přece opakovat gml kód pro každou WFS službu zvlášť. Při vygenerování tříd pro konkrétní implementaci, se však dostaneme do situace, že třída **`gml:FeaturePropertyType`** musí akceptovat objekty ze jmenného prostoru CP. To není možné, sestavení pro gml prostě „nezná“ objekty z tohoto prostoru. Možné řešení je naznačeno v následujícím diagramu tříd:

**"Inspire Cadastral Parcels" generovaný C# kód:**




**"Inspire Cadastral Parcels" validní C# kód:**



# 4. Efektivní přístup k prostorovým datům

## 4.1. NAJDI OBJEKT (A RYCHLE!!!)

Vyhledání objektů s jistými vlastnostmi je od počátků informatiky jádrem jakéhokoli informačního systému. Myslíme „efektivní“ vyhledání, tedy vyhledání s lepší časovou složitostí než lineární. Ta lze dosáhnout triviálně vždy, prostým průchodem přes všechny objekty. Touto „dovedností“ musí disponovat jak robustní datová úložiště (relační databáze), tak i klientské aplikace.

Představme si dva programátory, kteří vyvíjejí aplikaci, ve které přidávají a vyhledávají svoje objekty. Špatný programátor použije jednoduchý postup:

```
class MyObject
{
  public int Key { get; set; }
  //.. other properties
}

static void AddObject_Wrong(MyObject o)
{
    for (int i = 0; i < myObjects.Count; ++i)
    {
        if (myObjects[i].Key == o.Key)
        {
            myObjects[i] = o;
            return;
        }
    }
    myObjects.Add(o);
}

static int SearchObject_Wrong(MyObject o)
{
    for (int i = 0; i < myObjects.Count; ++i)
    {
        if(myObjects[i].Key == o.Key)
        {
            return i;
        }
    }
    return -1;
}
```

Dobrý programátor použije radši tento postup:

```
class MyObject:IComparable<MyObject>
{
    public int Key { get; set; }

    public int CompareTo(MyObject other)
    {
        return this.Key.CompareTo(other.Key);
    }
        //.. other properties
}


static void AddObject_Right(MyObject o)
{
    int idx = myObjects.BinarySearch(o);
    if (idx < 0)
    {
        // not found
        myObjects.Insert(~idx, o);
    }
    else
    {
        myObjects[idx] = o;
    }
}

static int SearchObject_Right(MyObject o)
{
    return myObjects.BinarySearch(o);
}
```

## 4.2. FORMALIZACE ÚLOHY VYHLEDÁNÍ

Efektivním přístupem k prostorovým datům rozumíme vyhledávací techniky nad prostorovými daty, kdy dotazy mají opět prostorový charakter. Například, vyhledej všechny objekty, které leží uvnitř obdélníku, který je vymezen obrazovkou počítače.

Problém vyhledání můžeme popsat termíny formální logiky prvního řádu (FOL). Za vybudování FOL vděčíme německému matematikovi a logikovi Gottlobu Fregemu (1848-1925), na jejím základě byl vybudován „betonový“ základ matematiky – Teorie množin. Pro naše účely postačí následující „zkratka“:

- Oblast zájmu (universum) je popsána souborem relací – predikátů, některé z nich mohou být funkce.

- FOL poskytuje aparát pro „správné“ sestavení výrazů a formulí s proměnnými, ty mohou nabývat hodnot prvků relací universa. Proměnné mohou být kvantifikovány (univerzálně – „pro každý“, existenčně – „existuje“)

Vyhledání v relaci $U$ podle formule $\Phi$ je potom podmnožina

$V \subseteq U$:

$$V = \{u \in U | \Phi(u)\}$$

Příklad – Problém příslušnosti prvku k množině:

Pro libovolnou množinu $V$ můžeme definovat jednoduchou formuli:

$$\Phi: x = a$$

kde $x$ je proměnná a $a$ je libovolná konstanta (z univerza).
Aplikací formule $V = \{u \in U | \Phi(u)\}$ dostaneme odpověď na to, zda $a \in U$, tj. $U$ obsahuje $a$.

Příklad – Rozsahový dotaz na uspořádané množině:

Nechť $(T, \leq)$ je úplně uspořádaná množina (její každé dva elementy lze porovnat) $U \subseteq T$. Rozsahovým dotazem potom rozumíme aplikaci formule ($a, b \in T$ jsou konstanty):

$$\Phi: a \leq x \leq b, \text{ kde } a \leq b$$

Příklad – Rozsahový dotaz na body ve 2D prostoru:

Nechť $(T, \leq)$ je úplně uspořádaná množina, $U \subseteq T^2$. 2D rozsahovým dotazem potom rozumíme formuli:

$$\Phi: x_{min} \leq x \leq x_{max} \wedge y_{min} \leq y \leq y_{max}$$

kde

$$x_{min}, x_{max}, y_{min}, y_{max} \in T, x_{min} \leq x_{max}, y_{min} \leq y_{max}$$

jsou konstanty.

Typ dotazů $\Phi$ jsou obdélníky (okna) rovnoběžné s osami souřadného systému). Aplikací formule dostaneme všechny body uvnitř „okna“.

Příklad – Rozsahový dotaz na obdélníky ve 2D prostoru:

Nechť $(T, \leq)$ je úplně uspořádaná množina $U \subseteq T^4$ s vlastností:

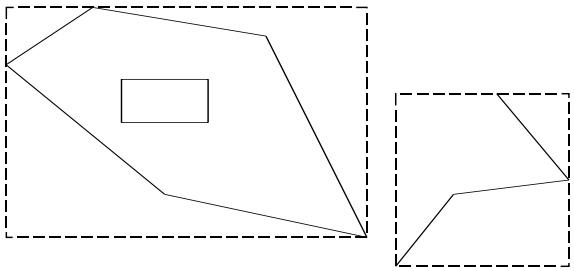
$$\forall [x_1, y_1, x_2, y_2] \in U \Rightarrow (x_1 \leq x_2 \wedge y_1 \leq y_2)$$

Prvky množiny $U$ jsou obdélníky ve 2D prostoru. Formule, která vyhledá všechny prvky z $U$, které mají neprázdný průnik (incidují) s 2D rozsahem (také obdélníkem) má tvar:

$$\Phi: x_1 \leq x_{max} \wedge y_1 \leq y_{max} \wedge x_2 \geq x_{min} \wedge y_1 \geq y_{min}$$

kde $x_{min}, x_{max}, y_{min}, y_{max}$ jsou konstanty s vlastnostmi z předešlého příkladu.

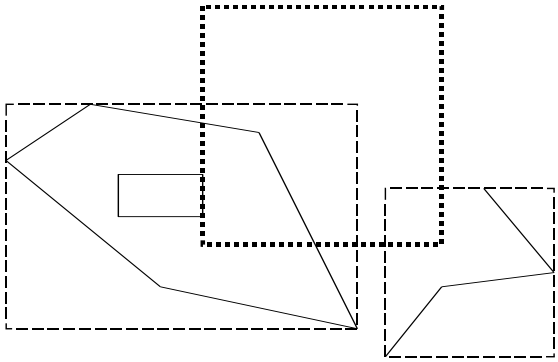
Jednotný zdroj pro prostorovou indexaci geometrických objektů: (tj. struktur pro efektivní vyhledání) je minimální omezující obdélník geometrického objektu rovnoběžný s osami souřadného systému – MBR (Minimal Bounding Rectangle):



tedy minima, resp. maxima lomových (definičních) bodů

$$[x_{min}, y_{min}, x_{max}, y_{max}]$$

V naprosté většině případů vystačíme s obdélníkovým dotazem:



Metoda, která realizuje tento dotaz je často nazývána primárním filtrem. Metoda, která realizuje přesnou odpověď je nazývána filtrem sekundárním.

Vraťme se ze světa logiky a množin do světa informatiky. Všechny uvedené příklady lze triviálně řešit jedním průchodem množiny/seznamu $U$, tedy v lineární časové složitosti $O(|U|)$. Uvádění jiných metod má tedy smysl pouze v případě, že tento základní odhad nějak zlepšíme.

Pro rozsahové výběry se většinou studuje časová složitost „zásahu“ prvního objektu, který splňuje podmínku rozsahového výběru.

Běžné indexovací metody (tj. ty které jsou implementovány v RDBMS – např. $B^+$ stromy) poskytují efektivní aparát pro vyhledávací problémy:

- příslušnosti k množině
- rozsahový dotaz

ale samy o sobě neposkytují aparát vhodný k prostorovým dotazům:

Příklad – incidence intervalů:

Představme si tabulku, kde bude za zaměstnance zaznamenán jeho nástup do firmy a konec jeho pracovního poměru. Vedení firmy bude zajímat, kdo všechno byl zaměstnán ve firmě v určitém časovém období. Formálně:

Máme soubor intervalů (1D obdélníků), a dotaz bude opět interval. Odpovědí budou všechny intervaly, které s dotazem incidují.

Lineární indexovací metoda (tj. uspořádání podle jednoho či více klíčů), nám nepomůže, neboť nejhorší případ dotazu vede prohledání celého souboru.

Problémy vyhledávání rozdělíme na dvě hlavní třídy:

- problém statický
- problém dynamický

Statický:

- $build(U)$ vybuduje podpůrné struktury pro množinu $U$
- $search(\Phi, U)$ odpoví na vyhledávací dotaz

Dynamický:

- $insert(x, U)$ vloží do množiny $U$ nový objekt $x$
- $delete(x, U)$ vymaže z množiny $U$ objekt $x$
- $search(\Phi, U)$ odpoví na vyhledávací dotaz

Dynamické řešení problému zároveň řeší statickou variantu (opakujeme funkci insert).

Funkce $search$ bývá většinou rozdělena na dvě části, a to

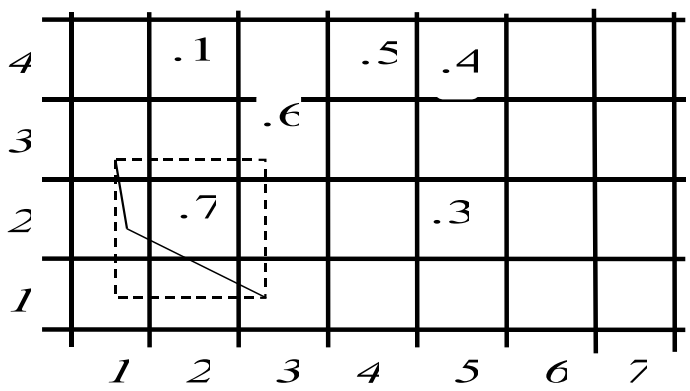
- $\Phi_i = init(\Phi, U)$ inicializace dotazu
- $o = fetch(\Phi_i)$ vrací jeden objekt z množiny $U$

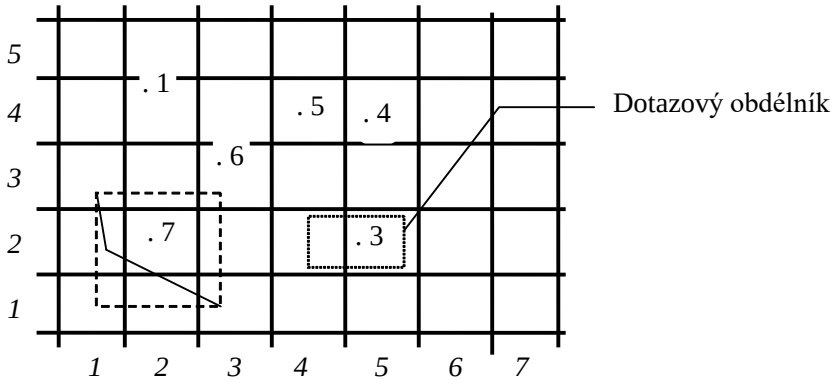
práce potom probíhá podle jednoduchého schématu:

```
var f=init(q,U);
 while((object x=fetch(f))!=null)
  {
   zpracuj_objekt(x);
  }
 close(f);
```

### 4.3. METODA „GRID“:

Spočívá v pravidelném rozdělení 2D prostoru, resp. zájmového území obdélníkovou sítí. Elementy sítě lze maticově indexovat, tím prostor „linearizovat“ a využít je tím pro primární prostorový filtr.



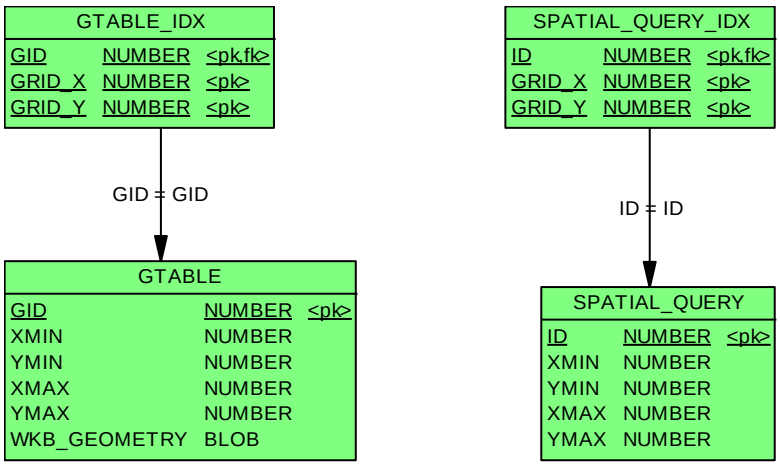



Prostorový dotaz v GRIDu“:

Prohledáváme pouze čtverce incidentní s dotazem, tedy (4,2) a (5,2), pro efektivní přístup ke čtvercům použijeme libovolnou vyhledávací metodu podporující rozsahový dotaz na úplně uspořádaných množinách. Tyto metody jsou standardně implementovány ve všech databázových strojích a lze je téměř okamžitě použít.

Realizace GRID metody v prostředí SQL:

Každé prostorové tabulce (tj. obsahující vektorovou geometrii např. ve formátu WKB) přiřadíme indexovací prostorovou tabulku.




Tabulka s prostorovými daty:

```
create table GTABLE
 (
  GID  number,
  XMIN number,
  YMIN number,
  XMAX number,
  YMAX number,
  WKB_GEOMETRY blob,
  ...
  constraint GTABLE_PK primary KEY (GID));
```

Tabulka grid indexů:

```
create table GTABLE_IDX
 (
  GID    number,
  GRID_X number,
  GRID_Y number
);
```

Omezení a prostorové indexy:

```
alter table GTABLE_IDX add constraint
 GTABLE_IDX_PK primary key (gid,grid_x,grid_y);

alter table GTABLE_IDX
add constraint GTABLE_IDX_fk1
foreign key (GID) references GTABLE(GID)
ON DELETE CASCADE;

create index GTABLE_IDX_I1
    on GTABLE_IDX(grid_x, grid_y);
```

Triggerem zajistíme vkládání prostorových indexů:

```
create trigger gtable_spatial
before insert or update of x,y on GTABLE for each row
begin
 xfrom:=GET_GRID_X(:NEW.XMIN);
 xto  :=GET_GRID_X(:NEW.XMAX);
 yfrom:=GET_GRID_Y(:NEW.YMIN);
 yto  :=GET_GRID_Y(:NEW.YMAX);

 pro xfrom<=i<=xto a yfrom<=j<=yto

 begin
  INSERT INTO GTABLE_IDX VALUES(:NEW.GID,i,j);
 end;
end;
/
```

(funkce **GET_GRID_X/Y** vrací gridové indexy)

Implementace prostorového dotazu:

Vytvoříme dotazovou tabulku:

```
create table SPATIAL_QUERY
  (
   id int,
   xmin int,
   ymin int,
   xmax int,
   ymax int,
   constraint SPATIAL_QUERY_PK primary key (id)
  );
```

a ostatní objekty (indexová tabulka, integritní omezení, trigger) stejně jako u tabulek s prostorovými daty.

Prostorový dotaz pro obdélník **[xmin,ymin,xmax,ymax]** provedeme následovně:

1. Identifikace dotazu: Z databáze získáme nový (jednoznačný) klíč dotazu **id**, například ze sekvence.

2. Inicializace dotazu:

   ```
   insert into spatial_query
   values (id,xmin,ymin,xmax,ymax),
   ```

   vlivem triggeru **SPATIAL_QUERY_SPATIAL** automaticky vloží identifikace gridových čtverců to tabulky **spatial_query_idx**

3. Prostorový dotaz:

   ```
   select …
   from
    gtable            A,
    gtable_idx        B,
    spatial_query_idx C
  where
   A.GID=B.GID       AND
   B.grid_x=C.grid_x AND
   B.grid_y=C.grid_y AND
   C.query_id=id;
   ```

Ukončení prostorového dotazu:

```
delete from spatial_query where id=id;
```

Jaký mechanismus odstraňuje řádky z tabulky **spatial_query_idx**?)

Výhody vs. nevýhody GRID metody.

