Fakulta Informatiky, Masarykova Universita v Brně

Fólie k přednášce

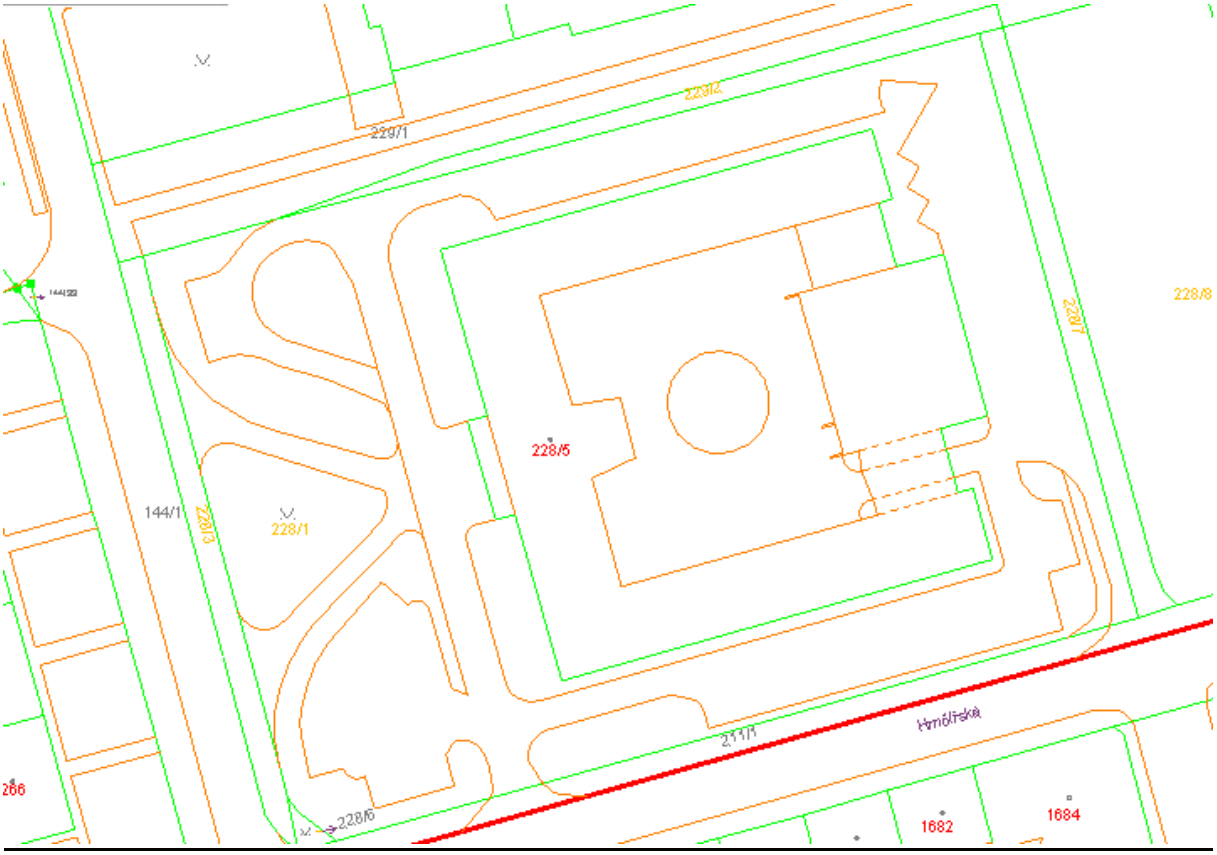
PV019 – Úvod do geoinformačních systémů

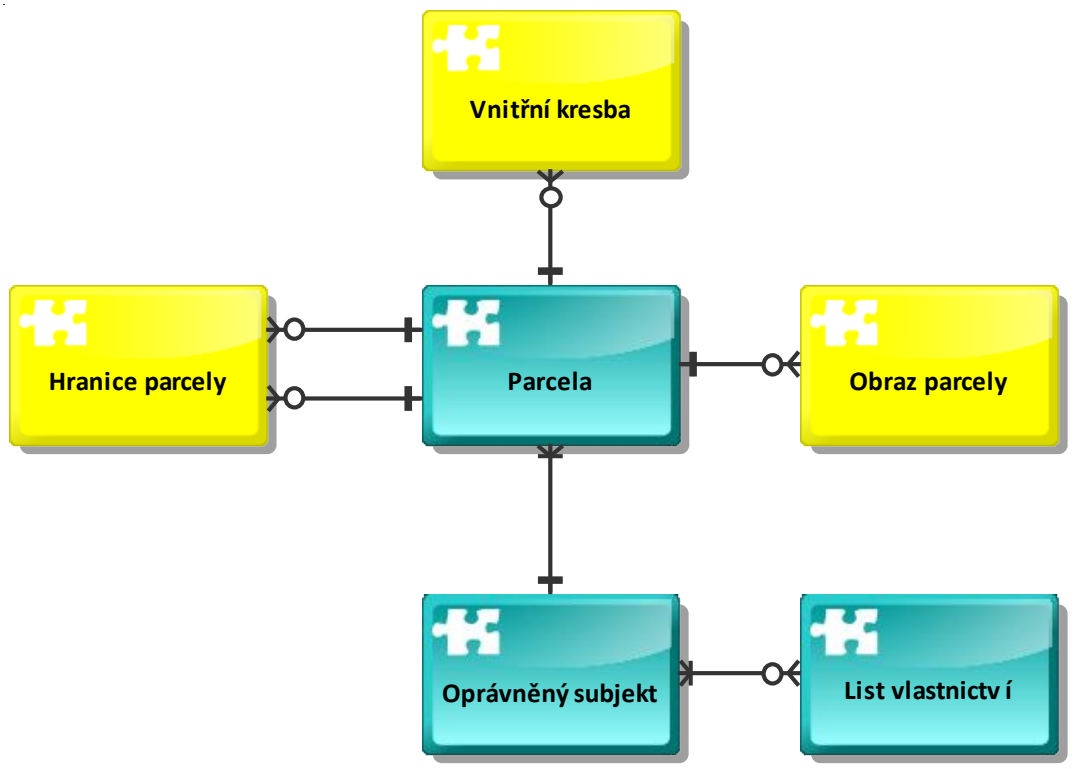
1. Geoinformační systém, místo na povrchu Země.

1.1. VYMEZENÍ POJMU GIS

Příklad 1 - Informační systém o nemovitostech

Katastrální úřady evidují vlastnické vztahy fyzických a právnických osob k nemovitostem (budovám a pozemkům). Poloha nemovitostí je zaznamenána v katastrálních mapách.





Uvažujme „klasický“ 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 parcely vlastní osoba ...?; jakou cenu mají parcely, které vlastní osoba …?

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

- kde se nalézá parcela ...?
- jaké typy parcel se nalézají v daném regionu ...?
- Jakou výměru parcel vlastní daný oprávněný subjekt

Vymezení pojmu GIS:

GIS je jakýkoliv manuálně nebo počítačově založený 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 - speciální případ GIS v podrobnosti velkého měřítka, který zahrnuje vlastnické vztahy (hranice parcel a informace o vlastnících parcel).

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

- Grafický informační systém - systém pracující s obrazovými daty, která nemá smysl lokalizovat v nějakém (jednotném) prostoru.

- V poslední době (10 – 15 let) toto dělení ztrácí smysl, po geoinformačních systémech je požadována komplexní funkcionalita.

1.2. MÍSTO NA ZEMSKÉM POVRCHU

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

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

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

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

Souřadnice [φ, λ] 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 změření zemského obvodu s popisem metody. Úhlová metoda, změření maximální výšky slunce na dvou místech na stejném poledníku ve stejný čas. Byla vybrána města Alexandrie a Syéné (dnešní Asuán).

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

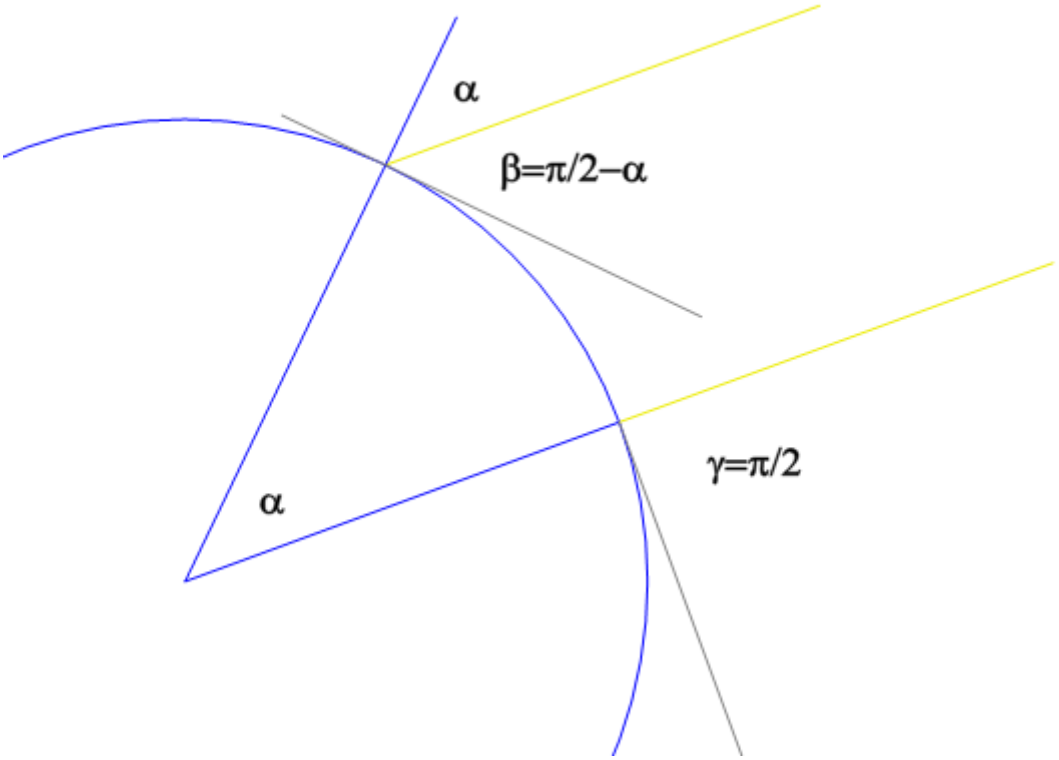
=

Poměr 2π a obvodu Země

Vzdálenost Alexandrie a Syéné byla změřena na 5000 stádií (záznamy obchodníků, pochodující legie), 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

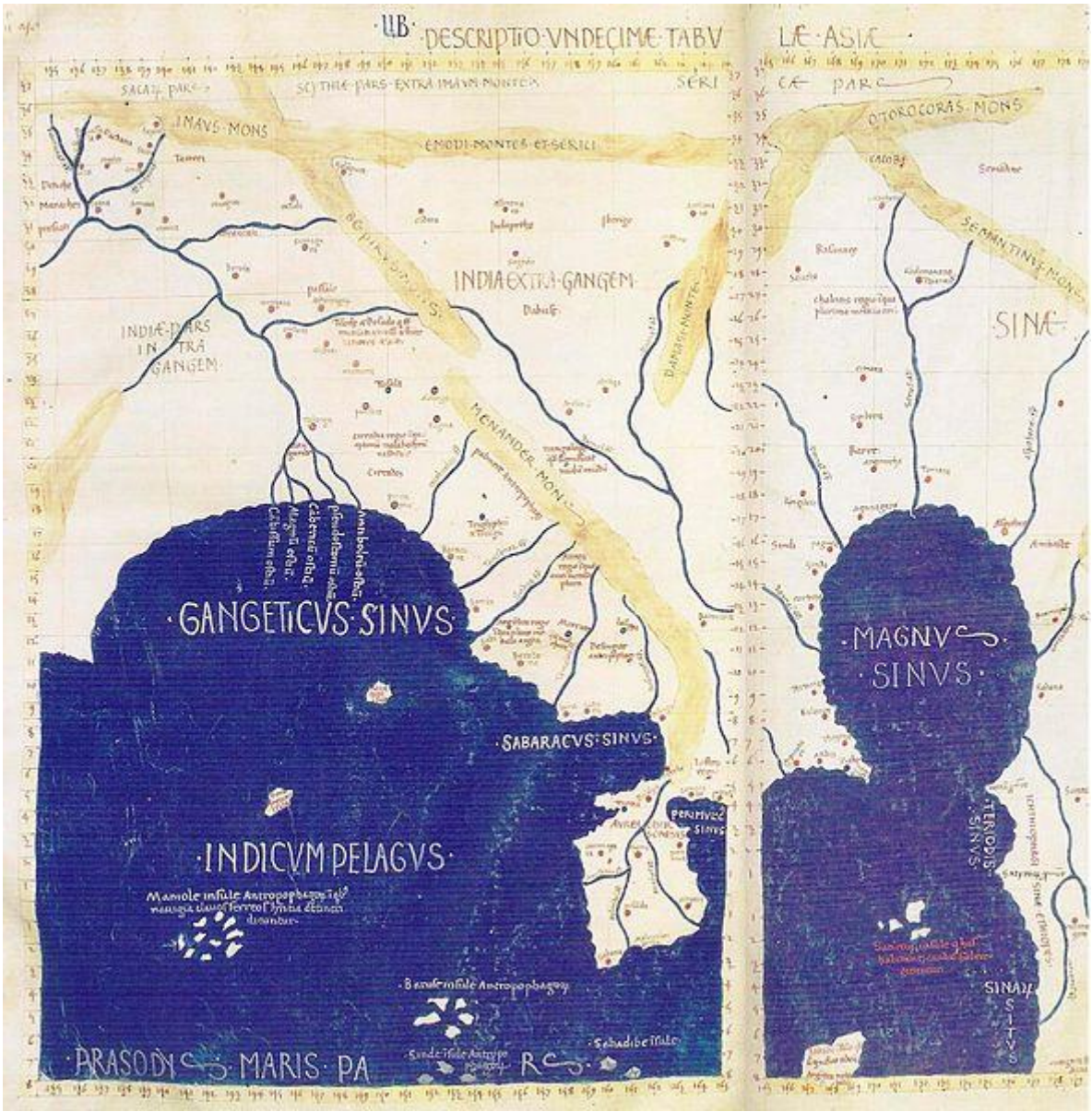
β - maximální výška slunce za slunovratu v Alexandrii
γ - maximální výška slunce za slunovratu v Syéné (= π/2)
α - středový úhel mezi Syéné a Alexandrií

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

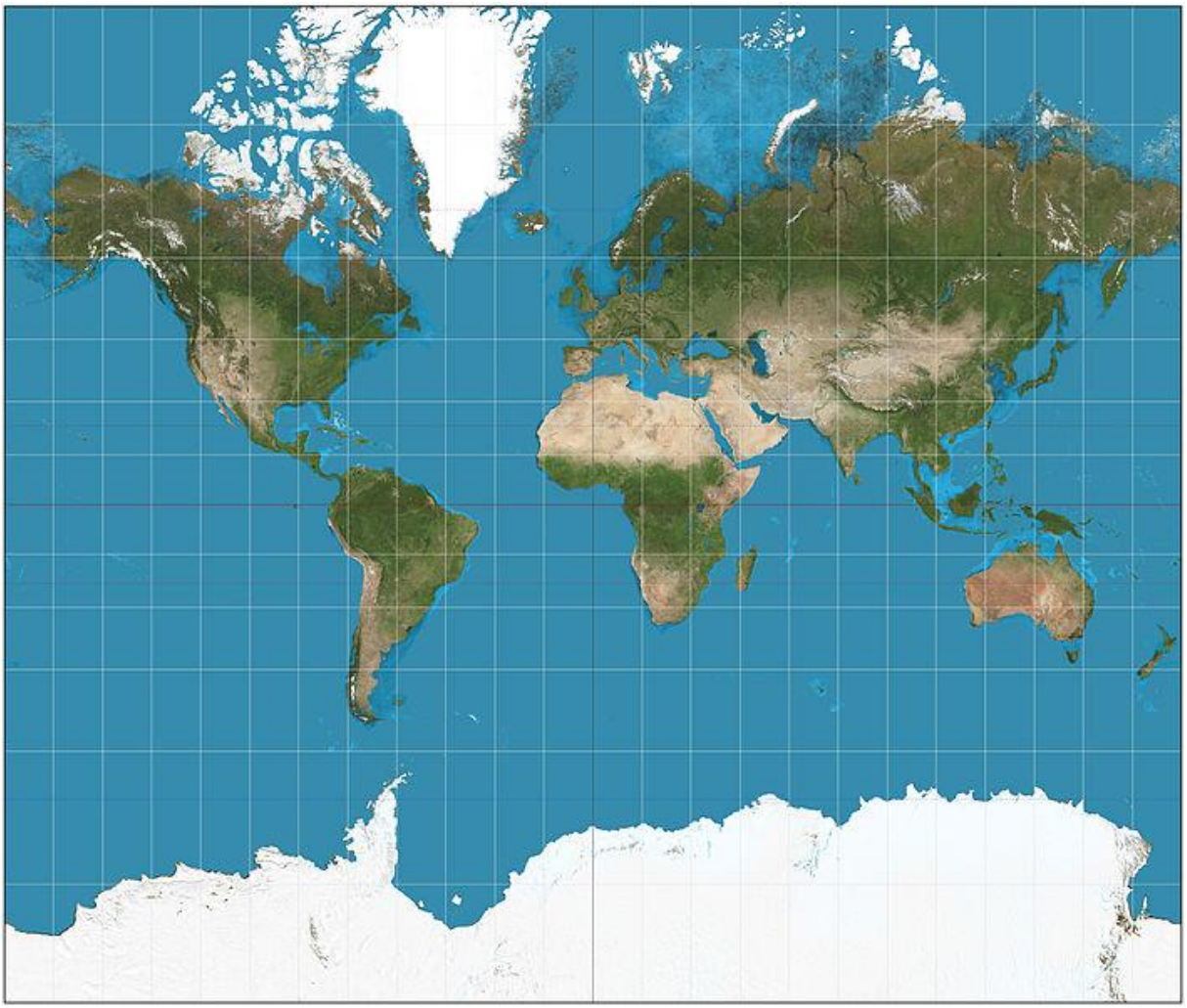
Jsou známy různé rozměry antického stadionu, od 157 do 185 m a není jasné se kterým údajem Eratosthenes pracoval. I přes cca 10% chybu se jedná o jeden z nejvýznamnějších výsledků antické vědy.

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 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 „rovinným zobrazením známé Země, se vším, co se na ní nachází.



