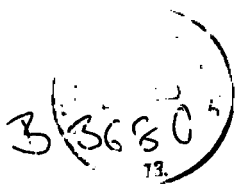
Dr. Katona Endre

Automatikus térkép-interpretáció

PhD értekezés

Szegedi Tudományegyetem

2000



Tartalomjegyzék

1. Bevezetés

Ha a szakember ránéz egy térképre, értelmezni tudja annak struktúráját, jelkulcsait, egyszóval interpretálja a térképet. Kérdés, hogy egy szkennerrel digitalizált, raszteres térképi állományon egy számítógépes program meg tudja-e tenni ugyanezt?

A fenti probléma különleges gyakorlati jelentőséggel bír. Világszerte igen sok munkaórát töltenek papírtérképek digitalizálásával, vagyis azzal, hogy az analóg nyersanyagot vektoros digitális formára alakítják, ezzel lehetővé téve annak térinformatikai felhasználását. Az eljárás vagy digitalizáló táblával történik, vagy a szkennelt térképet képernyőn rajzolják át vektoros formátumra.

Ezt az eljárást gyakran *vektorizálás*nak, vagyis *raszter-vektor konverzió*nak nevezik, pedig ez megtévesztő. Itt nem egyszerű formátum konverzióról van szó (az rutinfeladat lenne), hanem a térképet *interpretálni* kell, lokális és globális struktúráját felismerni, csak ennek alapján készíthető el a megfelelő vektoros állomány.

Az *automatikus térkép-interpretáció* olyan számítógépes eljárást jelent, amely minimális humán támogatással értelmezni tudja a térképet, és így automatikusan előállítja annak helyesen strukturált vektoros megfelelőjét. Ilyen eljárások kidolgozása jelentős anyagi haszonnal kecsegtet, ezért a világon számos kutatóhely és fejlesztő cég foglalkozik a kérdéssel. A munka eredményeként előálló szoftverek általában nagy bonyolultságú, költséges rendszerek, amelyek ugyanakkor csak bizonyos térképtípusokra és bizonyos feltételek mellett működnek hatékonyan.

A térkép-interpretációt az általános *dokumentum elemzés* (document analysis) szakterületbe szokták sorolni. Ezen belül több olyan alkalmazási terület van, amely többé-kevésbé kapcsolódik a térképek feldolgozásához:

– *Kézzel írott ill. nyomtatott karakterek felismerése* (OCR = Optical Character Recognition). Itt igen bőséges szakirodalom áll rendelkezésre, elég, ha csak az ICDAR konferenciákra és az IJDAR folyóiratra utalunk (International Conference/Journal on Document Analysis and Recognition). A publikált eljárások többsége tisztán szöveges dokumentumok (esetleg képekkel illusztrált dokumentumok) feldolgozására készült, így térképi feliratok felismerésére csak elvétve alkalmazhatók.

– *Nyomtatványok, űrlapok feldolgozása* (form analysis). Itt már fellép a vonalak és szövegek szétválasztásának igénye, de lényegesen speciálisabb formában, mint térképek esetén.

– *Műszaki rajzok feldolgozása.* Ez a terület áll legközelebb a térkép-interpretációhoz, de a műszaki rajzok specialitásai (méretvonalak, körívek, stb.) miatt az itt elért eredmények is csak korlátozottan hasznosíthatók térképek esetén.

– *Térkép-interpretáció.* A publikált megoldások elsősorban a nagy tömegben előforduló, nagyméretarányú térképek feldolgozására törekszenek.

Jelen dolgozat az automatikus térkép-interpretáció témakörének egészét kívánja áttekinteni, a terület szerteágazó volta miatt azonban mélységben csak egyes részterületek kifejtésére vállalkozhattunk. A hangsúlyt *saját térkép-interpretáló rendszerünkben, a MAPINT-ben* alkalmazott eljárások kifejtésére tettük.

A *2. fejezet* a raszter-vektor konverzió kérdésével foglalkozik, amelyen itt felismerés nélküli, nyers vektorizálást értünk. Először a szakirodalomban megjelent nagyszámú vektorizáló eljárásról adunk áttekintést, majd saját, a MAPINT rendszerben alkalmazott eljárásunkat ismertetjük.

A *3. fejezet* bemutatja a térkép-interpretáció fontosabb irányzatait néhány konkrét rendszer ismertetésén keresztül.

A *4. fejezet* az interpretációnál alkalmazható adatmodellek és indexelési technikák vizsgálatával foglalkozik. Az *5. fejezet*ben saját adatmodellünket, a DG (Drawing Graph) modellt mutatjuk be előbb elvi (logikai) majd implementációs (fizikai) szinten.

A *6. fejezet* részletesen ismerteti saját rendszerünket, a MAPINT-et. Mivel a rendszer jelenlegi változata elsősorban hazai kataszteri térképek feldolgozását támogatja, így a fejezet ezen térképállomány áttekintésével indul, majd az egyes feldolgozási lépések algoritmusainak ismertetése következik. A *7. fejezet*ben bemutatjuk azt a projektet, amelynek keretében a MAPINT illesztésre került a földhivatali adatfeltöltés folyamatába.

A domborzat a térképeken általában külön rétegben (azaz eltérő színnel) szintvonalas ábrázolásban jelenik meg, amelyet speciális jelkulcsi elemek egészítenek ki. Ennek interpretálásával foglalkozik a *8. fejezet*. Kérdés azonban, hogy a domborzati réteg vektorizálása, jelkulcsok felismerése tekinthető-e a domborzat teljes körű interpretációjának. Mivel szintvonalak között semmilyen magassági információnk nincs, így álláspontunk szerint nem. Ehhez terepmodell kell, amely minden pontban (vagy legalább egy rács minden pontjában) megadja a terepmagasságot, és így lehetővé teszi például a térkép 3-dimenziós perspektív megjelenítését.

A *9. fejezet* áttekinti azokat a módszereket, amelyek segítségével szintvonalrajzból digitális terepmodell állítható elő. Ezek közül a – véleményünk szerint – legpontosabb terepmodellt szolgáltató módszert, a végeselem alapú variációs spline illesztés elméleti alapjait tárgyalja részletesen a *10. fejezet*.

A *11. fejezet*ben egy teljes technológiai folyamatot ismertetünk raszteres terepmodell előállítására, a 10. fejezetben tárgyalt elméleti modell alapján. A maximális pontosság érdekében a szintvonalrajzot *nem vektorizáljuk*, hanem végig raszteres műveletek során jutunk el a nagy felbontású terepmodellhez. A fejezet végén röviden kitérünk a DDM-10 projektre, amelynek keretében az ismertetett technológiával állították elő Magyarország egész területét lefedő digitális terepmodelljét 10 x 10 méteres rácson.

Végül a *12. fejezet* a tézisszerűen foglalja össze a dolgozatban ismertetett új eredményeket.

A dolgozat interdiszciplináris területtel foglalkozik (leginkább *térinformatikai* témának tekinthető, amely a földrajz, geodézia és informatika határterületén helyezkedik el), ezért a tárgyalás során olyan fogalmakat is definiálunk, amelyek az egyes szakterületek specialistái részére evidensek. A dolgozatban szereplő apró betűs szövegrészek kiegészítő megjegyzéseket tartalmaznak, ezek nem képezik a tárgyalás lényeges részét. Ezzel a tipográfiai megoldással az anyag áttekinthetőségét igyekeztünk növelni.

2. Raszter-vektor konverzió

Papírtérképek átalakítása digitális, vektoros (CAD) formátumra az alábbi módokon lehetséges:

1. Manuális vektorizálás. A vektoros rajz előállítását teljes egészében az operátor végzi. Két fő módja van:

1.1. Digitalizáló tábla használata. Előnye a viszonylag szerény hardver szükséglet. Hátránya viszont, hogy az operátornak felváltva kell a táblára és a monitorra nézni, továbbá az, hogy az adatbevitel pontossága és teljessége csak nehezen ellenőrizhető.

1.2. Heads-up vektorizálás. A rajzot szkennerrel digitalizálják, és a nyert raszterképet a képernyőn manuálisan vektorizálják – ugyanúgy, mint a digitalizáló tábla esetén. Itt a digitalizáló tábla fent említett hátrányai megszűnnek: az operátor a képernyőn egymásra vetítve látja a szkennelt raszteres és az általa létrehozott vektoros rajzot, így a pontosság és teljesség könnyen ellenőrizhető. Az eljárás kb. kétszer hatékonyabb a digitalizáló táblánál (Falk, Voloncs 1992). Viszont A0-ás szkenner beszerzését kevesen engedhetik meg maguknak, ezért a legtöbb felhasználó szolgáltatókra van utalva a szkennelés tekintetében.

2. Félautomatikus vektorizálás. Annyiban különbözik a heads-up digitalizálástól, hogy a szoftver a képernyőn automatikusan követi a vonalakat és képezi le vektorokra, de elágazásnál megáll, és az operátor irányítására vár: hogyan tovább. Ez számos szoftver jellemző szolgáltatása, itt csak a MicroStation Descartes rendszert említjük. Az eljárás például olyankor előnyös, amikor nem kell a teljes rajzot vektorizálni. Az előállított eredmény azonban rendszerint manuális korrigálásra szorul.

3. Automatikus vektorizálás. Itt a szoftver operátori beavatkozás nélkül állítja elő a vektoros rajzot. Az eljárás viselkedése általában számos paraméter beállításával szabályozható, a piacon kapható vektorizáló rendszerek több-kevesebb interpretáló képességgel is rendelkeznek (szaggatott vonalak felismerése, szimbólumok elkülönítése, stb.). Az előállított vektoros rajz itt is manuális javításra szorul.

A továbbiakban csak automatikus vektorizáló eljárásokkal foglalkozunk.

2.1. Vektorizáló algoritmusok áttekintése

A szakirodalomban nagyszámú vektorizáló eljárás jelent meg, ezekről jó áttekintést ad Wenyin és Dori (1999). Megállapításuk szerint a vektorizáló eljárások többsége az alábbi három lépésből áll:

1. A vonalak középtengelyének meghatározása. Ez történhet vékonyítással (vázképzéssel), vagy más módon.
2. Vonalkövetés a középtengely mentén. A vonal ekkor már elemi vektorok sorozataként is leírható.
3. Poligonalizálás. Kritikus pontok (töréspontok) keresése a görbe vonalakon, ezek lesznek a végleges vektorok szögpontjai.

A továbbiakban a fontosabb vektorizálási módszereket tekintjük át. Kizárólag bináris képek feldolgozását vizsgáljuk, amelyek fehér alapon fekete vonalrajzot tartalmaznak. Megállapodunk, hogy 0 érték jelöli a fehér (háttér)pontot és 1 a fekete (alakzat)pontot.

2.1.1. Vékonyításon alapuló módszerek

Ezen eljárások első lépése a vékonyítás (más vázképző eljárás), erre számos módszer ismert (Lam és tsai 1992). Rendszerint olyan iteratív eljárást alkalmaznak, amely pixelek törlésével a vonalat addig "hámozza" mindkét oldalról, amíg egy pixel vastag vonal marad. Az eljárás meg kell hogy őrizze a topológiát, vagyis összefüggő alakzatok vékonyítás után is összefüggők maradnak.

A vékonyított rajzon már könnyen meghatározhatók a végpontok és elágazási pontok: minden pixelhez 8 szomszédot feltételezve az előbbieknek egy, az utóbbiaknak legalább három 1-es pixel szomszédjuk van. A kétszomszédos pixelsorozatok viszont olyan vonalláncokat alkotnak, amelyek töréspontjait *poligonalizálás*sal kell megkeresni. Erre számos módszert dolgoztak ki, néhány jellemző megoldás:

– Az egyik legrégibb eljárás (Douglas, Peucker 1973) a vizsgált görbe két végpontját egyenessel köti össze, és figyeli a görbe távolságát ettől az egyenestől. Ahol a maximális távolságot méri, ott – ha a távolság meghalad adott küszöböt – töréspontot vesz fel a görbén. Az eljárást rekurzívan ismétli addig, amíg a távolság már sehol sem haladja meg az adott küszöböt.

– Hasonlóan dolgozik Wall és Danielsson (1984) módszere, de távolság helyett az egyenes és a görbe közötti terület vonalegységre eső értékét figyeli. Az eljárás hátránya, hogy a sarokpontok elcsúszhatnak, mivel a módszer néhány pixel késéssel detektálja azokat.

– Ray B. K. és Ray K. S. (1992) minimális és maximális görbület figyelése alapján választ ki domináns pontokat.

– Janssen és Vossepoel (1997) a töréspontok iteratív módosításával finomítja a fenti eljárásokat.

Vékonyítás alapú vektorizálási módszerekkel a 2.2. fejezetben foglalkozunk részletesebben, saját eljárásunk ismertetése során.

2.1.2. A futam-gráf módszer

Az eljárást Di Zenzo és Morelli (1989) ötlete alapján Boatto és tsai (1992) alkalmazta olasz kataszteri térképek feldolgozására.

*Futam*nak (*run*) nevezzük a rajzon egy vonal vízszintes vagy függőleges metszetét, amelyet egy $R = (d,c,b,e)$ négyessel írhatunk le (1. ábra):

– d (direction) a futam iránya (0: vízszintes, 1: függőleges),

– c (coordinate) a futam koordinátája d-re merőlegesen ($d = 0$ esetén a függőleges, $d = 1$ esetén a vízszintes koordináta),

– b (begin) és e (end) a futam első és utolsó elemének koordinátái d irányban.

Belátható, hogy tetszőleges bináris kép információvesztés nélkül leírható futamok halmazaként. A futamokkal kapcsolatos további fogalmak:

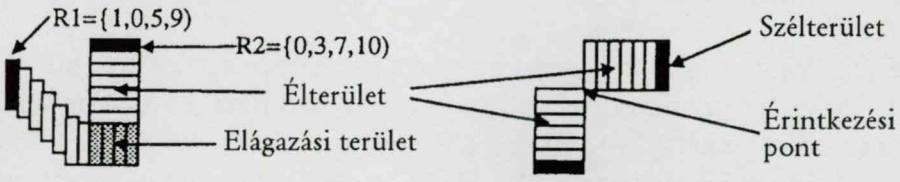
– Két futam *konjugált*, ha egymásra merőlegesek, és van közös pontjuk.

– Egy futamot *rövid*nek nevezünk, ha egyik konjugáltjánál sem hosszabb.

– Szomszédos rövid futamok maximális halmazát *élterület*nek nevezzük.

– Azt a futamot, amelynek az egyik oldalon nincs szomszédja, *szélterület*nek mondjuk.

– *Elágazási terület*nek nevezzük függőleges (rész)futamok összefüggő sorozatát, amely nem tartalmaz sem függőleges, sem vízszintes rövid futamot.



1. ábra. Futam-gráf szemléltetése (origó a bal felső sarok)

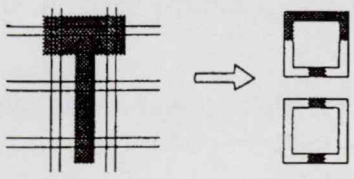
A fentiek alapján *futam-gráf*nak nevezzük a $G = (V,E)$ gráfot, ahol a szögpontok V halmazát a szélterületek és elágazási területek alkotják, az élek E halmaza pedig az élterületekből áll.