+

- velmi snadná implementace v prostředí RDBMS
- snadné rozšíření na více dimenzí (?)
- relativně snadná (resp. řešitelná implementace neobdélníkových dotazů)
- netriviální odhad velikosti GRIDových čtverců, špatná volba má dramatické důsledky
- nepravidelné chování při řádově rozdílné velikosti geometrických objektů

### 4.4. MODIFIKACE BINÁRNÍCH STROMŮ PRO PROSTOROVÉ VYHLEDÁVÁNÍ, KD STROMY

Definice – Binární strom:

- Nechť $(U, \leq)$ je úplně uspořádaná množina, $V \subseteq U$ a $|V| < \infty$
- Nechť $(V, E)$ je strom ve smyslu teorie grafů, takový, že každý jeho uzel obsahuje maximálně dva syny.
- Každý uzel $n$ vyjma kořene má vlastnost „pravý“ a “levý“, označme $n_L, n_R$ levé a pravé následníky uzlu $n$.
- $n_L \leq n$ pro libovolný uzel $n$.
- $n \leq n_R$ pro libovolný uzel $n$.

Implementaci binárního stromu si můžeme představit jako jednoduchou strukturu (c#):

```
class KeyType : IComparable
{
…
…
}

class BinTreeNode
{
    KeyType Key;
    BinTreeNode Left;
    BinTreeNode Right;
}
```

(Interface **`IComparable`** vynucuje „srovnatelnost“ libovolných instancí typu **`KeyType`**)

Algoritmus – Vyhledání klíče v binárním stromu:

1. Vstup: uzel stromu $n$, klíč $k$.
2. Je-li $n = \emptyset$ (strom je prázdný), potom končíme vyhledávání ”neúspěchem”.
3. Je-li $n = k$, potom končíme ”úspěchem”.
4. Je-li $k < n$, pokračujeme 1. pro levý podstrom $n_L$
5. Je-li $k > n$, pokračujeme 1. pro pravý podstrom $n_R$

Algoritmus – Vkládání klíče do binárního stromu:

1. Vstup: klíč $k$.
2. Procházíme strom, jako bychom hledali klíč $k$ dokud nenarazíme na volnou pozici, tedy končíme bodem 2 předešlého algoritmu.
3. Do volné pozice vložíme klíč $k$.

Algoritmus – Rozsahové vyhledání v binárním stromu:

1. Vstup: interval $[min, max]$, uzel stromu $n$.
2. Je-li $n = \emptyset$, potom konec.
3. Patří-li $n$ do intervalu $[min, max]$, pošleme jej na výstup a aplikujeme algoritmus na $n_L$ a $n_R$.
4. Je-li $max < n$, aplikujeme algoritmus na $n_L$.
5. Je-li $min > n$, aplikujeme algoritmus na $n_R$.

Povšimněme si kroků 4. a 5. Zlepšení časové složitosti spočívá v tom, že v určitých fázích algoritmů jsme schopni rozhodnout, kterou větev stromu můžeme bez rizika vynechat.

Potíže s touto jednoduchou strukturou způsobuje fakt, že v jistých případech může být strom degenerovaný (např. $n_L = \emptyset$ pro všechny uzly). Degenerace nastává tehdy, když jednotlivé prvky vstupují do stromu v nevhodném pořadí (jsou uspořádány).

V případě statické verze vyhledávacích problémů lze vybudovat tzv. optimální binární strom (na vstupu procedury $build$ známe celou množinu $V$).

Definice - Optimální strom:

Strom nazveme optimální, liší-li se počty uzlů v podstromech $|n_L|$ a $|n_R|$ maximálně o 1 pro jeho každý uzel $n$.

*Poznámka:*

Hloubka optimálního stromu $V$ obsahujícího $|V|$ klíčů je:

$$log_2(|V|)$$

Algoritmus - Vybudování optimálního binárního stromu:

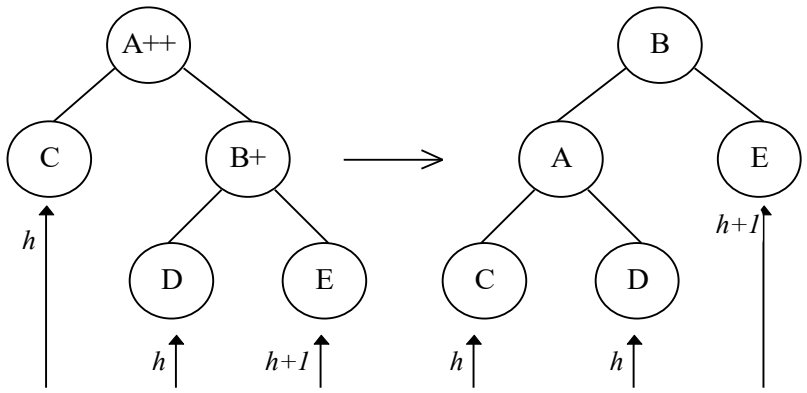
1. Vstup: množina klíčů $V$, uzel stromu $n$.
2. Je-li $V = \emptyset$, skonči.
3. Rozděl množinu $V$ na po dvou disjunktní množiny $V_1, \{med(V)\}, V_2$ tak, že $med(V)$ je medián množiny $V$, klíče z $V_1$ jsou menší než $med(V)$ a klíče z $V_2$ jsou větší než $med(V)$.
4. Polož $n = med(V)$.
5. Aplikuj algoritmus na množinu $V_1$ pro levý podstrom $n_L$.
6. Aplikuj algoritmus na množinu $V_2$ pro pravý podstrom $n_R$.

Definice – Vyvážené stromy:

Binární strom nazveme vyvážený, liší-li se hloubky $n_L$ a $n_R$ maximálně o 1 pro jeho každý uzel $n$ (hloubkou stromu rozumíme maximální délku cesty od kořene k listu).

Poznámka – Rotace ve vyvážených stromech:

Podmínku vyvážení lze udržovat dynamicky pomocí tzv. rotací (AVL stromy – Adelson, Velskij, Landis). Každý uzel si může zapamatovat hloubku levého a pravého podstromu. Při vložení nového klíče se strom v těch uzlech, ve kterých došlo k porušení podmínky vyvážení přeorganizujeme, např:



Počet uzlů v podstromu s kořenem $n$ označíme $|n|$.

Definice – $BB[\alpha]$ stromy:

Buď $0 < \alpha < \frac{1}{2}$. Binární strom patří do třídy $BB[\alpha]$ stromů, platí-li pro jeho každý uzel $n$:

$$\alpha < |n_L|/(|n_R| + |n_L|) < 1 - \alpha$$

Poznámka:
Podmínka platí triviálně i pro pravý podstrom:

$$\alpha < |n_R|/(|n_R| + |n_L|) < 1 - \alpha$$

Poznámka – význam parametru $\alpha$:

Pokud byl v nějakém okamžiku podstrom definovaný uzlem $n$ optimální, pak k porušení podmínky z definice $BB[\alpha]$ stromů musí dojít k minimálně $c \times |n|$ vložení/mazání uzlů do/z příslušného podstromu ($c$ je konstanta závislá pouze na parametru $\alpha$). To znamená, že čím více má podstrom uzlů, tím méně častěji dochází k porušení podmínky definice $BB[\alpha]$ a nutnosti jeho reorganizace.

Definice – $kD$ strom:

Úrovní $level(n)$ uzlu $n$ binárního stromu rozumíme délku cesty k tomuto uzlu od kořene stromu.

Buď $(S, \leq)$ úplně uspořádaná množina, $k > 0$,

$$x = (x_0, .., x_i, .., x_{k-1}), y = (y_0, .., y_i, .., y_{k-1}) \in S^k$$

Řekneme, že

$$x \leq_i y, \text{ jestliže } x_i \leq y_i$$

a pro $a \in S$

$$x \leq_i a \text{ jestliže } x_i \leq a$$

Analogicky jsou definovány operátory $<_i, \geq_i, >_i$

$kD$ stromem nad $S$ nazveme binární strom, jehož uzly jsou k-tice z $S^k$, a kde pro každý uzel $n$, jeho levý podstrom $n_L$ resp. $n_R$ platí:

$$n_l \in n_L \Rightarrow n_l \leq_i n, \qquad n_r \in n_R \Rightarrow n \leq_i n_r$$

kde

$$i = level(n)$$

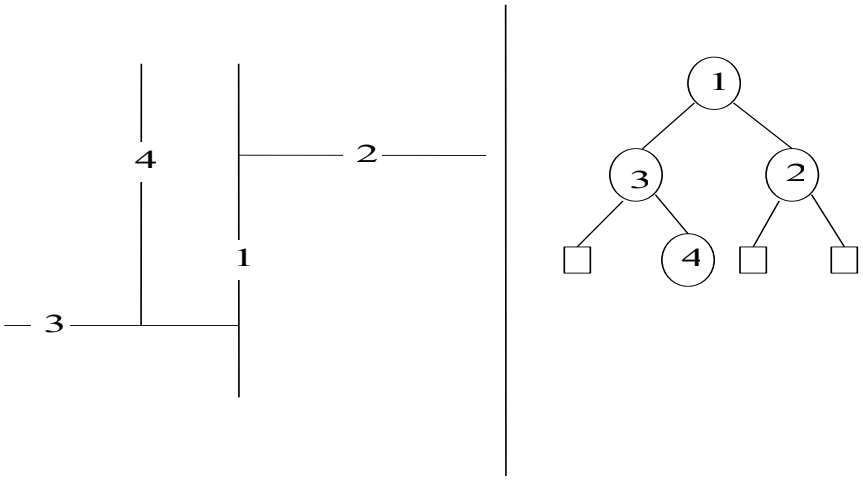
Algoritmus – Vyhledání bodu ve 2-d stromu:

Analogicky k binárním stromům, s tím rozdílem, že na každé úrovni použijeme jinou srovnávací metodu ($\leq_i$).

Algoritmus – Vložení bodu do 2-d stromu:

Analogicky k binárním stromům, hledáme ve stromu „bod“ dokud nenarazíme na volnou pozici. Do ní vložíme nový klíč.

Geometrická interpretace $2D$ stromu, každý vložený bod dělí, jistou část roviny na dvě „poloroviny“.

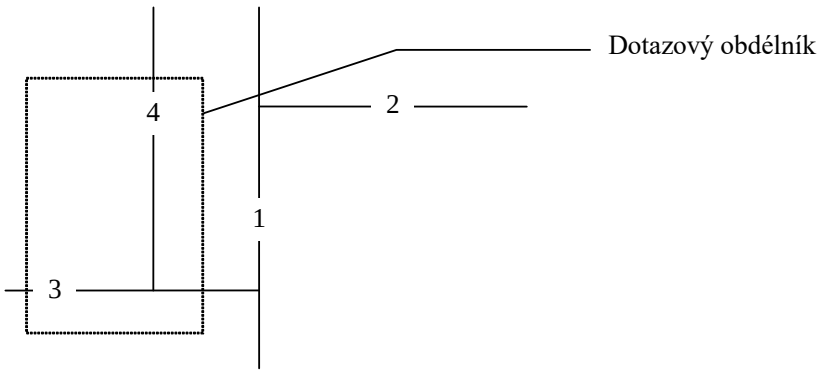



Algoritmus – Rozsahový dotaz pro body ve $2D$ stromu:

1. Vstup: Uzel $n$ a obdélník $[xmin, ymin, xmax, ymax]$.
2. Je-li $n = \emptyset$, konec.
3. Je-li $n$ (tj. „bod“ ve 2D prostoru) uvnitř dotazového obdélníku, pošli jej na výstup a aplikuj algoritmus na oba dva syny.
4. Vyber syny, pro které budeš aplikovat algoritmus, a to podle úrovně ve které se nachází uzel $n$, například je-li:

$$level(n)\ mod\ 2 = 0 \wedge\ n >_0 xmax$$

potom aplikuj algoritmus jen na větev $n_L$, analogicky pro další možné případy.



Vyvažování multidimensionálních stromů je komplikované. Nedají se totiž provádět rotace jako v klasických binárních stromech, protože v každém patře stromu měníme srovnávací kritérium, např. (obrázek rotace) z

$$B >_0 A \wedge D <_1 B \text{ neplyne } A <_0 B \wedge D >_1 A$$

Pomocí $BB[\alpha]$ techniky lze však $kD$ stromy udržovat vyvážené pomocí tzv. částečné reorganizace, tedy „hlídat“ v každém uzlu $kD$ stromu poměr počtu uzlů v jeho levém a pravém podstromu. V případě porušení podmínky $BB[\alpha]$ nahradit optimální podstrom.

Algoritmus – Vybudování optimálního $2D$ stromu:

1. Vstup: množina bodů $V$, uzel stromu $n$.
2. Je-li $V = \emptyset$, skonči.
3. $l = level(n)$, rozděl množinu $V$ na po dvou disjunktní množiny $V_1, \{med_l(V)\}, V_2$ tak, že $med_l(V)$ je takový bod, že jeho $l$-tá souřadnice je medián množiny $l$–tých souřadnic z $V$, $l$-té souřadnice z $V_1$ jsou menší než $med_l(V)$ a $l$-té souřadnice z $V2$ jsou větší než $med_l(V)$.
4. Polož $n = med_l(V)$.
5. Aplikuj algoritmus na množinu $V_1$ pro $n_L$ a $V_2$ pro $n_R$.

Poznámka:

Rozdělení množiny z kroku 3. lze realizovat snadnou modifikací algoritmu $QuickSort$.

Metodu $kD$ stromů lze použít i na obdélníky, které můžeme považovat za $4D$ body $[xmin, ymin, xmax, ymax]$. Použijeme tedy $4D$ strom.

Algoritmus – Rozsahový výběr pro obdélníky ve $4D$ stromu:

1. Vstup: kořen stromu $n$, dotazový obdélník $[xmin, ymin, xmax, ymax]$.
2. Je-li $n = \emptyset$, skonči.
3. Je-li obdélník $n$ incidentní s dotazem, pošli $n$ na výstup a aplikuj algoritmus na $n_L$ a $n_R$.
4. Jinak, podle úrovně $level(n)$, ve které se nacházíš ve stromu, se rozhodni, zda můžeš vynechat nějakou větev. Je-li např.:

$$level(node)\ mod\ 4 = 0 \wedge n >_0 xmax$$

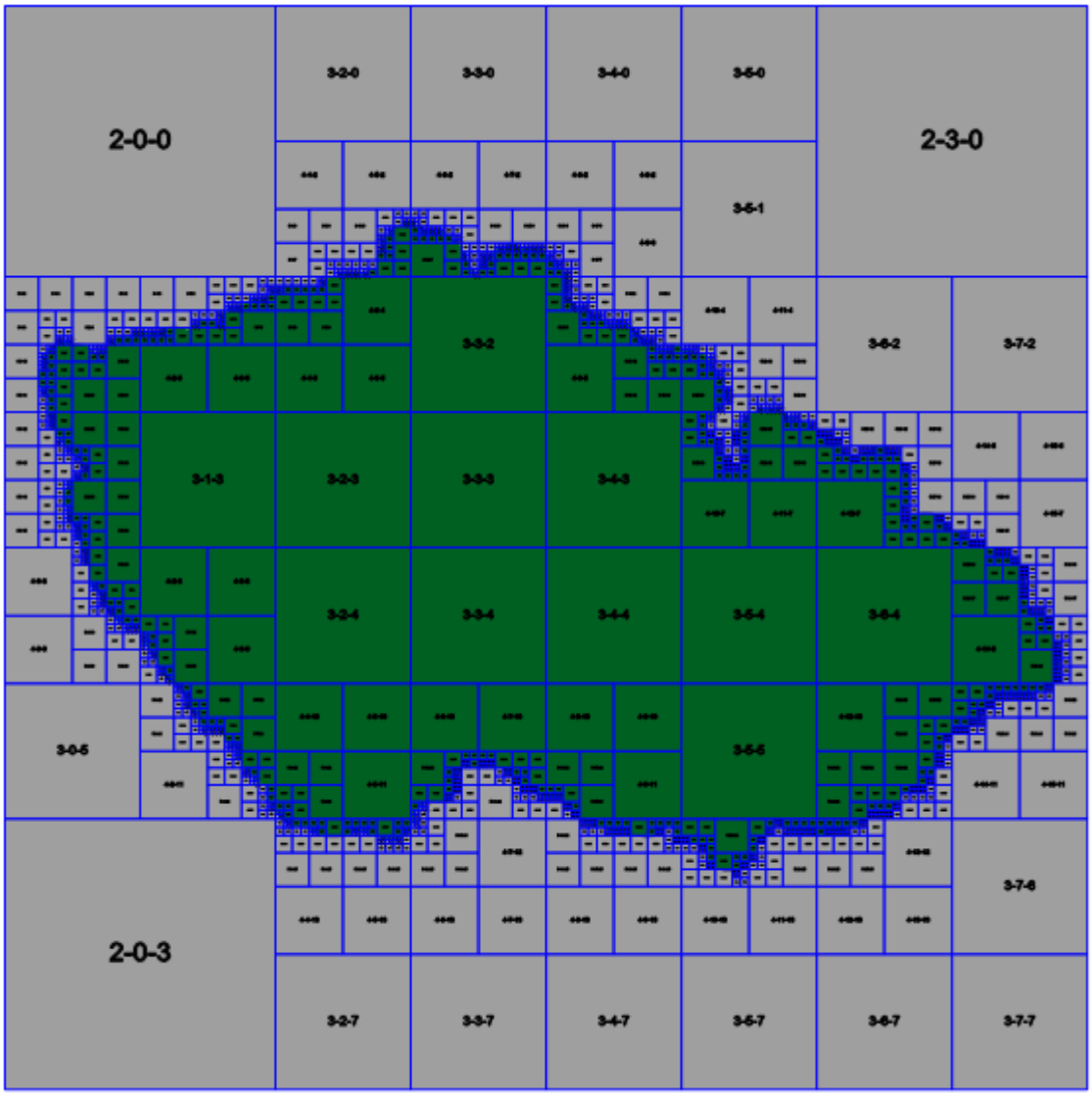
vynechej $n_R$. Analogicky pro další úrovně, v každé se dá za jistých podmínek jedna větev vynechat.