Gerhard Mercator (1512-1594) – Vlámský matematik a kartograf. Je autorem tzv. Mercatorova zobrazení, na kterém jsou poledníky a rovnoběžky k sobě kolmé. Zobrazení je velmi užitečné pro navigaci z místa A do místa B. Loxodromy jsou zobrazeny jako přímky, stačí tedy stanovit azimut cíle. Je 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)

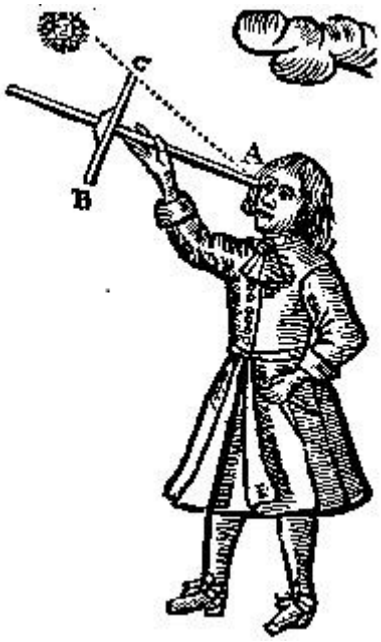
1.4. VÝVOJ NAVIGAČNÍCH POMŮCEK

Určování šířky

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.

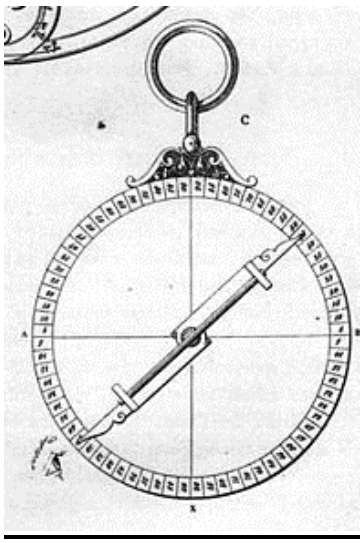
Kamal (obrázek z http://www.lode.cz/re.php)

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.



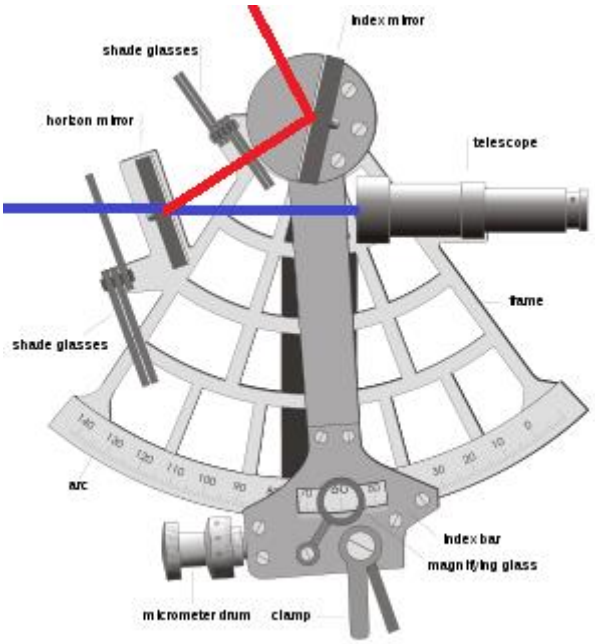
Jakubova hůl, John Sellers - Practical Navigation (1672)

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.



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

Určování délky

Je problém výrazně komplikovanější, tradiční metody vychází z měření místního času, resp. místního poledne.

První 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 uspokojivě vyřešen.

Přesné hodiny (chyba v minutách za týdny provozu) byly sestrojeny Johnem Harissonem (1693 - 1776).

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

K určení pozice na Zemi tedy stačí:

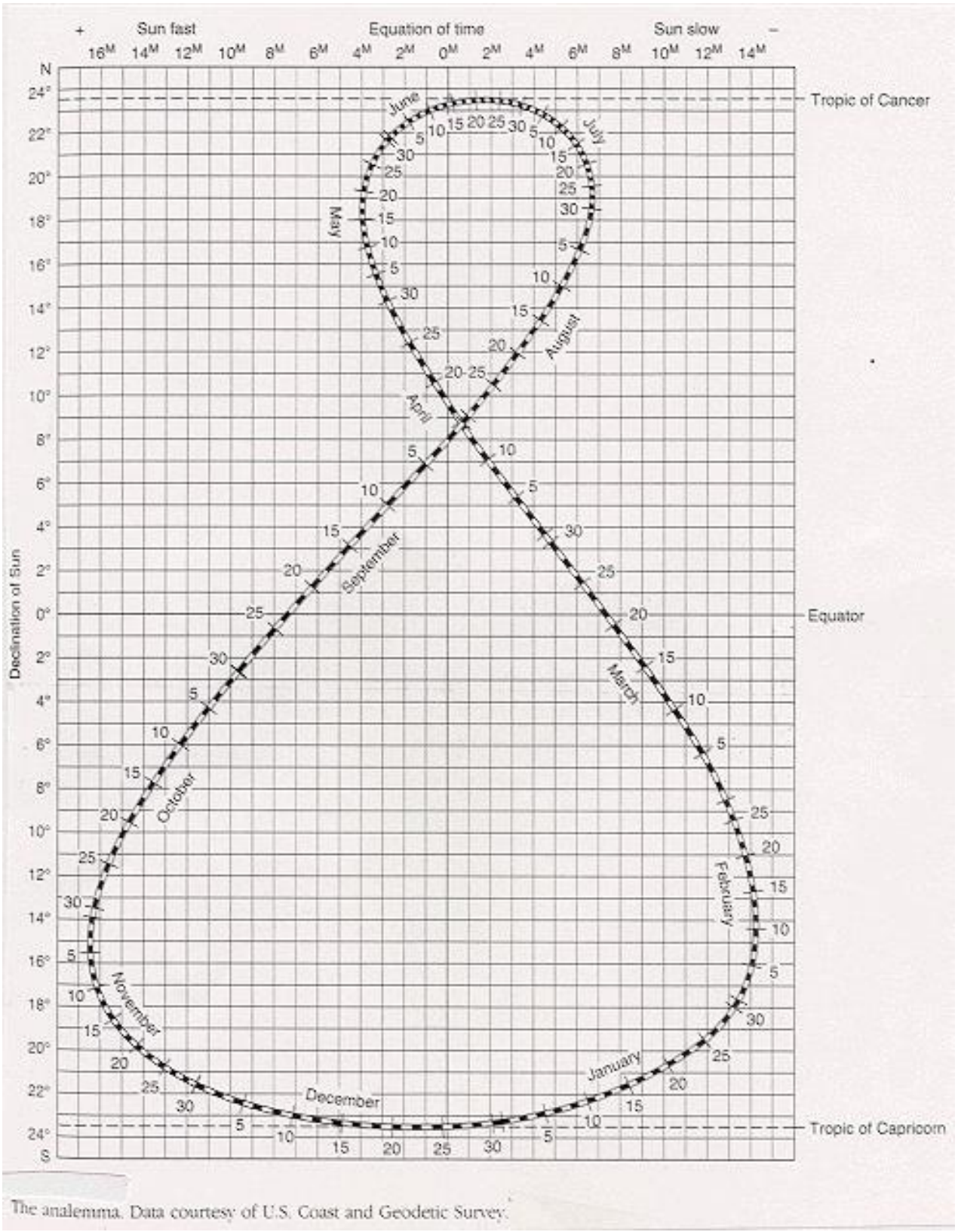
1) Hodiny ukazující přesný čas nultého poledníku.

2) Úhloměrné zařízení (sextant), pomocí, kterého změříme největší úhel slunce proti horizontu, zároveň si zaznamenáme čas, kdy maximum nastalo.

3) Znalost data, kdy měříme.

4) Korekční tabulky pro opravu času a úhlu vzhledem k datu.

Bod 4) je nutný, neboť vlivem sklonu zemské osy vůči rovině ekliptiky a 2. Keplerova zákona, má slunce v poledne různou výšku a místní poledne se různí od středního poledne až o cca. 16 minut (střední poledne je čas měřený přesnými hodinami). Korekční tabulku nahrazuje křivka ***analema***, kde na jedné ose je zobrazena diference výšky a na druhé časová diference místního poledne proti střednímu. Na křivce je potom vyznačený průběh roku.



The analemma. Data courtesy of U.S. Coast and Geodetic Survey.

Analema – U.S. Coast and Geodetic Survey

V současné době je nejmasovějším prostředkem pro určení polohy systém **GPS** (Global Positioning System). Je založen na 32 satelitech schopných vysílat svoji přesnou polohu a přesný čas. Čas je pro všechny satelity velmi přesně synchronizován. Z dat 4 satelitů jsme schopni dopočítat přesnou polohu (s přesností na 10-20 metrů).

Pomocí stacionárních GPS stanic a korekcí dosahuje přesnost měření až 0.1 m.

Přesto, navigátoři plavidel, letadel musí být schopni určit svou polohu (pro případ kolapsu elektroniky) čistě pomocí mechanických pomůcek.

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.

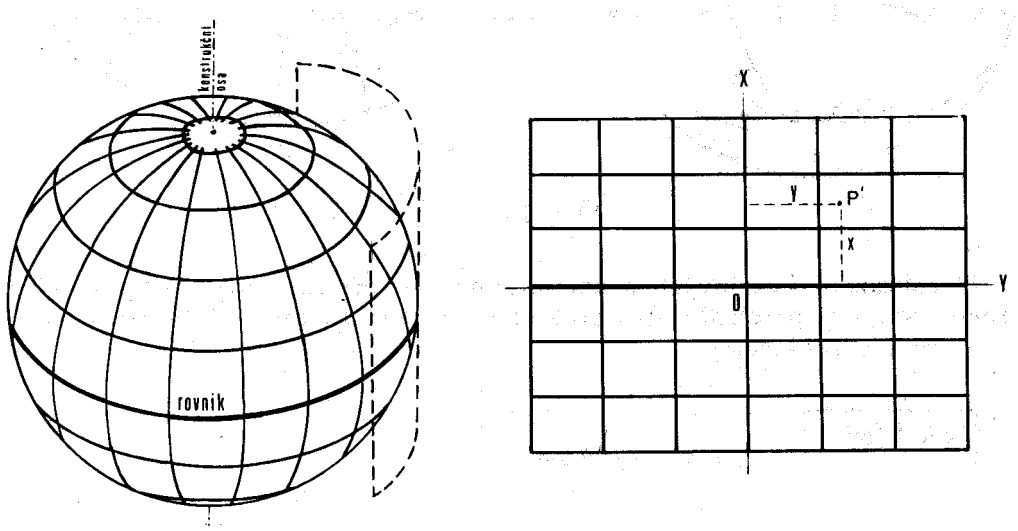
Ekvivalentní zobrazení - nezkresluje plochy, zkreslení úhlů je však zde poměrně značné, což se projevuje zejména ve tvarech ploch.

Geografické souřadnice – určení polohy bodu na ploše elipsoidu pomocí zeměpisné šířky *φ* a zeměpisné délky *λ*.

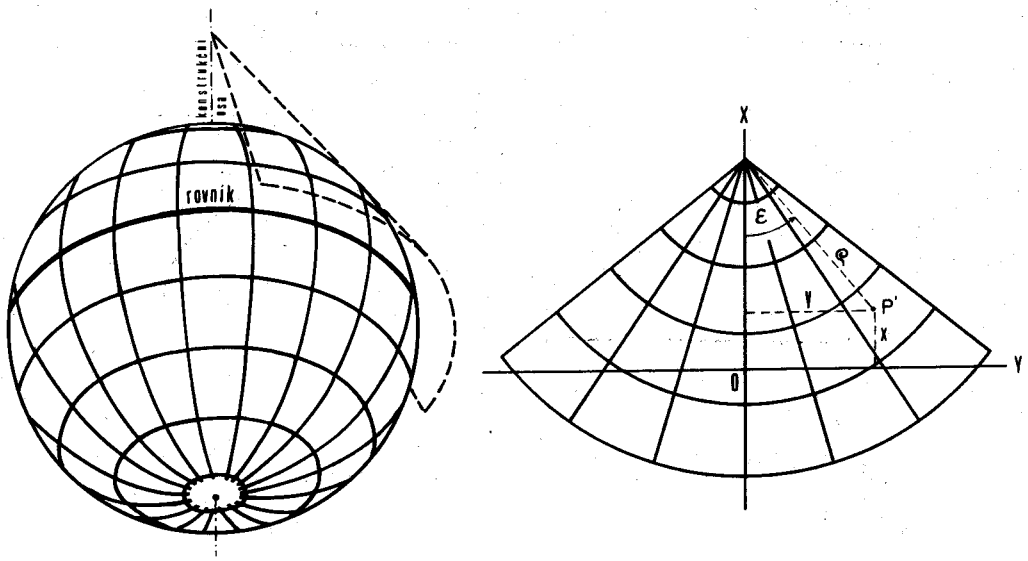
Geocentrické souřadnice *X,Y,Z* - prostorový souřadný systém s počátkem ve středu elipsoidu, osa *X* je vložena do průsečíku rovníku a roviny základního (nultého) poledníku, osa *Z* spojuje střed elipsoidu a severní pól a osa *Y* leží v rovině rovníku otočena o *90°* proti směru hodinových ručiček od 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) tj. rovinné souřadnice narůstající k východu resp. k severu.

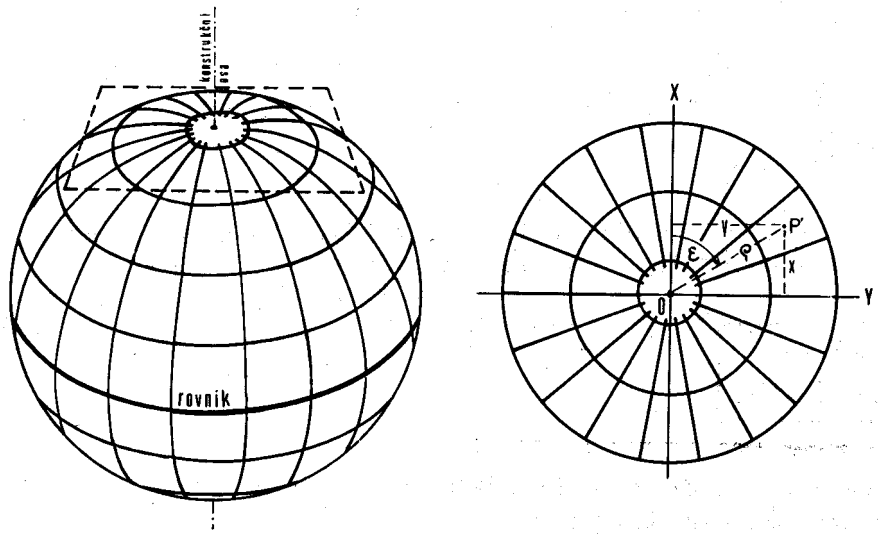
Základní typy převodu geografických do rovinných souřadnic:



Válcové zobrazení



Kuželové zobrazení



Azimutální zobrazení

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

- Geodetické datum (elipsoid, referenční bod, datum určení)
- Souřadný systém geografických souřadnic ***φ, λ***, (včetně volby základního poledníku)
- Zobrazovací rovnice do rovinných souřadnic
- Souřadný systém rovinných souřadnic ***X,Y***

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 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. Geofrafické 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 interpretovat jako:

- vektor posunu – [**Δx**, **Δy**, **Δz**]
- úhel otočení pro jednotlivé osy - [α, β, γ]
- 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 → EL2

použijeme postup:

EL1 → WGS84 → 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:

$$x = f(u,v) = a_1u + b_1v + c_1$$
$$y = g(u,v) = a_2u + b_2v + c_2$$

Bilineární:

$$f(u,v) = a_1u + b_1v + c_1uv + d_1$$
$$g(u,v) = a_2u + b_2v + c_2uv + d_2$$

Kvadratická, kubická, obecná polynomiální …

Nalezení koeficientů polynomiální transformace:

Vstup: Dva seznamy „odpovídajících“ si bodů:

$$\{[x_1,y_1]..\ [x_n,y_n]\}$$

$$\{[u_1,v_1]..\ [u_n,v_n]\}$$

Výstup: Seznam koeficientů polynomiální transformace zvoleného stupně.

Transformujeme $[u,v]$ do $[x,y]$ s co nejmenší chybou, tedy:

$$\Sigma dist^2([x_i,y_i],[f(u_i,v_i),g(u_i,v_i)]) = min$$

kde

$$dist^2([x_1,y_1],[x_2,y_2])=(x_1-x_2)^2+(y_1-y_2)^2$$

Lineární regrese (příklad pro lineární transformaci):

(Σ označuje sumu pro všechny dvojice odpovídajících si bodů)

$$\Sigma = \Sigma (a_1u_i+b_1v_i+c_1-x_i)^2+(a_2u_i+b_2v_i+c_2-y_i)^2 = \min$$

Výraz Σ derivujeme podle proměnných $a_1..c_2$ a derivaci položíme rovnu 0 (hledání extrémů funkcí více proměnných).

$$d\Sigma/da_1 = \Sigma\, 2(a_1u_i + b_1v_i + c_1 - x_i).u_i = 0$$

$$d\Sigma/db_1 = \Sigma\, 2(a_1u_i + b_1v_i + c_1 - x_i).v_i = 0$$

$$d\Sigma/dc_1 = \Sigma\, 2(a_1u_i + b_1v_i + c_1 - x_i) = 0$$

$$\vdots$$

Dostáváme soustavu tzv. normálních rovnic (pro $g(u,v)$ analogicky):

$$a_1 \Sigma u_i^2 + b_1 \Sigma u_iv_i + c_1 \Sigma u_i = \Sigma x_iu_i$$

$$a_1 \Sigma u_iv_i + b_1 \Sigma v_i^2 + c_1 \Sigma v_i = \Sigma x_iv_i$$

$$a_1 \Sigma u_i + b_1 \Sigma v_i + c_1 .n = \Sigma x_i$$

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/

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 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 družicích

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

- *Tématická 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, symbologie 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, postupně ji vytvářejí katastrální 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).