Az elágazási területek és a rövid futamok középpontjának meghatározásával nyerhető a rajz váza, amelyből a 2.1.1. pontban tárgyalt poligonalizálási eljárásokkal nyerhető vektoros adatstruktúra.

Wenyin és Dori (1999) értékelése szerint az eljárás hibája, hogy téves elágazási pontokat detektálhat futamirány változásnál és zajos vonalakon, ezért görbe vonalakat tartalmazó rajz feldolgozására nem alkalmas. Ugyanakkor Boatto és tsai (1992) sikerrel alkalmazta a módszert olasz kataszteri térképek interpretációjánál.

2.1.3. A mozaik (mesh pattern) módszer

Az eljárást Lin és tsai (1985) definiálta diagramok feldolgozására. A rajzterületet négyzetráccsal fedi le, és az egyes négyzetek határán a fekete pixelek eloszlását vizsgálja (2. ábra). Rendelkezésére áll egy mozaik adatbázis, amely jellemző mintákat tartalmaz (a konkrét alkalmazás esetén 51 ilyen minta van). Az eljárás a rajz egyes négyzeteit aszerint sorolja osztályba, hogy a mozaik adatbázis melyik mintájának felelnek meg.



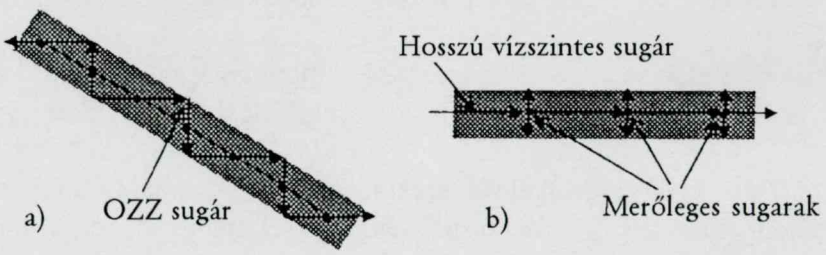
2. ábra. A mozaik módszer szemléltetése

A módszer kritikus pontja a rácsméret meghatározása. Az optimális rácsméret nagyobb a maximális vonalvastagságnál, de kisebb a minimális vonaltávolságnál. Ez a feltétel gyakran nem teljesíthető, ezért például pontozott vonalak eltűnhetnek a rajzról. Az eljárás ugyanakkor igen gyors, hiszen csak a pixeleknek egy részét (a négyzetek határát) vizsgálja.

A mozaik módszert Vaxiviere és Tombre (1992) alkalmazza műszaki rajzok feldolgozására. Az eljárást annyiban finomítják, hogy szükség esetén tovább osztják a vizsgált négyzetet.

2.1.4. OZZ vektorizálás

Az eljárást Wenyin és Dori (1996) dolgozta ki. Alapja az Orthogonal Zig-Zag (OZZ) vonalkövetési módszer: egy pixel vastag "fénysugarat" indítunk a vonal belsejében, amely a vonal határához érve mindig 90 fokkal elfordulva verődik vissza (3/a. ábra). Minden sugárszakasz középpontját feljegyezzük. Ha – például vízszintes vonal esetén – egy sugárszakasz hosszabb valamely adott küszöbnél (mondjuk 30 pixelnél), akkor megszakítjuk, és egy merőleges sugarat indítunk, ez utóbbinak középpontját jegyezzük fel (3/b. ábra). A feljegyzett értékek adják a vonalak középtengelyét.



3. ábra. Az OZZ vektorizálás alapelve

Az eljárás előnye, hogy az íveket valóban ívekként (és nem vonallánccként) detektálja, hátrányként említhető viszont a bonyolult paraméterezés: 12 értéket kell megfelelően beállítani (Janssen, Vossepoel 1997).

2.1.5. Hough-transzformáción alapuló vektorizálás

Az eljárás alapgondolata, hogy egy egyenesbe eső pontsorozatokat keresünk. Ehhez használható a Hough-transzformáció (leírását lásd például Gonzalez, Wintz (1987)), amelynek lényege a következő.

Az x, y koordinátatengelyekkel adott *normál tér*, és az a, b koordinátatengelyekkel adott ún. *paraméter tér* egyenesei és pontjai között létesítünk megfeleltetést a következőképpen. A normál tér $y = a \cdot x + b$ egyenlettel adott egyenesének a paraméter tér (a, b) pontja felel meg, és fordítva: a paraméter tér $b = -x \cdot a + y$ egyenesének a normál tér (x, y) pontja felel meg.

Következésképp, ha n pont a normál térben egy egyenesbe esik, akkor a nekik megfelelő n egyenes a paraméter térben egy ponton megy át.

Diszkrét esetben mind a normál, mind a paraméter tér mátrix lesz, jelölje ezeket N és P. Ekkor a Hough-transzformáció a következőképp számítható. Kezdetben a P mátrix minden eleme 0. Az N mátrix minden 1 értékű pixeléhez meghatározzuk a paraméter térben a megfelelő egyenest, és P azon elemeit, amelyeken az egyenes átmegy, 1-gyel inkrementáljuk. Az eredményként adódó P mátrixban egy elem értéke n, ha n egyenes megy át rajta. Ezen mátrixelemnek a normál térben egy egyenes felel meg, amely n 1-es értékű ponton megy át.

A Hough-transzformáció alapján történő vektorizálás (Dori 1997) előnye, hogy szakadozott, zajos egyenesek felismerésére is képes. Ugyanakkor pontatlan, a rajzot számottevő lokális torzulásokkal és hibákkal képezi le, ezért a gyakorlatban nem terjedt el.

2.2. A MAPINT rendszer vektorizáló eljárása

Saját vektorizáló eljárásunk, amelyet a MAPINT térkép-interpretáló rendszerben (6. fejezet) alkalmazunk, a *vékonyítás alapú* megoldások közé tartozik. Az eljárás eredményeként egy VECT nevű vektorfájl áll elő, amely egyszerűen felsorolva tartalmazza a vektorokat, ebből a későbbiekben hozunk létre topológikus adatstruktúrát (5.3. fejezet).

2.2.1. Vékonyítás

Szokásos pixel alapú, iteratív vékonyító algoritmust alkalmazunk (Pratt, Kabir 1985). *Paraméterként* a maximális vonalvastagságot kell megadni, az iterációs lépések száma ennek felével kell hogy megegyezzen. Ennél nagyobb lépésszám alkalmazása nem csak fölöslegesen növeli a futási időt, de eltünteti az esetleges tömör alakzatokat is (lásd alább).

2.2.2. Belső pixelek törlése

Törlünk minden olyan 1-es pixelt, amelynek mindnégy ortogonális szomszédja 1-es értékű. Ezzel az esetleges tömör alakzatok kontúrja marad meg, és kerül majd vektorizálásra.

Megjegyezzük, hogy a tömör alakzatok morfológiai műveletek segítségével is elkülöníthetők, Nagasamy és Langrana (1990) például ilyen módon az eredeti képből két képet hoz létre (vonalrajz, illetve tömör alakzatok), amelyeket a továbbiakban külön kezel. A fenti eljárás ennél lényegesen egyszerűbb és gyorsabb, és – hacsak nincsenek nagyon vastag vonalak a rajzon – a morfológiai eljárással lényegében egyenértékű eredményt ad.

2.2.3. Elemi gráf létrehozása

Tekintsük az 1-es pixeleket egy gráf szögpontjainak, a szomszédsági viszonyokat pedig éleknek: két szögpont akkor van összekötve egymással, ha 8-szomszédság szerint szomszédosak. Ilyen elemi gráfot alkalmaz Suzuki és Ueda (1991) vékonyításra, esetünkben csak a vektorizálás további részét alapozzuk ezen modellre. A gráfot – Moore (1992)

megoldásához hasonlóan – *szomszédmaszkok* segítségével kódoljuk: a kép minden egyes p pixelét egy b byte-tal helyettesítjük:

– Ha $p = 0$, akkor $b := 0$.

– Ha $p = 1$, de mind a 8 szomszédja 0 (izolált pont), akkor $b := 0$, és egy nulla hosszúságú vektort veszünk fel a létrehozandó VECT fájlba. (Az izolált pontok származhatnak zajból, de jelölhetnek ékezetet vagy tizedespontot is (a vékonyítás után nagyobb alakzatok is izolált ponttá zsugorodhatnak), ezért nem lehet őket egyszerűen törölni.)

– Ha $p = 1$, és van nemnulla szomszéd, akkor b a 8 szomszéd értékét tartalmazza:

$$b := (N, NE, E, SE, S, SW, W, NW)$$

ahol N az északi, NE az északkeleti, stb. szomszédot jelöli.

Ezután az elemi gráfon *élritkítás*t végzünk az alábbiak szerint. Ha az *A, B, C* pixelek

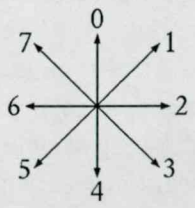
A
B C

elhelyezkedésűek, akkor az *A* és *C* közötti átlós élt töröljük. (Ez természetesen a fenti elhelyezkedés mindnégy elforgatottjára is érvényes.) A törlés nem változtatja meg a topológiát, hiszen az *A* és *C* közötti kapcsolat az *AB* és *AC* éleken keresztül továbbra is megmarad. A törlés a szomszédmaszkok lokális vizsgálatával elvégezhető. Élritkításra azért van szükség, hogy elkerüljük a redundáns csomópontokat (lásd alább). Hasonló megoldást alkalmaz Suzuki és Yamada (1990) a MARIS rendszerben (3.1. fejezet).

Csomópontok meghatározása. Az elemi gráf minden olyan szögpontját, amelyből kettőnél több vagy kevesebb él indul ki, csomópontnak tekintjük, és megjelöljük. A csomópont feltétel a szomszédmaszkon könnyen ellenőrizhető.

2.2.4. Vonalkövetés

Az elemi gráfot járjuk be csomóponttól csomópontig. Az élsorozatok a szokásos lánckódolásnak megfelelően interpretálhatók: minden egyes élt a 4. ábra szerinti iránykóddal helyettesítve egy számsorozat adódik.



4. ábra. A lánckódolásnál használt 8 iránykód

A vonalkövetésnél addig haladunk, amíg az érintett lánckód bizonyos értelemben egyenesnek tekinthető, ha ez már nem teljesül, töréspontot veszünk fel. Ehhez a következőt vesszük alapul:

1. Állítás (Freeman 1974, lásd még Álló és tsai 1989). Egy lánckód akkor és csak akkor reprezentál racionális meredekségű digitális egyenest, ha

– legfeljebb kétféle iránykód fordul elő benne, és ezek szomszédosak,

– kétféle iránykód, d_1, és d_2 esetében a d_1 irányú szakaszok többeleműek is lehetnek (főirány), a d_2 irányúak viszont mindig egyeleműek (mellékirány),

– a d_2 iránykódok eloszlása a láncon a lehető legegyenletesebb.

A fentiek alapján fogalmazzuk meg az alábbi definíciókat.

1. Definíció. *Homogén egyenes*nek nevezzük azt a lánckódot, amely $(d_1^n d_2)^m$ alakban írható, vagyis n főirány után 1 mellékirány következik, és mindez m-szer ismétlődik ($n \geq 0$, $m \geq 0$). A homogén egyenes nyilván a digitális egyenes speciális esete.

2. Definíció. *Homogén egyenesszakasz*nak nevezzük a $d_1^u d_2 (d_1^n d_2)^m d_1^v$ alakban írható lánckódot, ahol $n \geq 0$, $m \geq 0$, $0 \leq u \leq n$, $0 \leq v \leq n$.

A vonalkövetést egy (nem végesállapotú) *automatával* végezzük, amely bemenőjelként fogadja az egymás utáni iránykódokat, és mindaddig működik, amíg azok eleget tesznek a digitális egyenesszakasz feltételnek. Az automata *állapot*ának komponensei:

– s: felismerési fázis ($s = 0, ..., 6$),

– d_1, d_2: fő- és mellékirány kódja,

– u, n, v: a felismert főirányú szakaszok hossza.

Induláskor $s = 0$, d_1 és d_2 definiálatlan, $u = n = v = 0$.

Az automata *átmenetfüggvény*ét az alábbiak szerint definiáljuk (e az aktuális bemenőjelet (iránykódot) jelöli):

s=0 fázis: kezdőállapot. Átmenet:
$d_1 := e$, $u := 1$, $s := 1$

s=1 fázis: d_1 definiált, d_2 még definiálatlan. Átmenet:
Ha $e = d_1$, akkor $u := u+1$, $s := 1$
Ha e szomszédos d_1-gyel, akkor $d_2 := e$, $s := 2$
Egyéb esetekben $s' := 6$

s=2 fázis: d_1 és d_2 definiált. Átmenet:
Ha $e = d_1$ akkor $n := 1$, $s := 3$
egyébként ha $e = d_2$ és $u = 1$ akkor $u := 0$, $n := 2$, d_1 és d_2 felcserélendő, $s := 3$
Egyéb esetekben $s := 6$

s=3 fázis: d_1 a főirány, d_2 a mellékirány. Átmenet:
Ha $e = d_1$, akkor $n := n+1$, $s := 3$
Ha $e = d_2$ és $u \leq n$, akkor $s := 4$
Egyéb esetekben $s := 6$

s=4 fázis: az utolsó él mellékirányú volt. Átmenet:
Ha $e = d_1$, akkor $v := 1$, $s := 5$
Egyéb esetekben $s := 6$

s=5 fázis: az utolsó él főirányú volt. Átmenet:
Ha $e = d_1$ és $v < n$, akkor $v := v+1$, $s := 5$
Ha $e = d_2$ és $v = n$, akkor $s := 4$
Egyéb esetekben $s := 6$

s=6 fázis: végállapot. Az automata megáll.

2. Állítás. Egy $e_1, ..., e_n$ élsorozat akkor és csak akkor homogén egyenesszakasz, ha bejárása során a fenti automata nem kerül végállapotba.

Bizonyítás. Legyen $e_1, ..., e_n$ homogén egyenesszakasz, és tegyük fel, hogy e_1 főirányú. Ekkor a bemenő jelsorozat $f_1m_2f_3m_4f_5m_6....$ felépítésű, ahol f_i a főirányú, m_i a mellékirányú szakaszokat jelenti. Az automata konstrukciójából látható, hogy az egyes feldolgozási fázisok az egyes szakaszoknak az alábbiak szerint felelnek meg:
– s=1: f_1 szakasz feldolgozása
– s=2: m_2 szakasz feldolgozása
– s=3: f_3 szakasz feldolgozása
– s=4: m_4, m_6, stb. szakaszok feldolgozása
– s=5: f_5, f_7, stb. szakaszok feldolgozása
Ha a bemenő jelsorozat mellékiránnyal kezdődik, akkor f_1 elmarad, és az automata feldolgozási fázisai a következőképp alakulnak:
– s=1: m_2 szakasz feldolgozása
– s=2,3: f_3 szakasz feldolgozása
– s=4: m_4, m_6, stb. szakaszok feldolgozása
– s=5: f_5, f_7, stb. szakaszok feldolgozása
Az automata átmenetfüggvényéből jól látható, hogy amint a homogenitási feltétel sérül, az automata s=6 végállapotba megy át. Ebből az állítás mindkét irányban következik.