5. Aplikuj 1. pro vybrané následníky.

## 4.5. QUAD TREE – KVADRANTOVÉ STROMY

Kvadrantové stromy jsou prostorové vyhledávací struktury založené na pravidelném dělení části 2D prostoru (většinou čtverce) na čtyři stejné části. Každý uzel reprezentuje (čtvercovou) část oblasti a je opatřen příznakem, zda se areálový jev v této oblasti vyskytuje. Trik spočívá v tom, že je-li uzel označen jako prázdný/plný potom nemusíme budovat prázdný/plný podstrom – všechny jeho uzly „dědí“ tuto vlastnost.

V základní verzi tedy QuadTree mohou přímo definovat polygonální geometrii. $z-i-j$ na obrázku označuje: $z$ – hloubka uzlu $i, j$ maticové indexy vzhledem všem uzlům v dané úrovni.



Definice – kvadrantový strom:

Kvadrantový strom hloubky $h$ je kvartérní strom (tj. každý uzel má maximálně 4 následníky) s těmito vlastnostmi:

- Uzel $n$ je tvořen obdélníkem $R = [xmin, ymin, xmax, ymax]$ a příznakem $content(n) \in \{empty, full, half\}$.
- List $n$ má příznak $content(n) \in \{empty, full\}$.
- Nelistový uzel má čtyři následníky a příznak $content = half$.
- Následníci uzlu dělí obdélník rodičovského uzlu na čtyři „stejné“ části (podle středu).
- Maximální délka od kořene k listu je $h$.

Podle toho, jakou část rodičovského obdélníku reprezentuje uzel, označíme jej $TL, TR, BL, BR$ (Top/Bottom Left/Right).

Poznámka:

Struktura je velmi dobře použitelná v těch případech, kdy je prostor pevně rozdělen na „dlaždice“, například pro služby typu WMTS (viz datové sklady GIS). Každá úroveň podrobnosti $n$ je reprezentována maticí dlaždic $2^n \times 2^n$. Struktura potom může odpovídat na dotaz, zda je dlaždice obsazená (je na ní mapa/jev). Strukturu lze implementovat jednoduchým objektem:

```
public class QtreeNode
{ public QtreeNode TopLeft;
  public QtreeNode TopRight;
  public QtreeNode BottomLeft;
  public QtreeNode BottomRight;
  public QtreeContent Content;}
```

Struktura nemusí obsahovat souřadnice čtverce/obdélníku, který reprezentuje, stačí si pamatovat nultý čtverec/obdélník kořene stromu. Souřadnice ostatních uzlů jsou potom definovány cestou od kořene.

Poznámka:
Vlastnost $half$ je redundantní. Je ekvivalentní s „uzel není list“. Je zavedena pouze pro přehlednost.

Algoritmus – vyhledání dlaždice v QuadTree:

**1.** Vstup sloupec $col$, řádek $row$, podrobnost $zoom$ a uzel $n$. Nechť $c_i, i = 1,..,zoom$ je binární reprezentace sloupcového indexu $col$ a $r_i, i = 1,..,zoom$ je binární reprezentace řádkového indexu $row$.

2. $i = 0$

3. Je-li $content(n) \neq half$ vrať $content(n)$ a konec.

4. Je-li $i = zoom$ vrať $content(n)$ a konec.

5. Vyber pokračovací uzel $next$ takto:
   $r_i = 0$ - nahoru
   $r_i = 1$ - dolů,
   $c_i = 0$ - doleva,
   $c_i = 1$ – doprava

6. $n = next, i = i + 1$ a pokračuj 3.

Na struktuře lze velmi snadno a elegantně provádět množinové operace. Stačí implementovat operaci inverze a např. průniku, ostatní množinové operace lze provést pomocí těchto dvou.

Algoritmus – inverze QuadTree

1. Vstup uzel $n$.

2. Je-li $content(n) = full$ potom $content(n) := empty$ a návrat.

3. Je-li $content(n) = empty$ potom $content(n) := full$ a návrat.

4. Je-li $content(n) = half$ potom proveď kroky 2. a 3. pro všechny následníky.

Algoritmus – průnik QuadTree:

1. Vstup uzly dvou stromů $n_1, n_2$ (reprezentující stejnou plochu).

2. Je-li $content(n_1) = empty \vee content(n_2) = empty$, potom je výsledek uzel $n$ s obsahem $empty$.

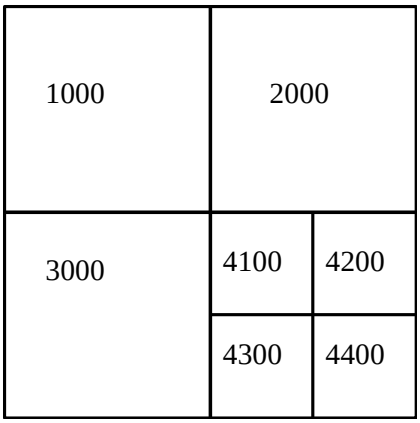
3. Je-li $content(n_1) = full$, potom je výsledek uzel $n_2$.

4. Je-li $content(n_2) = full$, potom je výsledek uzel $n_1$.

5. Jinak je výsledek nový uzel $n$ s obsahem $half$ a následníky z kroků 1. – 4.

Pevný kvartérní strom (non-pointer Quad Tree)

Zájmové území je postupně děleno na obdélníkové části a podle nich je jim přidělován „klíč“




Obr. - Číslování obdélníků-dlaždic v non-pointer Quad Tree.

Obdélník bude mít index takové dlaždice „pevné struktury“ která je jeho nadmnožinou a je nejmenší s touto vlastností.

- Pro libovolný obdélník $R$ označme $Q(R)$ jeho klíč v non-pointer QuadTree.
- Pro libovolný klíč $K$ označme jeho „nenulovou“ část, tedy levý podřetězec symbolem $NZ(K)$.
- Délku znakového řetězce $K$ označme $len(K)$.
- Podřetězec řetězce $K$ z levé strany délky $l$ označme $sub(K, l)$.

Tvrzení – incidence obdélníků n non-pointer QuadTree:

Buďte $A, B$ libovolné obdélníky, jejichž strany jsou rovnoběžné s osami souřadného systému. Nechť dále $A \cap B \neq \emptyset$.

Označíme-li,

$$l = min\{len(NZ(Q(A))), len(NZ(Q(B)))\}$$

potom:

$$sub(Q(A), l) = sub(Q(B), l)$$

tj. klíče $A$ a $B$ mají stejnou společnou nenulovou část.

Algoritmus - Vyhledání obdélníků v non-pointer QuadTree:

1. Vstup – obdélník $S = [xmin, ymin, xmax, ymax]$.

2. Pošli na výstup všechny obdélníky $A$, pro které:

$$sub\left(Q(A), len\left(NZ\left(Q(S)\right)\right)\right) = NZ\left(Q(S)\right) \wedge A \cap S \neq \emptyset$$

3. Pošli na výstup všechny obdélníky $A$, pro které:

$$Q(A) = P \wedge A \cap S \neq \emptyset$$

   kde $P$ jsou všechny klíče, které jsou na cestě od $Q(S)$ ke kořenu, tj. v $Q(S)$ zprava postupně nahrazujeme nenulové číslice nulami.

Tento postup má jednu nevýhodu. V případě, že například dotaz inciduje se středem území, potom procházíme v bodě 2 všechno. Této nevýhodě se vyhneme dekomponování dotazu.

Algoritmus - Dekompozice dotazu v non-pointer QuadTree:

1. Vstup – dotazový obdélník $S$.

2. Rozděl obdélník $S$ na obdélníky $S_1$ a $S_2$ ($S_1 \cup S_2 = S$) podle takové souřadnice $x$ resp. $y$, která způsobila klíčování v *non-pointer quadTree,* tj. takovou, která ohraničuje nějaký čtverec v *non-pointer quadTree* a prochází dotazovým obdélníkem $S$. V případě, že taková souřadnice neexistuje potom obdélník $S$ neděl a konec.

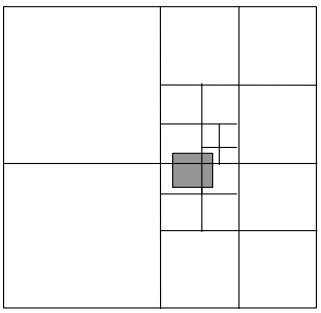
3. Aplikuj krok 2. na čtverce $S_1$ a $S_2$ podle druhé souřadnice.

Tímto postupem získáme maximálně 4 obdélníky na které aplikujeme algoritmus vyhledání obdélníků

Výhody této metody:

- Velmi snadná implementace v prostředí SQL – tedy relačních databází. Například norma OGC WKB nepředepisuje metodu efektivního výběru, tímto způsobem ji můžeme doplnit.

- „jeden objekt“ reprezentuje „jeden klíč“ znamená, že prostorová indexace je zabezpečena přímo v geometrické tabulce. Prostorový výběr nevyžaduje součin, či spojení s dalšími tabulkami.



Dekompozice dotazu – 4 obdélníky s klíči:
**2330000000,2343000000,4110000000,4120000000**

```
SELECT  ID FROM KM_ALL WHERE (
 (SPAT_KEY BETWEEN '2330000000' AND '2335000000')  OR
 (SPAT_KEY BETWEEN '2343000000' AND '2343500000')  OR
 (SPAT_KEY BETWEEN '4110000000' AND '4115000000')  OR
 (SPAT_KEY BETWEEN '4120000000' AND '4125000000')
) OR
SPAT_KEY IN
('0000000000', '2000000000', '2300000000',
 '2340000000', '4000000000', '4100000000'
)
AND (xmax>=-642646042) AND (ymax>=-1114990337) AND
    (xmin<=-569087654) AND (ymin<=-1070777051)
```

### 4.6. R-STROMY

Definice – B⁺-stromy:

B-strom řádu $m$ je strom s těmito vlastnostmi:

- Každý uzel má maximálně $m$ synů.
- Každý uzel, s výjimkou kořene a listů, má minimálně $m/2$ synů.
- Kořen má minimálně 2 syny, pokud není list.
- Všechny listy jsou na stejné úrovni.
- Nelistový uzel s $k$ syny obsahuje $k-1$ klíčů
- Klíče v uzlu $key_1, .., key_k$ jsou vzestupně uspořádány
- Uzly B⁺ stromu mají tvar $p_0 key_1 p_1 \ldots p_{k-1} key_k p_k$.
- ukazatel $p_i$ ukazuje na uzel, jehož všechny klíče jsou v intervalu $[key_i, key_{i+1}]$, formálně předpokládáme, že $key_0 = -\infty$ a $key_{k+1} = \infty$.

Algoritmus vložení klíče do B⁺ stromu:

1. Vstup klíč $key$ a kořen B⁺ stromu.

2. Není-li uzel list vyber dvojici klíčů $key_i, key_{i+1}$ s vlastností $key_{i-1} < key < key_i$ a pokračuj uzlem na který ukazuje $p_i$. Opakuj tento krok. Dokud uzel není list.

3. Vlož uzel do listu na správnou pozici. Je-li počet uzlů větší než $m$, rozděl uzel.

Algoritmus dělení uzlů v B⁺ stromu řádu $m$ :

1. Vstup uzel s $m + 1$ klíči.

2. Nechť $k = m/2 + 1$**,** vytvoř dva nové uzly:

$$p_0 key_1 p_1 \dots p_{k-2} key_{k-1} p_{k-1}$$

a

$$p_k key_{k+1} p_{k+1} \dots p_m key_{m+1} p_{m+1}$$

   a ukazatele na ně $q$ resp. $r$

3. Je-li uzel kořen, vytvoř nový prázdný kořen:

$$q\ key_k\ r$$

   Jinak, nechť $p_i$ je ukazatel z otcovského uzlu. Vlož klíč $key_k$ do otcovského uzlu na pozici:

$$..key_i\ p_i\ key_{i+1} \rightarrow key_i\ q\ key_k\ r\ key_{i+1}$$

4. Má-li otcovský uzel $m + 1$ klíčů, opakuj pro něj kroky 1.-3.

Definice R-strom:

Analogie k B-stromům, klíče jsou obdélníky.

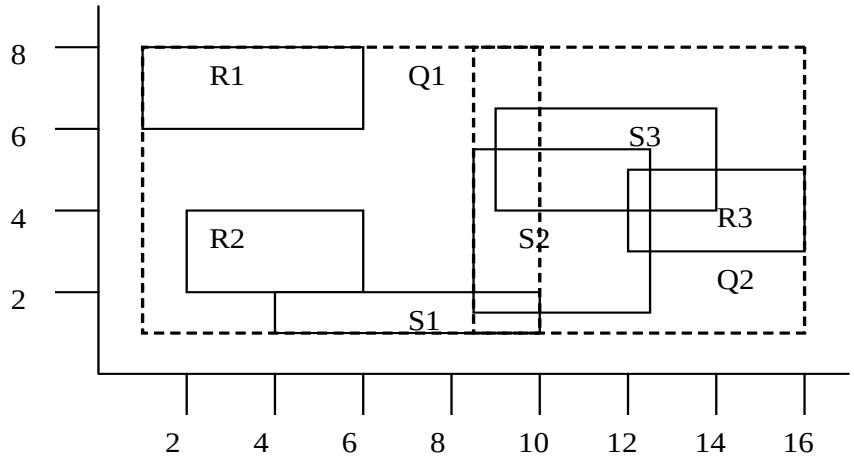
$M$ – maximální počet klíčů v uzlu,
$m \leq M/2$ – minimální počet klíčů v uzlu

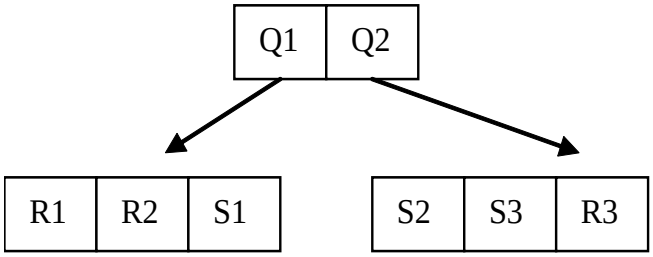
- Každý uzel obsahuje minimálně $m$ klíčů a maximálně $M$ klíčů, pokud není kořen.

- Klíče v R-stromech jsou obdélníky s ukazateli na synovské uzly, v listech obdélníky s ukazateli na geometrické prvky. Následníka obdélníku $K$ označíme $next(K)$, rodičovský uzel uzlu $n$ označíme $prev(n)$.

- Pro synovské uzly platí, že jejich klíče (tj. obdélníky) jsou uvnitř “otcovského” obdélníku.

- Listy stromu jsou na téže úrovni.
- Kořen obsahuje minimálně dva klíče, pokud není list.




Algoritmus – Vyhledání klíčů obdélníků v R- stromech:

1. Vstup uzel R-stromu $n$. Dotazový obdélník $Q$.
2. Je-li $n$ list, potom všechny klíče z $n$ incidentní s $Q$ na výstup.
3. Jinak aplikuj algoritmus na syny takových klíčů z $n$, pro které je klíč incidentní s $Q$.

Algoritmus – Vkládání klíčů do R- stromů:

1. Vstup, klíč $K$ (obdélník)
2. Vyhledej list $n$:
   a. Polož $n := root$ (kořen stromu).
   b. Je-li $n$ list pokračuj 3., jinak c).
   c. Nechť $k$ je klíč v $n$, jehož obdélník vyžaduje nejmenší rozšíření takové, aby obsahoval $K$. Rozšiř $k$ o klíč $K$, polož $n := next(n)$ a pokračuj b.
3. Přidej $k$ do vybraného listu $n$.

4. Je-li počet klíčů v $n$ menší, nebo roven $M$ konec. Jinak rozděl uzel $n$ na dva nové uzly. Je-li $n$ kořen, vytvoř nový kořen se dvěma novými klíči, jinak odstraň z rodičovského uzlu původní klíč a nahraď jej dvěma novými klíči a polož $n = prev(n)$.