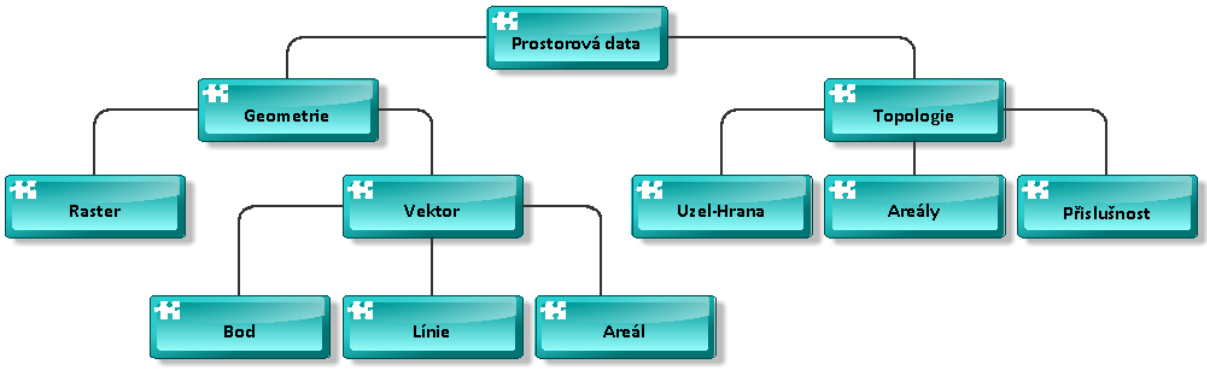
Mapy středních měřítek 1 : 10000 až 1 : 200 000

- 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 GŠ ČSA, 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 tématické 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 tématická část informace je získávána na základě předem definované sítě, kterou je rozděleno zájmové území.

Topologie - vymezuje vztahy mezi entitami (objekty) systému, aniž by obsahovala 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. 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

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

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

Příklad **TFW** souboru:

```
12.701100762066045723962743437653
0
0
-12.701100762066045723962743437765
-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.

3.3. VEKTOROVÉ DATOVÉ SKLADY

Pro fyzickou reprezentaci je možné použít vlastních datových struktur a ukládat je přímo ve file 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, MySQL).

Není 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. Norma neobsahuje symbologii (barva, síla, styl linií, výplň polygonu). Tu obsluhuje aplikace na základě popisných informací. Norma neobsahuje „heterogenní“ kolekce geometrií a tím i 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 *.dfb 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 línie se mohou skládat z různých segmentů – např. lomených čar a oblouků.
- Geometrie obsahuje symbologii geometrických primitiv.
- Typ **CELL** může opět obsahovat buňku.

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) a složitějších zobrazovacích symbolik (např, linie vzorovaná symbolem). Dalším problémem jsou operace na složitějšími geometriemi (např. průnik polygonů, jejichž hranice se skládají z lomených čar, 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 „default“ symbologii kresby.

ORACLE 7x (Spatial Data Option):

Geometrie je reprezentována čtyřmi tabulkami:

_SDOLAYER - obsahuje služební 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	více linií
MUTIPOLYGON	více areálů

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

Open GIS Consortium, Inc.

Je sdružení soukromých, veřejných organizací (universit, 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 MySQL je přímou implementací OGC), i když původní proklamace zní s odstupem času až úsměvně.

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:

```
// 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 mapu, neobsahují:

- Symbologie (grafická reprezentace – barva, síla, tloušťka) geometrických elementů
- reprezentace 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 mapových symbolů.

GML – Geographic Markup Language:

Původně norma WKB v XML formátu, od verze 3.2 značně rozšířená (křivky…). V rámci EU (projekt INSPIRE) převzatá jako norma pro výměnu vektorových prostorových dat.

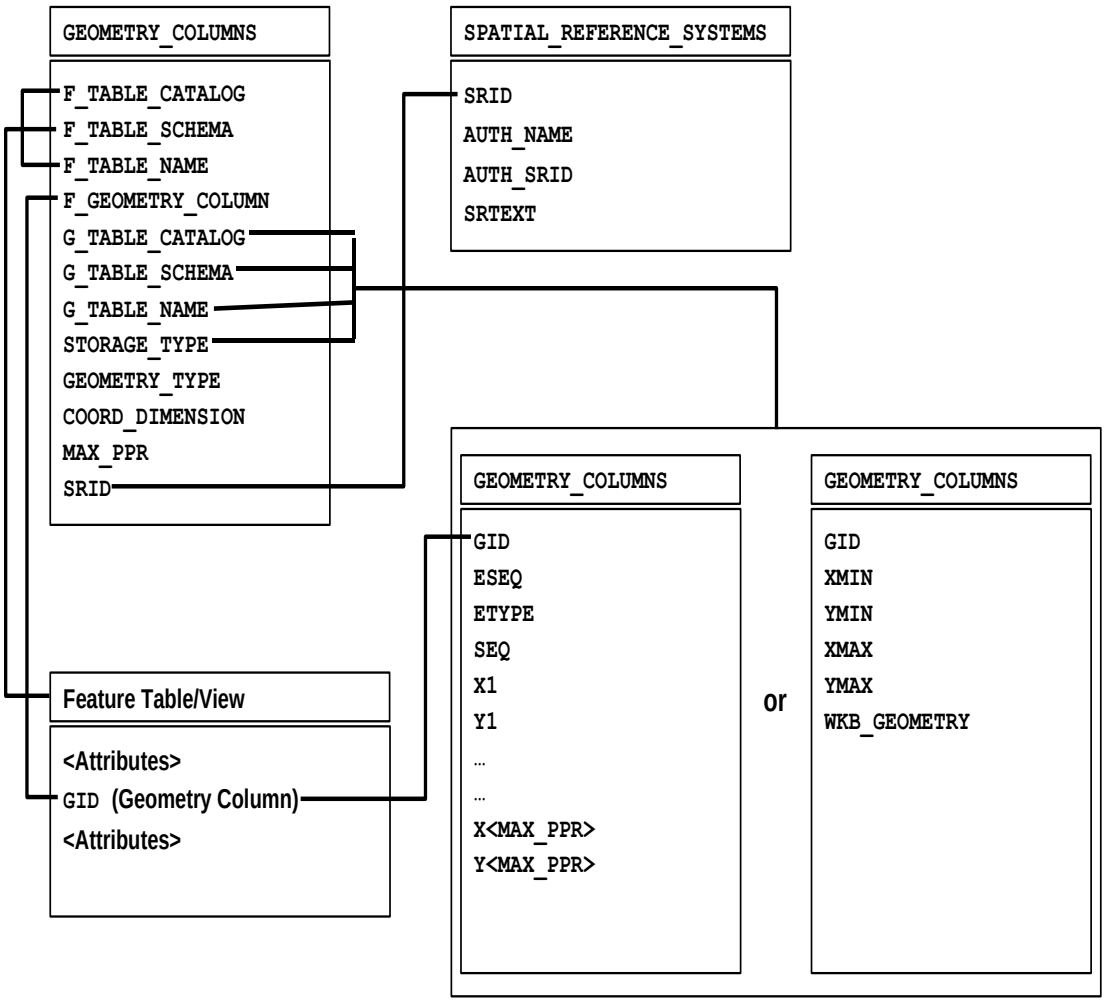
Formát GML definuje XML reprezentaci geometrie, popisnou část informace nijak neomezuje. Vzhledem k masivní podpoře XML parserů mnoha vývojových prostředí se zřejmě jedná výměnný „formát budoucnosti“. Pro ukládání prostorových informací je však nevhodný, jeho velikost je oproti binárnímu formátu až desetinásobná.

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>
  </gml:exterior>
 </gml:Polygon>
</CP:geometry>
```

OGC specifikace pro ukládání prostorových informací v relačních databázích:

OpenGIS Simple Features Specification For SQL metamodel



3.4. WEBOVÉ SLUŽBY A JEJICH STANDARDY

3.4.1. OGC – WEB MAP SERVICE (WMS):

Je webová služba poskytující mapu v zadaném výřezu a zadané kartografické projekci. Obsahuje 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
```

3.4.2. 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 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:102067</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

3.4.3. OGC WEB FEATURE SERVICE (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

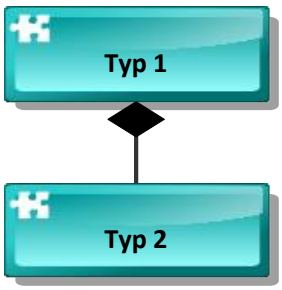
```
<StoredQueryDescription
id="CadastralBoundary_BBOX_Query">
<Abstract>Cadastral Parcel Boundaries</Abstract>
<Parameter type="xsd:double" name="XMIN"/>
<Parameter type="xsd:double" name="YMIN"/>
<Parameter type="xsd:double" name="XMAX"/>
<Parameter type="xsd:double" name="YMAX"/>
<Parameter type="xsd:string"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

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

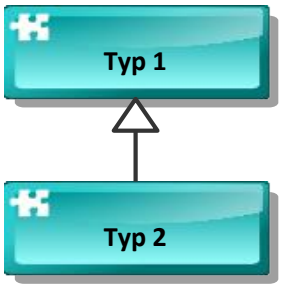
```
<Filter…>
 <BBOX>
  <gml:Envelope
    srsName="urn:ogc:def:crs:EPSG::2065">
     <gml:lowerCorner>557658 106440</gml:lowerCorner>
     <gml:upperCorner>557489 106330</gml:upperCorner>
   </gml:Envelope>
 </BBOX>
</Filter>

<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>
```

Příklad – implementace tříd akceptujících typy objektů směrnice INSPIRE (EU):



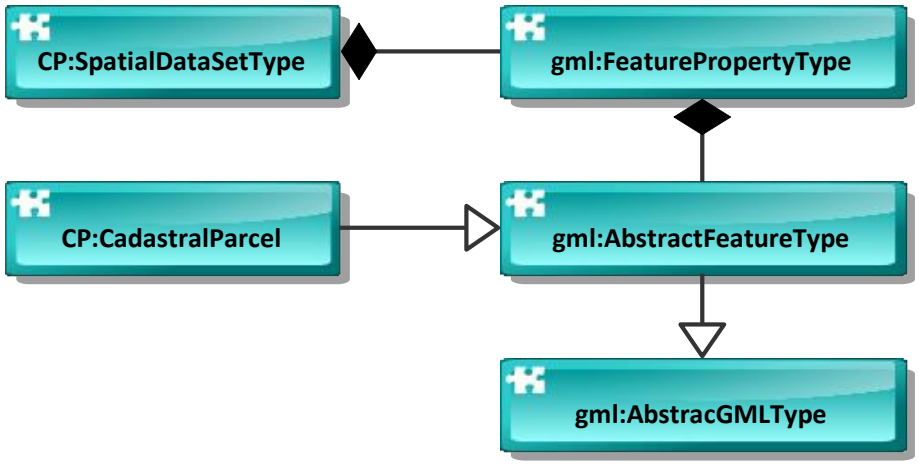
Typ "Typ 1" obsahuje atribut, vlastnost typu "Typ 2".



Typ "Typ 2" je potomkem typu "Typ 1" (dědí jeho vlastnosti)

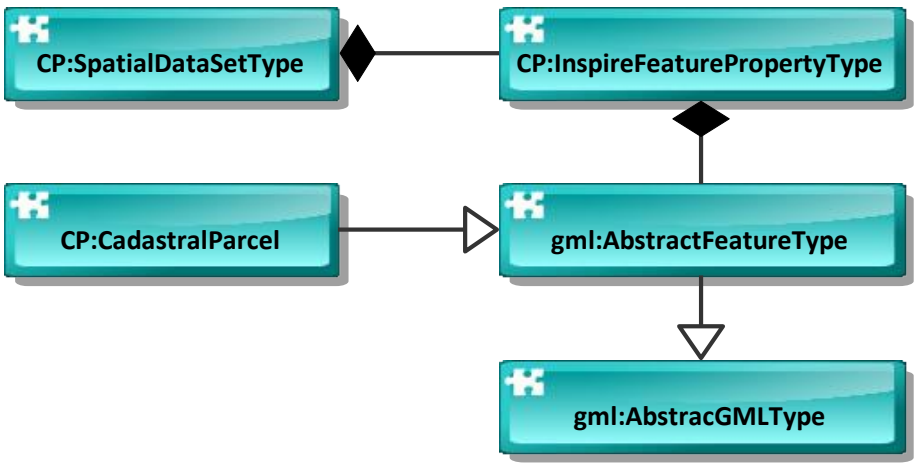
"Inspire Cadastral Parcels" generovaný C# kód:

Musí se serializovat NameSpace **CP**



"Inspire Cadastral Parcels" validní C# kód:

Kopie z gml, akceptuje objekty z **CP**



4. Efektivní přístup k prostorovým datům

4.1. VYMEZENÍ PROBLÉMU

Problém vyhledání:

- Pole V (seznam) objektů stejného typu T_1, ve které vyhledáváme.
- Množina Q (potenciálně nekonečná) typu T_2, definující možné dotazy (tj. „povolené“ typy predikátů nad T_1).
- Množina R (typu T_3) definující možné odpovědi.
- Funkce $check(r,q): R \times Q \rightarrow \{true,false\}$, která určuje, zda $r \in R$ vyhovuje dotazu $q \in Q$.

Problém vyhledání je libovolná funkce $search(q,V)$, která pro dotaz q nad množinou Q vrací všechny možné vyhovující odpovědi:

$$search(q,V) = \{r \in R \mid check(r,q)=true\}$$

Příklad - Problém příslušnosti prvku k množině:

- Položme $T_1 = T_2$ (typ vyhledávací množiny je totožný s typem dotazu)
- $V \subseteq Q$ a $|V| < \infty$
- $R = \{true, false\}$
- $check(r,q) = true \Leftrightarrow r = true \wedge q \in V$

potom říkáme, že funkce ***search*** řeší problém příslušnosti na typu T_1.

Příklad - Rozsahový dotaz na uspořádané množině:

- Nechť $(U,\leq)$ je úplně uspořádaná množina s prvky typu T_1, vyhledávací množina je úplně uspořádaná (její každé dva elementy lze porovnat).

- $V \subseteq U$ a $|V| < \infty$

- $Q \subset U \times U$, $[low,high] \in Q \Leftrightarrow low \leq high$

- $R = U$

- $check(r,[low,high]) = true \Leftrightarrow$

  $\Leftrightarrow r \in V \wedge low \leq r \leq high$

potom říkáme, že funkce $search$ řeší problém rozsahového výběru na typu T_1.

Příklad - Rozsahový dotaz na body ve 2D prostoru:

- $V \subset E_2$ a $|V| < \infty$ (konečná množina bodů euklidovského 2D prostoru).

- $Q \subset E_2 \times E_2$ taková, že

$$[xmin, ymin, xmax, ymax] \in Q \Rightarrow$$
$$xmin < xmax \wedge ymin < ymax$$

  Typ dotazů jsou obdélníky (okna) rovnoběžné s osami souřadného systému).

- $R = V$

- $check([x,y],[xmin,ymin,xmax,ymax]) = true \Leftrightarrow$

$$\Leftrightarrow [x,y] \in V \wedge xmin \leq x \leq xmax \wedge ymin \leq y \leq ymax$$

Funkce $search(q,V)$ řeší problém rozsahového dotazu na body ve 2D prostoru.

Příklad Rozsahový dotaz na obdélníky ve 2D prostoru:

- $V \subset E_2 \times E_2$ a $|V| < \infty$, taková, že

  $[xmin, ymin, xmax, ymax] \in V \Rightarrow$
  $\Rightarrow xmin < xmax \wedge ymin < ymax$

  Vyhledávací množina je tedy konečná množina obdélníků ve 2D prostoru jejichž strany jsou rovnoběžné s osami souřadného systému.

- $Q \subset E_2 \times E_2 < \infty$, taková, že

  $[xminQ,yminQ,xmaxQ,ymaxQ] \in Q \Rightarrow$
  $\Rightarrow xminQ<xmaxQ \wedge yminQ<ymaxQ$

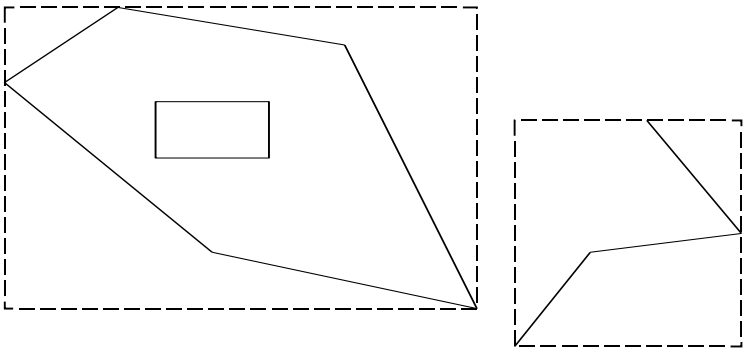
  Dotazem je libovolný obdélník ve 2D prostoru jehož strany jsou rovnoběžné s osami souřadného systému.

- $R = V$

- $check([xmin,ymin,xmax,ymax],$
  $[xminQ,yminQ,xmaxQ,ymaxQ])=true$
  $\Leftrightarrow$
  $[xmin,ymin,xmax,ymax] \in V \wedge$
  $xmin \leq xmaxQ \wedge ymin \leq ymaxQ \wedge$
  $xminQ \leq xmax \wedge yminQ \leq ymax$

Funkce **search** tedy vrací takové obdélníky, které incidují (mají neprázdný průnik) s obdélníkem dotazu. Taková funkce řeší problém rozsahového dotazu na obdélnících 2D prostoru.

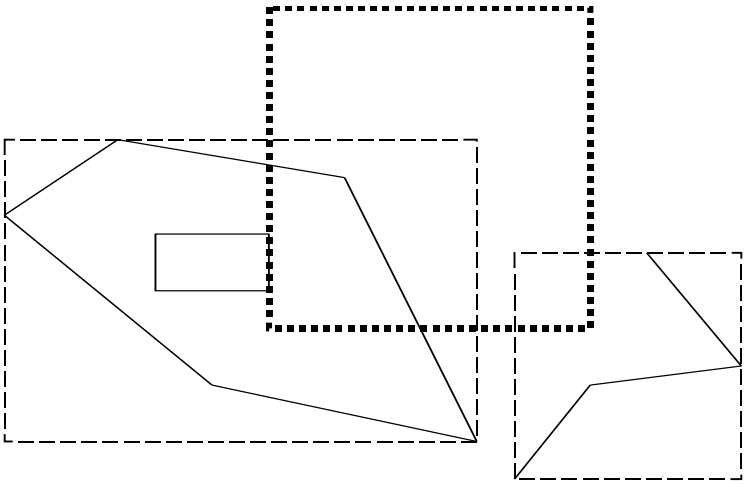
Jednotný zdroj pro prostorovou indexaci geometrických objektů: (tj. struktur pro efektivní vyhledání) je minimální omezující obdélník geometrického objektu rovnoběžný s osami souřadného systému – MBR (minimal bounding rectangle):



tedy minima, resp. maxima lomových (definičních) bodů

`[xmin,ymin,xmax,ymax]`

V naprosté většině případů vystačíme s obdélníkovým dotazem:



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

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:

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

Podmínka pro incidenci:

$$[xmin,xmax] \cap [xminQ,xmaxQ] \neq \emptyset$$
$$\Leftrightarrow$$
$$xmin \leq xmaxQ \wedge xmax \geq xminQ$$

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(V)`** vybuduje podpůrné struktury pro množinu **`V`**
- **`search(q,V)`** odpoví na vyhledávací dotaz

Dynamický:

- ***`insert(x,V)`*** vloží do množiny ***`V`*** nový objekt ***`x`***
- ***`delete(x,V)`*** vymaže z množiny ***`V`*** objekt ***`x`***
- ***`search(q,V)`*** odpoví na vyhledávací dotaz

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

Funkce **`search`** bývá většinou rozdělena na dvě části, a to

- ***`init(q,V)`*** inicializace dotazu
- ***`fetch(x)`*** vrací jeden objekt z množiny ***`V`***

práce potom probíhá podle jednoduchého schématu:

```
init(q,V);
while(fetch(x)==SUCCESS)
 {
   zpracuj_objekt(x);
 }
```

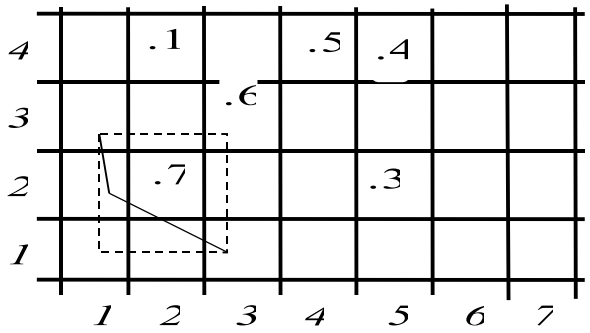
Poznámka:

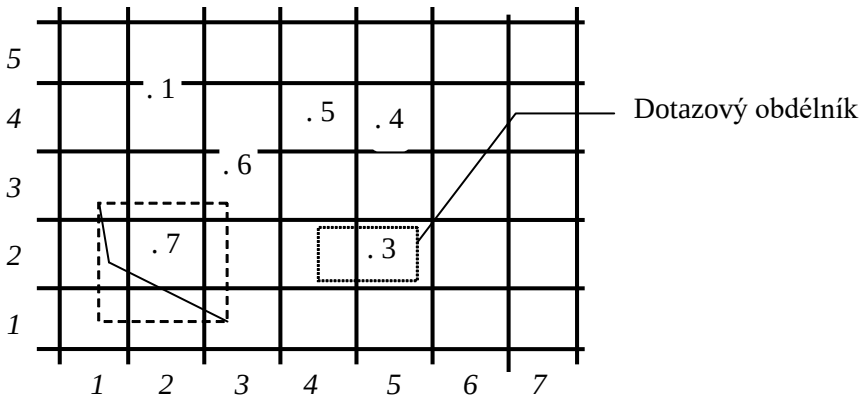
Všechny uvedené příklady lze triviálně řešit jedním průchodem množiny V, tedy v lineární časové složitosti $O(|V|)$. Uvádění jiných metod má tedy smysl pouze v případě, že tento základní odhad nějak zlepšíme.

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

4.2. 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.



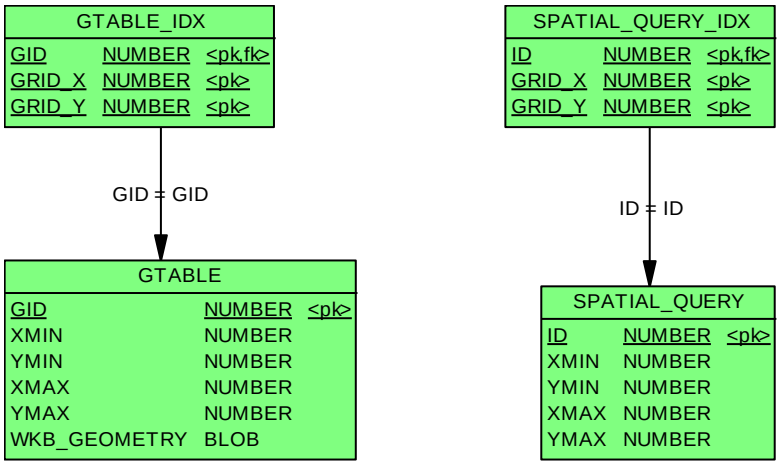


Prostorový dotaz v GRIDu“:

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

4.3. MODIFIKACE BINÁRNÍCH STROMŮ PRO PROSTOROVÉ VYHLEDÁVÁNÍ, K-D 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 syn má vlastnost „pravý“/“levý“.

- Všechny uzly „levého“ podstromu libovolného uzlu jsou menší, než tento uzel.

- Všechny uzly „pravého“ podstromu libovolného uzlu jsou větší, než tento uzel.

Implementaci binárního stromu si můžeme představit jako jednoduchou strukturu:

```
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: kořen stromu ***nod***, klíč ***key***.
2. Je-li ***nod=null*** (strom je prázdný), potom končíme vyhledávání ”neúspěchem”.
3. Je-li ***nod.Key=key***, potom končíme ”úspěchem”.
4. Je-li ***key<nod.Key***, pokračujeme krokem 1 pro ***nod.Left***
5. Je-li ***key>nod.Key***, pokračujeme krokem 1 pro ***nod.Right***

Algoritmus - Vkládání klíče do binárního stromu:

1. Vstup: klíč ***key***.
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 ***key***.

Algoritmus - Rozsahové vyhledání v binárním stromu:

1. Vstup: interval ***[min,max]***, kořen stromu ***nod***.
2. Je-li ***nod=null*** (strom je prázdný), potom konec.
3. Patří-li ***nod.Key*** do intervalu ***[min,max]***, pošleme jej na výstup a aplikujeme algoritmus na ***nod.Left*** a ***nod.Right***.
4. Je-li ***max<nod.Key***, aplikujeme algoritmus na ***nod.Left***.
5. Je-li ***min>nod.Key***, aplikujeme algoritmus na ***nod.Right***.

Zlepšení časové složitosti spočívá v tom, že v určitých fázích algoritmů jsme schopni rozhodnout, kterou větev stromu můžeme bez rizika vynechat. Potíže způsobuje skutečnost, že v jistých případech může být strom degenerovaný (např. ***nod.Left=null*** 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 $|nod.Left|$ a $|nod.Right|$ maximálně o 1 pro jeho každý uzel ***nod***.

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

$$log(|V|)$$

Algoritmus - Vybudování optimálního binárního stromu:

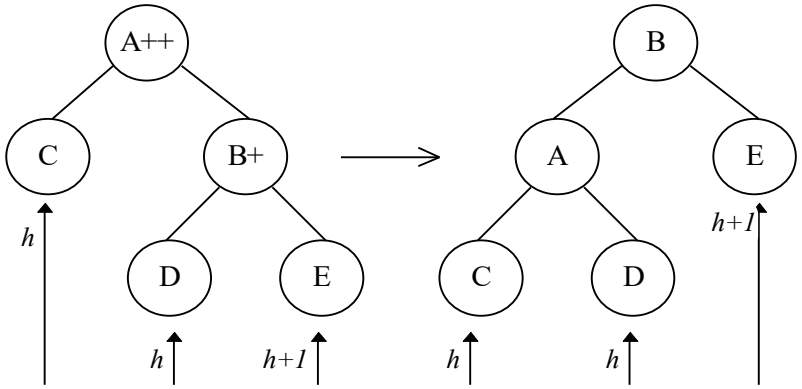
1. Vstup: množina klíčů ***V***, kořen stromu ***nod***.
2. Je-li $V = null$, skonči.
3. Rozděl množinu ***V*** na po dvou disjunktní množiny $V_1, \{med(V)\}, V_2$ tak, že $med(V)$ je medián množiny ***V***, klíče z V_1 jsou menší než $med(V)$ a klíče z V_2 jsou větší než $med(V)$.
4. Definuj kořen stromu jako $med(V)$.
5. Aplikuj algoritmus na množinu V_1 pro levý podstrom ***nod.Left***.
6. Aplikuj algoritmus na množinu V_2 pro pravý podstrom ***nod.Right***.

Definice - Vyvážené stromy:

Binární strom nazveme vyvážený, liší-li se hloubky ***nod.Left*** a ***nod.Right*** maximálně o 1 pro jeho každý uzel ***nod*** (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). 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 přeorganizuje, např:



Počet uzlů v podstromu s kořenem **nod** označíme **|*nod*|.**

Definice - ***BB[α]*** stromy:

Buď ***0 < α < 1/2.*** Binární strom patří do třídy ***BB[α]*** stromů, platí-li pro jeho každý uzel ***nod***

$$\alpha < |nod.Left|/(|nod|) < 1-\alpha$$

Poznámka – smysl $BB[\alpha]$:

Pokud byl v nějakém okamžiku podstrom definovaný uzlem nod optimální, pak k porušení podmínky z definice $BB[\alpha]$ stromů musí dojít k minimálně $c*|nod|$ vložení/mazání uzlů do/z příslušného podstromu (c je konstanta závislá pouze na parametru α).

Definice **k-D** strom:

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

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

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

Říkáme, že

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

k-D stromem nad S nazveme binární strom, jehož uzly jsou k-tice z S^k, a kde pro každý uzel nod, jeho levý podstrom $nod.Left$ a všechny uzly tohoto podstromu $nodL$ platí:

$$nodl \leq_i nod \text{ kde } i = level(nod) \bmod k$$

Analogická podmínka musí být splněna i pro pravý podstrom $nod.Right$.

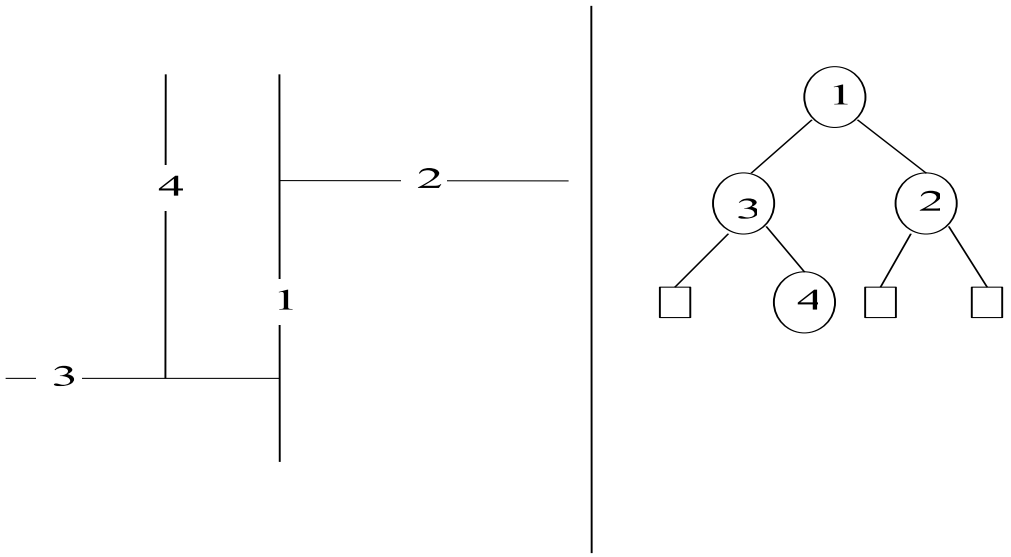
Algoritmus – Vyhledání bodu ve 2-d stromu:

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 **2-D** stromu, každý vložený bod dělí, jistou část roviny na dvě „poloroviny“.

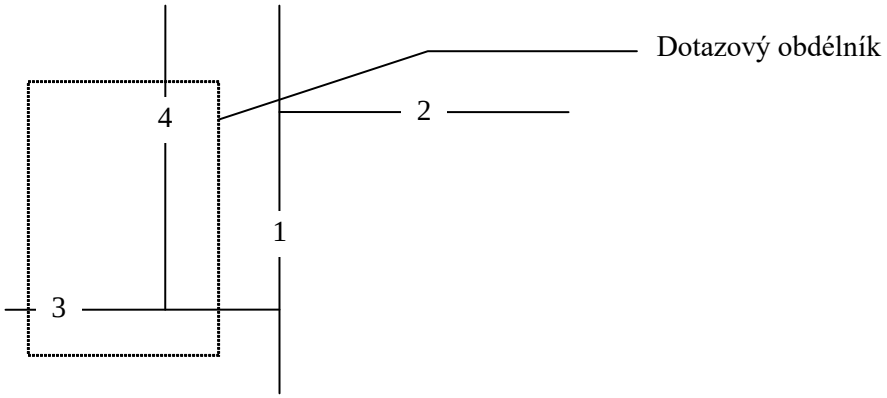


Algoritmus - Rozsahový dotaz pro body ve 2-d stromu:

1. Vstup: Kořen **nod** a obdélník **q=[xmin,ymin,xmax,ymax]**.
2. Je-li **nod=null**,konec.
3. Je-li **nod.Key** (tj. bod ve 2D prostoru) dotazovém 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 ***nod***, například je-li:

$$\mathbf{level(nod)\ mod\ 2=0 \wedge nod.Key.x > xmax}$$

potom aplikuj algoritmus jen na větev ***nod.Left***, 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.Key.x>A.Key.x ∧ D.Key.y<B.Key.y neplyne

A.Key.x<B.Key.x ∧ D.Key.y>A.key.y

Pomocí ***BB[α]*** techniky lze však **k-D** stromy udržovat vyvážené pomocí částečné reorganizace, tedy „hlídat“ v každém uzlu **k-D** stromu poměr počtu uzlů v jeho levém a pravém podstromu a v případě porušení podmínky ***BB[α]*** nahradit vybudovat optimální strom.

Algoritmus -Vybudování optimálního **2-D** stromu:

1. Vstup: množina bodů ***V***, kořen stromu ***nod***, úroveň uzlu ***l*** ∈ **{'x','y'}**
2. Je-li ***V = null***, skonči.
3. 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***-ová souřadnice je medián množiny ***l***-ových souřadnic z ***V***, ***l***-ové souřadnice z V_1 jsou menší než $med_l(V)$ a ***l***-ové z V_2 jsou větší než $med_l(V)$.
4. Definuj kořen stromu jako $med_l(V)$.
5. Je-li ***l*** rovno '***x***'potom přiřaď ***l='y'*** jinak ***l='x'***
6. Aplikuj algoritmus na množinu V_1 pro levý podstrom ***nod.Left***.
7. Aplikuj algoritmus na množinu V_2 pro pravý podstrom ***nod.Right***.

Poznámka:

Rozdělení množiny z kroku 3. lze realizovat modifikaci metody **`QuickSort`**.

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

Algoritmus - Rozsahový výběr pro obdélníky ve 4-d stromu:

1. Vstup: kořen stromu ***`nod`***, dotazový obdélník ***`q=[xmin,ymin,xmax,ymax]`***.
2. Je-li ***`nod = null`***, skonči.
3. Jsou-li obdélníky ***`q`*** a ***`nod.Key`*** incidentní, pošli ***`nod`*** na výstup a aplikuj algoritmus na ***`nod.Left`*** a ***`nod.Right`***.
4. Podle úrovně, ve které se nacházíš ve stromu, se rozhodni, zda můžeš vynechat nějakou větev, např. Je-li:

   `level(nod) mod 4=0 a nod.Key.xmin > q.xmax`

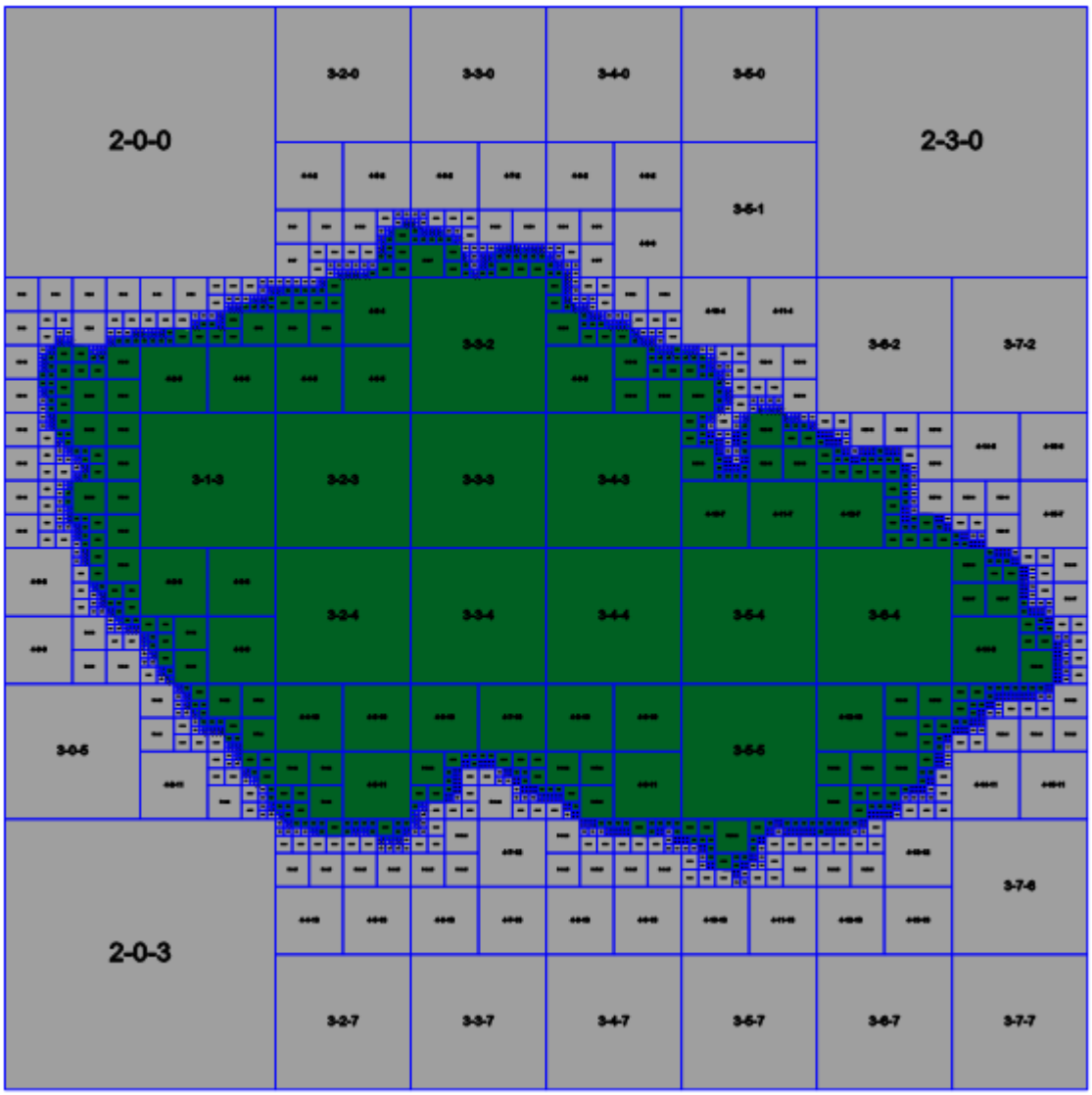
   aplikuj algoritmus pouze pro ***`nod.Left`***.