Ezek után a vonalkövetés menete a következő. Az elemi gráfon csomóponttól csomópontig haladunk a fenti automatával. Ha az automata végállapotba kerül, akkor a detektált egyenesszakasznak megfelelő vektort felvesszük egy munkatömbbe, és így haladunk tovább egészen addig, amíg a láncot lezáró csomópontot el nem érjük. A vonalkövetés során érintett éleket töröljük az elemi gráfból. Az eljárás addig tart, amíg valamennyi él el nem fogy.

Megjegyzendő, hogy a fenti automata az 1. állításnak megfelelő digitális egyeneseket általában csak több szakaszban ismeri fel, ezen szakaszok egyesítése az alább tárgyalt lánc optimalizálási művelet feladata.

Műveletigény. Az automata konstrukciójából következik, hogy a vonalkövetés lineáris időben végezhető.

2.2.5. Vonallánc optimalizálás

A vonalkövetés eredményeként két csomópont között egy $P_1, P_2, ..., P_n$ pontsorozattal adott vonallánc keletkezik (P_1, P_n a csomópontok). Az optimalizálás célja, hogy a közel egy egyenesbe eső szakaszokat egyesítsük, ezzel csökkentve a generált vektorok számát.

Paraméter: az eljáráshoz egy T tolerancia értéket kell megadni.

Algoritmus:

Az eljárás során törlés jelzéssel látunk el egy P_i pontot, ha a $P_{i-1}P_{i+1}$ szakaszok egyesíthetők. A pontok végleges törlésére csak az eljárás végén kerül sor.

1. Végigmegyünk a P_2, ..., P_{n-1} pontokon (pontosabban azokon, amelyek még nincsenek törölve). Minden egyes P_i pontra a következőket végezzük:

– megkeressük legközelebbi P_{i-j} és P_{i+k} nem törölt pontokat.

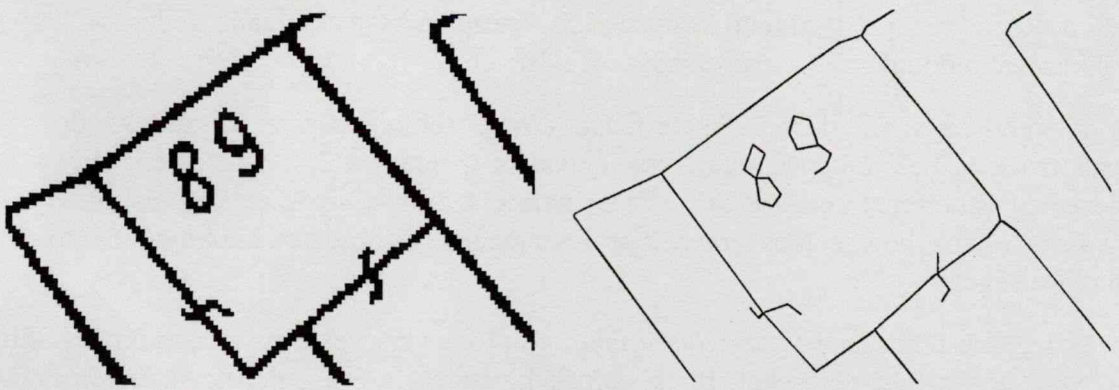
– megnézzük, hogy P_{i-j} és P_{i+k} közötti összes pont (törölteket is figyelembe véve) közül melyik van a legmesszebb a $P_{i-j}P_{i+k}$ egyenestől, a távolság értéke legyen max_i.

2. Vesszük a kapott max_i értékek minimumát, legyen ez *min*. Ha $min < T$, akkor töröljük azt a P_i pontot, amelyre a minimális max_i érték adódott, és az eljárást 1-gyel folytatjuk. Ha $min \geq T$, akkor az eljárás véget ér.

A műveletigény a pontok helyzetétől függően $O(n)$ és $O(n^2)$ között változik, de mivel a vizsgálandó pontok száma nagyságrenddel kevesebb, mint a vonalkövetésnél, így ennek nincs különösebb jelentősége.

2.2.6. Összefoglalás

A MAPINT rendszer vektorizáló algoritmusa szokásos, vékonyítás alapú eljárásnak tekinthető, amelyben egyedi megoldást a vonalkövetési technika jelent. A teljes vektorizálási folyamatban a vékonyítás ideje a meghatározó, ehhez képest az összes többi művelet elhanyagolható idejű. Az algoritmus által előállított nyers vektoros rajz a 5. ábrán látható.



5. ábra. Nyers vektorizálás. Baloldalt a szkennelt raszterkép, jobboldalt a vektorizálás eredménye

Megjegyezzük, hogy a MAPINT rendszernél a hangsúlyt a felismerő algoritmusokra helyeztük. A vektorizálás szerepe itt csupán egy *nyers vektoros adatstruktúra* előállítása, amely alapanyagul szolgál a felismeréshez. Ezért a vektorizálás során nem törekedtünk például a sarkoknál vagy T-elágazásoknál fellépő torzulások kiküszöbölésére (5. ábra), hanem ezt a felismerési fázisra hagytuk (6.4.9. alfejezet).

3. Interpretáló rendszerek áttekintése

Az utóbbi évtizedben számos térkép-interpretációs rendszerről számolt be a szakirodalom. Ebben a fejezetben olyan rendszereket mutatunk be, amelyek nem csak egy-egy részfeladat megoldását, hanem adott térképtípus többé-kevésbé teljes körű interpretációját tűzik ki célul, és valamennyi fontosabb kutatási/fejlesztési irányt lefednek. Ezek többsége *adott térképtípusra specializált.* Jellemzően minden országban a nagy tömegben előforduló, nagyméretarányú térképállományokat célozzák meg, amelyek a következők:

– *Földmérési alaptérképek,* más néven *kataszteri térképek.* Elsődleges céljuk a földhivatali ingatlan nyilvántartás, vagyis az épületek és földrészletek (telkek) geometriai viszonyainak ábrázolása. Méretarányuk jellemzően 1:500 és 1:5000 között változik.

– *Topográfiai térképek.* Általános célú térképek, amelyek a domborzat, vízrajz, út- és vasúthálózat, települések ábrázolását tartalmazzák. Méretarányuk 1:10 000 és 1:100 000 között mozog.

– *Közműtérképek.* Az egyes közművállalatok vezetékrendszerét és szerelvényeit ábrázolják (víz, gáz, villany, telefon, kábeltelevízió, stb.). Jellemző méretarányuk 1:500.

Az országonként eltérő térképi szabványok miatt az interpretáló rendszerek többsége "nemzeti sajátosságokat" mutat.

Természetesen számos kutató törekszik *univerzális megoldásokra.* Ezt az irányt képviseli a MAGELLAN rendszer (3.6. fejezet), amely a térkép jelkulcsainak automatikus megtanulására épül. Az univerzalitásra törekvés másik irányát a szemantikus háló modell (4.5. fejezet) alkalmazása jelenti, ahol egyfajta tudásbázisban rögzítik a térkép felépítési szabályait. Erre épül a 3.4. fejezetben bemutatott rendszer. A gyakorlatias megközelítések közül a RoSy rendszert mutatjuk be a 3.5. fejezetben, itt speciális fejlesztő környezet kialakításával oldották meg a testreszabás lehetőségét.

A rendszerek általában fekete-fehér anyag feldolgozására készültek, de többen próbálkoznak színes térképek interpretációjával is (például a 3.4. fejezetben vagy a 8.2.2. alfejezetben ismertetett rendszerek). Véleményünk szerint a színszétválasztás annyi többlet hibát visz a rajzba, hogy színes térképek professzionális feldolgozása csak nyomdai fóliánként külön lehetséges.

A legtöbb rendszer *vektorizálásra épül*: először nyers vektoros adatstruktúrát állít elő, majd ezen végez felismeréseket. Itt is vannak kivételek: a 3.6. fejezetben *tisztán raszteres* feldolgozásra is látunk példát.

3.1. A MARIS rendszer

A rendszer neve a *MAp Recognition Input System* rövidítése, *japán nagyméretarányú térképek* feldolgozására készült (Suzuki, Yamada 1990). Az első komplex, részletesen dokumentált rendszerek közé tartozik.

A térképek 1:2500 méretarányúak, 60 x 80 cm keretméretűek, és jellemzően az alábbi rétegeket tartalmazzák:

– épületek,
– szintvonalrajz,
– vasút, út és vízrajz.

A japán térképek sajátossága, hogy az épület poligonok DK irányból napfénnyel megvilágított oldalát 0.3 mm, a többi oldalát 0.1 mm vastag vonallal rajzolják. A felismerő algoritmus ezt ki is használja.

A térképeket 16 pixel/mm felbontással szkennelik, majd 70 partícióra bontva tárolják. Ezután vektorizálás, majd automatikus felismerés következik, végül manuális korrekcióval zárul a feldolgozás.

A *vektorizálás* lépései:

1. Vonalvastagság meghatározása, minden egyes pixelt címkéznek a vonalvastagság értékével.
2. Vékonyítás.
3. Elemi gráf képzése és élritkítás (v.ö. 2.2.3. alfejezet).
4. Ezután az algoritmus sorfolytonosan járja be a raszterképet. Ha csomópontot talál (amelyből nem két él indul), akkor megszakítja a bejárást, és vonalkövetést végez, majd folytatja a bejárást. A poligonalizálás módját a cikk nem ismerteti.

A vektorizálás eredményét a rendszer *három relációs adattáblában* tárolja a következő felépítésben:

– A *csomópont tábla* egy rekordja egy csomópont azonosítóját, koordinátáit, és a kiinduló vonalláncok azonosítóit tartalmazza, a raszteres 8-szomszédságnak megfelelően:
NODE (id, x, y, $line_1$, ..., $line_8$)

– A *vonal tábla* egy rekordja egy (két csomópontot összekötő) vonallánc azonosítóját, a kezdő és záró csomópontok azonosítóit, valamint a kezdő és záró vonalszegmens azonosítóit tartalmazza:
LINE (id, $node_1$, $node_2$, s_1, s_2)

– A *szegmens tábla* egy rekordja egy vonalszegmens azonosítóját, kezdő- és végpontjának koordinátáit, a szegmenst tartalmazó vonal azonosítóját, és a vonalvastagságot tartalmazza:
SEGMENT (id, x_1, y_1, x_2, y_2, line, width)

A fenti vektoros adatstruktúra elemzésére egy *szegélykövető* (border tracing) algoritmust alkalmaz a rendszer, amelynek segítségével minden vonalat "mindkét oldalán" bejár. A eljárás eredményeként önmagába záródó *külső szegély*ek (outer border) és *belső szegély*ek (hole border) keletkeznek.

Ezután a *hosszú vonalak* felismerése következik. Adott L vastagságú éleken halad végig, elágazásnál azon az élen folytatja, amelyik legkisebb szögben törik (és a szög kisebb egy adott küszöbnél). Ilyen módon különíti el a szintvonalakat, vasútvonalakat. A vonalat csak akkor fogadja el, ha összhossza nagyobb egy adott T küszöbnél.

Az épületek felismerésénél a rendszer kihasználja a japán térképekre jellemző – már említett – kétféle vonalvastagságot. Egy "belső szegélyt" akkor tekint épületnek, ha az alábbi feltételek együtt teljesülnek:

(i) Van olyan rész-poligonja, amely 0.3 mm vonalvastagságú, hosszabb adott küszöbnél, és tartalmaz egy bal vagy felső "nyitott derékszögű" csúcsot. Ez utóbbi feltételt a koordinátákból képezett 2 x 2-es determinánsokkal definiálja.

(ii) A poligon éleinek összhossza kisebb adott küszöbnél.

(iii) A poligon belső területe nagyobb adott küszöbnél.

(iv) A poligon ún. "szórtsági értéke" nagyobb adott küszöbnél. Ez azt fejezi ki, hogy az épület poligonok nem túlságosan komplexek és nem túlságosan nyújtottak.

Az épületek felismerése után az eljárás töröl minden olyan vonalat, amely csak épülethez tartozik, és nem út/utca határvonal része. Ennek módja: az épület élekhez folytatást keres a hosszú vonal felismerő algoritmussal, és ha talál olyan folytatást, amely nem része épületnek, akkor ezt a vonalat meghagyja.

A MARIS rendszert ismertető cikk érdeme, hogy részletesen dokumentálja az algoritmusok többségét. Ugyanakkor az eljárások helyenként túlbonyolítottak, például a szegélykövető algoritmus helyettesíthető lenne a jelen dolgozat 6.4.6. alfejezetében ismertetett, sokkal egyszerűbb poligon struktúra bejárással. A cikk továbbá nem foglalkozik megírások elkülönítésével és felismerésével, és a mellékelt mintatérkép is túlságosan idealizáltnak tűnik (a beépített rész nem tartalmaz szintvonalat, és fordítva).

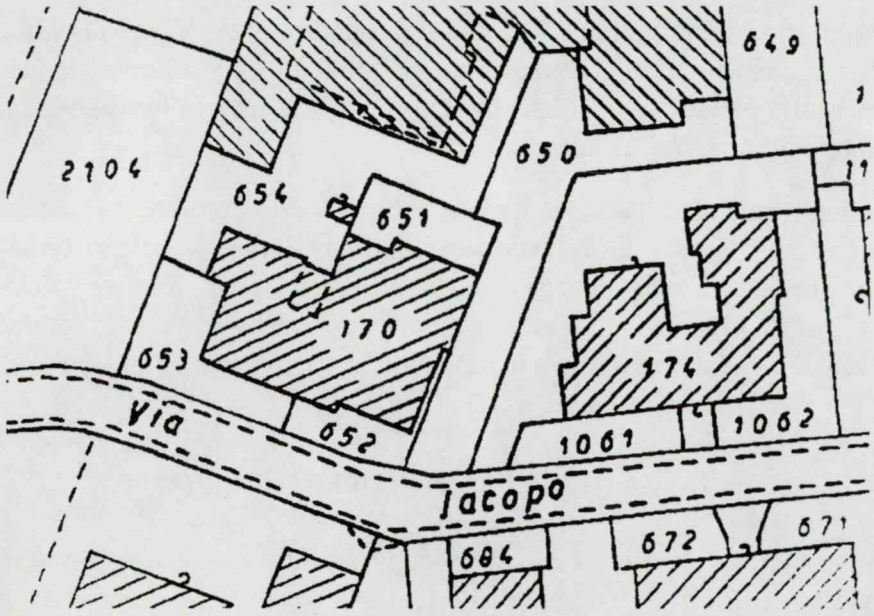
3.2. Olasz kataszteri térképek interpretációja

A Boatto és tsai (1992) által bemutatott komplex interpretáló rendszer olasz kataszteri térképeket dolgoz fel (6. ábra).