5. Opakuj 4.

Problém rozdělení uzlu v R-Tree:

Najdi dva obdélníky (možná incidentní) s následujícími vlastnostmi (NP – úplný problém):

- Sjednocení obou obdélníků je původní obdélník

- Oba obdélníky obsahují zhruba stejný počet klíčů, splňují podmínku z definice R-tree

- Oba obdélníky se překrývají co nejméně

Algoritmus dělení uzlu R-stromu (kvadratická složitost):

1. Vyber první dva obdélníky

   a) Pro každou dvojici klíčů $k, l$ vytvoř minimální obdélník $j$ obsahující oba klíče a polož:

$$p(k,l) = Plocha(j) - Plocha(k) - Plocha(l)$$

   b) Vyber dvojici obdélníků $k, l$ s maximem $p(k,l)$, zařaď je do první a druhé skupiny.

2. V případě, že jedna skupina obsahuje tak málo obdélníků, že pro zachování podmínky minima $m$ musí obsahovat všechny nezařazené obdélníky, zařaď do ní zbývající obdélníky a konec.

3. Pro všechny nezařazené obdélníky spočítej rozdíl ploch, o které se zvětší obdélníky první a druhé skupiny začleněním nezařazeného obdélníku.

4. Vyber obdélník z 3. který má maximální rozdíl ploch a zařaď ho do té skupiny, jejíž celkový obdélník se rozšíří méně. Pokračuj krokem 2.

Algoritmus dělení uzlu R-stromu (lineární složitost):

1. Vyber první dva obdélníky:

    a. Pro každou dimenzi najdi klíče s maximem minima a minimem maxima, stanov „separační vzdálenost“ mezi těmito klíči (minimum minus maximum).

    b. Normalizuj separační vzdálenost tak, že vzdálenost intervalů podělíš rozsahem všech klíčů v dané dimenzi.

    c. Vyber dvojici $k, l$ s největší normalizovanou separační vzdáleností, zařaď je do první a druhé skupiny.

2. V případě, že jedna skupina obsahuje tak málo obdélníků, že pro zachování podmínky minima $m$ musí obsahovat všechny nezařazené obdélníky, zařaď do ní zbývající obdélníky a konec.

3. Vezmi další nezařazený klíč a zařaď jej do takové skupiny, jejíž MBR vyžaduje menší rozšíření.

R-Tree je nejpoužívanější metoda prostorové indexace, je nezávislá na velikosti objektů, nemusíme znát rozsah území, poměrně snadno je modifikovatelná na polygonální dotazy (viz algoritmus dotazu, dotazový obdélník můžeme nahradit dotazovým polygonem).

# 5. Funkce a operace nad geometrickými objekty

## 5.1. KONVERZNÍ FUNKCE (OGC):

Funkce umožňující konverze formátů, například vkládání do RDBMS obecným klientem SQL.

```
AsText(g Geometry)          : String
AsBinary(g Geometry)        : Binary
```

Používají pro vkládání a výběr geometrických objektů.

## 5.2. MĚŘÍCÍ FUNKCE

Délka:

```
Length(l LineString)        : double
Length(l Polygon)           : double
```

Plocha areálu:

Předpokládáme, že polygony (hranice) jsou “správně” orientovány,

```
Area(l Polygon) : Double Precision
```

$$A = 1/2 \sum_i (x_i - x_{i+1})(y_i + y_{i+1})$$

## 5.3. POLOHOVÉ FUNKCE

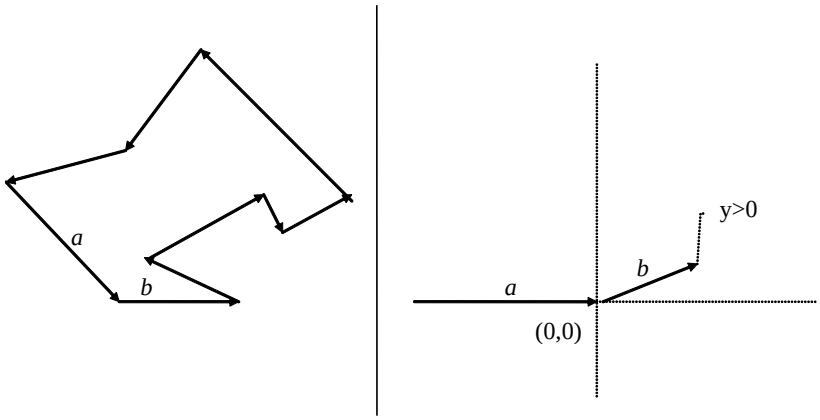
Určení orientace polygonu:

Algoritmus - Určení orientace polygonu:

1. Vyber z hranic oblastí takovou hranici a tři po sobě jdoucí její body tak, aby střední bod měl minimální souřadnici *y* (ze všech souřadnic $y$ v polygonu) a první bod měl souřadnici $y$ větší než bod prostřední.

2. Rotuj souřadnou soustavu tak, aby orientovaná úsečka definovaná prvními dvěma body splynula s osou $x$ v kladném směru.

3. Znaménko souřadnice *y* posledního bodu určuje orientaci polygonu.




Poloha bodu vůči polygonu:

Algoritmus - Bod v polygonu:

1. Najdi v bodech polygonu bod, jehož $y$ souřadnice je různá od souřadnice bodu, který testujeme. Nechť $pp$ je polopřímka vycházející z testovaného bodu rovnoběžná s osou $x$ v kladném směru. $nPrus := 0$. Od vybraného bodu postupně procházej všechny úsečky a proveď body 2 – 5.

2. Leží-li testovaný bod na úsečce, ukonči proceduru s výsledkem $NA\_HRANICI$.

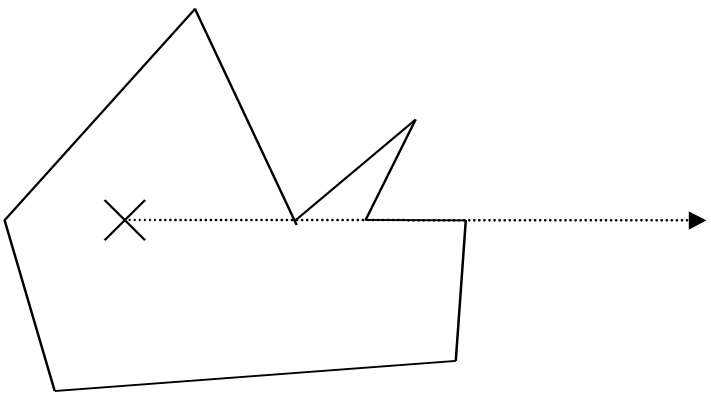
3. Má-li úsečka vlastní průsečík s polopřímkou $pp$, potom $nPrus := nPrus + 1$.

4. Končí-li úsečka na polopřímce a začíná-li mimo polopřímku, stanov podle počátku úsečky
   $odkud := POD$ nebo $odkud := NAD$.

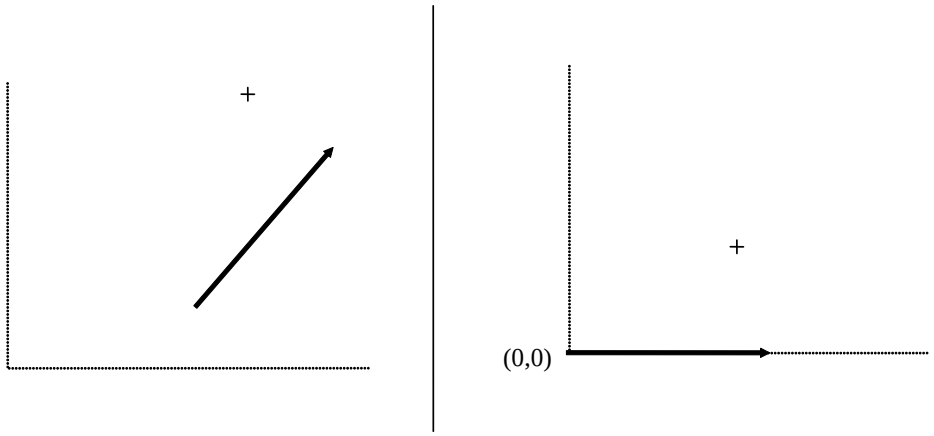
5. Začíná-li úsečka na polopřímce a končí-li mimo polopřímku a pokračuje-li do jiné poloroviny, než je stav proměnné $odkud$, potom $nPrus := nPrus + 1$.

6. Je-li $nPrus$ sudý, ukonči proceduru s výsledkem $VNĚ$.

7. Je-li $nPrus$ lichý, ukonči proceduru s výsledkem $UVNITŘ$.



Poloha bodu vůči úsečce:




Rotujeme souřadnou soustavu tak, aby její počátek splynul s počátečním bodem úsečky a koncový bod ležel na **x**-ové souřadnici v kladném směrů. Určení polohy je potom triviální operace (nalevo, napravo, minimální vzdálenost…)

$$x' = x.cos(\alpha) - y.sin(\alpha)$$
$$y' = x.sin(\alpha) + y.cos(\alpha)$$

Vzdálenost geometrických objektů:

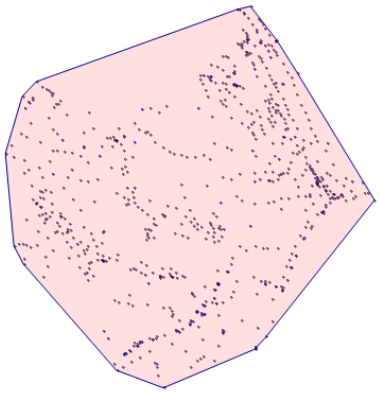
```
Distance(g1 Geometry, g2 Geometry) : Double
```

Kombinace metod podle typu geometrií: Vzdálenost bod-bod, poloha bodu vůči úsečce, průsečík úseček, bod v polygonu …

## 5.4. GEOMETRICKÉ OPERÁTORY

Konvexní obal množiny bodů:

Je nejmenší konvexní polygon s takovou vlastností, že všechny vstupní body leží uvnitř něj nebo na jeho hranici.



Konvexní polygon je polygon s touto vlastností: Každý vnitřní bod úsečky, jejíž krajní body leží na hranici polygonu, je i vnitřním bodem polygonu.

Nechť polygon $C = [s_0, .., s_n]$ je konvexní obal množiny bodů $S$, $s_0$ má minimální $y$ souřadnici a maximální $x$ souřadnici (v případě více minim $y$). Nechť dále $[\alpha_1, .., \alpha_n]$ jsou úhly mezi $x$ osou a úsečkou $[s_0, s_n]$. Potom:

- Bod $s_i$ leží nalevo od přímky definované body $[s_{i-2}, s_{i-1}]$, (úhel $[s_{i-2}, s_{i-1}, s_i]$ je konvexní).

- Polygon $C$ je tvořen nejvíce body z $S$ s předešlou vlastností
- $\alpha_i \leq \alpha_{i+1}$.

Algoritmus – konvexní obal (Graham scan - $O(Nlog(N))$:

1. Vstup – soubor $N$ bodů $S$.

2. Vyber z $S$ bod $P_0$ s minimální $y$-souřadnicí, ten nejvíce vpravo (max. $x$-souřadnice).

3. Setřiď body podle úhlu, které svírá přímka procházející daným bodem a bodem $P_0$ do pole $P[]$, bod $P_0$ bude prvním bodem pole.

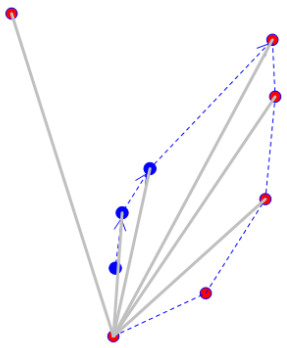
4. Vlož do zásobníku $R$ bod $P[0] = P_0$ a bod $P[1]$.

5. Pro $1 < i < N$, kde $N$ je počet bodů v $P$:

6. Nechť $p_1$ je první bod v zásobníku $R$, $p_2$ druhý bod

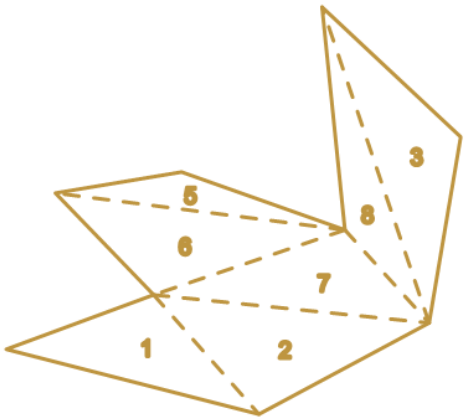
7. Je-li $P[i]$ napravo od přímky $p_1 \rightarrow p_2$ vlož $P[i]$ do zásobníku a $i := i + 1$, jinak odstraň $p_1$ ze zásobníku a znovu 6.

8. Zásobník $R$ obsahuje konvexní obal bodů.



Triangulace polygonu:

Je postup, kterým získáme sadu trojúhelníků, které pokrývají vstupní polygon.




Algoritmus triangulace „Ear clipping“:
(David Eberly - http://www.geometrictools.com/)

1. Vstup polygon bez děr. Vytvoř prázdný seznam trojúhelníků $T$ a pokračuj 2.

2. Vstup polygon bez děr, seznam trojúhelníků $T$.

3. Zorientuj polygon tak, aby měl kladnou orientaci.

4. Je-li polygon trojúhelník, přidej ho do seznamu a konec – výstup $T$.

5. Procházej postupně trojice po sobě jdoucích lomových bodů $p_i, p_{i+1}, p_{i+2}$. Zjisti, zda je úhel bodů $p_i, p_{i+1}, p_{i+2}$ konvexní, použijeme algoritmus polohy bodu $p_{i+2}$ vůči úsečce, $[p_i, p_{i+1}]$), po posunu a rotaci musí mít poslední bod kladnou $y$ souřadnici. Není-li tomu tak, pokračuj další trojicí bodů.

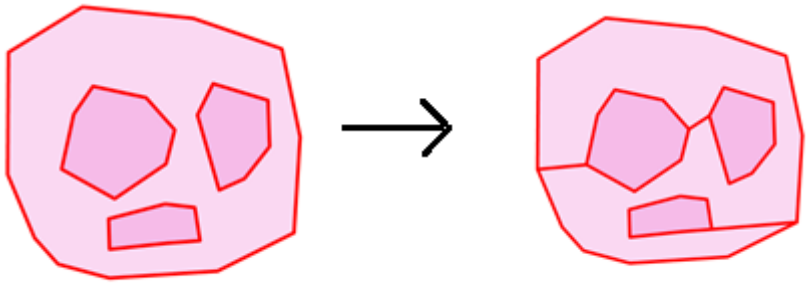
6. Otestuj, zda pro trojúhelník $t$ tvořený zkoumanou trojicí platí – bod $p_i$ je „viditelný“ z bodu $p_{i+2}$):

a) Žádný lomový bod vstupního polygonu neleží uvnitř $t$.
b) Žádná úsečka z hranice vstupního polygonu neprotíná žádnou úsečku hranice $t$.

7. V případě 5. je splněno, přidej do seznamu $T$ trojúhelník $t$. Odstraň bod $p_{i+1}$ z hranice polygonu a pokračuj 2. (snížili jsme počet bodů v hranici a rekurzivně aplikujeme týž postup).

Algoritmus „Ear clipping“ – pro polygony s dírami

Postupujeme tak, že polygon zorientujeme, vytvoříme z hranic jedinou hranici vhodným zřetězením a aplikujeme předešlý algoritmus. Například:



Vhodným zřetězením hranice rozumíme takový postup, že můžeme vkládat ‚spojnice‘ mezi jednotlivé hranice pouze mezi body, které jsou vzájemně přímo viditelné, tj. novou spojnici neprotíná žádná úsečka z řetězených hranic polygonu.

Poznámka:

Pro velké polygony vytvoříme prostorový index na všechny úsečky hranic a lomové body. Indexových struktur využijeme v kroku 6.

Poznámka:

Zřetězením hranic polygony s dírami obdržíme ‚nekorektní‘ hranici, je tečná sama se sebou, to ale pro navazující algoritmus není podstatné, vlivem zorientování zůstává zachována základní vlastnost konvexity úhlů.

**Množinové operace:**

```
Geometry Intersection (g1 Geometry, g2 Geometry)
Geometry Difference (g1 Geometry,g2 Geometry)
Geometry Union (g1 Geometry, g2 Geometry)
Geometry SymDifference(g1 Geometry, g2 Geometry)
Geometry Buffer (g1 Geometry, d Double)
```

Bod – bod:

Triviální operace. Pozor, pro příslušnost bodu k množině je nutné použít vhodnou přístupovou metodu k prostorovým datům.

Bod – lomená čára:

Vzájemná poloha úsečka X bod.

Bod – oblast:

Poloha bodu vůči polygonu

Lomená čára – lomená čára:

Poloha dvou úseček.

Lomená čára – oblast:

Algoritmus – Průnik lomené čáry s oblastí.

1. Vstup: oblast a lomená čára.

2. Ze vstupní lomené čáry vytvoř seznam $P$ segmentů lomené čáry takových, které buď neprotínají hranice oblasti, nebo jsou celé tečné.

3. Ze seznamu $P$ vytvoř seznam $S \subseteq P$ takový, že libovolný vnitřní bod každého segmentu z $S$ leží uvnitř oblasti.