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

4.4. QUAD TREE – KVADRANTOVÉ STROMY

Kvadrantové stromy jsou prostorové vyhledávací struktury založené na pravidelném dělení části 2D prostoru (většinou čtverce) na čtyři stejné části.

V základní verzi mohou přímo definovat polygonální geometrii.



Definice – kvadrantový strom:

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

- Uzel je tvořen obdélníkem **R=[xmin,ymin,xmax,ymax]** a příznakem **content∈{empty,full,half}.**
- List má příznak **content∈{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 odpovídá 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 neobsahuje souřadnice čtverce/obdélníku, který reprezentuje, stačí si pamatovat nultý čtverec/obdélník kořene stromu. Souřadnice ostatních uzlů jsou potom definovány cestou od kořene.

Algoritmus – vyhledání dlaždice v QuadTree:

1. Vstup sloupec **`col`**, řádek **`row`**, podrobnost **`zoom`** a kořen **`node`**.

2. **`i=zoom`**

3. Je-li **`node.Content≠half`** vrať **`node.Content`** a konec.

4. Je-li **`i==0`** vrať **`node.Content`** a konec.

5. Nechť col_i je **`i`**-tá binární cifra čísla **`col`** zprava, row_i je **`i`**-tá binární cifra čísla **`row`** zprava.

6. Je-li $col_i=0$ pokračuj nahoru, je-li $col_i=1$ pokračuj dolů, Je-li $row_i=0$ pokračuj doleva, je-li $row_i=1$ pokračuj doprava. Přiřaď uzlu **`node= node`** „pokračování“, **`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 kořen **node.**

2. Je-li **node.Content==full** potom **node.Content=empty** a návrat.

3. Je-li **node.Content==empty** potom **node.Content=full** a návrat.

4. Je-li **node.Content==half** potom proveď kroky 2. a 3. pro všechny následníky.

Algoritmus – průnik QuadTree:

1. Vstup kořeno dvou stromů **node1** a **node2.**

2. Je-li **node1.Content==empty** nebo **node2.Content==empty** vrať uzel **node** s obsahem **node.Content= empty.**

3. Je-li **node1.Content==full** a **node2.Content==full** vrať uzel **node** s obsahem **node.Content=full.**

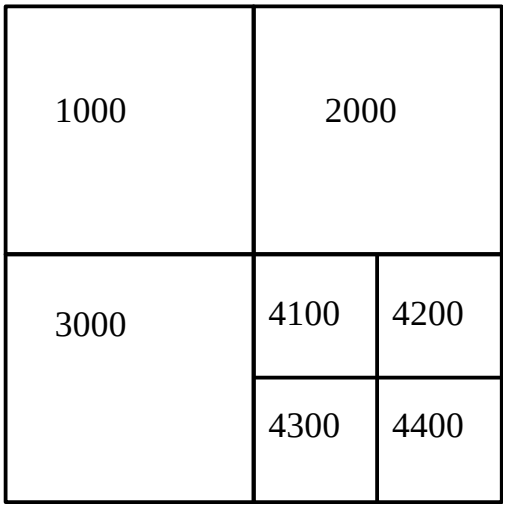
4. Je-li **node1.Content==half a node2.Content==full** vrať **node1**.

5. Je-li **node2.Content==half a node1.Content==full** vrať **node2**.

6. Jinak vytvoř nový uzel **node** a následníkům přiřaď uzly opakováním kroků 2. – 6. pro odpovídající dvojice následníků uzlů **node1** a **node2**. Vrať nový uzel **node.**

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 **`substr(K,l)`**.

Tvrzení – incidence obdélníků n non-pointer QuadTree:

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

Označíme-li,

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

potom:

$$\mathbf{substr(Q(A),l)=substr(Q(B),l)}$$

Algoritmus - Vyhledání obdélníků v non-pointer QuadTree:

1. Vstup – obdélník **S=[xmin,ymin,xmax,ymax]**.
2. Pošli na výstup všechny obdélníky **A**, pro které:

$$\mathbf{substr(Q(A),len(NZ(Q(S))))= NZ(Q(S))}$$
a
$$\mathbf{A \cap S \neq \emptyset}$$

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

$$\mathbf{Q(A)=P}$$
a
$$\mathbf{A \cap S \neq \emptyset}$$

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 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.

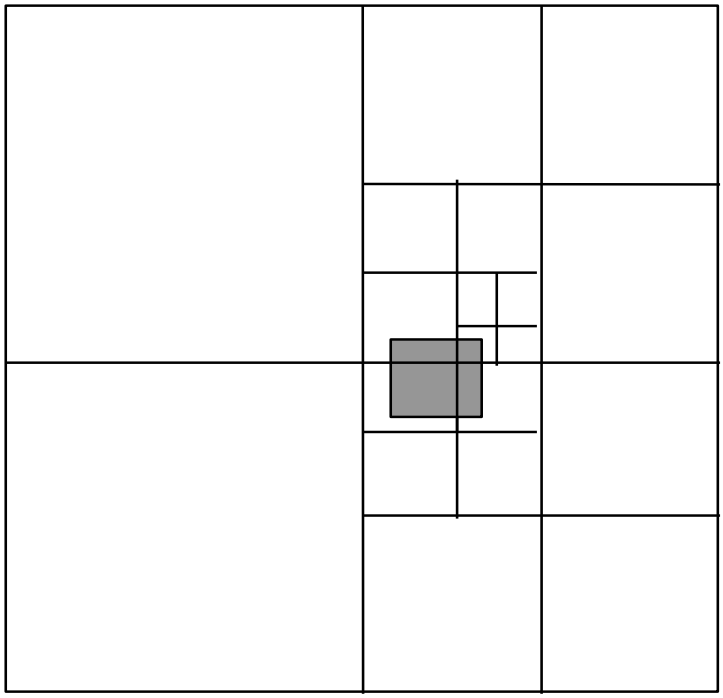
3. Aplikuj krok 2. na čtverce s_1 *a* s_2 podle druhé souřadnice.

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 **WKB** nepředepisuje metodu efektivního výběru, tímto způsobem ji můžeme doplnit.

- „jeden objekt“´=„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 sklíči:

```
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)
```

4.5. SB+ 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
- pro klíče v uzlu $\mathbf{key_1,..,key_k}$ jsou vzestupně uspořádány
- ukazatel $\mathbf{p_i}$ ukazuje na uzel, jehož všechny klíče jsou v intervalu $\mathbf{[key_i,key_{i+1}]}$ (formálně předpokládáme, že $\mathbf{key_0}=-\infty$ a $\mathbf{key_{k+1}}=\infty$).

Uzly B+ stromu mají tedy tvar $\mathbf{p_0 key_1 p_1 ... p_{k-1} key_k p_k}$.

Algoritmus vložení klíče do B+ stromu:

1. Vstup klíč **key** a kořen B+ stromu.
2. Není-li uzel list vyber dvojici klíčů $\mathbf{key_i,key_{i+1}}$ s vlastností $\mathbf{key_{i-1}<key<key_i}$ a pokračuj uzlem na který ukazuje $\mathbf{p_i}$. Opakuj tento krok. Dokud uzel není list.
3. Vlož uzel do listu na správnou pozici. Je-li počet uzlů>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:

   $\mathbf{p_0 key_1 p_1 \ldots p_{k-2} key_{k-1} p_{k-1}}$ a $\mathbf{p_k key_{k+1} p_{k+1} \ldots p_m key_{m+1} p_{m+1}}$

   a ukazatele na ně **q** resp. **r**

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

   $$\mathbf{q\ key_k\ r}$$

   jinak nechť $\mathbf{p_i}$ je ukazatel z otcovského uzlu. Vlož klíč $\mathbf{key_k}$ do otcovského uzlu na pozici:

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

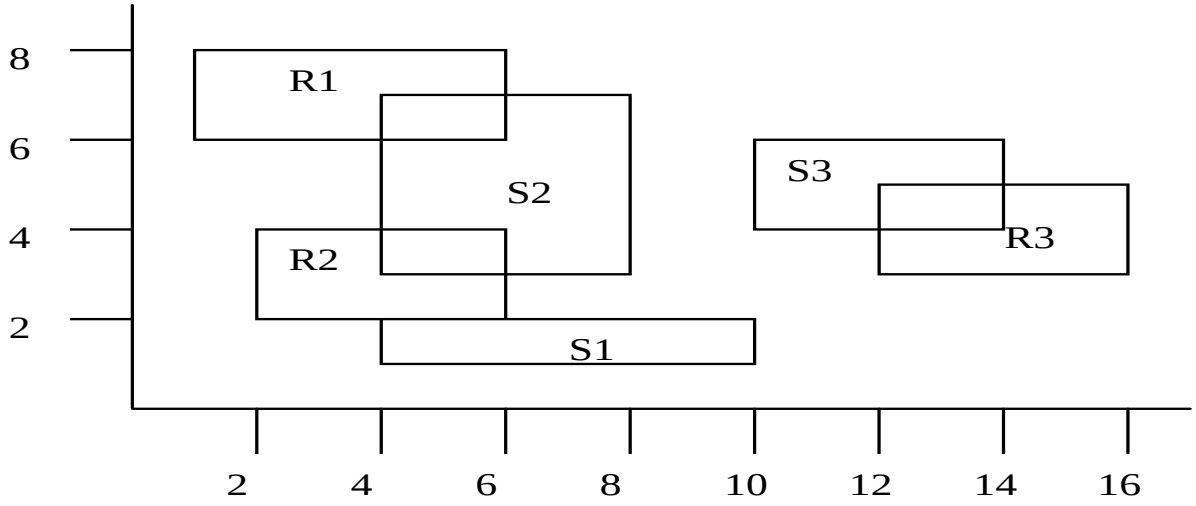
4. Má-li otcovský uzel **m+1** klíčů, opakuj pro něj kroky 1.-3.

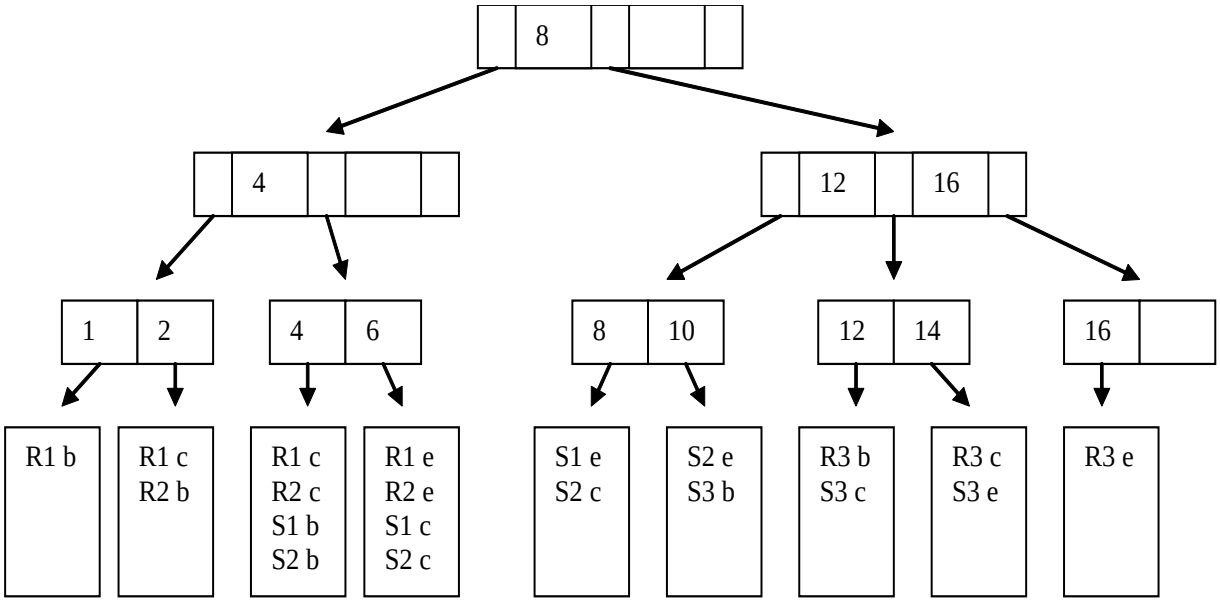
Na základě struktury B^+ byl definován SB^+ strom:

SB^+ strom je B^+ strom z počátečních a koncových bodů intervalů a navíc:

- V SB^+ stromech jsou k listům přidány seznamy identifikátorů intervalů, které jsou incidentní s klíčem v listu (tj. nějakým počátkem resp. koncem nějakého intervalu).

- S každým identifikátorem je pamatován příznak, který označuje zda v se jedná o počáteční hodnotu intervalu, koncovou hodnotu intervalu, popřípadě zda interval touto hodnotou prochází.


8
6
4
2
R1
S2
S3
R3
R2
S1
2
4
6
8
10
12
14
16


8
4
12
16
1
2
4
6
8
10
12
14
16
R1 b
R1 c
R2 b
R1 c
R2 c
S1 b
S2 b
R1 e
R2 e
S1 c
S2 c
S1 e
S2 c
S2 e
S3 b
R3 b
S3 c
R3 c
S3 e
R3 e

Algoritmus – Incidence intervalů v SB^+ stromech:

1. Vstup dotazový interval **`[xmin,xmax]`**. Existující SB^+ strom **`S`**.

2. Najdi ve stromu takový list, že pro bod **`ip`** který reprezentuje tento list platí:

$$ip = \min\{i;\ i \in S, i > xmin\}$$

3. Pro všechna **`i`** s vlastností:

$$ip < i < xmax$$

4. Pošli na výstup identifikace intervalů ze seznamu listu reprezentovaným bodem i (identifikace se mohou opakovat, posíláme jen jednou).

Algoritmus – Vkládání intervalů do SB^+ stromu:

1. Vstupní interval **[xmin,xmax]**, jeho identifikace **I**.

2. Najdi ve stromu takový list, že pro bod **ip** který reprezentuje tento list platí **ip=xmin**.

3. Jestliže v kroku 2. jsme takový list nenašli, potom:

   3.1. Vlož do stromu bod **xmin** standardní metodou pro B^+ stromy

   3.2. Nechť **pip** je bezprostřední předchůdce **xmin**, **nip** bezprostřední následník **xmin** v **SB^+** stromu.

   3.3. Polož:

   xmin.seznam = pip.seznam $\cap$ **nip.seznam**

   bez ohledu na příznak typu incidence.

   3.4. Polož příznak typu incidence =**'c'** pro všechny intervaly z **xmin.seznam**.

4. Kroky 2.-3. pro xmax.

5. Pro všechny listy SB^+ stromu takové, že pro jejich body ip platí **xmin** $\leq$ **ip** $\leq$ **xmax**:
   5.1. Je-li **ip=xmin** potom přidej do **ip.seznam** identifikaci **I** a příznak typu incidence '**b**'.
   5.2. Je-li **ip=xmax** potom přidej do **ip.seznam** identifikaci **I** a příznak typu incidence '**e**'.
   5.3. Je-li **xmin<ip<xmax** potom přidej do **ip.seznam** identifikaci **I** a příznak typu incidence '**c**'.

Poznámka:

Vícerozměrný problém řešíme vybudováním indexových struktur pro každou osu. Pro vícerozměrný výběr potom musíme vytvořit výstup jako průnik výstupů pro každou osu.

Poznámka:

Strukturu SB^+ stromu můžeme velmi efektivně použít na řešení incidence objektů v dotazovém okně, tedy na dotaz typu: Všechny dvojice objektů, které mohou mít neprázdný průnik a leží v daném dotazovém okně. Takový dotaz řešíme snadnou modifikací algoritmu v kroku 3.

Poznámka:

Metoda SB^+ stromu je okamžitě použitelná v relačních databázích indexovými tabulkami typu:

```
create table table_idx
(
 idInterval int,
 point      int,
 incidence  varchar(1)
);
```

4.6. R-STROMY

Analogie k B-stromům, klíče jsou obdélníky.

M – maximální počet klíčů v uzlu,
m ≤ M/2 – minimální počet klíčů v uzlu

Definice R-strom:

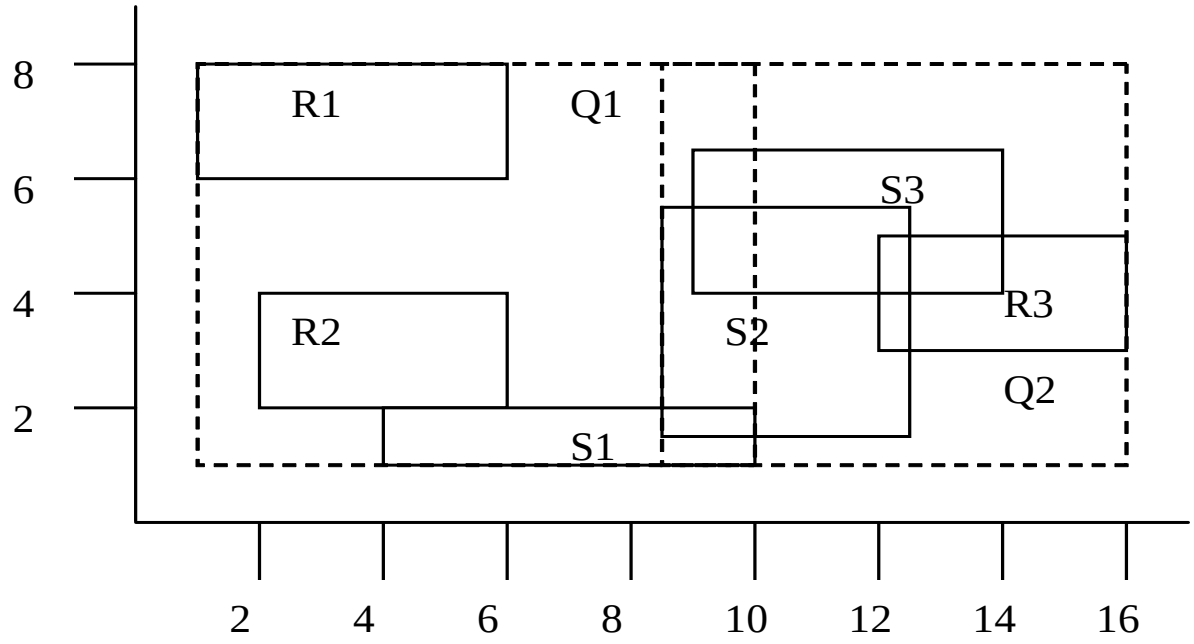
- Každý uzel obsahuje minimálně **m** klíčů a maximálně **M** klíčů pokud není kořen.