A térképek egyszínűek, a szokásos kataszteri objektumokat (épületek, földrészletek) tartalmazzák – természetesen számos más jelkulccsal kiegészítve. Sajátosságuk, hogy az épületeket vonalkázott területként ábrázolják. Méretarányuk 1:500 és 1:5000 között változik, keretméretük 100 x 70 cm.

A feldolgozás lépései a következők:

1. Előfeldolgozás. A szkennelt állományt zajszűrésnek vetik alá, majd a 2.1.2. alfejezetben ismertetett futam-gráf módszerhez hasonló eljárással egy ún. *képgráf*ot állítanak elő: a gráf szögpontjainak és éleinek összefüggő pixelhalmazok felelnek meg a raszterképen. Az így nyert képgráfon végeznek minden további felismerést.



6. ábra. Olasz kataszteri térkép részlete

2. Szegmentálás. Elkülönítésre kerülnek a folytonos vonalak, szaggatott vonalak, szimbólumok és vonalkázott területek (épületek). A részműveletek:

– A vonalkázott részek behatárolása heurisztikus algoritmussal történik (operátori segítséget igényelhet), ezután a vonalkázást törlik és csak a terület keretét vektorizálják.

– Összefüggő gráf komponenseket keresnek, méret alapján választják szét a vonalakat a szimbólumoktól. Ekkor még a szimbólumokhoz sorolódnak a szaggatott vonal darabok is.

– Vonalak vektorizálása. A vonalak minden esetben a raszteren belül maradnak, így nyilvánvalóan megfelelnek az olasz földhivatal által megkövetelt 0.4 mm pontosságnak.

3. Felismerés.

– Szaggatott vonalak felismerése: egymáshoz közeli, mindkét végén végpontban végződő vektorok keresésére épül.

– Karakter felismerés: raszteresen történik. Minden egyes karaktertípushoz egyedi jellemzőhalmazt állítanak össze. A jellemzők között típikus a (konvex burok részét képező) burkoló vonalak száma, aránya, szöge.

A karakter felismerés után már tisztán vektoros adatstruktúrával dolgozik a rendszer (vektorgráf).

4. Geometriai kapcsolatok meghatározása. Ezen fázis fő célja a földrészletek (telkek) felismerése. Egy földrészlet általában több részpoligonból áll, amelyeket kapcsolójel (kis kampószerű vonal, 6. ábra) köt össze. A feldolgozás lépései:

– Minimális köröket keresnek a vektorgráfban, ezek lesznek a részpoligonok.

– Másodlagos gráf létrehozása, amelynek szögpontjai a fenti poligonok, élei pedig a kapcsolójelek.

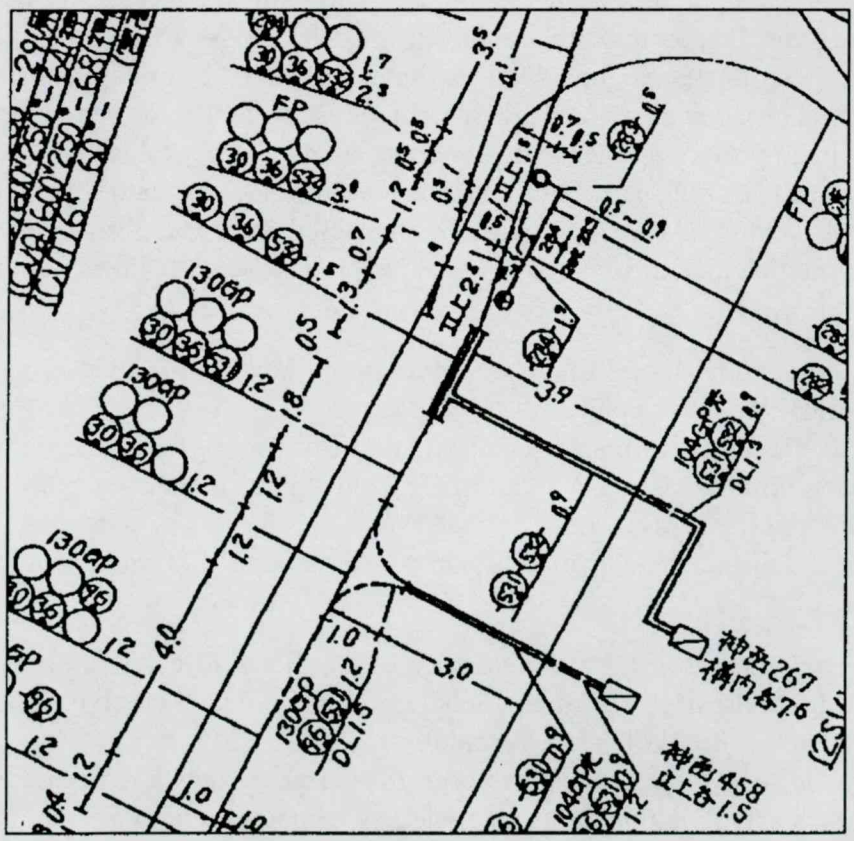
– Ezen gráf összefüggő komponensei felelnek meg a földrészleteknek.

Futási eredmények. Az automatikus feldolgozás ideje szelvényenként kb. 30 perc, ehhez mintegy 3 óra manuális munka társul, mivel hibátlan digitális térkép előállítása csak folyamatos operátori ellenőrzéssel biztosítható. Az eljárás így is lényegesen hatékonyabb a teljesen manuális digitalizáláshoz képest.

Az ismertetett rendszer hatalmas befektetett munkamennyiséget takar (erre utal, hogy a cikknek 12 társszerzője van!), és valóban alkalmas lehet az olasz térképek hatékony feldolgozására. Érdekes tulajdonsága a részben raszteres, részben vektoros képgráf adatstruktúra. Hiányosságként említhető, hogy a karakterfelismerés nem tanítható, továbbá a cikk nem részletezi az adatstruktúrát és a műveletigények vizsgálatával sem foglalkozik.

3.3. Japán közműtérképek feldolgozása

A közműtérképek jelentik az egyik legnagyobb kihívást a térkép-interpretáció terén. Rajzolatuk gyakran zsúfolt és kusza, gyakori a gyenge minőségű másolat és az utólagos javítások. Ezért is tiszteletre méltó az a rendszer, amelyet Shimotsuji és tsai (1992) ismertet japán földalatti elektromos hálózatok kábelezését ábrázoló térképek interpretációjára (7. ábra).



7. ábra. Japán elektromos hálózati térkép részlete

A cikk vázlatosan tárgyalja a feldolgozás lépéseit, ezek a következők:

1. Vektorizálás. A vonalaknak nem csak a középtengelyét, hanem a kontúrját is vektorizálják.

2. Vonalszakadások automatikus összekötése.

3. Hosszú egyenesek felismerése. Az ilyen egyeneseket elemi élek halmazaként kezelik.

4. Körívek felismerése. Először lánckód alapján ún. k-görbület meghatározásával válogatják ki a szóbajöhető élsorozatokat, majd legkisebb négyzetek módszerét alkalmazzák.

5. Szimbólumok felismerése. A szimbólumokat zárt körök, végpontok és kitöltött részek együtteseként írják le. Ha egy szimbólumkomponenst talál a rendszer, megpróbálja erre felépítve felismerni a szimbólumot.

6. Karakterek felismerése. Rövid vonalak csoportjait keresik. Mivel a karaktersorozatok gyakran vonal mentén helyezkednek el (7. ábra), ezért ilyen vonalat keresnek a közelben, ez alapján határozzák meg a karaktersorozat irányát.

7. Vonalak interpretációja. Egy vonal stílusából (folytonos, szaggatott, stb.) az adott térképtípusnál még nem állapítható meg, hogy annak a vonalnak mi a jelentése, ehhez a környezetet is vizsgálni kell. Az eljárást Hori és tsai (1992) ismerteti részletesen. A vizsgált térképfajta (7. ábra) esetén 5 vonaltípust különböztetnek meg:

– kábel,
– magyarázó szöveg segédvonala,
– aláhúzás,
– lezáró vonal,
– utat jelző vonal.

7.1. Először minden vonalszakaszhoz a $c_1,...,c_5$ címkéket rendelik, ahol c_i annak a valószínűsége, hogy az adott él i típusú. A címkék kezdőértékét a környezetüknek megfelelően (például kapcsolódó szimbólumok alapján) választják meg. Ha nincs környezeti információ, akkor minden címke egyenlő értékű.

7.2. Címkék módosítása relaxációval. Három szögponttípust különböztetnek meg: metszés, elágazás, törés. Mindegyikhez egy-egy együtthatómátrix tartozik. Minden él címkéinek új értékét a kapcsolódó élek címkéinek függvényében számítják az együtthatómátrixok segítségével.

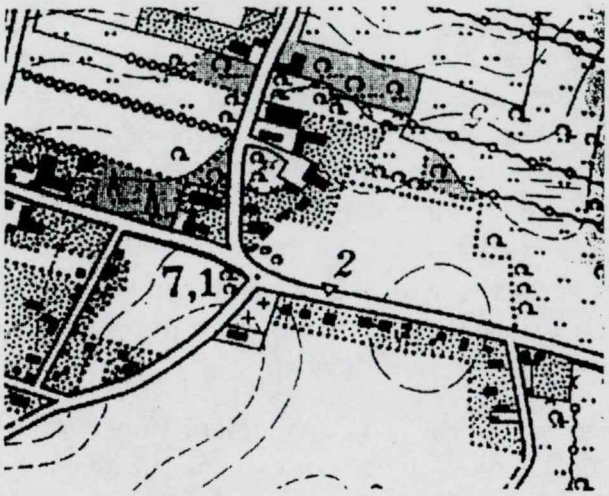
A rendszer fő érdekessége ez utóbbi technikában rejlik, de kérdéses, hogy más térképtípusok esetén ez mennyire használható.

3.4. Német topográfiai térképek feldolgozása

Az Ebi (1995) által publikált FRIMAP (FRame-based Interpretation of MAPs) elnevezésű prototípus rendszer szemantikus hálókat (4.5. fejezet) alkalmaz 1:25000 méretarányú színes topográfiai térképek feldolgozására (8. ábra).

Az előfeldolgozó modul elkülöníti és vektorizálja a fekete, zöld és barna rétegeket. Ezután következik szemantikus háló modell alapján történő interpretáció. Az alábbi objektumtípusok (concept-ek) kerülnek definiálásra: rét (legelő), tűlevelű erdő, lombhullató erdő, vegyes erdő, cserje, magányos fa, épület, útszakasz, kereszteződés, szintvonal,

vízterület. A szerző megoldást ad *rekurzív struktúrák* (például szaggatott vonal, kitöltő minták) leírására a szemantikus háló modellben.



8. ábra. Német topográfiai térkép részlete. A kivágat eredeti mérete 27 x 22 mm

A cikk a felismerés két jellegzetes problémájára hívja fel a figyelmet:

1. A réteket a német topográfiai térképek kitöltő mintázattal jelölik, de pontos határvonalukat nem adják meg. A felismerő algoritmus ezért az összefüggő kitöltő mintázat minimális befoglaló poligonját képezi, ez azonban nem mindig esik egybe azzal a területtel, amit szakember olvasna le a térképről.

2. A szintvonalak helyenként hiányosak. Hosszabb szakadást a rendszer nem képes pótolni, viszont figyelmeztet a vakon végződő szintvonalra. A szükséges korrekciót manuálisan kell elvégezni.

Karakter felismeréssel a rendszer nem foglalkozik. A cikk tárgyalásmódja rendkívül vázlatos, így további részletek a rendszerről nem derülnek ki.

Megjegyezzük, hogy Maderlechner és Mayer (1994) német kataszteri térképekre alkalmazza a szemantikus háló modellt.

3.5. A RoSy rendszer

A rendszert egy müncheni cég fejleszti (M.O.S.S. Computer Grafik System GmbH, internetes honlap: www.moss.de). A cég nagy volumenű digitalizálási munkákkal foglalkozik, ennek támogatására folyamatosan bővítik a RoSy lehetőségeit. A cég a MUSKAR-projekt (Mustererkennung in der Kartographie) keretében hosszú távú kutatási-fejlesztési együttműködést folytatott a Karlsruhei Müszaki Főiskolával, a JATE és a KMF közötti kapcsolat keretében volt lehetőségünk a rendszert közelebbről megismerni.

Általános jellemzés:

– A RoSy egy sok modulból álló, nagy bonyolultságú, nyitott rajzfeldolgozó rendszer. A nyitottság azt jelenti, hogy saját fejlesztő környezetével adott térkép- vagy rajztípushoz testreszabott alkalmazások hozhatók létre. Ezen munka elvégzése azonban hosszabb betanulási időt és komoly programozási munkát igényel.

– A rendszer megvásárolható, de rendkívül drága. (A fontosabb modulok ára együtt mintegy 10 millió forintot tesz ki.) Ezért – és a fent említettek miatt – a fejlesztők elsősorban cégen belüli használatban alkalmazzák a rendszert.

– A feldolgozás nyers vektorizálással indul, majd automatikus felismerés és végül manuális javítás történik.

– A RoSy fejlett raszter-vektor editálási képességekkel rendelkezik. Ez érthető, hiszen az automatikus felismerés mindig kíván manuális korrekciót, rossz minőségű nyersanyag esetén pedig az automatikus interpretáció egyáltalán nem alkalmazható.

A rendszer jellegzetes komponensei:

– *OGS nyelv* (Objektorientierte Graphische Sprache, Kern (1996)): egyes parancsai *komplex* felismerő műveleteket valósítanak meg. Régebbi fejlesztés, ma már szerepét az EASI (lásd alább) vette át. Az OGS programok interpreterrel batch-ben futtathatók.

– *EASI nyelv* (Erweiterbarer Applications SprachInterpreter, MOSS (1998): egyes parancsai *elemi* felismerő műveleteket valósítanak meg. Lehetőségei jóval tágabbak az OGS-nél. Folyamatosan fejlesztik, az OGS funkciókat beépítik. Az EASI programok interpreterrel batch-ben futtathatók.

– *HEDI modul* (MOSS 1996): Komplex raszter-vektor rajzszerkesztő rendszer sokféle szolgáltatással. OGS és EASI programok a HEDI-ből is futtathatók.

Az RoSy rendszer különféle alkalmazásairól, OGS és EASI fejlesztésekről számos diplomamunka és cikk számol be, néhány ezek közül:

– Klauer (1993) német kataszteri térképek feldolgozását vizsgálja.

– Kern, Mezősi és Garay (1997) magyar kataszteri térképek feldolgozási tapasztalatain keresztül mutatja be a rendszert.

– Hudra és Kern (1999) párhuzamos vonalak (például utak) felismerésére vonatkozó fejlesztésekről számol be.

– Hudra (2000) magyar kataszteri térképek feldolgozásához fejlesztett eljárásokat, ezeket hasonlítja össze a MAPINT rendszerrel (erre a 6.5. fejezetben még visszatérünk).