4. Zřetěz segmenty z $S$ do “co nejdelších” lomených čar, a výsledek pošli na výstup

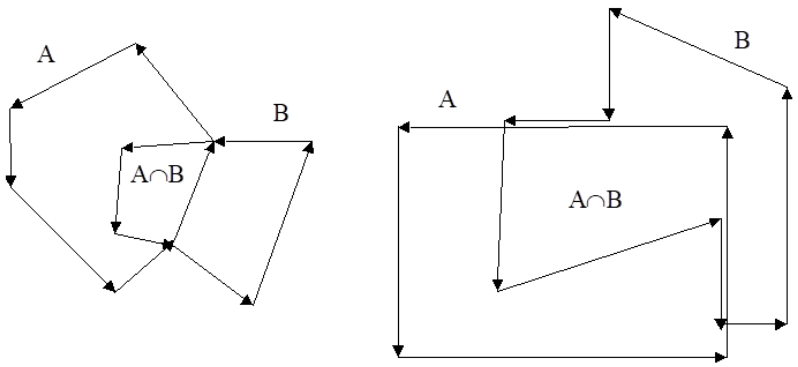
Oblast – oblast:

Množinové operace. Pro ty stačí algoritmizovat např. operaci doplňku a průniku, ostatní lze jimi vyjádřit.

Doplněk:

Prostá změna orientace hranic oblasti.

Průnik dvou oblastí:



Předpokladem je, že hranice vstupních polygonů jsou správně orientovány. Tyto doplníme o vnitřní body tak, že mimo lomové body se hranice vstupních polygonů neprotínají.

Leží-li část hranice oblasti uvnitř oblasti druhé, potom je i hranicí průniku. V jejím levém okolí se totiž nachází vnitřní body první oblasti a v pravém nikoli, zatímco v jejím levém i pravém okolí se nachází vnitřní body oblasti druhé. Část hranice je tedy součástí hranice průniku.

Obdobnou úvahou odvodíme, že pokud část oblasti splývá s částí oblasti druhé, potom je tato část hranicí operace průniku právě když splývající části mají totožnou orientaci.

Algoritmus – Průnik dvou oblastí:

1. Vstup: dvě orientované oblasti.

2. Všechny hrany hranic oblastí modifikuj tak, aby se vzájemně neprotínaly, mohou však splývat s hranami hranic z druhé oblasti. Potom mají tyto vlastnosti

   - hrana splývá s jinou z druhé oblasti
   - hrana leží celá uvnitř druhé oblasti
   - hrana leží celá vně oblasti

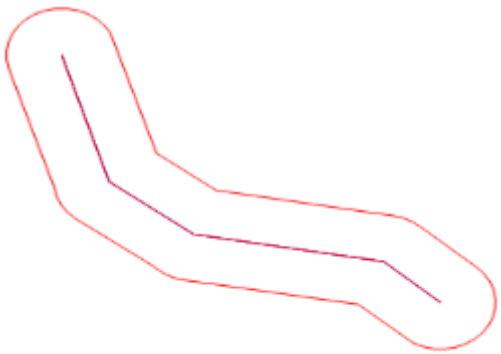
3. Do seznamu zařaď ty hrany, které buď, leží celé v druhé oblasti, nebo splývají s nějakou hranou z druhé oblasti, se kterou mají stejnou orientaci, (totožné hrany jen jednou).

4. Z vybudovaného seznamu zřetěz hranice výsledné oblasti a výsledek pošli na výstup.

(Nástin důkazu, že 4. Je uskutečnitelný…)

Obalová zóna poloměru $m$ (buffer zone):

Je polygon, jehož vnitřní body mají vzdálenost od vstupního objektu maximálně $m$.



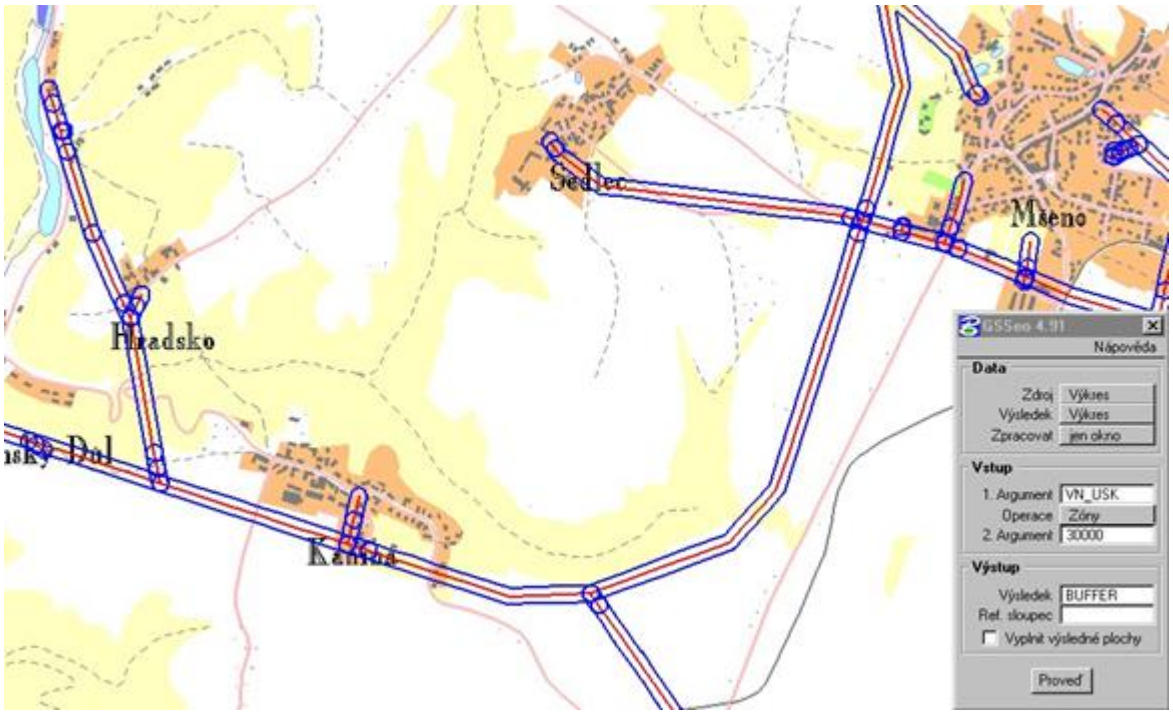
Algoritmus – Obalová zóna linie:

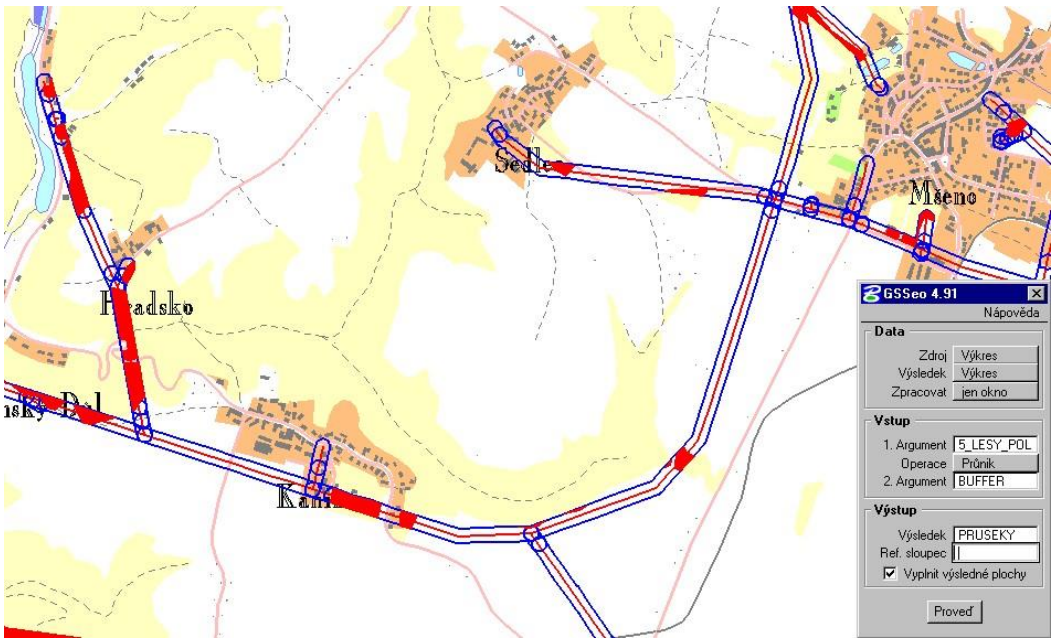
1. Obalíme jednotlivé segmenty (úsečky) lomené čáry.
2. Obaly segmentů sjednotíme.

Algoritmus – Obalová zóna oblasti:

1. Obalíme všechny hranice předešlým algoritmem.
2. Výsledek sjednotíme se vstupem (polygonem, oblastí)

Příklad: V GIS systému máme (mimo jiné...) vrstvu lesů reprezentovanou jako areály a vrstvu venkovních úseků vysokého napětí. Zajímá nás, kde lesy zasahují do bezpečnostního pásma 10 metrů kolem úseků vysokého napětí.
Jednotlivé úseky obalíme buffer zónou o daném poloměru.
Obalové zóny úseků nemusí být disjunktní, sjednotíme je.
Výsledek pronikneme s vrstvou lesů.



# 6. Rastrová data v GIS


Typy rastrových dat používaných pro GIS technologie jsou stejná jako v počítačové grafice:

- Binární
- Polotónová
- Víceúrovňová

Pro další práci očíslujeme sousedy pixelu P následujícím způsobem:

| | | |
|---|---|---|
| **3** | **2** | **1** |
| **4** | **P** | **0** |
| **5** | **6** | **7** |

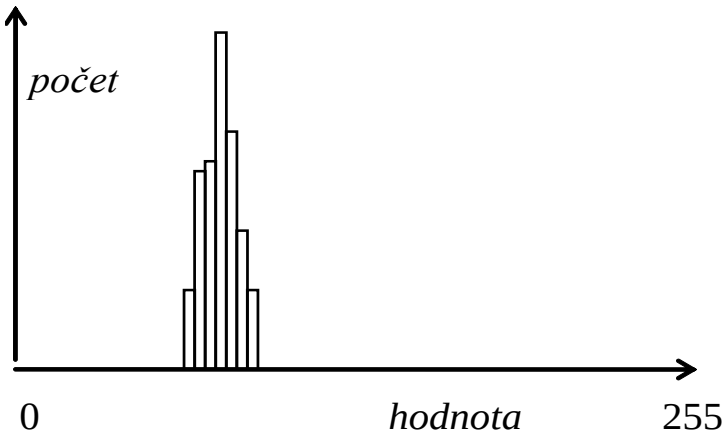
Sousedé 0,2,4,6 se nazývají přímí (d-) sousedé pixelu p.
Sousedé 1,3,5,7 se nazývají nepřímí (i-) sousedé pixelu p.

Pro polotónová obraz $f$ označme $f(i,j)$ barvu jeho pixelu na pozici $i,j$.

### 6.1. KVANTITATIVNÍ CHARAKTERISTIKY OBRAZU

Definice - Histogram obrazu:

Nechť $f$ je polotónový obraz barev $1..M$. Jeho *histogramem* rozumíme konečnou posloupnost $h(f) = (h_1..h_M)$, kde, $h_i$ je počet pixelů s barvou $i$.




Obr. 18 - Histogram obrazu

Definice - Matice sousednosti:

Nechť $f$ je polotónový obraz barev $1..M$. Jeho maticí sousednosti rozumíme čtvercovou $MxM$ matici $CM(f) = \{cm_{i,j}\}$, kde, $cm_{i,j}$ je počet (přímo) sousedících pixelů o barvě $i$ a $j$.

## 6.2. OPERACE NAD OBRAZY

Lineární filtrace:

Buď $f$ polotónový obraz, $M > 0$. Položme $g(x,y) = H(M,x,y)$

kde $H$ je libovolná funkce, která v konstantním čase počítá novou hodnotu pixelu $g(x,y)$ z okolí pixelu $x,y$ o rozměru $M$.

Funkce $H$ bývá většinou vyjádřena váženým průměrem pixelů a lze ji vyjádřit:

$$H(M,x,y) = \Sigma_{i=-M,M}\Sigma_{j=-M,M}\, h(i,j) * f(x+i, y+j)$$

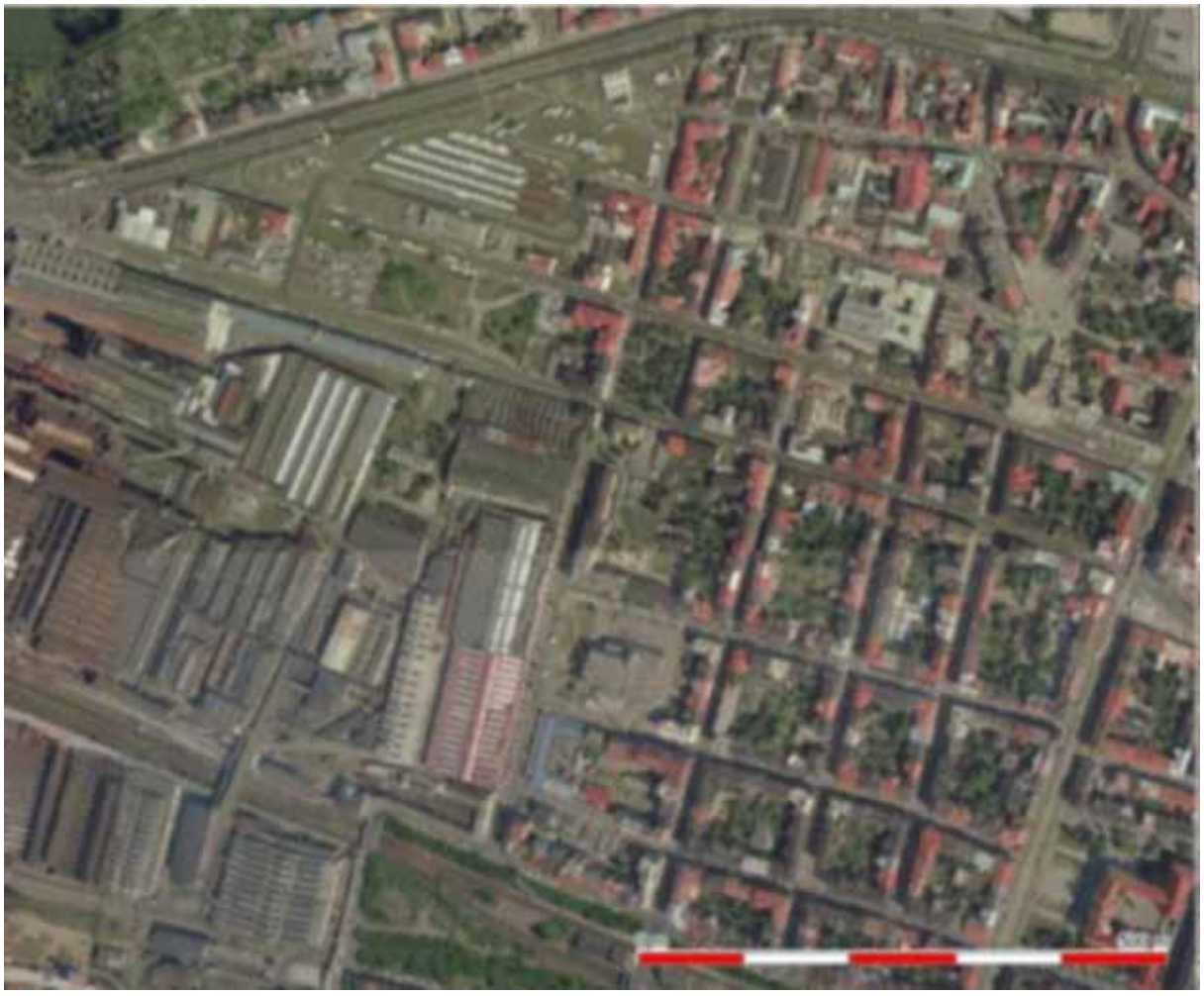
Zhlazující filtry:

a)

| | | |
|---|---|---|
| **1** | **1** | **1** |
| **1** | **1** | **1** |
| **1** | **1** | **1** |

b)

| | | |
|---|---|---|
| **1** | **2** | **1** |
| **2** | **4** | **2** |
| **1** | **2** | **1** |



Zostřující filtry:

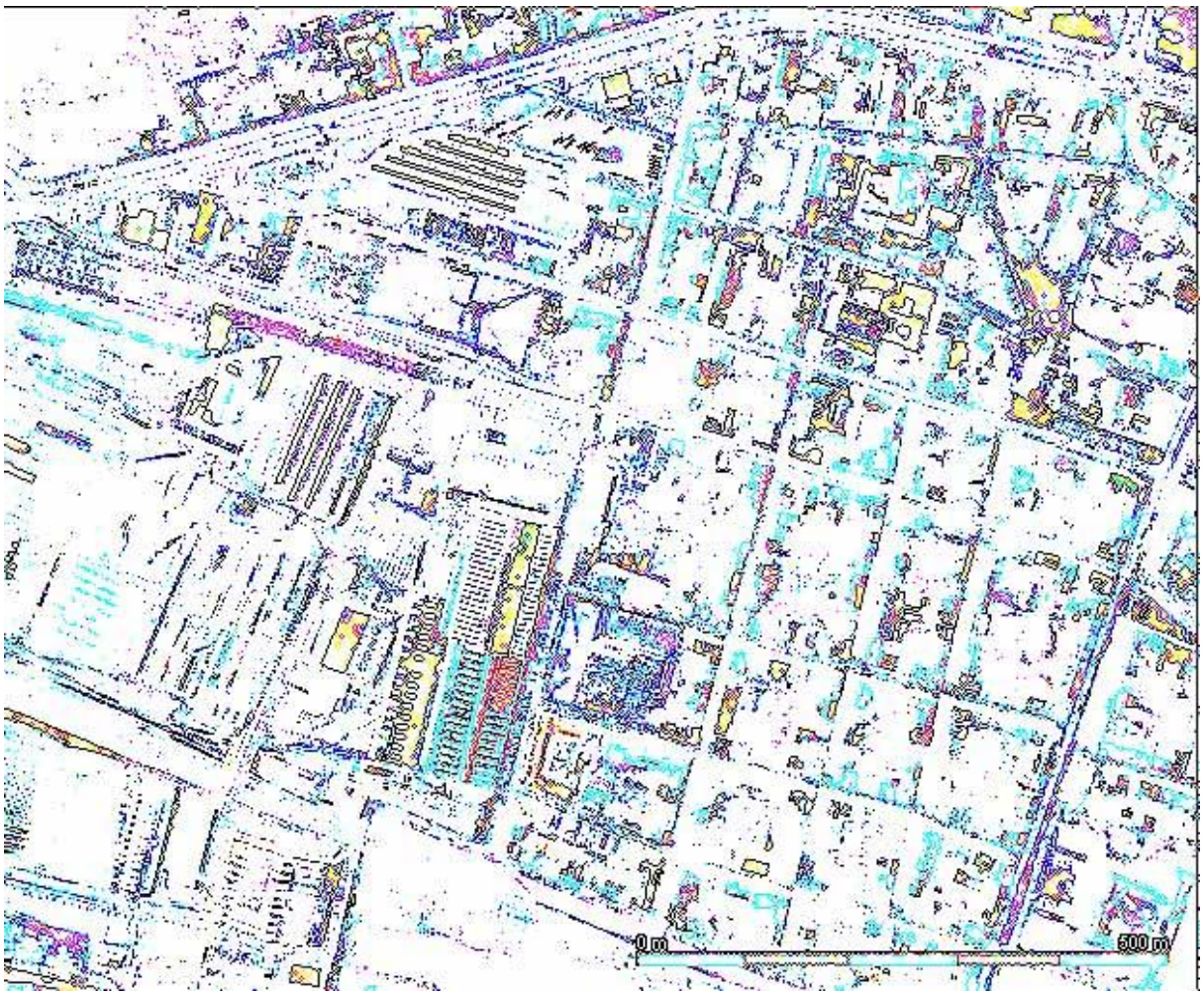
| | | |
|---|---|---|
| **-1** | **-1** | **-1** |
| **-1** | **n** | **-1** |
| **-1** | **-1** | **-1** |

Detektory hran:

Základním filtr této třídy přiřadí novému pixelu největší absolutní hodnotu ze dvou výsledků:

| | | |
|---|---|---|
| **1** | **2** | **1** |
| **0** | **0** | **0** |
| **-1** | **-2** | **-1** |

| | | |
|---|---|---|
| **1** | **0** | **-1** |
| **2** | **0** | **-2** |
| **1** | **0** | **-1** |




Původní obraz

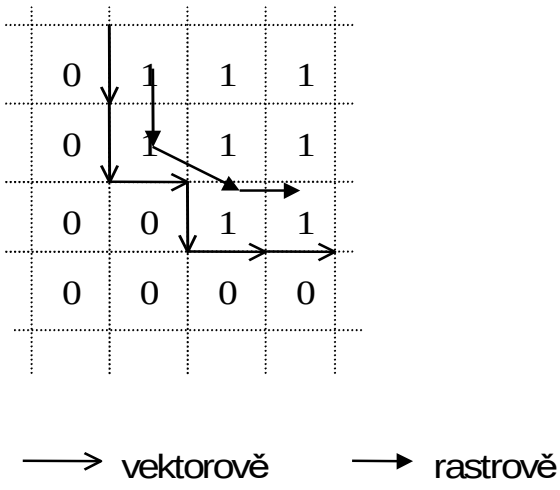
Zhlazující filtr

Zostřující filtr

Detektor hran:

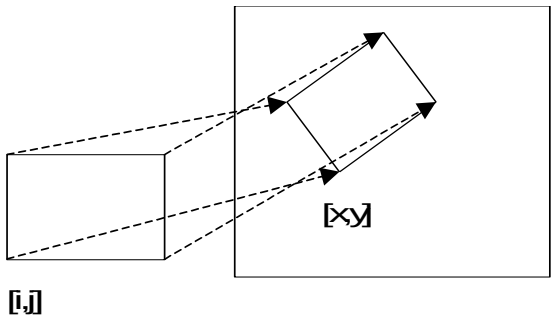
<u>Kontura (hranice) oblasti v binárním obrazu:</u>

Nechť *O* je libovolná oblast (množina složená z jedniček) v binárním obrazu, Konturou (hranicí) oblasti *O* rozumíme všechny pixely patřící této oblasti, které mají nulového d-souseda.



## 6.3. TRANSFORMACE RASTROVÝCH DAT

Velmi častá úloha, zejména pro umisťování obrazu do dané kartografické projekce.




1. Určíme bodové objekty ve zdrojovém obrazu, jejichž (kartografické) souřadnice jsou známé. Například přesně odečtené z mapy, geodeticky zaměřené apod.

2. Určíme transformační funkce z nového do starého obrazu komerční produkty většinou poskytují polynomiální transformaci založené na metodě nejmenších čtverců, je však možné použít i přesnou kartografickou transformaci.

$$i = F(x, y)$$
$$j = G(x, y)$$

   kde $(i, j)$ značí souřadný systém originálního obrazu, $(x, y)$ souřadný systém obrazu nového.

3. Procházíme cílový obraz, pomocí transformačních funkcí $F$ a $G$ se „díváme“ do originálu a počítáme hodnotu pixelu. Podle toho, z jakého okolí zdrojového pixelu určujeme výslednou hodnotu pixelu nového, se hovoří o metodách:

   - nejbližší soused
   - bilineární transformace
   - konvoluce okolí $MxM$

První případ prostě přenese hodnotu pixelu do nového obrazu, v dalších případech se výsledná hodnota se počítá z jistého okolí pixelu v originálním obrazu.

Velmi často se setkáváme s následující situací (například v klientech WMS služby).

- Známe omezující obdélník cílové kartografické projekce a rozměr obrazu v pixelech
- Známe sadu projekcí, které poskytuje (WMS) server
- Potřebujeme dostat mapovou kompozici do “okna” na klientskou stanici.

V tomto případě postupujeme takto:

1. Vybereme vhodnou serverovou projekci (např. $WGS84$, tu poskytuje každý WMS server).

2. Transformujeme dotazový obdélník do serverové projekce, upravíme pixelový rozměr obrazu. K tomuto účelu použijeme ”přesný” převod souřadnic:

$$rovina_1 \rightarrow elipsoid_1 \rightarrow centr_1 \rightarrow centr_2 \rightarrow elipsoid_2 \rightarrow rovina_2$$

   (viz transformace souřadných systémů).

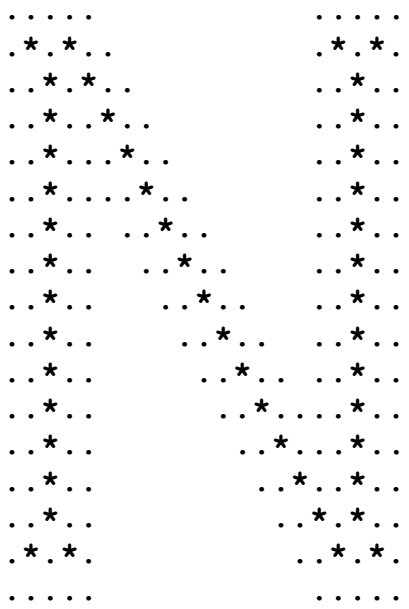
3. Provedeme dotaz, získáme obraz v souřadnicích $(i, j)$**.**

4. Vybereme sadu zdrojových pixelových souřadnic (většinou pravidelná mřížka $NxN$**,** nebo rohy obrazu). Tyto převedeme do zdrojových souřadnic, poté “přesnou” kartografickou transformací do souřadnic cílové projekce a cílových souřadnic pixelových $(x, y)$. Takto získáme sadu dvojic “odpovídajících si” pixelů, kterou použijeme pro získání parametrů polynomiální transformace obrazu.

Tento postup “šetří” výpočtovou náročnost tím, že nahradí náročné přepočty kartografických souřadnic polynomiální transformací.

Skelet binárního obrazu:

Při vektorizaci liniových prvků je prvním krokem zúžení rastrových linií tak, aby byly maximálně dva pixely široké.



Definice – Skelet:

Nechť $R$ je množina pixelů, $B$ její hranice (kontura), $P$ bod v $R$. Nejbližší soused bodu $P$ *na hranici* $B$ je bod $M$ z $B$ takový, že pro každý bod $M'$ z $B$ je vzdálenost $PM'$ větší nebo rovna vzdálenosti $PM$. Má-li bod $P$ více než jednoho nejbližšího souseda, nazývá se *bodem skeletu množiny* $R$. Sjednocení všech bodů skeletu tvoří *skelet množiny* $R$.

Algoritmus - Určení skeletu:

1. Urči hranici (konturu) $B(R)$ množiny $R$.

2. Urči množinu násobných pixelů $M(R)$ v hranici $B(R)$

3. Je-li $B(R) = M(R)$, skonči.

4. Polož $R = R - (B(R) - M(R))$ a pokračuj krokem 1.

(a)

| | | |
|---|---|---|
| A | A | A |
| 0 | P | 0 |
| B | B | B |

(b)

| | | |
|---|---|---|
| A | A | A |
| A | P | 0 |
| A | 0 | 2 |

(c)

| | | |
|---|---|---|
| A | A | C |
| 0 | P | 2+ |
| B | B | C |

Pixel označený 2 je hraniční pixel, pixel označený 2+ je hraniční nebo násobný pixel.

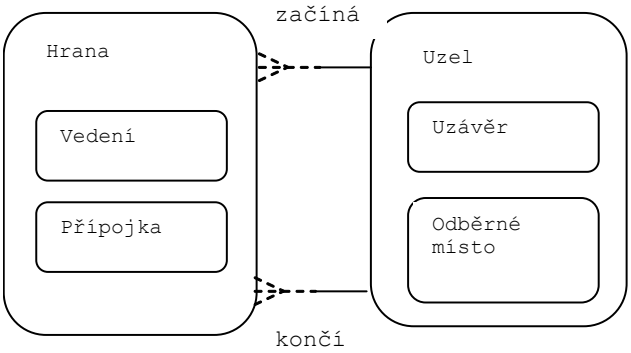
(a), (b): Alespoň jeden pixel ze skupiny pixelů A, B je nenulový

(c): Alespoň jeden pixel ze skupiny C musí být nenulový. Pokud jsou oba nenulové, pak může být hodnota pixelů ve skupinách A i B libovolná. Pokud je jeden pixel skupiny C nulový, musí být alespoň jeden pixel skupiny A i skupiny B nenulový.

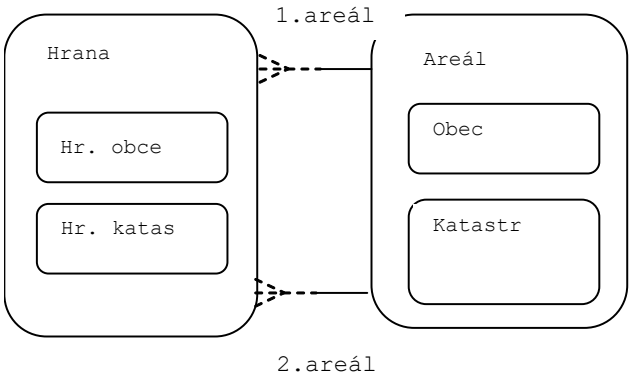
# 7. Topologické úlohy v GIS

Byly efektivně řešeny před tím, než byly vůbec GIS technologie vymezeny, přesto je můžeme považovat za součást analytického jádra topologicky orientovaných GIS.

Základní datová struktura síťového grafu:




Základní datová struktura areálového grafu:




Nejčastější úlohy:

Trasování grafu:

Vyber všechny uzly/hrany, které jsou "napájeny" z daného uzlu, hodně používaná úloha pro dispečery sítí, modelování situací „co se stane, když?".
Nejkratší cesta z uzlu do uzlu:

Používá se klasický Dijkstrův algoritmus. Lze výhodně využít vlastnosti, že uzly grafu jsou prostorově lokalizovány.

Algoritmus „Minimální cesta (Best search):

Nechť $(U, H)$ je graf s nezáporně ohodnocenými hranami, váhu libovolné hrany $h \epsilon H$ označme $w(h)$.
Nechť dále každý uzel $u_i \epsilon U$ má 2D souřadnice $(x_i, y_i)$.
Pro libovolné uzly $u, v$ označme $d(u, v)$ jejich vzdálenost v $E_2$. Pro libovolnou hranu $h_i = (u_i, v_i)$ nechť dále platí trojúhelníková nerovnost metrického prostoru $E_2$.

$$d(u_i, v_i) \leq w(h)$$

Potom pro libovolné uzly $u_0, v \epsilon U$**,** následující postup vede k nalezení minimální cesty z $u_0$ do $v$.

1. Inicializace: Pro každý uzel $u_i \epsilon U, u_i \neq u_0$ položme $d\_path_i = \infty$, $d\_path_0 = 0$, dále položme každý uzel $u_i \epsilon U$ $c\_path_i = null.$

2. Výběr pivota: Položme $c\_path_i = d\_path_i$ pro takový $u_i$, pro který je $c\_path_i = null, d\_path_i < \infty$ a pro který je $d\_path_i + d(u_i, v)$ minimální. Když neexistuje – končíme, cesta neexistuje. Je-li $u_i = v$, potom konec, $c\_path_i$ je délka minimální cesty a provedeme „zpětný“ chod.

3. Expanze: Pro všechny uzly $u_k \epsilon U$ takové, že existuje hrana $h_k \epsilon H, h_k = (u_i, u_k)$ ($u_i$ je pivot z předchozího kroku) položme $d\_path_k = min\{d\_path_k, c\_path_i + w(h_k)\}$ a pokračujeme 2.

Problém obchodního cestujícího:

Hamiltonovská kružnice v grafu, extrémně obtížná úloha, dosud nebyla uspokojivě vyřešena (jedná se o NP úplný problém).

Topologicko-geometrické úlohy:

- Vytváření topologických vazeb na základě geometrických vlastností objektů; jedná se o vytvoření příslušného typu grafu (uzel-hrana, hrana-hrana, areálový graf). Hojně se využívá přístupových metod pro geometrické objekty.

- Generování oblastí z hran areálového grafu.

- Identifikace hran areálového grafu.

- Generování vyšších územních celků areálového grafu.

- Kontrola konzistence geometrických a topologických vlastností dat (kontrola shody umístění uzlu a konce hrany s ním incidentní, kontrola křížení hran, kontrola stupňů uzlových bodů a další kontroly).

# 8. 3D geometrie v GIS

## 8.1. 3D Geometrické primitivy

Jsou zcela analogické 2D s tím rozdílem, že vycházejí z 3 dimensionálních bodů $[x, y, z]$.

## 8.2. Odhad normály množiny 3D bodů, kovarianční matice bodů

1) $ax + by + cz + d = 0$ je rovnice roviny v prostoru.
2) $\left[x_i^p, y_i^p, z_i^p\right], i = 1, .., n$ jsou body vstupní body
3) Úloha: Pro body $\left[x_i^p, y_i^p, z_i^p\right], i = 1, .., n$ hledáme $a, b, c, d$ , co nejlepší aproximaci roviny z 1), resp. její normálový vektor $[a, b, c]$.
4) Nechť bod $[\bar{x}, \bar{y}, \bar{z}] = [\frac{\sum x_i^p}{n}, \frac{\sum y_i^p}{n}, \frac{\sum z_i^p}{n}]$ je centroid bodů z 2).
5) Položíme $[x_i, y_i, z_i] = \left[x_i^p - \bar{x}, y_i^p - \bar{y}, z_i^p - \bar{z}\right], i = 1, .., n$. Triviálně, posunem bodů „do centroidu“ se normálový vektor nezmění. Dále, po této úpravě máme:

$$\sum x_i = \sum y_i = \sum z_i = 0$$

6) Hledáme nejlepší řešení soustavy rovnic $(n \geq 4)$:

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & z_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}$$

Opět použijeme metodu nejmenších čtverců, dostaneme soustavu rovnic:

$$\begin{bmatrix} \sum x_i x_i & \sum y_i x_i & \sum z_i x_i & \sum x_i \\ \sum x_i y_i & \sum y_i y_i & \sum z_i y_i & \sum y_i \\ \sum x_i z_i & \sum y_i z_i & \sum z_i z_i & \sum z_i \\ \sum x_i & \sum y_i & \sum z_i & n \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

7) Z 5) a posledního řádku matice soustavy máme $d = 0$. Tedy hledáme netriviální řešení (tj. $[a, b, c] \neq [0,0,0]$):

$$\begin{bmatrix} \sum x_i x_i & \sum y_i x_i & \sum z_i x_i \\ \sum x_i y_i & \sum y_i y_i & \sum z_i y_i \\ \sum x_i z_i & \sum y_i z_i & \sum z_i z_i \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Povšimněme si, že tato soustava má vždy triviální řešení, netriviální řešení dostaneme pouze tehdy, je-li matice této soustavy singulární. O triviální řešení ale zájem nemáme, zajímá nás řešení netriviální.