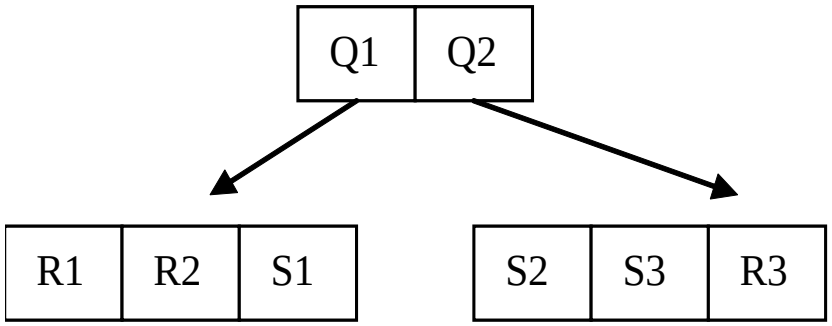
- Klíče v R-stromech jsou obdélníky s ukazateli na synovské uzly, v listech obdélníky s ukazateli na geometrické prvky.

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

Algoritmus – Vkládání klíčů do R- stromů:

1. Vstup, klíč **`Key=(MBR,ID)`**

2. Vyhledej list **`N:`**

   `a.` Polož **`N=kořen stromu.`**

   b. Je-li **`N`** list pokračuj 3, jinak c)

   c. Nechť klíč **`F`** v **`N`** jehož obdélník vyžaduje nejmenší rozšíření takové, aby obsahoval MBR vstupujícího klíče. Rozšiř jeho MBR o klíč **`Key`** a pokračuj d.

   d. **`N`**=synovský uzel na který ukazuje **`F`**, pokračuj b.

3. Přidej **`Key`** 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=rodič(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, splňujípodmínku 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. Vpřípadě, 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 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. Vpřípadě, 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:

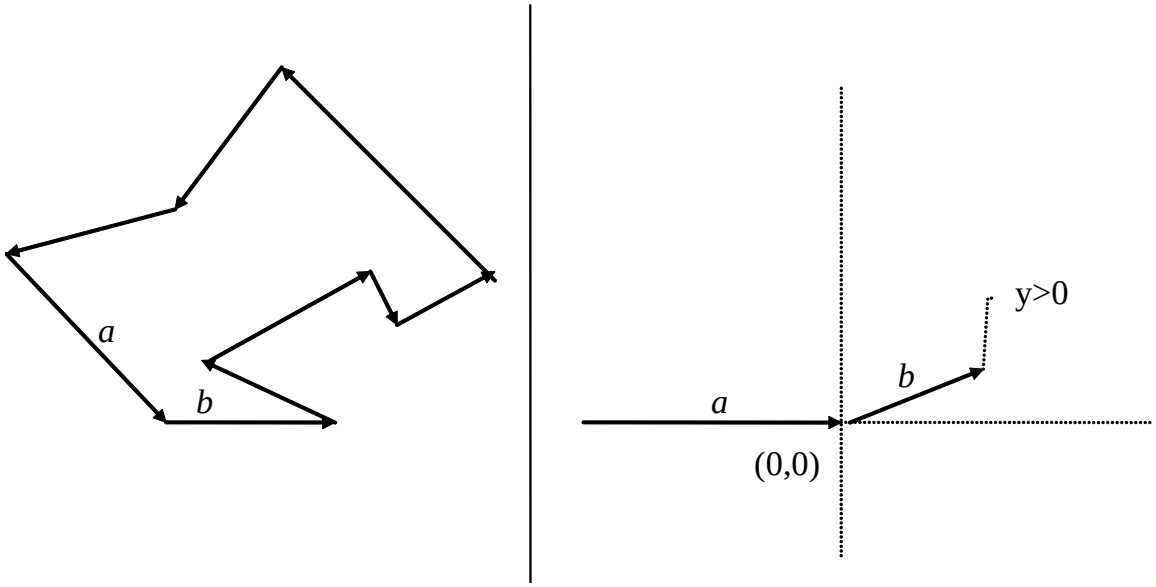
Předpokládáme, že polygony (hranice) jsou “správně” orientovány,

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

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

5.3. POLOHOVÉ FUNKCE

Určení orientace polygonu:



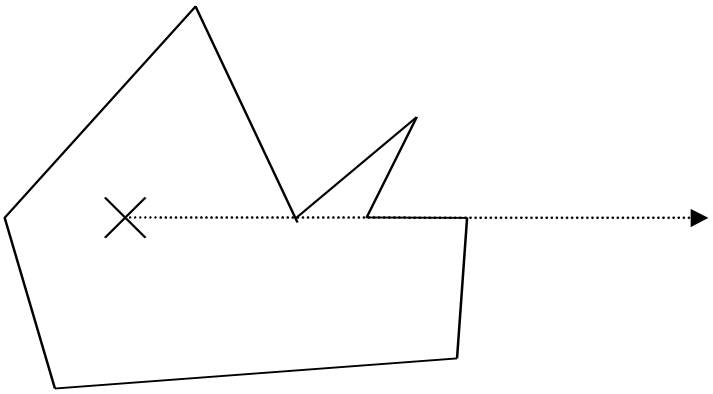
Algoritmus - Určení orientace polygonu:

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*-ová 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.***

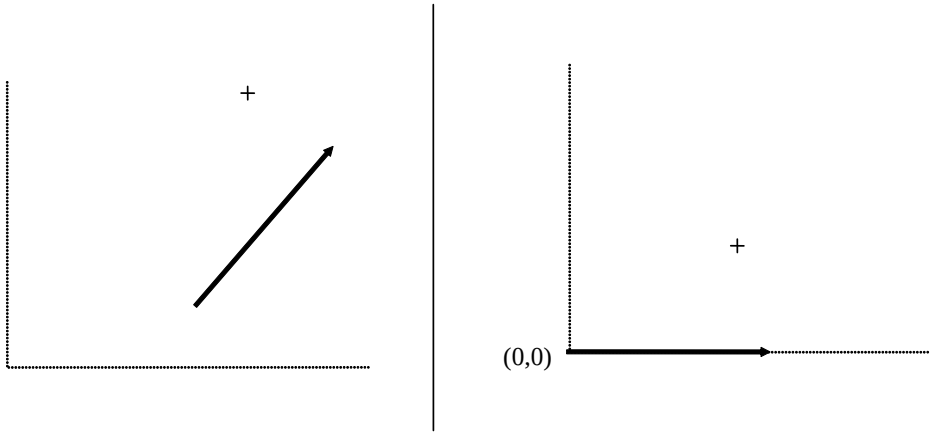
4. Končí-li úsečka na polopřímce a začíná-li mimo polopřímku, stanov podle počátku úsečky ***odkud:=POD*** nebo ***odkud:=NAD***.

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:

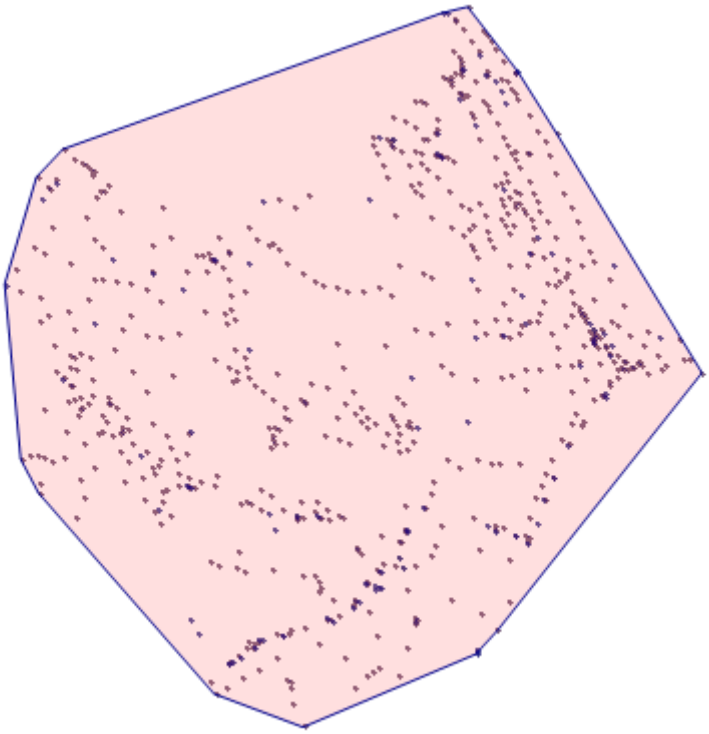
```
Distance(g1 Geometry, g2 Geometry) : Double
```

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.



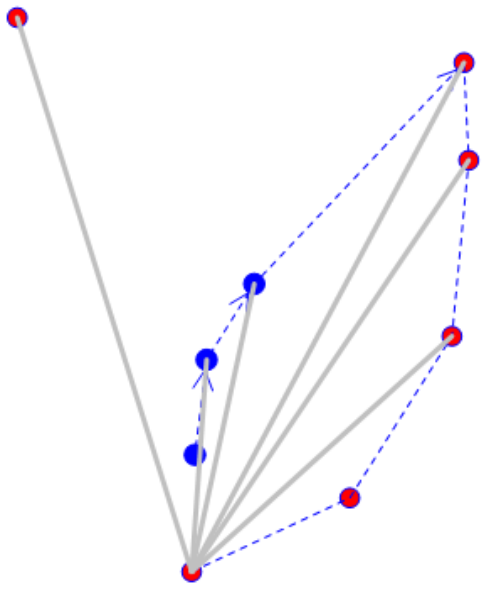
Konvexní polygon je polygon s touto vlastností: Každý vnitřní bod úsečky, jejíž krajní body leží na hranici polygonu, je i vnitřním bodem polygonu.

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 dale $\{\alpha_1,..,\alpha_n\}$ jsou úhly mezi x osou a úsečkou $[s_0,s_n]$. Potom:

- Bod s_i leží nalevo od přímky definované body $[s_{i-2},s_{i-1}]$
- Polygon C je tvořen nejvíce body z S s předešlou vlastností
- $\alpha_i \leq \alpha_{i+1}$

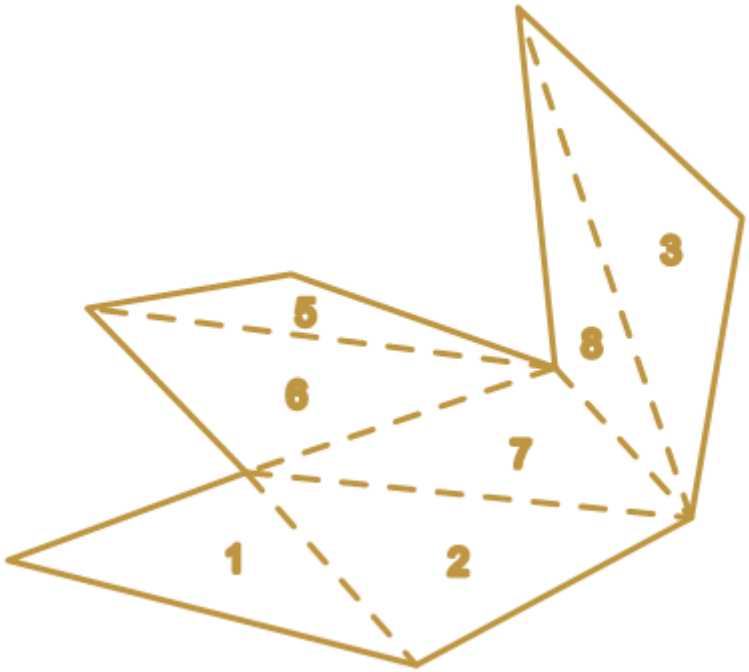
Algoritmus – konvexní obal (Graham scan - *O(N*log(N)*):

1. Vstup - soubor **N** bodů **S**
2. Vyber z **S** bod **P0** s minimální y-souřadnicí ten nejvíce vpravo (max. x-souřadnice).
3. Setřiď body podle úhlu, které svírá přímka procházející daným bodem a bodem **P0** do pole **P[]**, bod **P0** bude prvním bodem pole.
4. Vlož do zásobníku **R** bod **P[0]**=**P0** a bod **P[1]**.
5. Pro **1 < i < N**, kde **N** je počet bodů v **P:**
6. Nechť **PT1** je první bod v zásobníku **R**, **PT2** druhý bod
7. Je-li P[i] napravo od přímky **PT1→PT2**, vlož **P[i]** do zásobníku a **i:=i+1**, jinak odstraň **PT1** 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 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.

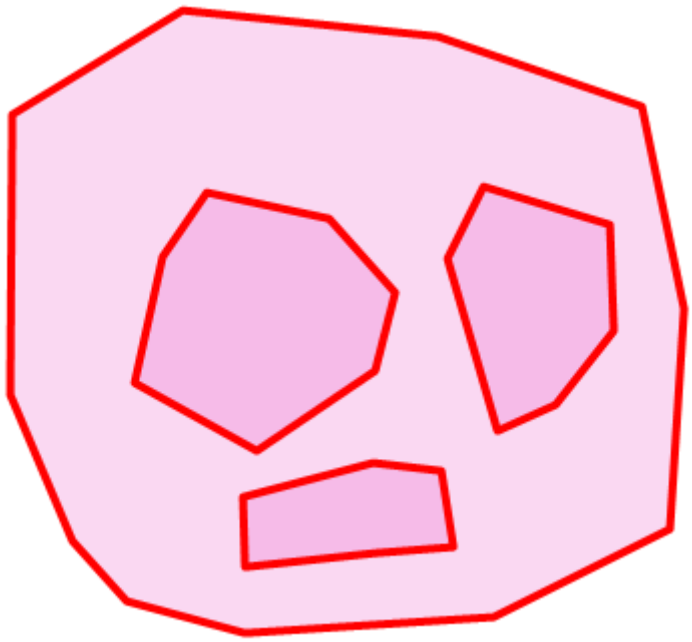
6. Otestuj, zda pro trojúhelník ***t*** tvořený zkoumanou trojicí platí (bod p_i je ‚viditelný' z bodu p_{i+2}):

   A) Žádný lomový bod vstupního polygonu neleží uvnitř ***t***.
   B) Žádná úsečka z hranice vstupního polygonu neprotíná žádnou úsečku hranice ***t***.

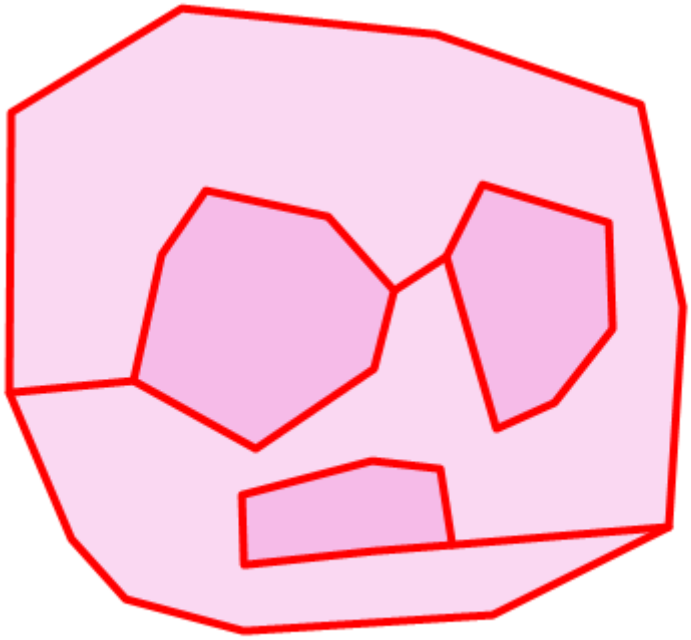
7. V případě 5. je splněno, přidej do seznamu **T** trojúhelník ***t***. Odstraň bod p_{i+1} z hranice polygonu a pokračuj 2. (tj. snížili jsme počet bodů v hranici a rekurzivně aplikujeme týž postup).

Algoritmus „Ear clipping" – pro polygony s dírami

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



převedeme na:



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 5.

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:

```
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 Precision):
Geometry
```

Bod – bod:

Triviání 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.

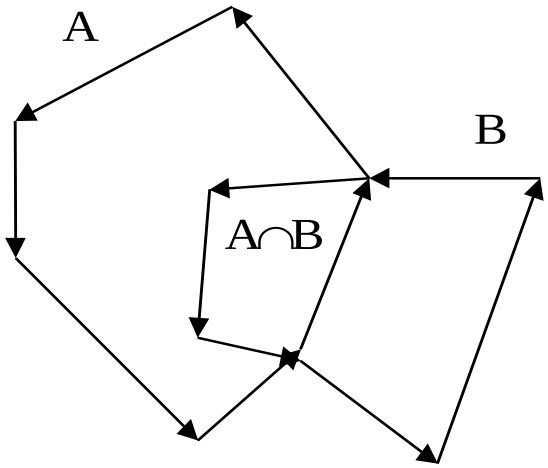
4. Zřetěz segmenty z S do “co nejdelších” lomených čar, a výsledek pošli na výstup

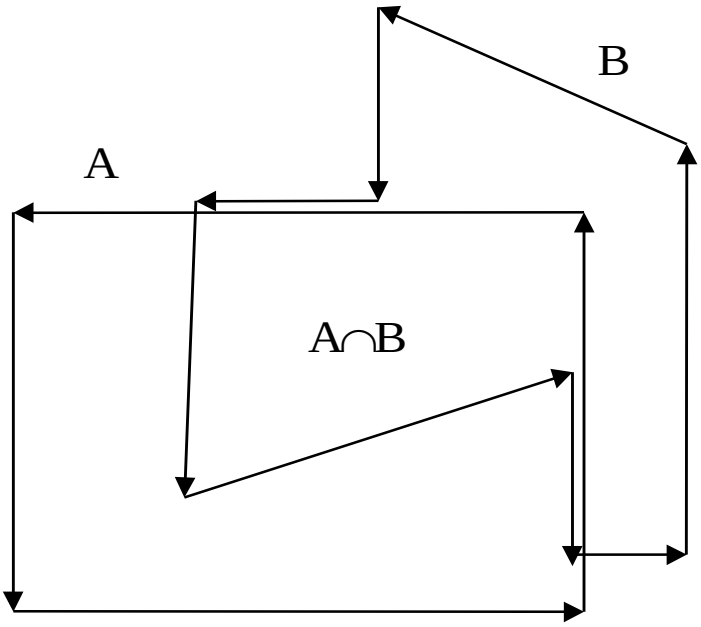
Oblast – oblast:

Doplněk:

Změna orientace hranic oblasti.