3.6. További rendszerek

A MAGELLAN rendszer (Map Aquisition of GEographic Labels by Legend ANalysis, Samet és Soffer (1998)) a térkép jelkulcsainak automatikus megtanulására épül. Ezen megközelítés korlátja, hogy a térképkészítés szabályrendszerét általában komplex szabályzatok rögzítik (magyar térképek esetén például a T3 (1981) és F7 (1983) szabályzatok), pusztán a jelkulcs alapján csak korlátozott interpretáció lehetséges.

Vektorizálás nélkül, tisztán raszteres algoritmusokkal végez felismeréseket Yamada és tsai (1994) 1:25 000 méretarányú japán topográfiai térképeken. Vonalak és szimbólumok

elkülönítését végzi, és magassági megírásokat ismer fel. A eljárás bitsíkok közötti műveletekre épül. Hátránya, hogy ilyen jellegű algoritmusok implementációja hatékonyan csak nagy párhuzamosságú (massively parallel) processzorral lehetséges.

Tan és Ng (1998) szintén raszteres eljárással különíti el a feliratokat a térkép vonalrajzától. Először a nagy, összefüggő alakzatokat eltávolítja (ezek ugyanis feltehetően nem feliratok), majd multirezolúciós piramist használ (hasonlóan a 11.3.1. alfejezet terepmodellező algoritmusához): a kicsinyített képen egy felirat karakterei összefüggővé válnak, így detektálhatók.

Li és tsai (1999) szintén a feliratok és vonalrajz elkülönítését veszi célba. A vizsgált 1:24000 méretarányú USA topográfiai térképekre jellemzőek a vonalrajzon átírt utcanevek, a cikk erre a problémára koncentrál. Szétválasztás után az utcahálózatot vektorosan, a megírásokat raszteresen dolgozza fel.

Chen és tsai (1996) 1:1000 méretarányú kínai kataszteri térképek interpretációjával foglalkozik. A helyrajzi számok arab számmal, a földrészlet jellege kínai írásjellel szerepel a térképen. Vékonyítás után a vázon a végpontok és az elágazási pontok sűrűségét vizsgálja, a sűrű területek jelentik a kínai írásjeleket.

Lladós és tsai (1999) vonalkázott területek felismerésére használja a Hough-transzformációt (2.1.5. alfejezet), eljárását francia kataszteri térképeken teszteli.

4. Az adatmodell kérdése

Egy alakfelismerő rendszer hatékonysága szempontjából meghatározó jelentőségű, hogy milyen adatstruktúrát, adatmodellt alkalmaz. Alább a térinformatika klasszikus adatmodelljeit tekintjük át (lásd például Detrekői, Szabó (1995)), kiegészítve az előzőekben tárgyalt interpretáló rendszerek által használt modellekkel és egyéb szóbejöhető megoldásokkal.

Sajnos a térkép-interpretáció szakirodalma csekély súlyt helyez az adatstruktúrák dokumentálására és értékelésére, ezt is szeretnénk pótolni ezzel a fejezettel. A saját interpretáló rendszerünk részére kidolgozott DG adatmodellt majd a következő fejezet ismerteti.

4.1. A spagetti modell

Eza a legegyszerűbb adatmodell: rajzelemek halmaza, a rajzelemek között nincs hivatkozási kapcsolat. Ilyen a CAD rendszerek (AutoCAD, MicroStation) és az egyszerűbb térinformatikai rendszerek (Mapinfo) adatstruktúrája. Előfordul, hogy az alaprendszer (például MicroStation) spagetti modellt használ, de egyes erre épülő modulok már topológiát építenek (MicroStation Geographics, lásd Bentley (1998)).

Az spagetti modell előnye, hogy könnyen kezelhető, egyszerű az adatok karbantartása. Ezért az előnyért viszont számos hátránnyal kell fizetni:

a) Az egymást keresztező vonalak metszéspontjában nem feltétlenül van csomópont (a vonalak ilyenkor "nem tudják", hogy metszik egymást).

b) A szomszédos poligonok (például telkek) határvonala kétszer tárolódik, ami egyrészt redundanciát jelent, másrészt módosításkor zavarokat okozhat.

A fenti jellegű problémák miatt az adatintegritás ellenőrzése, elemzések elvégzése nehézkes és lassú, ezért alakfelismerési alkalmazásokhoz a spagetti modell semmiképp sem javasolt.

4.2. Topológikus modellek

Ha az adatstruktúra nem csak a rajzelemeket, hanem azok térbeli kapcsolódási struktúráját (azaz a topológiát) is tartalmazza, akkor topológikus adatmodellről beszélünk. Az ilyen modelleknél általában minden rajzelemnek egyedi azonosítója (id) van, ennek segítségével az egyes rajzelemek egymásra hivatkozhatnak. A topológikus modelleknél általában megkövetelik, hogy

a) az egymást metsző vonalak metszéspontjában csomópont kell hogy legyen,

b) a szomszédos poligonok határvonala egyszeresen tárolódjon.

A topológikus modell kezelése akkor a leggyorsabb, ha az egyes rajzelemek közvetlen pointer-hivatkozásokkal érik el egymást.

A klasszikus topológikus modellek 1970 körül jelentek meg, Márkus (1994) 30. fejezete több ilyen modellt ismertet. Ezek közül az Arc/Info rendszer adatmodellje bizonyult a

legsikeresebbnek, amelyet a következő alfejezetben tárgyalunk. Utána egy interpretáló rendszer (RoSy, lásd 3.5. fejezet) topológikus adatstruktúráját mutatjuk be.

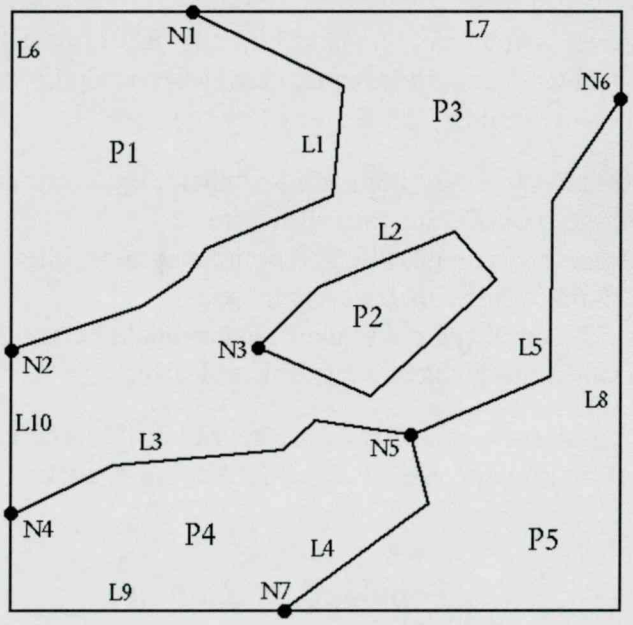
4.2.1. Az Arc/Info adatmodell

Itt minden rajzelemhez két azonosító tartozik:

– *belső azonosító,* amelynek az Arc/Info ad értéket és módosíthatja, a felhasználó nem fér hozzá.

– *felhasználói azonosító,* amelynek az Arc/Info ad kezdőértéket, de továbbiakban értékét a felhasználó módosíthatja (az Arc/Info ezt nem változtatja).

A komplex, több fájlon tárolt adatstruktúrát ESRI (1994a) ismerteti részletesen. Itt csak a legjellemzőbb komponenseket mutatjuk be: azokat, amelyek egy *poligon fedvény* (diszjunkt tartományokból álló fedvény) leírásához szükségesek (9. ábra).



9. ábra. Tartománytérkép. A csomópontokat N_i, a vonalakat L_i, a poligonokat P_i jelöli.

Két szomszédos tartomány határát a modell vonallánccal írja le, ezek találkozási pontjai a csomópontok. Magukat a tartományokat (poligonokat) a határoló vonalak sorozataként definiálja. Az adatstruktúra elemei:

a). Csomópontok. Leírásukat a LAB fájl tartalmazza, ennek egy rekordja:

label	belső azonosító (sorszám)
user-id	felhasználói azonosító
x	csomópont x koordinátája
y	csomópont y koordinátája

b). Vonalak. Leírásukat az ARC fájl tartalmazza, ennek egy rekordja:

line	belső azonosító (sorszám)
user-id	felhasználói azonosító
$x_1, y_1, ..., x_n, y_n$	a vonal töréspontjainak koordinátái
$node_1$	kiinduló csomópont azonosítója
$node_2$	végcsomópont azonosítója
lpoly	baloldali poligon azonosítója
rpoly	jobboldali poligon azonosítója

c). Poligonok. Leírásukat a PAL fájl tartalmazza, ennek egy rekordja:

poly	belső azonosító
id	felhasználói azonosító
$line_1, ..., line_n$	határoló vonalak azonosítói

Ha egy tartomány szigete(ke)t tartalmaz, akkor a sziget(ek) határvonalait is fel kell venni a poligon rekord listájára.

Az *lpoly* és *rpoly* azonosítók a tartománytérképek hatékony algoritmikus kezelését szolgálják. Pontos jelentésük: az adott határvonal az *lpoly* és *rpoly* tartományokat választja el, éspedig ha a $node_1$ kezdőpontból haladunk a $node_2$ végpont felé, akkor *lpoly* bal oldalon, *rpoly* pedig jobb oldalon fekszik.

4.2.2. Az SGD adatstruktúra

A 3.5. fejezetben bemutatott RoSy rendszer saját, SGD-nek nevezett adatszerkezetet használ. Ez egy meglehetősen bonyolult felépítésű struktúra, raszter- és vektoradatokat egyaránt tartalmaz. Amíg az Arc/Info egy fedvény adatait is több fájlon tárolja, addig itt egyetlen *.sgd* kiterjesztésű fájl tartalmaz mindent, még az esetlegesen kapcsolt raszteres adatokat is. A vektoros adatstruktúra topológikus, vagyis minden rajzelemnek egyedi azonosítója (ISN) van, amely segítségével hivatkozhatók a kapcsolódó rajzelemek.

Az adatstrukrúra részletes dokumentációja nem hozzáférhető, ezért itt csak vázlatos bemutásra szorítkozhatunk. Az SGD 17-féle adattípust használ, ezek a következők:

sgdPoint (POINT2D): Pont
sgdPolyline (POLYLINE2D): Vonallánc, vektor
sgdCircle (ELLIARC2D): kör
sgdCircleArc (ELLIARC2D): körív
sgdEllipse (ELLIARC2D): ellipszis
sgdEllipseArc (ELLIARC2D): ellipszisív
sgdElemArea (LASSO2D): elemi felület (széleinek pontjaival megadva)
sgdTextLine (TEXTLINE2D): szöveg
sgdRaster (QTINVOKE): raszter
sgdCompLine (COMPLINE): összetett vonal
sgdSimpleArea (SIMPLEAREA): egyszerű felület (határoló objektummal megadva)
sgdConnArea (CONNECTAREA): felület
sgdCompArea (COMPAREA): felület, lehet lyukas is
sgdSet (SET): halmaz (elemeinek sorrendje nem számít)

sgdSequence (SEQUENZ): szekvencia (elemeinek sorrendje számít)
sgdSegment (SEGMENT): szegmens
sgdSymbol (SYMREF2D): szimbólumhivatkozás

A RoSy igen fejlett rajzszerkesztő (CAD) funkciókkal rendelkezik, ebből adódnak még az alábbi jellemzők:

– Minden rajzelemhez *több rétegjelzés* és további (egyszerű vagy összetett) *attribútumok* tartozhatnak. Ezeket *.def* szöveges állományokban lehet definiálni.

– A felsorolt 17 típus mellett még ún. segédrajzelemek is vannak, például ideiglenes kijelölésekhez, nagyításhoz, stb. Ezekhez nem tartozik rétegjelzés.

– Adott réteghez megjelenési jellemzőket lehet definiálni. (Például, ha egy rajzelem több réteghez tartozik, akkor a *displayPriority* paraméter értéke szabályozza a megjelenítését.) Ezeket egy vagy több *.asp* (aspect) kiterjesztésű, szöveges állományban kell megadni.

– A font-állományok *.dat* kiterjesztésűek, és a fontok mellett színek, kitöltési minták, vonaltípusok, pontszimbólumok, stb. megadását tartalmazzák.

A fentiek is érzékeltetik, hogy az adatstruktúra megismeréséhez és a rendszer használatának elsajátításához hosszú betanulási idő szükséges.

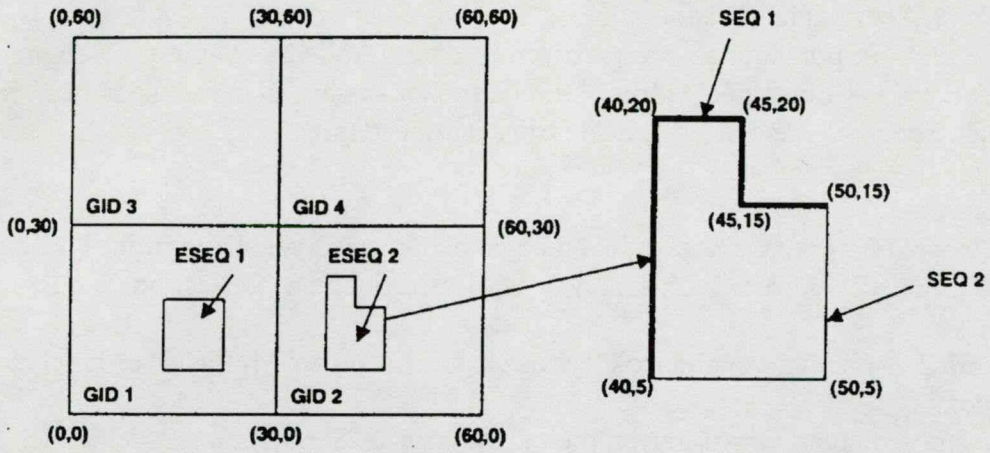
4.3. A relációs modell

A relációs adatbázisok klasszikus adatmodellje nem igazán illeszkedik grafikus adatok leírásához. Mégis számos alkalmazás ezzel a modellel dolgozik, mivel ekkor egy kommersz relációs adatbázis-kezelő szoftver (RDBMS) használható az adatok kezelésére, és ezzel megtakarítható az adatstruktúra-kezelő algoritmusok és programok előállításának munkája. Példaként említhető a MARIS térkép-interpretáló rendszer (3.1. fejezet), vagy Sárközi (1999) által bemutatott *micro GIS* adatmodell (ez utóbbi az Arc/Info adatmodell átalakításának tekinthető relációs adattáblákra).

A relációs modellnél hátrányt jelent, hogy egy tábla rekordjainak mezőszáma állandó, így például poligonok tárolása körülményessé válik. Erre vonatkozó ajánlást ad az *OpenGIS konzorcium* (a szervezet a különféle GIS rendszerek közötti együttműködés támogatására jött létre), a javasolt megoldást a 10. ábrán mutatjuk be (OpenGIS 1997).