8) Je-li $[a, b, c, 0]$ řešením 1), potom je triviálně i $k[a, b, c, 0], k \epsilon \boldsymbol{R}$ řešením 1)

10) Dále, v netriviálním řešení musí platit, že buď $a \neq 0$, nebo $b \neq 0$, nebo $c \neq 0$. Předpokládejme tedy, že $c \neq 0$ a vzhledem k 9) můžeme předpokládat $c = 1$. Z 8) tedy dostáváme:

$$\begin{bmatrix} \sum x_i x_i & \sum y_i x_i & \sum z_i x_i \\ \sum x_i y_i & \sum y_i y_i & \sum z_i y_i \\ \sum x_i z_i & \sum y_i z_i & \sum z_i z_i \end{bmatrix} \times \begin{bmatrix} a \\ b \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}..$$

a tedy:

$$\begin{bmatrix} \sum x_i x_i & \sum y_i x_i \\ \sum x_i y_i & \sum y_i y_i \end{bmatrix} \times \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} -\sum z_i x_i \\ -\sum z_i y_i \end{bmatrix}$$

V případě nulového determinantu této matice zaměníme předpoklad z 10) $c = 1$ za $b = 1$, resp. $a = 1$. Máme řešení úlohy z 3).

## 8.3. 3D POLYGON

Ve 2D případě požadujeme po hranicích polygonů, aby se vzájemně neprotínaly. Ve 3D variantě požadujeme, aby tato vlastnost byla splněna pro ortogonální projekci do ‚nejlepší' proložené roviny z předešlého odstavce. Tato vlastnost nám především umožňuje jejich kresbu pomocí triangulace. Postup je následující:

1) Posuneme polygon do jeho centroidu.

2) Provedeme odhad „nejlepší" normály.

3) Rotujeme polygon v rovině XY a poté v rovině XZ tak, aby jeho normála z 2) byla rovnoběžná s osou Z.

4) Provedeme triangulaci, zohledňujeme jen souřadnice XY.

5) Pro výsledné trojúhelníky provedeme zpětnou transformaci z kroků 3) a 1).

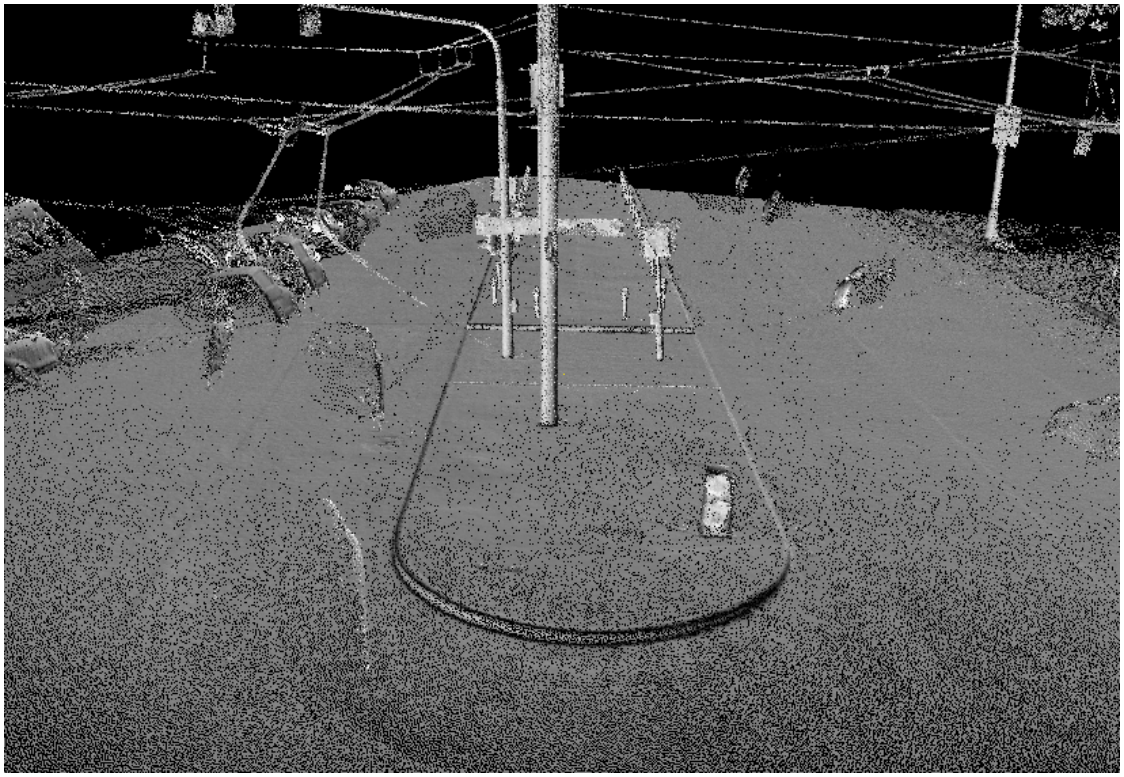
## 8.4. MRAČNA BODŮ

Mračna bodů jsou moderním zdrojem pro získávání geometrické (geograficky vztažené) informace. Vznikají snímáním (leteckým/pozemním) zemského povrchu laserovými scannery, výsledkem jsou 3D body s doplňující informací (intensita odrazu, barva). Jsou definovány standardy formátů, ve kterých jsou mračna poskytována (LAS).

*Zobrazení mračna bodů v centrální projekci*

## 8.5. OSVĚTLENÍ MRAČNA PODLE NORMÁL

V některých případech je například vhodné pro vizualizaci mračna použít jeho přebarvení podle prostorových vlastností. Základní metodou pro zvýraznění je ‚osvětlení' z určitého zdrojového bodu (vektoru), tj. jednotlivé body obarvit podle úhlu normály lokální roviny vůči vektoru osvětlení. Lokální rovinou rozumíme opět nejlepší proložení rovinou jistým (relativně malým) okolím jednotlivých bodů. Zdůrazněme, že tyto metody vyžadují velmi výkonnou prostorovou indexaci mračna bodů. Není výjimkou, že zpracováváme řádově $10^7$ bodů a bez prostorové indexace, a tedy s kvadratickou časovou složitostí se dostáváme mimo rozumný výkon ($10^{14}$ je hodně, i když počítače jsou výkonné). Pro každý bod mračna totiž potřebujeme hodně rychle vybrat všechny body z jeho okolí.



*Zobrazení mračna bodů obarveného podle úhlu normálových vektorů*

## 8.6. VLASTNÍ HODNOTY A VEKTORY KOVARIANČNÍ MATICE

Připomeňme si z lineární algebry, co jsou to vlastní vektory a vlastní hodnoty matice.

Buď $A$ čtvercová matice, vlastním vektorem $u$ nazveme takový vektor, pro který platí:

$$Au = \lambda u$$

kde $\lambda$ je číslo, které nazýváme vlastní hodnotou matice $A$.

Vlastní vektor je tedy takový vektor, který aplikací transformace reprezentované maticí $A$ nemění směr, mění se pouze jeho velikost faktorem $\lambda$.

Některé užitečné vlastnosti vlastních čísel a vektorů matic:

- Nula je vlastním číslem matice právě tehdy, když je matice singulární.
- Je-li matice symetrická a reálná (tj. obsahuje pouze reálná čísla), pak všechna její vlastní čísla jsou reálná. (*)
- Vlastní vektory reálné symetrické matice jsou po dvou ortogonální (tj. jejich skalární součin je nulový, jsou na sebe kolmé). (*)

(*) jsou základních vlastností Hermitovských (samosdružených) operátorů v Hilbertově vektorovém prostoru.

Matici nazveme positivně definitní, resp. positivně semidefinitní, jsou-li všechna její vlastní čísla kladná, resp. nezáporná.

Položme, viz 8. 2 8):

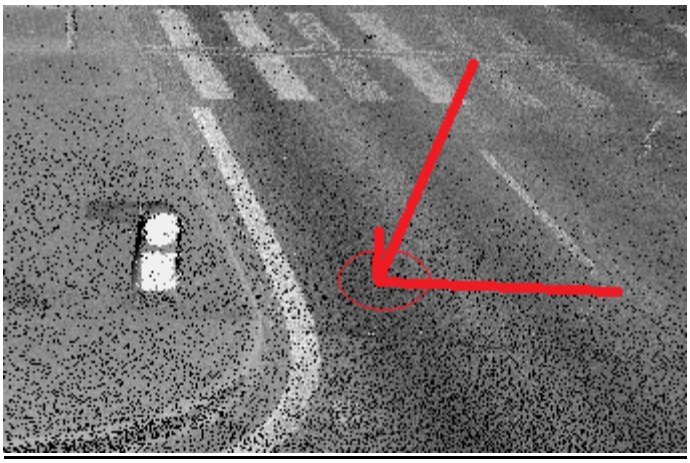
$$C = \begin{bmatrix} \sum x_i x_i & \sum y_i x_i & \sum z_i x_i \\ \sum x_i y_i & \sum y_i y_i & \sum z_i y_i \\ \sum x_i z_i & \sum y_i z_i & \sum z_i z_i \end{bmatrix} \times \begin{bmatrix} \frac{1}{n-1} & 0 & 0 \\ 0 & \frac{1}{n-1} & 0 \\ 0 & 0 & \frac{1}{n-1} \end{bmatrix}$$

$C$ se nazývá kovarianční maticí. Platí např. následující tvrzení.
0 je vlastní hodnota matice $C$ právě když $[x_i, y_i, z_i], i = 1, .., n$ leží v rovině (tj. existuje jejich dvouprvková báze).

Dále, všechny vlastní hodnoty matice $C$ jsou reálné, neboť $C$ je symetrická a reálná. Kovarianční matice $C$ je navíc positivně semidefinitní, takže každá její vlastní hodnota $\lambda \geq 0$.

Poznámka: Vlastní hodnoty matice lze spočítat např. Jacobiho iterační metodou, viz např.:
https://en.wikipedia.org/wiki/Jacobi_eigenvalue_algorithm

Z pohledu zpracování mračna bodů má kovarianční matice 3D bodů a zejména její vlastní hodnoty a vektory jednu velmi cennou vlastnost. Její nenulové vlastní vektory jsou totiž „rozumnou“ ortogonální bází vstupních bodů, jinými slovy tyto body lze vyjádřit pomoci lineární kombinace této báze. Jsou-li navíc odpovídající vlastní hodnoty příslušné těmto vektorům relativně malé vůči ostatním, můžeme je zanedbat.



*Vlastní vektory kovarianční matice okolí bodu, velikosti podle poměru odpovídajících vlastních hodnot*

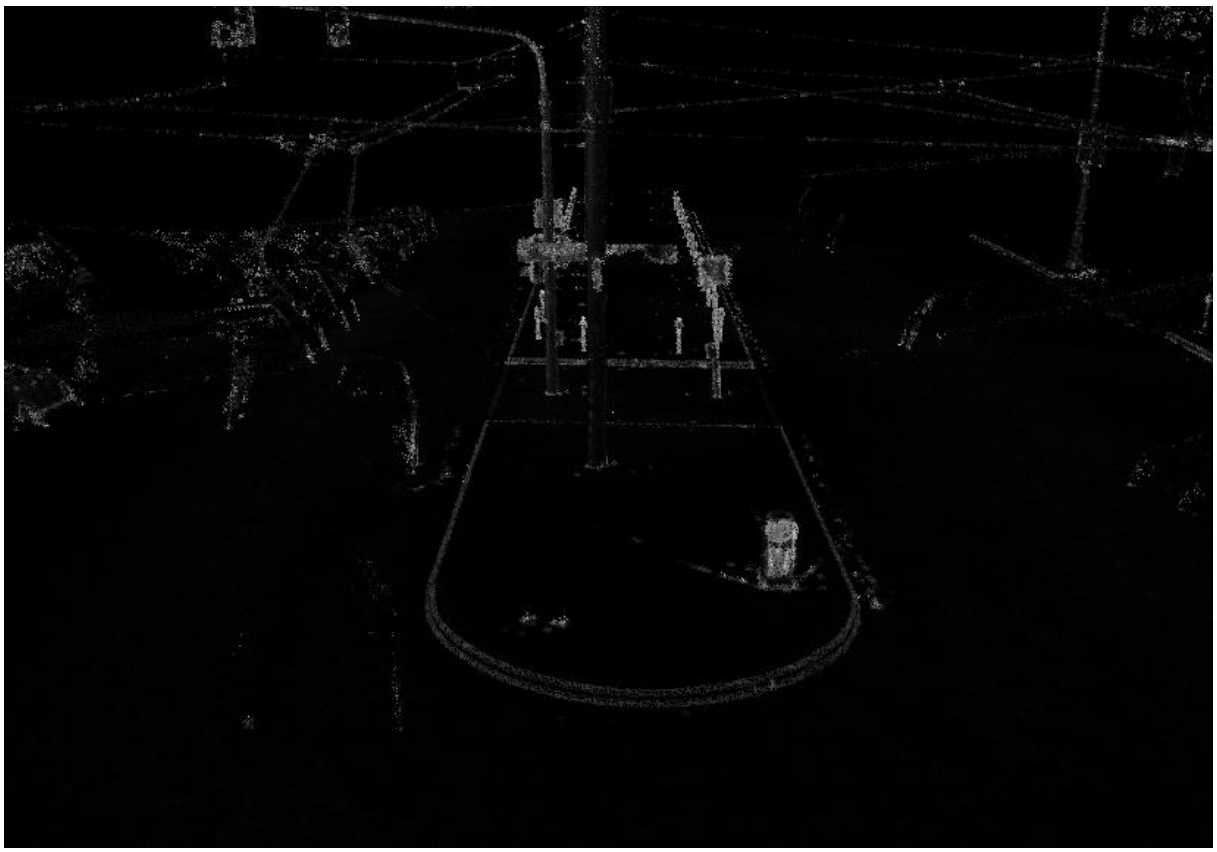
## 8.7. VARIANCE POVRCHU

Nechť $\lambda_0, \lambda_1, \lambda_2$ jsou vlastní hodnoty kovarianční matice $C$ z bodů z okolí bodu $p$, $\lambda_0 \leq \lambda_1 \leq \lambda_2$. Položme:

$$\sigma(p) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}$$

Operátor $\sigma(p)$ nazveme *povrchovou variancí* bodu $p$. $\sigma(p) = 0$ právě když body v okolí $p$ leží v (ideální) rovině. Čím je hodnota $\sigma(p)$ větší, tím je větší „nerovnost“ jeho okolí. Triviálně:

$$\sigma(p) \epsilon \left[0, \frac{1}{3}\right]$$

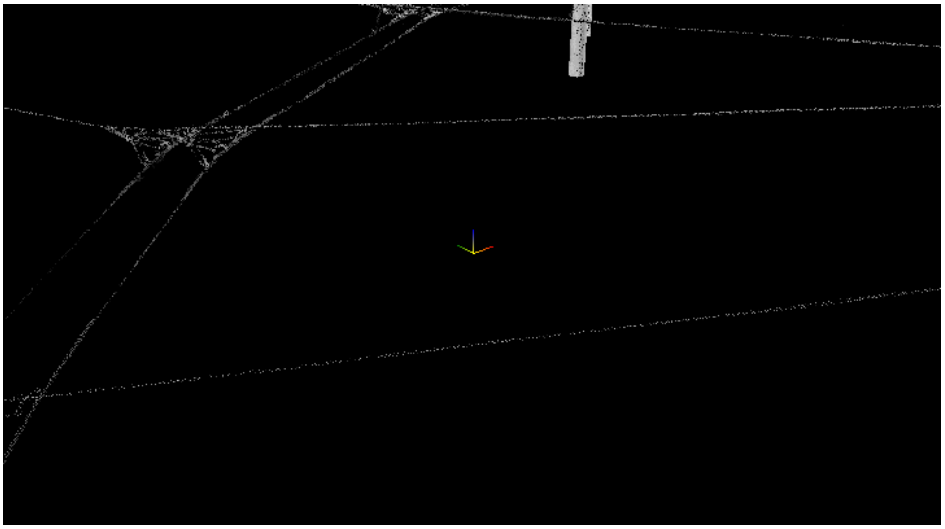


*Vizualizace variance povrchu – čím světlejší, tím větší míra nerovnosti*

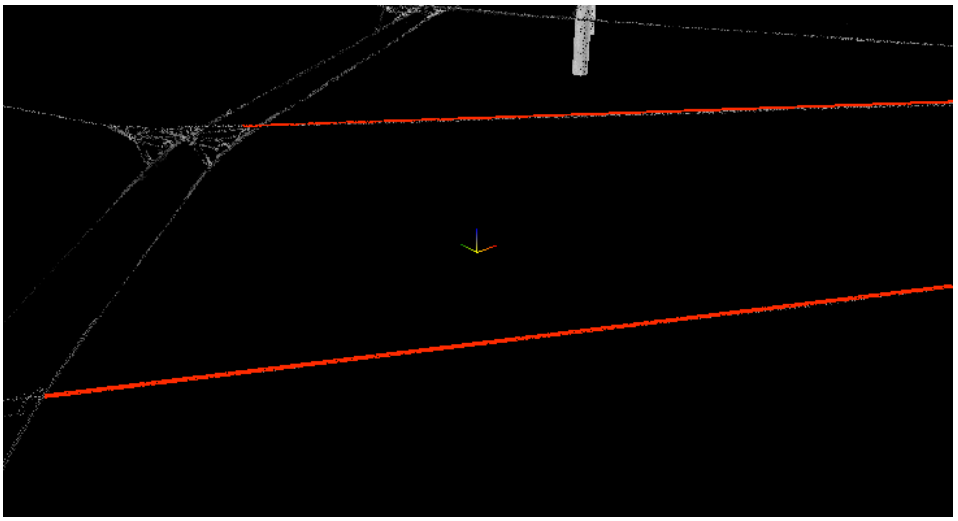
## 8.8. AUTOMATICKÉ TRASOVÁNÍ LINIÍ V MRAČNU BODŮ