Průnik dvou oblastí:





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

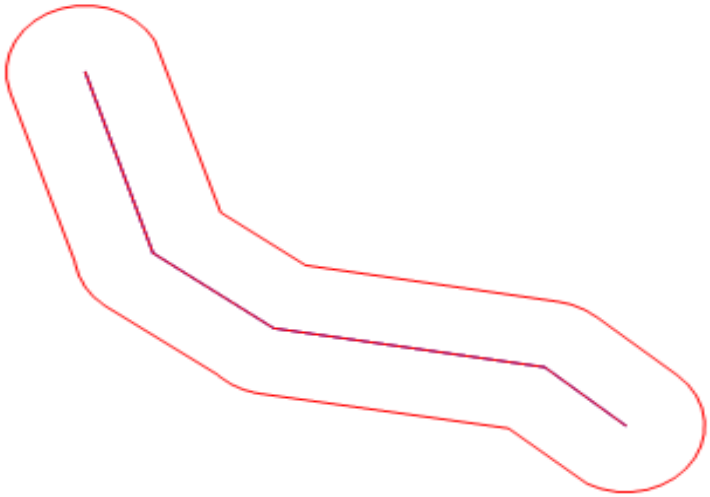
3. Do seznamu zařaď ty hrany, které buď, leží celé v druhé oblasti, nebo splývají s nějakou hranou z druhé oblasti, se kterou mají stejnou orientaci, (totožné hrany jen jednou).

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



Algoritmus – Obalová zóna linie:

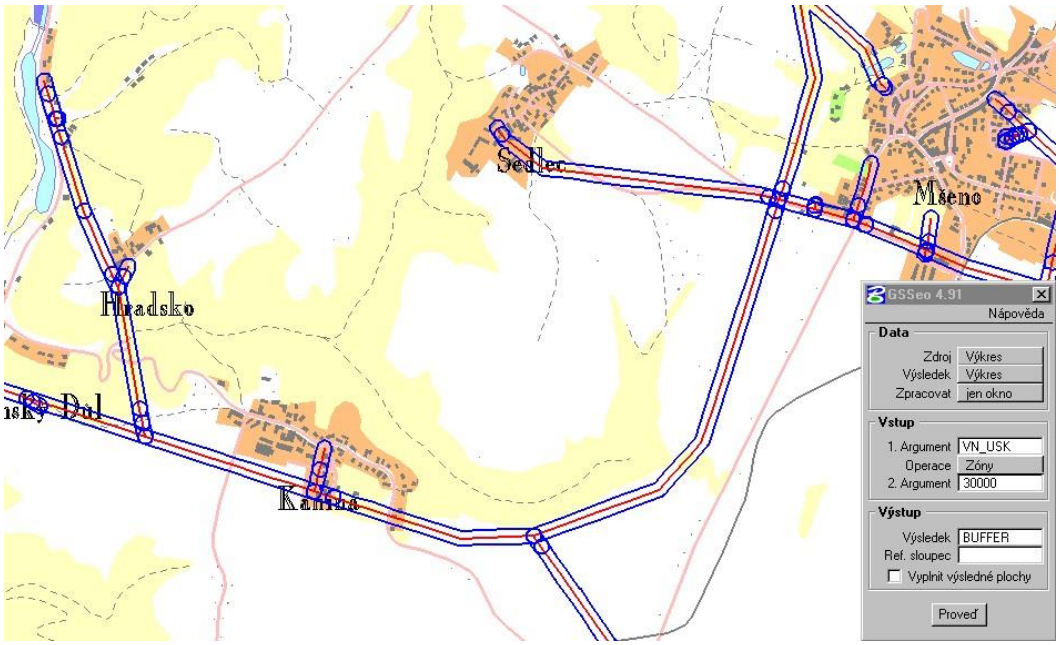
1. Obalíme jednotlivé sementy (úsečky) lomenné čáry.
2. Obaly segmentů sjednotíme.

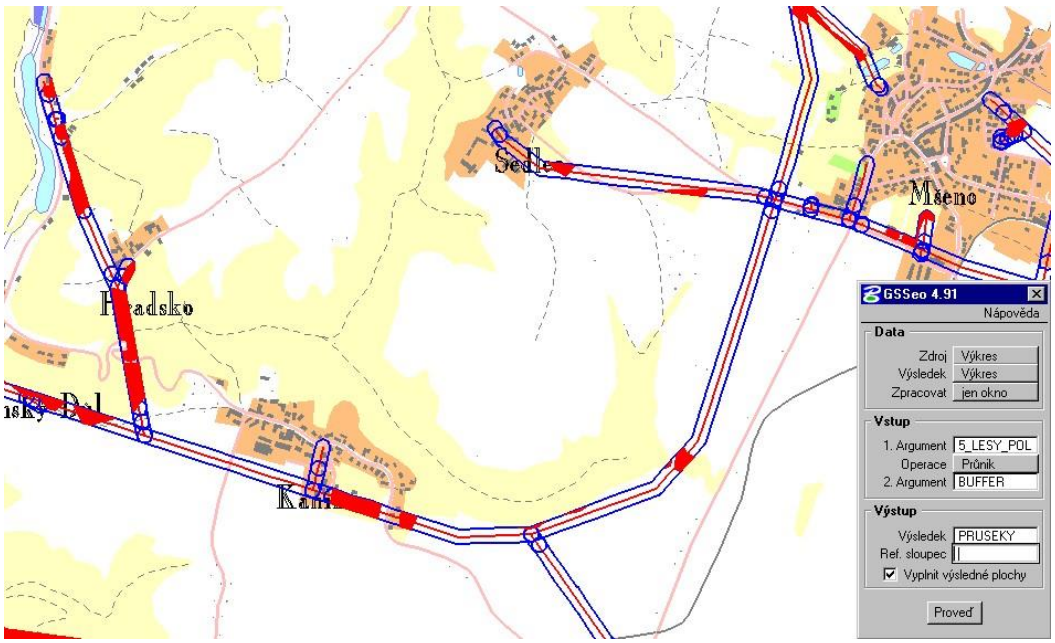
Algoritmus – Obalová zóna oblasti:

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 useků 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:

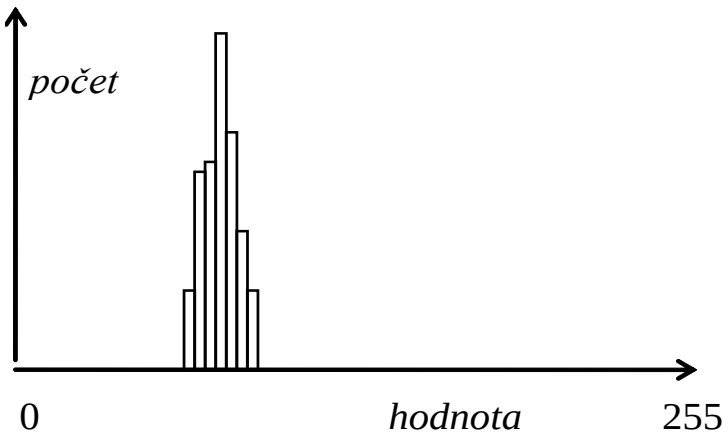
3	**2**	**1**
4	**P**	**0**
5	**6**	**7**

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

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 $CM(f)=\{cm_{ij}\}$, kde, cm_{ij} 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 někdy vyjádřena váženým průměrem pixelů a lze ji vyjádřit:

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

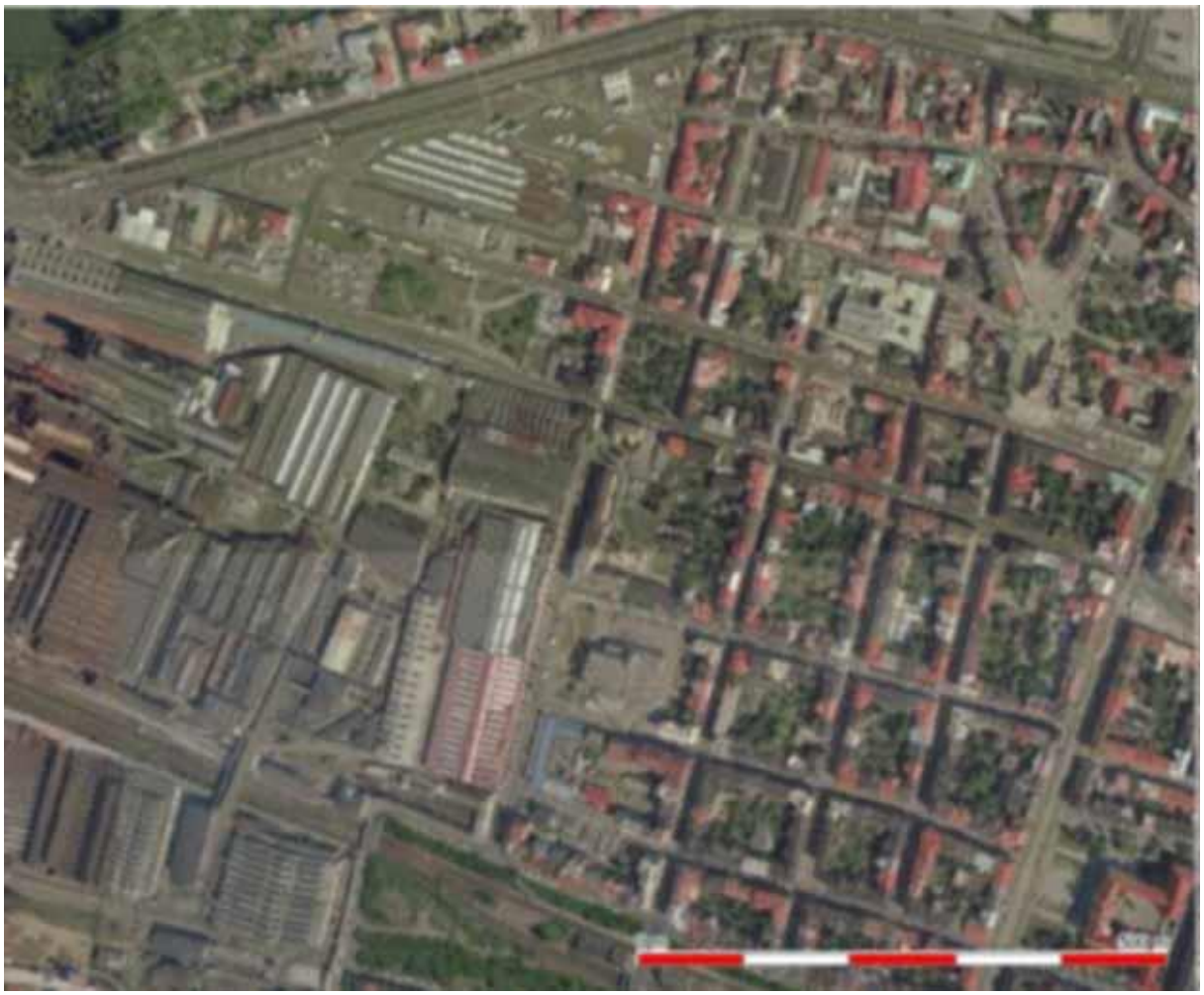
Zhlazující filtry:

a)

1	**1**	**1**
1	**1**	**1**
1	**1**	**1**

b)

1	**2**	**1**
2	**4**	**2**
1	**2**	**1**



Zostřující filtry:

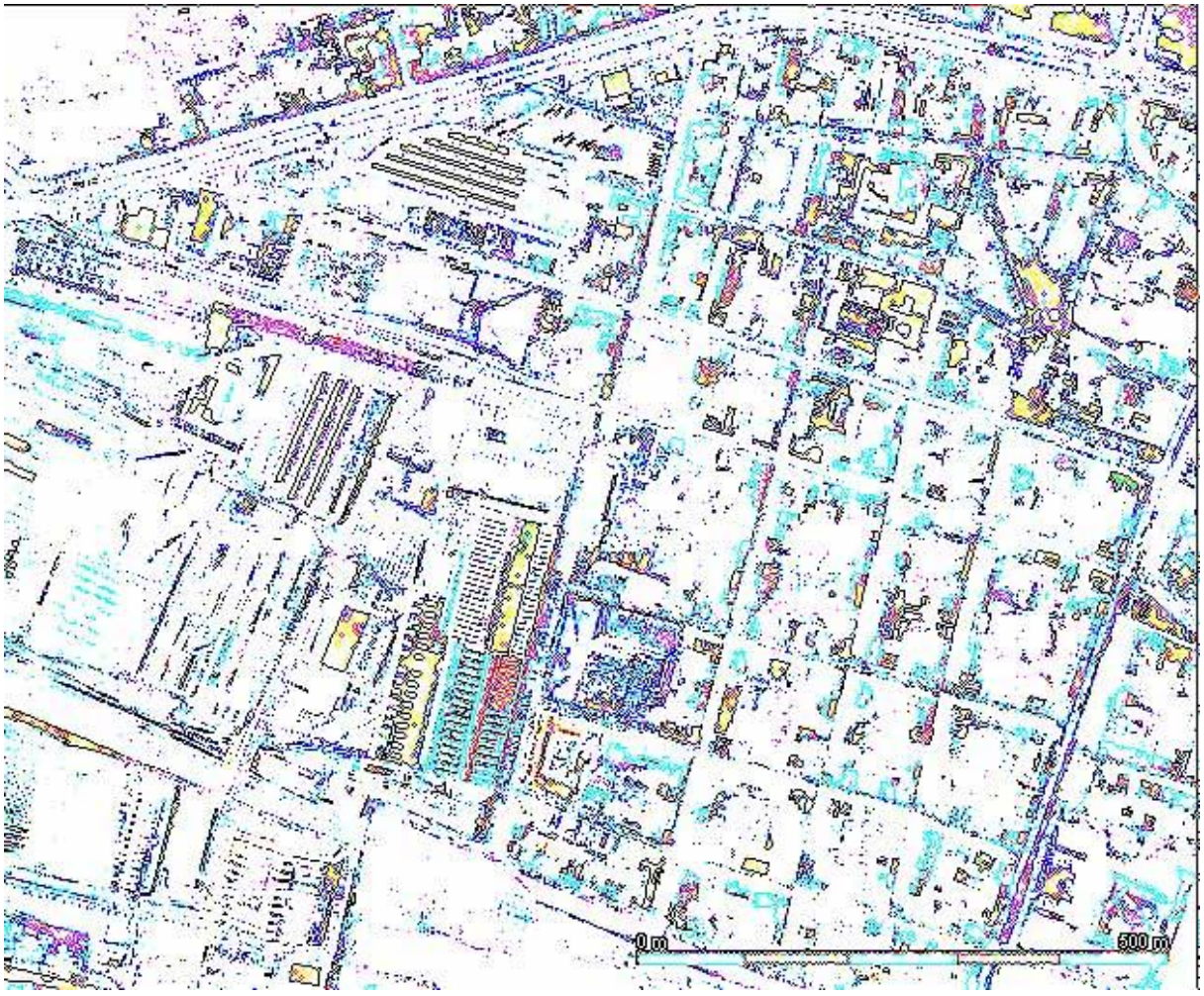
-1	**-1**	**-1**
-1	**n**	**-1**
-1	**-1**	**-1**

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

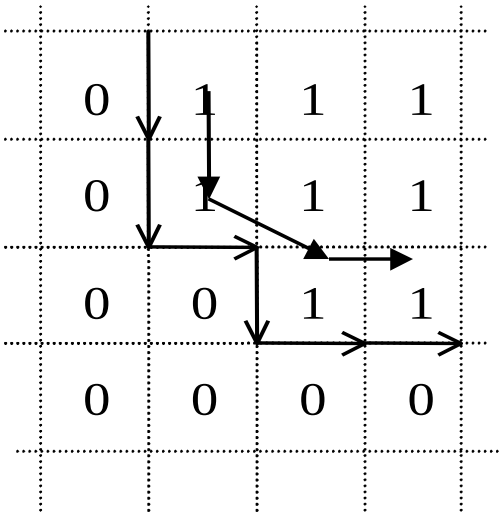
Zhlazovací filtr:

Zostřující filtr:

Detektor hran:

Kontura (hranice) oblasti v binárním obrazu:

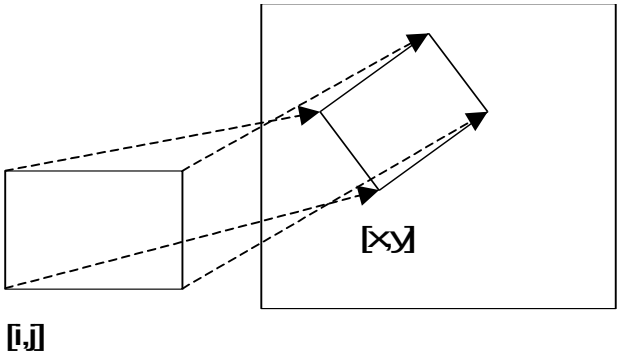
Nechť *O* je libovolná oblast (množina složená z jedniček) v binárním obrazu, Konturou (hranicí) oblasti *O* rozumíme všechny pixely patřící této oblasti, které mají nulového d-souseda.



→ vektorově → rastrově

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. V této fázi 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).