A bemutatott példán egy fix szélességű relációs adattábla négy térbeli objektumot tartalmaz (egyszerű és összetett poligonok), amelyek azonosítására a GID mező szolgál (Geometry IDentifier). Egy térbeli objektum komponenseit az ESEQ mező sorszámozza (Element SEQence number). A térbeli objektum típusát az ETYPE mező adja meg (esetünkben ETYPE=3, ami poligon típust jelent). Végül az egyes poligonokat definiáló koordinátasorozatok 5-ös csoportokban kerülnek elhelyezésre: a tábla fix szélességű, egy poligon annyi sort foglal el, ahány 5-ös csoportra van szükség a leírásához. Az 5-ös csoportok sorszámozására a SEQ mező szolgál.

Megjegyezzük, hogy a fenti megoldást alkalmazza az *Oracle* relációs adatbázis-kezelő rendszer *Spatial Cartridge* elnevezésű modulja, amely kimondottan térinformatikai adatok relációs adatbázisban való tárolását és kezelését támogatja (Oracle 1997/a).



GID	ESEQ	ETYPE	SEQ	X0	Y0	X1	Y1	X2	Y2	X3	Y3	X4	Y4
1	1	3	1	0	0	0	30	30	30	30	0	0	0
1	2	3	1	10	10	10	20	20	20	20	10	10	10
2	1	3	1	30	0	30	30	60	30	60	0	30	0
2	2	3	1	40	5	40	20	45	20	45	15	50	15
2	2	3	2	50	15	50	5	40	5	Nil	Nil	Nil	Nil
3	1	3	1	0	30	0	60	30	60	30	30	0	30
4	1	3	1	30	30	30	60	60	60	60	30	30	30

10. ábra. Az OpenGIS (1997) által adott példa poligonok tárolására relációs adattáblában

Amint láttuk, relációs modell alkalmazása esetén is egyedi azonosítót kaphatnak a rajzelemek, a topológikus kapcsolatok ilyenkor *relációs kapcsolatok* formájában valósíthatók meg. A feldolgozás indexeléssel, rendezéssel gyorsítható, de sebessége mindenképp elmarad egy direkt pointereket használó topológikus modellétől.

4.4. Objektum-orientált modellezés

Az objektum-orientált modell nagyfokú flexibilitása folytán megszűnnek a relációs modell korlátai (Atwood és tsai, 1994). Ez a korszerű térinformatika természetes adatmodellje. Az OpenGIS konzorcium "Geometry Object Model" címen az alábbi térbeli objektumtípusokat definiálja (OpenGIS 1998):

– *Geometry:* ez az osztály-hierarchia gyökere, az "ős-objektum". Absztrakt, nem interpretálható típus.

– *GeometryCollection:* geometry objektumok halmaza.
– *Point:* koordinátáival adott pont.
– *MultiPoint:* ponthalmaz, vagyis olyan GeometryCollection, amelynek elemei pontok.
– *Curve:* egydimenziós görbe, amelyet szokásosan alappontokkal és formulával definiálnak. Speciális esetei a Line, LineString és LinearRing:
– *Line:* egyenesszakasz.
– *LineString:* vonallánc.
– *LinearRing:* önmagába záródó, önmagát nem metsző vonallánc (mint 1D objektum).
– *MultiCurve:* görbék halmaza, vagyis olyan GeometryCollection, amelynek elemei görbék.
– *MultiLineString:* vonalláncok halmaza, vagyis olyan MultiCurve, amelynek elemei vonalláncok.
– *Surface:* felület, vagyis zárt görbék által határolt 2D objektum. A felületnek egy külső és tetszőleges számú belső határoló görbéje lehet (ez utóbbi akkor, ha lyukakat tartalmaz).
– *Polygon:* olyan felület, amelyet LinearRing-ek határolnak.
– *MultiSurface:* felületek halmaza, vagyis olyan GeometryCollection, amelynek elemei felületek.
– *MultiPolygon:* poligonok halmaza, vagyis olyan MultiSurface, amelynek elemei poligonok.

Az egyes objektum-típusok részletes leírását és a hozzájuk kapcsolódó metódusokat itt nem tárgyaljuk, ezek OpenGIS (1998)-ban megtalálhatók.

Az objektum-orientált modell gyakorlati alkalmazása esetén két lehetőség közül választhatunk:
1. Szert teszünk egy – ma még ritkábban használatos – objektum-orientált adatbázis-kezelő rendszerre (OODBMS), vagy legalábbis egy objektum-relációs rendszert kell választanunk (Oracle, 1997/b).
2. Valamely objektum-orientált programfejlesztési környezetben (például C++) saját OO-adatmodellt valósítunk meg.

Interpretáló rendszerünk esetében mindkét változatot elvetettük. Ennek oka, hogy noha az objektum-orientált környezet hatékony alkalmazás-fejlesztést tesz lehetővé, tapasztalatok szerint ennek árát sebességben kell megfizetni. Térkép-interpretációnál, ahol a hangsúly nagy komplexitású, alacsony szintű elemző algoritmusok alkalmazásán van, egy pointerekre épített topológikus adatstruktúrától lényegesen jobb teljesítmény várható.

4.5. Szemantikus hálók

A szemantikus háló (semantic network) nem térinformatikai adatmodell, hanem egy általános tudásreprezentációs forma (Russell, Norvig 2000). Itt elsősorban azért foglalkozunk vele, mert német kataszteri és topográfiai térképek interpretációjára az ERNEST elnevezésű szemantikus háló modellt alkalmazták (lásd 3.4. fejezet).
Az ERNEST-et Niemann és tsai (1990) írja le részletesen. A modell elemei:

Node: a szemantikus háló egy csomópontja, típusai:
– *concept:* valamely tárgy, esemény, fogalom, stb. absztrakt leírása.
– *instance:* a concept egy konkrét példánya, megjelenési formája.
– *modified concept:* az egyes példányokból nyert információ alapján módosított concept.

Link: node-ok közötti kapcsolat leírására szolgál. Link típusok:
– *instance:* concept és instance közötti kapcsolat.
– *specialization:* kapcsolat egy általános concept és annak egy specializációja között.
– *part:* egy concept és annak valamely komponense közötti kapcsolat.
– *concrete:* különböző absztrakciós szinthez tartozó concept-ek közötti kapcsolat.

Niemann és tsai (1990) részletesen leírja a fenti modellt megvalósító adatstruktúrát, majd az alábbi alkalmazásokat mutatja be:
– szívműködést rögzítő képsorozatok elemzése,
– ipari gyártósoroknál tárgy felismerése és helyzetének megállapítása,
– beszédfelismerés.

A fentiekkel a kutatás egy lehetséges irányát kívántuk bemutatni, amely egyelőre még nem jellemző a térkép-interpretáció világában.

4.6. Térbeli indexelés

Ahogy a relációs adatmodellhez is hozzátartoznak a gyors keresést biztosító indexelési eljárások, úgy térbeli adatok hatékony kezeléséhez is elengedhetetlen valamilyen index alkalmazása. A térbeli keresés alapelveit Oracle (1997/a) alapján tekintjük át.

A térbeli keresés általában két lépésben történik:
– *Elsődleges szűrő* (primary filter) alkalmazásával kiválasztjuk az N elemű teljes adathalmaznak egy M elemű részét ($N >> M$), amely garantáltan tartalmazza a keresett objektumo(ka)t. Az elsődleges szűrés igen gyors kell hogy legyen, általában O(log(N)) műveletet igényel.
– *Másodlagos szűrő* (secondary filter) segítségével választjuk ki az M adatból a keresette(ke)t. Ennek ideje már lineáris (azaz O(M)) is lehet.

A sokféle térbeli keresési/lekérdezési művelet közül a legfontosabbak visszavezethetők a következőre: keressük egy adott téglalap által részben vagy egészben tartalmazott M db adatelemet. Ezt kell alkalmazni például, ha
– képernyőn meg akarunk jeleníteni egy adott kivágatot,
– adott objektumhoz legközelebbi objektumot keressük egy bizonyos környezeten belül.
Ilyenkor az elsődleges szűrő feladata, hogy kiválasszon egy O(M) elemű halmazt, amely már tartalmazza a keresett M elem mindegyikét, de azon kívül még fölös elemeket is tartalmazhat. A másodlagos szűrő ezután akár egyenkénti vizsgálattal választhatja ki a keresett elemeket.

Maga a térbeli index egy speciális adatstruktúra, amely az egyes adatelemek azonosítóit (id) tartalmazza alkalmas elrendezésben. Kétféle index létezik:

– *Egyszeres index* (SESI = single entry spatial index): egy adatelem azonosítója csak egyszer szerepel az indexben.

– *Többszörös index* (MESI = multiple entry spatial index): egy adatelem azonosítója több helyen is szerepelhet az indexben.

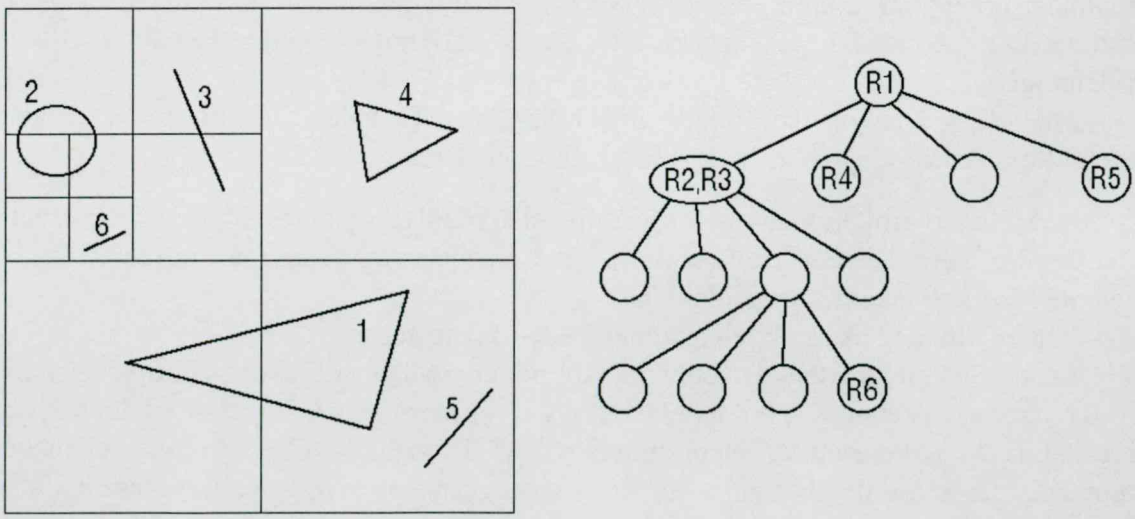
Pontszerű objektumok az (x,y) koordinátáik alapján kerülnek elhelyezésre az indexben. Nem pontszerű objektumokat a befoglaló téglalapjuk (MMB = minimum bounding box) alapján szokták elhelyezni.

A két leggyakrabban alkalmazott indexelési eljárás a *négyesfa* és a *grid index*. Ezeket ismertetjük a következőkben.

4.6.1. Négyesfa (quadtree) index

Alapelv: a teljes rajzterületet alkotó téglalapot négy egyenlő részre osztjuk, majd az egyes negyedeket tovább negyedeljük, stb. Így egy fastruktúra keletkezik, amelynek gyökere a teljes rajzterületet reprezentálja, szögpontjai pedig a negyedeléssel kapott egyes szegmenseket. Négyesfát képfeldolgozásban, raszteres és vektoros térinformatikában egyaránt használnak (Kollányi, Prajczer 1995).

A négyesfa egyszeres (SESI) és többszörös (MESI) indexként is felépíthető. A 11. ábrán egy SESI megvalósítást mutatunk be. Minden rajzelemet a négyesfa egy (és csak egy) szögpontjához rendelünk: ahhoz a szögponthoz, amelyhez tartozó szegmensbe a rajzelem befoglaló téglalapja teljes egészében belefér, de annak egyik negyedében sem fér már el. Egy szögponthoz több elem is tartozhat, ez beágyazott listák alkalmazásával oldható meg. Itt az elsődleges szűrő a négyesfa megfelelő szögpontjának kiválasztását, a másodlagos szűrő pedig a listán a megfelelő elem(ek) kivásztását jelenti.



11. ábra. Egyszeres négyesfa index. R_i jelöli az i sorszámú rajzelemet

Az Oracle rendszer Spatial Cartridge nevű térbeli adatbázis-kezelő modulja a négyesfa index egy MESI megvalósítását alkalmazza (Oracle 1997/a), amely jól illeszkedik az Oracle standard B^+-fa indexeléséhez, és az SQL nyelv kiterjesztésével térbeli lekérdezéseket tesz lehetővé (például spatial join).

A négyesfa számos változatát a kapcsolódó algoritmusokkal Samet (1989) tárgyalja.

4.6.2. Grid index

Ez az indexelés bizonyos értelemben a tördelőtáblázat (lásd például Aho és tsai, 1982) kétdimenziós változatának tekinthető. Alapelve, hogy a rajzterületet r x r méretű négyzetekre (általános esetben téglalapokra) osztjuk fel. Minden négyzethez egy listát generálunk, amely azon adatelemek azonosítóit foglalja magába, amelyeket részben vagy egészben tartalmaz az adott négyzet (Samet 1989). A grid index szükségképpen *többszörös index*, hiszen egy objektum azonosítója annyi listára kerül fel, ahány négyzetet lefed ill. metsz.

A grid indexet alkalmazza például az ESRI (Environmental Systems Research Institute) által fejlesztett *Spatial Data Engine* térbeli adatbázis-kezelő modul (ESRI 1998). Az adatok egyenlőtlen eloszlása esetén két vagy három, különböző felbontású gridet alkalmaznak.

Az indexbe pontszerű objektumot konstans időben lehet felvenni, hiszen egyszerű formulával meghatározható, hogy melyik négyzetbe esik. Nem pontszerű objektumnál viszont az idő területarányosan nő, mivel több listára kell azt felvenni.

Az adatelérés költsége *egyenletes adateloszlás* esetén a legkedvezőbb: ha a keresési téglalap mérete nagyságrendben megegyezik a rácsmérettel, akkor M db elem kiválasztásának ideje $O(M)$, vagyis egy elemre *konstans elérési idő* jut. Hátrányként jelentkezik viszont a redundancia: mivel egy objektum azonosítója több listán is szerepelhet, így egy elem többször is kiválasztásra kerülhet.

A grid index egy konkrét megvalósításával és elemzésével az 5.6. fejezet foglalkozik.

4.6.3. Összehasonlítás

A négyesfa és grid indexelést összehasonlítva megállapíthatjuk:

– A négyesfa általában *logaritmikus* elérési időt biztosít.

– A grid index legrosszabb esetben *lineáris*, egyenletes adateloszlás és a gridméretnek megfelelő keresési környezet esetén viszont *konstans* idejű elérést tesz lehetővé.

– A négyesfa egyszeres (SESI) indexelést is lehetővé tesz, míg a grid indexnél csak MESI alkalmazható, ami redundanciát jelent.

– A grid index annál előnyösebb, minél kisebb az objektumok átlagos mérete.

– A grid index egyszerűbben programozható, ami futási időben kisebb overhead-et és nagyobb megbízhatóságot jelent.