Tyto metody se snaží z mračna bodů, ve kterém jsou klasifikovány jisté body jako kandidáti potenciální trasy linie, převést ‚mračnovou‘ linii do 3D lomené linie. Kandidáty lze získat metodami variance povrchu (viz předešlý odstavec), výškovým omezením v případě nadzemního vedení aj. Princip spočívá v tom, že metody

klasifikace kandidátů musí od sebe jednotlivé linie oddělit (nejjednodušším případem jsou nadzemní, např. trolejová vedení – ty jsou od sebe oddělena přirozeně). Opět využijeme vlastností kovarianční matice okolí zkoumaného bodu. Na rozdíl od metod variance povrchu, kde jsme hledali dva dominantní vektory tvořící bázi plochy, nyní hledáme jeden dostatečně dominantní vektor, který určuje lokální směr linie. Zkoumaný bod posunujeme podle takto zjištěných vektorů.



*Zdrojové mračno pro trasování linií*



*Poloautomaticky trasované linie*

Reference ke 3D:

[1] Pauly, M., Gross, M., Kobbelt, L. *Efficient Simplification of Point-Sampled Surfaces. Visualization*, VIS. IEEE, pp. 163-170, 2002.

[2] Pauly, M., Keiser, R., Gross, M. *Multi-scale feature extraction on pointsampled surfaces. Computer Graphics Forum*, 22(3), pp. 281-289, 2003.

[3] Dena Bazazian, Josep R. Casas, Javier Ruiz-Hidalgo. *Fast and Robust Edge Extraction in Unorganized Point Clouds*, https://imatge.upc.edu/web/sites/default/files/pub/cBazazian15.pdf

# 9. Přílohy

## 9.1. Metoda řešení nelineárních rovnic, Newton–Raphson

Snažíme se vyřešit rovnici $f(x) = 0$ kde $f(x)$ má první derivaci a my ji navíc dokážeme spočítat.

Uvažme Taylorův rozvoj funkce:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 \ldots$$

V bodě $a$ můžeme nahradit funkci $f$ její tečnou

$$f(x) \approx f(a) + \frac{f'(a)}{1!}(x-a)$$

Rovnici:

$$f(a) + \frac{f'(a)}{1!}(x-a) = 0$$

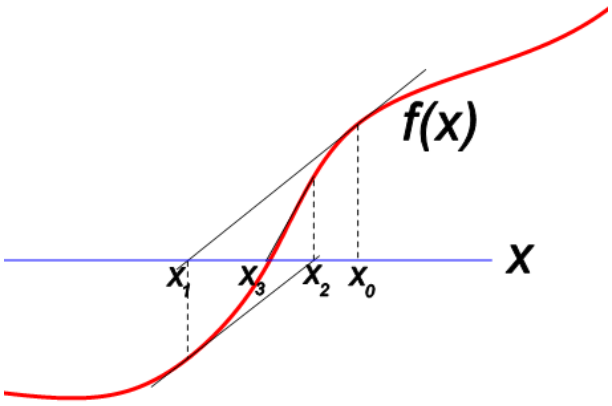
$$x = a - \frac{f(a)}{f'(a)}$$

už řešit umíme.

Postup:

Iterace lineární aproximace rovnice. Odhadneme první hodnotu $a = x_0$ a řešíme lineární rovnici, výsledek bude $x_1$. Položíme $a = x_1$ a postup opakujeme tak dlouho dokud rozdíl $|x_i - x_{i-1}|$ nebude dostatečně malý.

*Proces NEMUSÍ konvergovat! U některých funkcí hodně záleží na počátečním odhadu!*




Iterace Newton–Raphson metody funkce jedné proměnné

Příklad 1:

$f: y = 3x + 8$

$f': y = 3$

Položme $x_0 = a = 10$ a dosazením do

$$x_1 = a - \frac{f(a)}{f'(a)} = 10 - \frac{38}{3} = -\frac{3}{8}$$

V případě lineární funkce je tečna totožná z funkcí, stačí jedna iterace.

Příklad 2:

$f: y = \frac{x^2}{4} + \sqrt{x} - x - 2$

$f': y = \frac{1}{2}x + \frac{1}{2\sqrt{x}} - 1$

S počáteční hodnotou $x_0 = 12345$ potřebujeme cca 18 iterací:

| a | a-f(a)/f´(a) | f(a) | f´(a) |
|---|---|---|---|
| **12345** | 6173.486983 | 38087520.36 | 6171.5045 |
| 6173.486983 | 3087.725365 | 9521888.466 | 3085.749855 |
| 3087.725365 | 1544.837609 | 2380477.824 | 1542.87168 |
| 1544.837609 | 773.3844645 | 595123.2765 | 771.4315258 |
| 773.3844645 | 387.6458883 | 148783.3078 | 385.7102115 |
| 387.6458883 | 194.761798 | 37197.37652 | 192.8483394 |
| 194.761798 | 98.3030846 | 9300.2334 | 96.41672663 |
| 98.3030846 | 50.05846472 | 2325.485817 | 48.20197201 |
| 50.05846472 | 25.93059835 | 581.4792084 | 24.09990173 |
| 25.93059835 | 13.88925597 | 145.2605939 | 12.06348838 |
| 13.88925597 | 7.956261183 | 36.06543106 | 6.078790288 |
| 7.956261183 | 5.202262665 | 8.689946344 | 3.155392528 |
| 5.202262665 | 4.18901228 | 1.844468463 | 1.820348149 |
| 4.18901228 | 4.006273118 | 0.244651359 | 1.33880092 |
| 4.006273118 | 4.000007362 | 0.00785062 | 1.252940754 |
| 4.000007362 | 4 | 9.20245E-06 | 1.250003451 |
| 4 | 4 | 1.27027E-11 | 1.25 |
| **4** | **4** | 0 | 1.25 |

V případě funkcí více proměnných a systému nelineárních rovnic

$$f_1(x, y, z, ..) = 0$$
$$f_2(x, y, z, ..) = 0$$
$$.$$

je postup stejný. Funkce linearizujeme

$$f_i(x, y, z, ..) \approx f_i(x_0, y_0, z_0, ..) +$$

$$\frac{\partial f_i}{\partial x}(x_0, y_0, z_0, ..)(x - x_0) +$$

$$\frac{\partial f_i}{\partial y}(x_0, y_0, z_0, ..)(y - y_0) +$$

$$\frac{\partial f_i}{\partial z}(x_0, y_0, z_0, ..)(z - z_0) \ldots$$

V každém kroku iterace potom řešíme systém lineárních rovnic. Označíme-li $\boldsymbol{X_i} = [x_i, y_i, z_i, \ldots]$ a $\Delta\boldsymbol{X} = [x - x_i, y - y_i, z - z_i, \ldots]$, potom v každém kroku řešíme soustavu:

$$\begin{bmatrix} \frac{\partial f_1}{\partial x}(\boldsymbol{X_i}) & \frac{\partial f_1}{\partial y}(\boldsymbol{X_i}) & \frac{\partial f_1}{\partial z}(\boldsymbol{X_i}) & \ldots \\ \frac{\partial f_2}{\partial x}(\boldsymbol{X_i}) & \frac{\partial f_2}{\partial y}(\boldsymbol{X_i}) & \frac{\partial f_2}{\partial z}(\boldsymbol{X_i}) & \ldots \\ \frac{\partial f_3}{\partial x}(\boldsymbol{X_i}) & \frac{\partial f_3}{\partial y}(\boldsymbol{X_i}) & \frac{\partial f_3}{\partial z}(\boldsymbol{X_i}) & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \times \Delta\boldsymbol{X} = \begin{bmatrix} -f_1(\boldsymbol{X}_i) \\ -f_2(\boldsymbol{X}_i) \\ -f_3(\boldsymbol{X}_i) \\ \vdots \end{bmatrix}$$

Je-li řešením této soustavy $\Delta\boldsymbol{x}_i$, položíme $\boldsymbol{X}_{i+1} = \boldsymbol{X}_i + \Delta\boldsymbol{x}_i$ a pokračujeme, dokud $|\Delta\boldsymbol{x}_i| \approx 0$.

## 9.2. METODA NEJMENŠÍCH ČTVERCŮ – LINEÁRNÍ REGRESE

Hledáme „co nejlepší“ řešení soustavy lineárních rovnic ($M > N$):

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \approx \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{bmatrix}$$

$$\boldsymbol{A}x \approx b \qquad (1)$$

To provedeme tak, že budeme hledat takové řešení, pro které je součet čtverců odchylek jednotlivých rovnic co nejmenší (Carl Friedrich Gauss 1795), tedy:

$$\Delta x = (\boldsymbol{A}x - b)^T(\boldsymbol{A}x - b) = min \quad (2)$$

kde operátor $\boldsymbol{A}^T$ označuje transpozici matice $\boldsymbol{A}$ – vzájemnou výměnu řádků a sloupců. Připomeňme jeho základní vlastnosti:

$$(\boldsymbol{A}^T)^T = \boldsymbol{A}, \;\; (\boldsymbol{A} + \boldsymbol{B})^T = \boldsymbol{A}^T + \boldsymbol{B}^T, \;\; (\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$$

roznásobením (2) dostaneme:

$$\Delta x = (\boldsymbol{A}x)^T\boldsymbol{A}x - (\boldsymbol{A}x)^T b - b^T\boldsymbol{A}x + b^T b =$$

$$x^T\boldsymbol{A}^T\boldsymbol{A}x - x^T\boldsymbol{A}^T b - b^T\boldsymbol{A}x + b^T b$$

matice $b^T\boldsymbol{A}x$ má rozměr $1 \times 1$, tedy $b^T\boldsymbol{A}x = (b^T\boldsymbol{A}x)^T = (\boldsymbol{A}x)^T b = x^T\boldsymbol{A}^T b$ a:

$$\Delta x = x^T\boldsymbol{A}^T\boldsymbol{A}x - 2x^T\boldsymbol{A}^T b + b^T b$$

Hledáme minimum $\Delta x$, tedy derivujeme $\Delta x$ podle $x = [x_1, \dots, x_N]$ a výraz položíme roven $0$. Položme tedy:

$$\frac{\partial \Delta x}{\partial x} = \begin{bmatrix} \frac{\partial \Delta x}{\partial x_1} \\ \vdots \\ \frac{\partial \Delta x}{x_N} \end{bmatrix} = 2\boldsymbol{A}^T\boldsymbol{A}x - 2\boldsymbol{A}^T b = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

(derivace $\frac{\partial \Delta x}{\partial x}$ není na první pohled úplně zřejmá, dá se však po jistém úsilí odvodit).

Dostáváme soustavu tzv. normálních rovnic, jejímž řešením je hledaná hodnota $x$.

$$\boldsymbol{A}^T\boldsymbol{A}x = \boldsymbol{A}^T b$$

Vraťme se zpátky k zadání (1):

$$\boldsymbol{A}x \approx b$$

„Nejlepší“ řešení soustavy rovnic $\boldsymbol{A}x \approx b$ dostaneme, když její každou stranu zleva násobíme transponovanou maticí soustavy $\boldsymbol{A}^T$.

Příklad:

Body $[1,0], [2,3], [3,2], [4,5]$ proložˇ přímku $y = ax + b$.

Maticový zápis této úlohy je:

$$\begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 2 \\ 5 \end{bmatrix}$$

$$\boldsymbol{A}^T\boldsymbol{A} = \begin{bmatrix} 30 & 10 \\ 10 & 4 \end{bmatrix}, \qquad \boldsymbol{A}^T b = \begin{bmatrix} 32 \\ 10 \end{bmatrix}$$

Řešením:

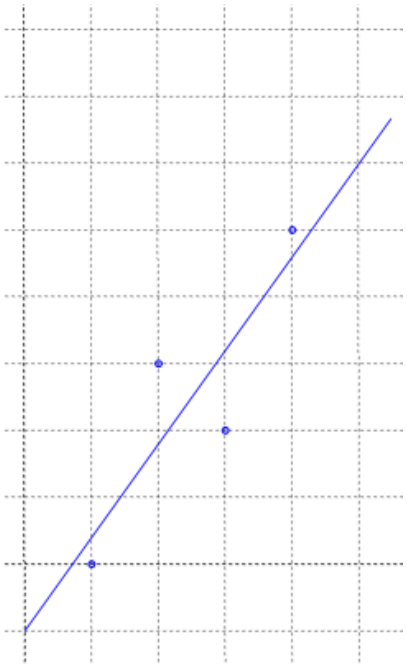
$$\begin{bmatrix} 30 & 10 \\ 10 & 4 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 32 \\ 10 \end{bmatrix}$$

dostaneme výsledek

$$a = \frac{7}{5}; \;\; b = -1$$

a tedy

$$y = 1.4x - 1$$



Příklad:

Máme dvě „odpovídající si“ sady bodů v rovině $[x_i, y_i]$ a $[u_i, v_i]$.

$$[u_i, v_i] \rightarrow [x_i, y_i]$$

$$[1,1] \rightarrow [0,5]$$
$$[3,1] \rightarrow [4,11]$$
$$[1,3] \rightarrow [0,11]$$
$$[5,4] \rightarrow [20,50]$$
$$[3,5] \rightarrow [12,39]$$

Vypočtěte koeficienty $a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2$ pro bilineární transformaci:

$$x = a_1 u + b_1 v + c_1 uv + d_1$$
$$y = a_2 u + b_2 v + c_2 uv + d_2$$

Maticový zápis úlohy je:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 3 & 1 & 3 & 1 \\ 1 & 3 & 3 & 1 \\ 5 & 4 & 20 & 1 \\ 3 & 5 & 15 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ 0 \\ 20 \\ 12 \end{bmatrix} \text{ a } \begin{bmatrix} 1 & 1 & 1 & 1 \\ 3 & 1 & 3 & 1 \\ 1 & 3 & 3 & 1 \\ 5 & 4 & 20 & 1 \\ 3 & 5 & 15 & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \\ c_2 \\ d_2 \end{bmatrix} = \begin{bmatrix} 15 \\ 11 \\ 11 \\ 50 \\ 39 \end{bmatrix}$$

Aplikací metody nejmenších čtverců $\boldsymbol{A}^T \boldsymbol{A} x = \boldsymbol{A}^T b$ dostáváme:

$$\begin{bmatrix} 45 & 42 & 158 & 13 \\ 42 & 52 & 168 & 14 \\ 158 & 168 & 644 & 42 \\ 13 & 14 & 42 & 5 \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \end{bmatrix} = \begin{bmatrix} 148 \\ 144 \\ 592 \\ 36 \end{bmatrix} \text{ a } \begin{bmatrix} 45 & 42 & 158 & 13 \\ 42 & 52 & 168 & 14 \\ 158 & 168 & 644 & 42 \\ 13 & 14 & 42 & 5 \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \\ c_2 \\ d_2 \end{bmatrix} = \begin{bmatrix} 416 \\ 444 \\ 1656 \\ 116 \end{bmatrix}$$

Řešením těchto dvou systémů lineárních rovnic dostaneme:

$$a_1 = 1, b_1 = -1, c_1 = 1, d_1 = -1, a_1 = 1, b_2 = 1, c_2 = 2, d_1 = 1$$

a

$$x = u - v + uv - 1$$
$$y = u + v + 2uv + 1$$