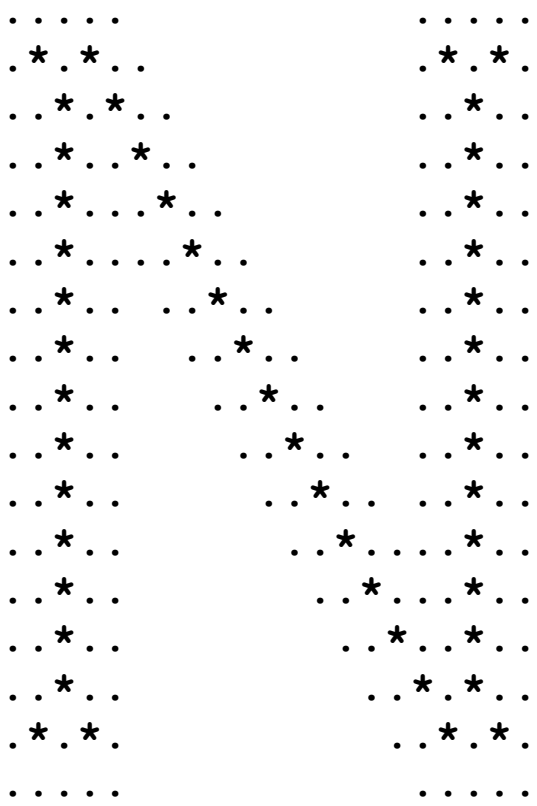
2. Transformujeme dotazový obdélník do serverové projekce, upravíme pixelový rozměr obrazu. K tomuto účelu použijeme ”přesný” převod souřadnic (**`rovina_1`**→ **`elipsoid_1`**→ **`centr_1`**→ **`centr_2`** → **`elipsoid_2`**→ **`rovina_2`**, viz. transformace souřadných systémů).

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:



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.

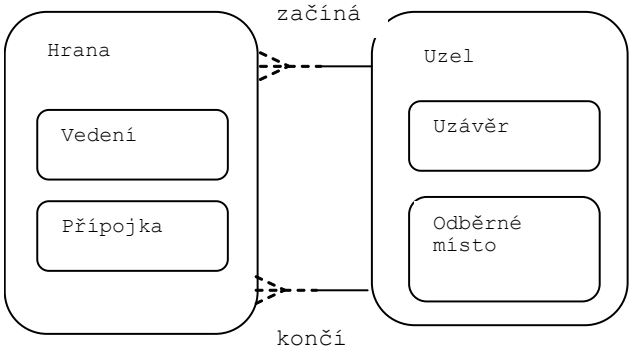
(a), (b): Alespoň jeden pixel ze skupiny pixelů A, B je nenulový

(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.

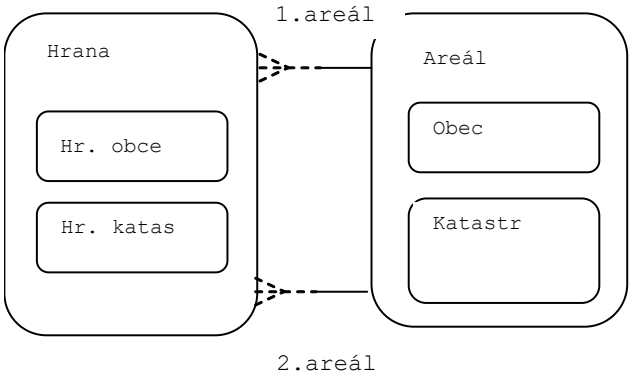
7. Topologické úlohy v GIS

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 $h \in H$ označme $w(h)$. Nechť dále každý uzel $u_i \in U$ 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 je $d(u_i,v_i) \leq w(h)$ – platí trojúhelníková nerovnost metrického prostoru E_2. Potom pro libovolné uzly $u_0, v \in U$, následující postup vede k nalezení minimální cesty z u_0 do v.

1. Inicializace: Pro každý uzel $u_i \in U$, $u_i \neq u_0$ položme $d\_path_i=\infty$, $d\_path_0=0$, a položme každý uzel $u_i \in U$ $c\_path_i=NULL$.

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 $u_k \in U$ takové, že existuje hrana $h_k \in H$, $h_k=(u_i,u_k)$ (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 dimenzionálních 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) $[x_i^p, y_i^p, z_i^p], i = 1,..,n$ jsou body vstupníbody

3) Úloha: Pro body $[x_i^p, y_i^p, z_i^p], i = 1,..,n$ 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 $[x_i, y_i, z_i] = [x_i^p - \bar{x}, y_i^p - \bar{y}, z_i^p - \bar{z}], i = 1,..,n$.
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) Použijeme metodu nejmenších čtverců:

$$r = \sum (ax_i + by_i + cz_i + d)^2 = min$$

7) Derivujeme r z 6) podle a, b, c a d a dostaneme soustavu rovnic (4x4):

$$\frac{\partial r}{\partial a} = \frac{\partial r}{\partial b} = \frac{\partial r}{\partial c} = \frac{\partial r}{\partial d} = 0$$

Maticově:

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

8) 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. To nastane pouze v případě, jsou-li body na vstupu v jedné (obecné) rovině. O triviální řešení ale zájem nemáme, zajímá nás co ‚nejlepší' netriviální řešení.

9) 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 $c = 1$. Ze 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

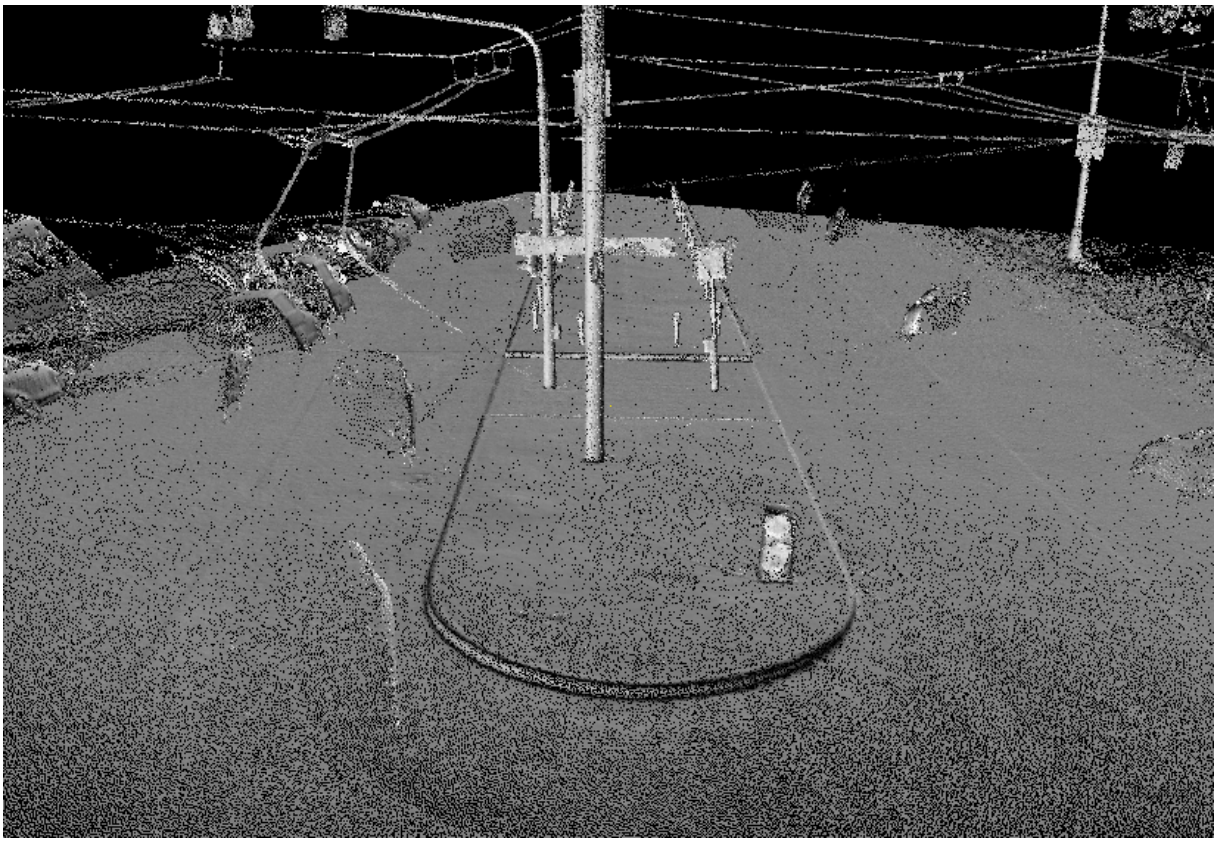
Mračna bodů jsou moderním zdrojem pro získávání geometrické (geograficky vztažené) informace. Vznikají snímáním (leteckým/pozemním) zemského povrchu laserovými scannery, výsledkem jsou 3D body s doplňující informací (intensita odrazu, barva). Jsou definovány standardy formátů, ve kterých jsou mračna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

Zobrazení mračna bodů v centrální projekci

8.5. STÍNOVÁNÍ 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 celkové kvadratické časové složitosti 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 $\boldsymbol{A}$ čtvercová matice, vlastním vektorem $\boldsymbol{u}$ nazveme takový vektor, pro který platí:

$$\boldsymbol{Au = \lambda u}$$

kde $\boldsymbol{\lambda}$ je číslo, které nazýváme vlastním hodnotou matice $\boldsymbol{A}$.

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

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).

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

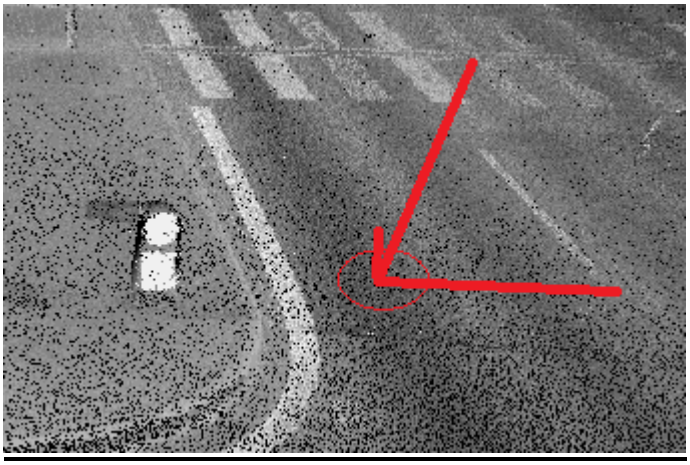
Položme (viz 8. 2. 8):

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

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 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

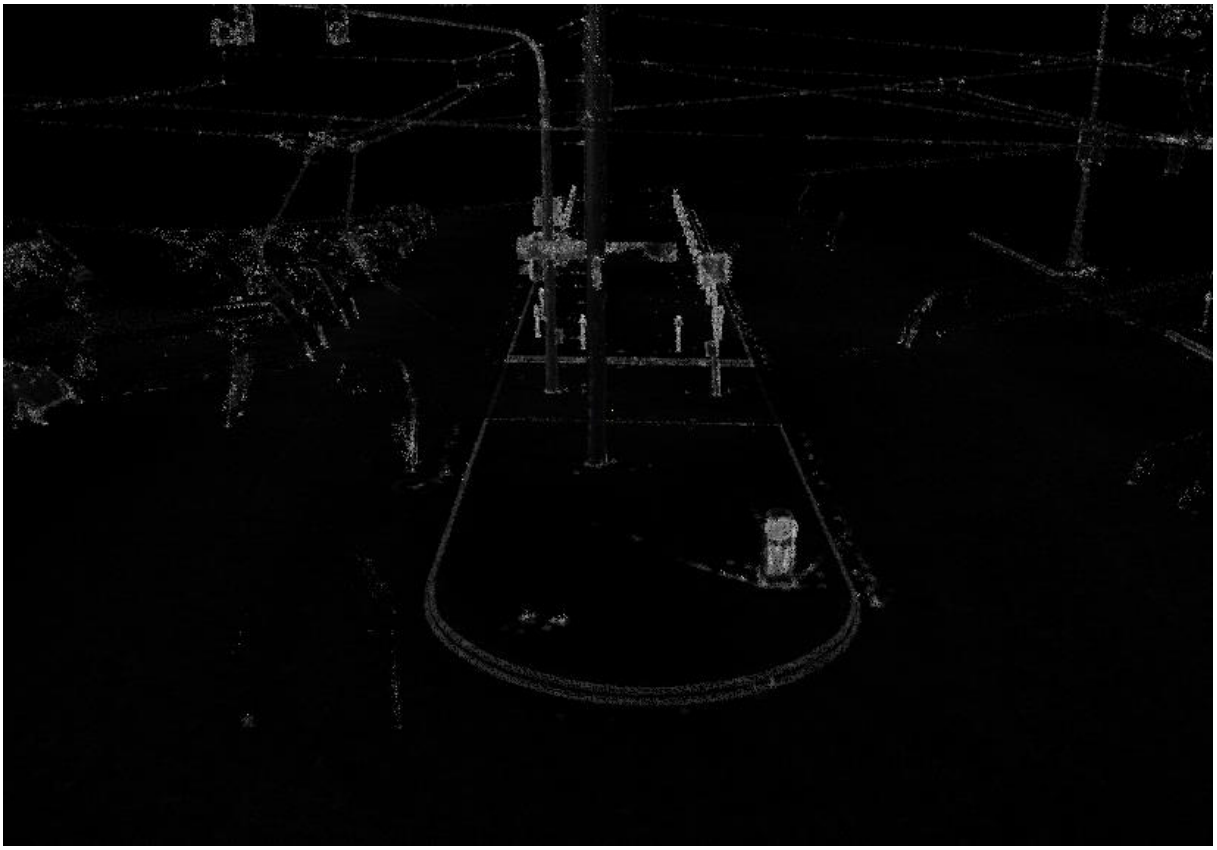
8.7. VARIANCE POVRCHU

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

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

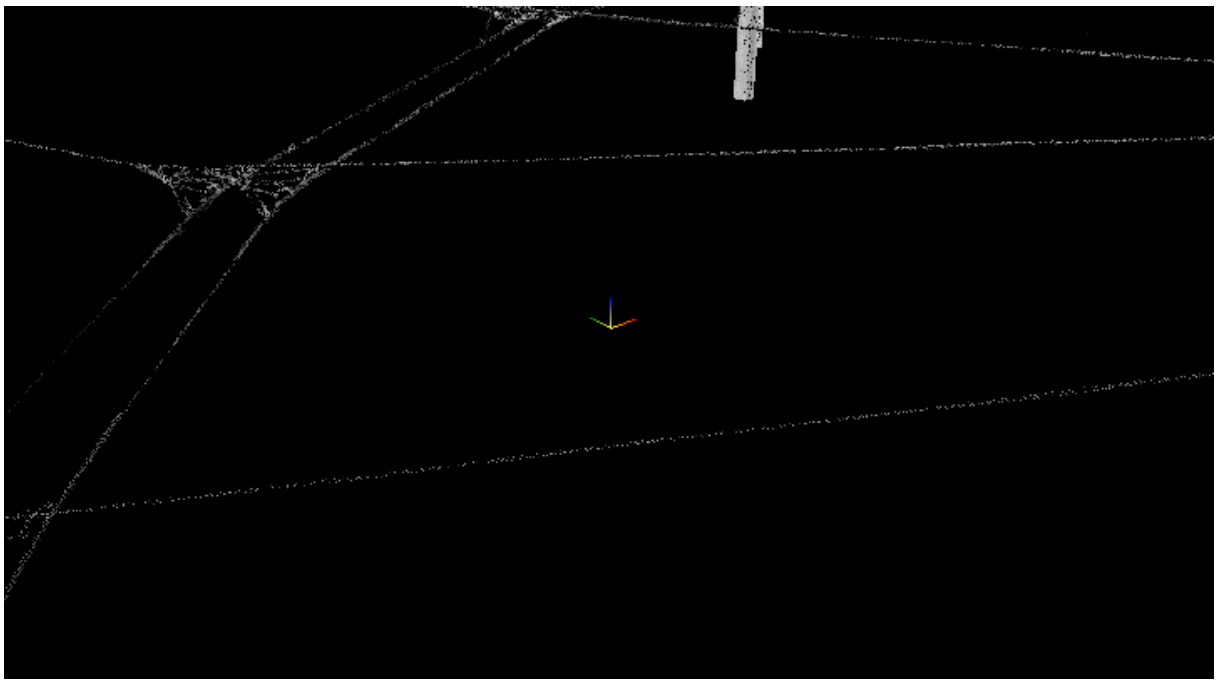


Vizualizace variance povrchu – čím světlejší, tím větší míra nerovnosti

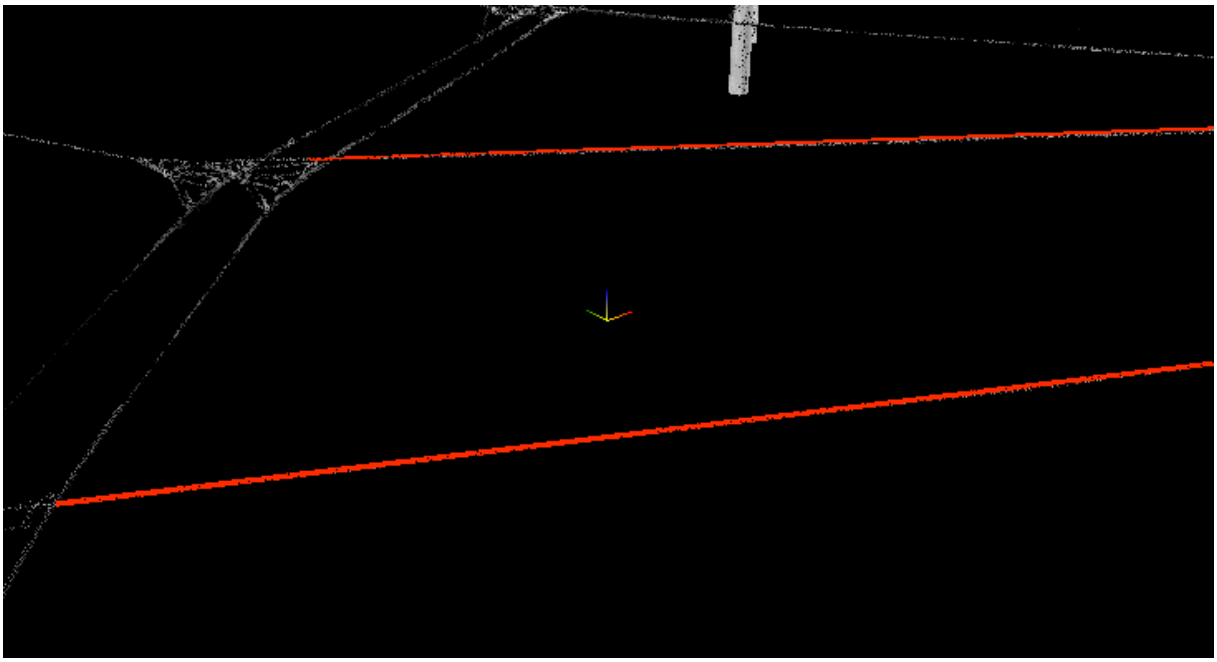
8.8. AUTOMATICKÉ TRASOVÁNÍ LINIÍ V MRAČNU BODŮ

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 je 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



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