5. A DG adatmodell

A DG (= Drawing Graph) adatmodellt kimondottan térkép-interpretáció támogatására dolgoztuk ki. A cél olyan *topológikus modell* definiálása volt, amely

– univerzális, vagyis tetszőleges térképtípus és felismerő eljárás esetén használható,

– hatékony algoritmikus kezelést tesz lehetővé, ezzel biztosítva a valós idejű feldolgozást,

– kevés adattípust használ, így a felhasználó számára is könnyen áttekinthető.

Ez utóbbi szempont azért lényeges, mert adott térképtípushoz a konkrét felismerő technológia kialakítása részben a felhasználó feladata, és ezt megkönnyíti az adatmodell ismerete. (A bonyolult adatstruktúrákat gyakran eltakarják a felhasználó elől, ezzel viszont korlátozzák annak lehetőségeit.) Továbbá, az adatstruktúrát kezelő programmodulok hatékonyabban és biztonságosabban programozhatók, ha nem kell sokféle adattípusra sokféle kezelő algoritmust készíteni.

Az adatmodellt két szinten tárgyaljuk:

– *Logikai modell:* absztrakt modell, amely segítségével leírhatók a felismerő algoritmusok. Egy interpretáló rendszer felhasználója számára a logikai modell ismerete elegendő.

– *Fizikai modell:* a számítógépes megvalósítás szintje. Ezen a szinten írhatók le a hatékonyságot garantáló elvi és gyakorlati megoldások, amelyek tehát az adatmodell lényeges részének tekintendők. Megmutatjuk, hogy a kezdeti topológia létrehozása $O(N \cdot \log(N))$ időt igényel, míg az adatstruktúrát kezelő összes többi művelet lineáris időben végezhető.

5.1. Logikai adatmodell

A DG nem objektum-orientált modell. Ennek ellenére az adatelemeket *DG-objektumok*nak nevezzük, itt tehát az objektum szó általánosabb értelmét használjuk. A DG-ben mindössze négy objektumtípus (adattípus) van, ezek segítségével a felismerés során fellépő bonyolult struktúrák is leírhatók, amint azt a későbbiekben látni fogjuk.

Minden DG-objektumhoz egy *rétegszám* tartozik. Ez egy szokásos, CAD értelemben vett rétegre utal, amelyhez megjelenítési információk (szín, vonaltípus és vonalvastagság) rendelhetők. A 0 rétegszámot definiálatlan rétegnek tekintjük. A felismerés kezdetén minden DG-objektum rétegszáma 0.

A négy DG-objektumtípus a következő:

NODE: csomópont. (x, y) koordinátákkal adott, pontszerű objektumot vagy él(ek közös) végpontját jelenti. Egy rétegben nem lehet két azonos koordinátájú csomópont (ha ilyenek keletkeznek, ezek összevonandók).

EDGE: él. Egyenesszakasz, amely két csomópontot köt össze. Két NODE között egy rétegben legfeljebb egy élet engedünk meg.

TEXT: felirat. A rajzon elhelyezett szöveg, vagy speciális karakter(sorozat)ként kezelhető jelkulcsi elem. ASCII jelsorozattal írható le, amelyhez (x, y) koordináta (beillesztési pont), méret (magasság) és elforgatási szög tartozik. A TEXT jellemzően felismerés eredményeként keletkezik, de ezen túlmenően sokoldalúan használható (például őrkeresztek megadására, szintvonal magasságértékének tárolására, stb.).

PAT: alakzat (pattern). Tetszőleges DG-objektumok (rendezett) halmaza. Itt tipikusan élek halmazára kell gondolnunk, amelyek együtt valamilyen alakzatot képeznek, de az adatmodell tetszőleges objektumhalmazt megenged, amelyet sok esetben ki is használunk. Hierarchikus struktúrák is leírhatók, ha egy PAT objektum másik PAT objektumot tartalmaz. Értelemszerű korlátozás, hogy ez a rekurzió ciklusmentes legyen, vagyis egy PAT ne tartalmazhassa önmagát sem közvetlenül, sem közvetve.

Jelölések: A logikai modell szintjén az DG-objektumokat az alábbiak szerint jelöljük:
– NODE (x, y)
– EDGE ($node_1$, $node_2$)
– TEXT (string, x, y, height, α)
– PAT (obj_1, ..., obj_n)
Ha például egy alakzat két EDGE és egy TEXT objektumot tartalmaz komponensként, akkor a PAT($EDGE_1$, $EDGE_2$, TEXT) jelölést alkalmazzuk.

> *Megjegyzések:*
> 1. Előbbiekben az alakzatot rendezett objektumhalmazként definiáltuk. Ennek oka, hogy a gépi adattárolás jellegéből adódóan egy halmaz elemei mindig adott sorrendben kerülnek tárolásra. Alkalmazástól függ, hogy az elemek tárolási sorrendjét kihasználjuk-e vagy sem.
> 2. Ha egy TEXT objektum *height* attribútumának értéke 0, akkor kirajzoláskor a TEXT nem jelenik meg. Ezt *rejtett szöveg*nek nevezzük, és belső adattárolásra használhatjuk.
> 3. A TEXT objektum tulajdonképpen *rajzi makróhívás*nak is tekinthető. Ugyanis szöveg megjelenítésekor valamely vektoros font kerül meghívásra, és a megfelelő méretben ill. elforgatási szögben kirajzolásra. Ez általánosítható úgy, hogy tetszőleges vektoros rajzrészletet PAT-ként definiálunk (*makró definíció*), és egy TEXT objektummal erre hivatkozunk (*makró hívás*). Az ilyen TEXT-ben ASCII jelsorozat helyett a megfelelő PAT azonosítója szerepel. Ez a technika a CAD rendszerekben általánosan használatos (AutoCAD-nél *blokk*nak, MicroStation-nél *cell*nek nevezik), és TEXT segítségével a DG-ben is megvalósítható, az eddigi alkalmazásoknál azonban nem volt rá szükség.

A DG modellben a NODE és EDGE objektumok együtt egy klasszikus *gráf-adatstruktúrát* alkotnak, amelyen a gráf-algoritmusok könnyen programozhatók. A többi térinformatikai adattípust mind a PAT objektumtípussal valósítjuk meg, amint ezt alább megmutatjuk.

5.1.1. OpenGIS objektumok megvalósítása DG-ben

Az OpenGIS (1998) által specifikált objektum típusokat (lásd 4.4. fejezet) az alábbiak szerint interpretáljuk:

– *Geometry:* absztrakt, nem interpretálható típus
– *GeometryCollection:* PAT
– *Point:* NODE
– *MultiPoint:* PAT($NODE_1$, ..., $NODE_n$)
– *Curve:* nem interpretáljuk
– *Line:* EDGE
– *LineString:* PAT($EDGE_1$, ..., $EDGE_n$), vagy a kezdő- és végpont feltüntetésével PAT($NODE_1$, $NODE_2$, $EDGE_1$, ..., $EDGE_n$), vagy csak a töréspontok feltüntetésével PAT($NODE_1$, ..., $NODE_n$)
– *LinearRing:* lásd LineString
– *MultiCurve:* nem interpretáljuk
– *MultiLineString:* PAT(PAT_1, ..., PAT_n)
– *Surface:* nem interpretáljuk
– *Polygon:* PAT($EDGE_1$, ..., $EDGE_n$), vagy PAT(PAT_1, ..., PAT_n), ahol PAT_i a poligont határoló LinearRing
– *MultiSurface:* nem interpretáljuk
– *MultiPolygon:* PAT(PAT_1, ..., PAT_n)

Egy *görbe* objektumtípus bevezetésével a nem interpretált objektumok is megvalósíthatók lennének. Ennek azonban nem éreztük szükségét, mivel a görbéket a térinformatikai gyakorlatban általában vonalláncal közelítik.

Ugyanakkor az OpenGIS specifikációból hiányzik a TEXT típus, amely a DG-ben nem csak feliratok, hanem rajzi makrók kezelésére is alkalmas.

5.2. Fizikai adatmodell

Az adatmodell topológikus, ezért minden objektumnak egyedi azonosítószáma van (*id*), a topológikusan kapcsolt objektumok egymás id-jére hivatkoznak. Ennek figyelembe vételével az egyes objektumoktípusokhoz a 12. ábrán megadott attribútumok tartoznak. A szögletes zárójelbe tett attribútumok elhagyhatók, amennyiben a felismerő algoritmusok során ezekre nincs szükség.

> Az egyes objektumtípusok felépítését úgy választottuk meg, hogy minden típus csak egy változó hosszúságú listát tartalmazzon, és ez az objektum végén helyezkedjen el. Ez megkönnyíti az adatstruktúra programozástechnikai kezelését.

Térkép-interpretációnál egyszerre egy szelvényt dolgozunk fel, az ehhez tartozó DG adatstruktúra általában 10 MB alatt van, így a mai gépek memóriaméretei mellett a teljes adatstruktúra a memóriában tárolható. A továbbiakban ezzel a feltételezéssel élünk.

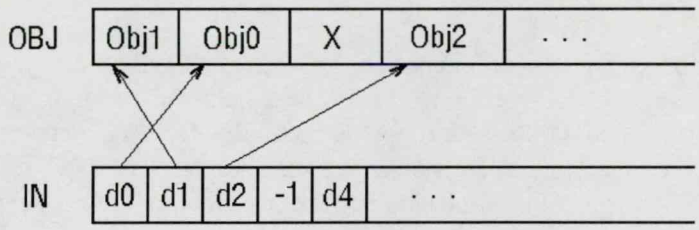
A DG-objektumokat egy *Obj* nevű tömbben tároljuk. Az egyes objektumok változó hosszúságúak, közvetlenül egymás után tárolódnak az Obj tömbben.

Az egyes objektumok *nem* tartalmazzák a saját azonosítójukat. Ehelyett egy *In* tömböt veszünk fel, amelynek *i*-edik eleme az *i* azonosítójú objektum *offset* értékét (vagyis az Obj tömbön belüli kezdőcímét (byte-sorszámát)) tartalmazza (13. ábra). Ezzel a technikával közvetlenül elérhetők a hivatkozott objektumok.

Megjegyzendő, hogy az Obj tömbben az objektumok nem feltétlenül az azonosítók növekvő sorrendjében helyezkednek el, amint azt a 13. ábra is szemlélteti.

NODE:	x, y	a pont koordinátái
	$edge_1$, ..., $edge_n$	a pontból kiinduló élek azonosítói
EDGE:	$node_1$	az él kezdőpontjának azonosítója
	$node_2$	az él végpontjának azonosítója
	[pat_1, ..., pat_n]	az élt tartalmazó PAT-ok azonosítói
TEXT:	x, y	a text középpontjának koordinátái
	pat	a texthez kapcsolt objektum azonosítója
	height	betűmagasság
	angle	elforgatási szög
	string	ASCII jelsorozat, maga a szöveg
PAT:	x, y	az alakzat referenciapontjának koordinátái
	obj_1, ..., obj_n	az alakzathoz tartozó DG-objektumok azonosítói

12. ábra. A DG objektumtípusok felépítése



13. ábra. Az Obj és In tömbök kapcsolata. Az X egy törölt objektum helyét, –1 a rá való hivatkozás helyét jelöli

A DG-objektumok egységes fejrésszel (header) rendelkeznek, amely az alábbi komponensekből áll:

Length: az objektum hossza byte-ban. Ebből számítható a következő objektum offset értéke, és az objektum végén lévő változó hosszúságú lista mérete.

Type: az objektum típusát adja (NODE, EDGE, TEXT vagy PAT).

Bitmask: különböző ideiglenes jelzésekre szolgálnak. Legfontosabb közűlük az objektum törölt voltát jelző bit.

Layer: az objektum rétegszáma. Háromféle lehet:

– *Definiálatlan rétegszám*, ezt 0 érték reprezentálja. Kezdetben minden objektum a 0 rétegbe tartozik.

– *Saját rétegszám:* 1 és 127 közötti érték.

– *Öröklött rétegszám:* 128 és 255 közötti érték. Ilyet azon 0 rétegszámú EDGE objektumok kapnak, amelyek valamely PAT-nak elemei. Ha a PAT rétegszáma *r*, akkor az

EDGE *r*+128 értéket kap. Ha az EDGE több PAT-nak is eleme, akkor a nagyobb rétegszámú PAT rétegét örökli. Az öröklött rétegszám jelentőségét többek között a grafikus megjelenítésnél fogjuk látni.

Az öröklött rétegszámok beírásáról egy *SetPatLayer* nevű rekurzív eljárás gondoskodik. Időigénye az aktuális adatstruktúra bonyolultságától függ: elvileg nem lineáris idejű, de mivel ritkán fordul elő, hogy egy él több PAT-nak is eleme, így lineáris idejűnek tekinthető (a gyakorlatban igen gyors). Ezt a műveletet csak akkor kell végrehajtani, ha az adatstruktúra változott, tehát például felismerő műveletek után.

A Type, Layer és Bitmask egy közös *Ltype* 16-bites mezőben helyezkedik el, az egyes részek jelentése:

– 0...3. bit: az objektum típusát adja (0=NODE, 1=EDGE, 2=PAT, 3=TEXT, a 4...15 értékek nem használatosak.).

– 4...11. bit: rétegszám.

– 12. bit: azt jelzi, hogy egy automatikus felismeréssel létrehozott objektumot a felhasználó elfogadott-e vagy sem (1 = elfogadott, 0 = nem)

– 13. bit: azt jelzi, hogy egy automatikus felismeréssel létrehozott objektumot a felhasználó ellenőrzött-e vagy sem (1 = ellenőrzött, 0 = nem).

– 14. bit: az automatikus felismerés során létrehozott, de még fel nem dolgozott objektumok jelzésére szolgál (1 = felismert, 0 = nem). (A 13. bit csak akkor lehet 1, ha a 14. bit is 1.)

– 15. bit: törölt objektum jelzésére szolgál (1=törölt, 0=nem).

A gyorsabb számolás érdekében a DG-ben belül *32-bites fixpontos koordinátákat* használunk (16 bit egészrész, 16 bit törtrész alakban, vagyis az 1 koordináta érték hexadecimálisan 00010000 alakú). A felhasználó felé ugyanakkor lebegőpontos koordinátákat mutat a rendszer. A fixpontos (x, y) és a lebegőpontos (x', y') számábrázolás között az egész DG-re globális eltolási faktorok és skálafaktorok segítségével történik az átszámítás:

$x' = x/s_x + e_x \qquad x = (x' - e_x)*s_x$

$y' = y/s_y + e_y \qquad y = (y' - e_y)*s_y$

Mindez lehetővé teszi a DG-beli számok normalizálását, és így maximális pontosság elérését.

Megjegyzendő, hogy a MicroStation rendszer is 32-bites fixpontos számábrázolást alkalmaz, ez a pontosság tehát nagyobb CAD/GIS alkalmazásokhoz is elegendőnek bizonyult.

A teljes DG adatstruktúra a következő komponensekből áll:

– *header:* a teljes adatstruktúra paramétereit tartalmazza, többek között az eltolási és skálafaktor értékét.

– *layers:* rétegek leírása.

– *objektum tömb* (Obj): DG-objektumok.

– *index tömb* (In): az egyes objektumok offsetjét (Obj-beli kezdőcímét) tartalmazza.

Az adatstruktúra mágneslemezen egy DG kiterjesztésű fájl formájában tárolódik, amelynek felépítése hasonló a memóriában tárolt DG adatstruktúrához, de az adattömböknek csak a ténylegesen feltöltött részét tartalmazza

5.3. Kezdeti topológia létrehozása

A DG adatmodell felépítésénél rendezetlen vektorhalmazból indulunk ki (spagetti modell). Ez származhat valamely vektorizáló programtól (például a 2.2. fejezetben leírt nyers vektorizáló eljárásunkból), vagy más, például DXF formátumú bemenetből. A DG adatstruktúra felépítésének lépései a következők:

1. Kezdeti feltöltés. Üres DG struktúrát hozunk létre, amelyet fokozatosan töltünk fel. Minden vektorhoz két NODE és egy EDGE objektum kerül a DG-be (a két végpontnak és magának az egyenesszakasznak megfelelően). Itt végezzük el a koordináták skálázását, vagyis leképezését a 32-bites fixpontos DG-számábrázolás szerint. Ekkor a DG még spagetti-modellként tárolja az adatokat.

2. Csomópontok rendezése. Koordináta szerint sorbarendezzük a NODE-okat, ezzel egymás mellé kerülnek az azonos NODE-ok, amelyek ezután összevonhatók. Rendezésnél nem a NODE objektumokat mozgatjuk, hanem egy *Sort* tömböt hozunk létre, amely a NODE-ok azonosítóit tartalmazza, és ezt a tömböt rendezzük halomrendező (heapsort) algoritmussal. Az algoritmus időigénye $O(N \cdot \log(N))$, ahol N a NODE-ok száma.

3. Azonos csomópontok összevonása. A művelet a *Sort* tömb segítségével lineáris időben végezhető. Eredményeként topológikus adatstruktúra áll elő.

4. Kettős élek eltávolítása. Ha két NODE között egynél több él fut, akkor ezek közül csak egyet tartunk meg. Erre a műveletre vektorizálási hibák miatt lehet szükség.

A fenti műveletek közül a rendezés a meghatározó, a többi mind lineáris időben végezhető, tehát az összes időigény N vektor esetén $N \cdot \log(N)$ nagyságrendben adódik.

5.4. Az adatstruktúra karbantartása

Minden adatmodell esetén *három aktualizálási műveletet* kell megvalósítani: új elem felvétele, elem törlése és elem módosítása. Megmutatjuk, hogy mindhárom művelet konstans időben végezhető, vagyis az időigény független a DG-objektumok számától.

– *Új objektum felvétele.* Az új objektum az Obj tömb végére kerül, részére a *maxid*+1 azonosító kerül kiosztásra, és az In tömbbe megtörténik a megfelelő bejegyzés.

– *Objektum törlése.* Az objektum fejrészébe egy "törölt" jelzés kerül, az In tömbben pedig az azonosítójához –1 értéket írunk (13. ábra).

– *Objektum módosítása.* A módosítást helyben végezzük, ha azzal az objektum hossza nem változik. Ellenkező esetben az objektum "törölt" jelzést kap, és módosított hosszal újra felvesszük az Obj végén. Azonosítója nem változik, de az In-beli bejegyzést módosítani kell az új offset értéknek megfelelően.

> Topológikus modellnél a fenti műveletekhez hozzátartozik a topológikus kapcsolatok aktualizálása is. Ha például két, már meglévő NODE-ot új EDGE-el kötünk össze akkor az alábbi műveleteket kell elvégezni:
> – EDGE objektum felvétele.
> – mindkét NODE edge-listájára felvenni az új EDGE azonosítóját.
> Belátható, hogy az In tömb közvetlen pointer hivatkozásainak köszönhetően ezek a műveletek is lineáris időben végezhetők.

A fenti műveletek biztosítására mind az Obj, mind az In tömb *túlcsordulási terület*tel rendelkezik, méretét a kiindulási adatstruktúra százalékában adjuk meg (általában 20% és 100% között). Ha a túlcsordulási terület betelik, akkor az adatstruktúrát tömöríteni vagy újraallokálni kell.

Az *újraallokálás*t az operációs rendszer végzi, erre csak ritkán van szükség.

A *tömörítést lineáris időben, helyben végezzük*, vagyis nem szükséges külön munkaterület allokálása. Ez lényeges, mivel az adatstruktúra igen nagy is lehet.

> A megoldás nem triviális, az alapötleteket alább részletezzük.
>
> *Az Obj tömb tömörítése* (törölt objektumok helyének kihagyása). A művelet lépései:
> 1. Az In tömbön haladunk végig. Minden objektum fejrészébe beírjuk a saját azonosítóját (a Length, Ltype mezők helyére), egyidejűleg a length, type értékeket mentjük az In tömbbe (az offset értékek helyére, mivel azok úgyis változni fognak). Mindez azért lehetséges, mert Length és Ltype együtt 4 byte-ot tesz ki, és az objektum azonosítókat szintén 4 byte-on tároljuk.
> 2. Az Obj tömbön haladunk végig, kihagyjuk a törölt objektumok helyét.
> 3. Az Obj tömbön haladunk végig. A Length, Ltype értékeket visszaállítjuk az In tömbből, egyidejűleg In-be beírjuk az új offset értékeket.
>
> *Az In tömb tömörítése.* Ez tulajdonképpen azt jelenti, hogy az objektumok újrasorszámozásával minden objektum új azonosítót kap, ezzel a törölt objektumok indexei felszabadulnak, és az In tömb mérete csökken. (Az In tömörítésére jóval ritkábban kerül sor, mint az Obj tömörítésére.) A művelet lépései:
> 1. Megegyezik az Obj tömörítés 1. lépésével.
> 2. Az In tömbben *azonosító-átszámító táblát* hozunk létre, vagyis In[id] értéke az *id* azonosítójú objektum *új id*-jét fogja adni. Ehhez az Obj tömbön haladunk végig, közben generáljuk az egyes objektumok új id-jét (amely az objektumok Obj-beli sorrendjének megfelelő sorszám). Mivel most minden objektum fejrésze a régi *id*-t tartalmazza (lásd 1. lépés), így In-be beírható az új id, egyidejűleg Length és Ltype visszaállítható az In tömbből.
> 3. Az Obj tömbön haladunk végig, az összes id-hivatkozást lecseréljük az In-ben létrehozott átszámító tábla segítségével.
> 4. Az Obj tömbön haladunk végig, In-be beírjuk az új offset értékeket.

Összefoglalás. Mivel N objektumot tartalmazó DG túlcsordulási területének beteléséhez legalább $O(N)$ aktualizálási művelet kell, és mivel a DG-t tömörítő műveletek lineáris idejűek, így megállapíthatjuk, hogy egy aktualizálási műveletre – a tömörítést is figyelembe véve – konstans idő jut.

5.5. Grid index alkalmazása

A DG adatmodell esetén grid indexelés (4.6.2. fejezet) mellett döntöttünk. Ennek indokai a következők:

– Térkép-interpretációnál *egyszerre egy szelvény* feldolgozásával foglalkozunk, így a négyesfa flexibilitása csak korlátozottan érvényesülne.

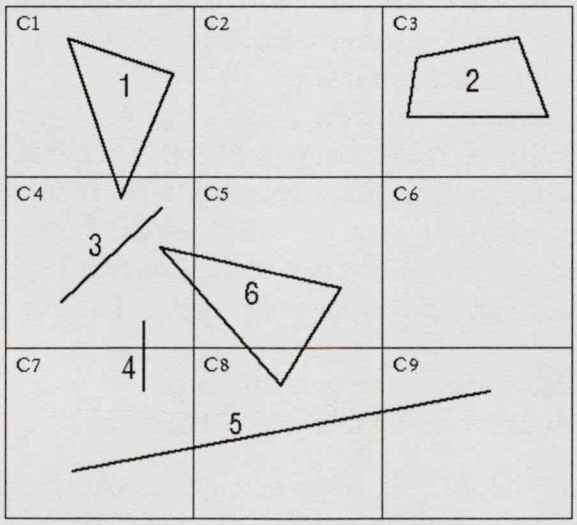
– Adott méretarányú szelvényen *az adateloszlás viszonylag egyenletes* – olyan értelemben, hogy a térkép adott területén lehetetlen bizonyos mennyiségűnél több adatot összezsúfolni az áttekinthetőség megőrzése mellett. Ez egy felső korlátot jelent az adatsűrűségre, ennek megfelelően optimalizálhatjuk az index méretezését.

– Térkép-interpretációnál általában *kis méretű objektumokkal* dolgozunk ezek hatékonyan kezelhetők grid indexszel. (Nagyobb objektumok csak a feldolgozás végén jönnek létre.)

5.5.1. Adatstruktúra

A DG esetén alkalmazott grid index adatstruktúrát a 14. és 15. ábra szemlélteti. Minden négyzethez egy indexlistát rendelünk, amely az adott négyzetbe – részben vagy egészben – beleeső rajzelemek indexeit tartalmazza. Ha egy rajzelem több négyzetben is szerepel, akkor szükségképpen több indexlistán fog szerepelni (15/a. ábra). Az összes indexlista egy közös láncolt listán helyezkedik el, amelynek első m eleme az egyes listák kezdete (m a listák száma). Minden listaelem egy *val* értékből és egy *next* pointerből áll. *Val* a listán szereplő rajzelem azonosítója, –1 üres listát jelent. *Next* a következő listaelem indexe, –1 a lista végét jelenti (15/b. ábra).

A fentiek mellett még a $last_1$, ..., $last_m$ változókat használjuk, amelyek az egyes listák utolsó elemeinek indexét tartalmazzák.



14. ábra. Grid indexelés. C1,...,C9 az egyes négyzeteket jelölik, 1,...,6 pedig az egyes rajzelemek azonosítói

a)

C1	C2	C3	C4	C5	C6	C7	C8	C9
R1		R2	R1	R6		R4	R5	R5
			R3			R5	R6	
			R4					
			R6					

b).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Val	R1	-1	R2	R1	R6	-1	R4	R5	R5	R3	R4	R5	R6	R6
Next	-1	-1	-1	10	-1	-1	12	14	-1	11	13	-1	-1	-1

15. ábra. Indexlisták a 14. ábra szerinti grid indexhez. *a)* a 9 lista logikai felépítése (R1, ..., R6 a rajzelem azonosítók), *b)* a fizikai adattömb

A grid index lineáris időben létrehozható. Pontos műveletigénye $R \cdot N$, ahol N a rajzelemek száma, R a tárolási redundancia (lásd a következő alfejezetben).

5.5.2. Optimális rácsméret meghatározása

Vizsgálatainknál magyar kataszteri térképeket veszünk alapul. Minden mennyiséget egységesen a térképen mért mm-ben adunk meg, a könnyebb tárgyalás kedvéért téglalap helyett mindenütt négyzetet veszünk. Az alábbi mennyiségeket vezetjük be:

– r a grid felbontása, vagyis r x r méretű négyzetekből áll az index. Ennek optimális értékét szeretnénk meghatározni.

– s az átlagos DG-objektum méret, vagyis egy objektum átlagosan s x s méretű négyzetbe foglalható be. Méréseink alapján nyers vektoros rajzon s = 0.5 mm, teljesen feldolgozott térképen s = 4 mm. A kis értékek abból adódnak, hogy az átlagos értékbe a nulla méretű NODE objektumok is beszámítanak.

– k az aktuális keresőablak mérete, vagyis egy k x k méretű terület objektumait kell kikeresni a DG-ből. k értéke széles határok között mozoghat, például szaggatott vonalak felismerésénél értéke 5 mm, míg grafikus megjelenítésnél átlagosan 40 mm.

– D a rajzsűrűség, vagyis egy mm^2-re eső objektumok átlagos száma. Méréseink alapján sűrű rajzolatú kataszteri szelvényen nyers vektorizálás után legfeljebb 40 vektor/cm^2 sűrűség adódik. Átlagos sűrűségként 15 vektor/cm^2 vehető. Tehát a jellemző értékek D=0.4 és D=0.15 .

– R a tárolási redundancia, ami azt adja meg, hogy egy objektum átlagosan R-szer szerepel az indexben. Értékét a fenti mennyiségekből származtatjuk (lásd alább).

A vizsgálatoknál feltételezzük, hogy az objektumok egyenletes térbeli eloszlás szerint helyezkednek el, és minden objektum s méretű.

1. Állítás. A grid index tárolási redundanciája $R = (r+s)^2/r^2$.

Bizonyítás. Legyen $(n-1)r < s \leq n \cdot r$, másképpen $s = (n-1)r+h$ $(0 < h \leq r)$. Ha egy s x s méretű objektum bal felső sarkát egy grid négyzeten belül mozgatjuk, akkor a négyzetet az alábbi T1, T2 és T3 tartományokra oszthatjuk:

– T1 esetén az objektum $n \cdot n$ grid négyzetet,
– T2 esetén az $(n+1) \cdot n$ grid négyzetet,
– T3 esetén az $(n+1) \cdot (n+1)$ grid négyzetet foglal le.

A T1, T2 és T3 tartományok területe rendre $(r-h)^2$, $2h(r-h)$ és h^2, ezért az objektum bal felső sarkának az egyes tartományokba esési valószínűsége $(r-h)^2/r^2$, $2h(r-h)/r^2$ és h^2/r^2. Innen az objektum által foglalt négyzetek számának várható értéke

$$R = [\ (r-h)^2n^2 + 2h(r-h)(n+1)n + h^2(n+1)^2\ ]/r^2 = (r \cdot n+h)^2/r^2 = (r+s)^2/r^2$$

Jellemző redundancia értékek:

rácsméret (r)	*objektumméret: s=0.5*	*s=4*
50 mm	1.02	1.16
20 mm	1.05	1.44
5 mm	1.21	3.24
2 mm	1.56	9.00

2. Állítás. Egy k x k méretű terület objektumainak megkereséséhez $(r+k)^2/r^2$ grid négyzetet kell megvizsgálni.

Bizonyítás. Pontosan az 1. állítás mintájára történik.

3. Állítás. Egy k x k méretű terület objektumainak megkereséséhez $T = D(r+s)^2(r+k)^2/r^2$ rajzelemet kell megvizsgálni, ahol D a rajzsűrűség.

Bizonyítás. Egy r x r területre $D \cdot r^2$ rajzelem esik, a tárolási redundanciát is figyelembe az 1. állítás alapján egy grid négyzet listáján $RDr^2 = D(r+s)^2$ rajzelem kell hogy legyen. A 2. állítás szerint $(r+k)^2/r^2$ négyzetet kell megvizsgálni, innen adódik a 3. állítás.

A *keresési hatékonyság*ot egy T/T_0 hányadossal mérhetjük, ahol $T_0 = Dk^2$ a keresett elemek száma. Innen

$$T/T_0 = (r+s)^2(r+k)^2/(r^2k^2)$$

A keresési hatékonyság értéke k=5 mm esetén (például szaggatott vonal felismerésnél):

rácsméret(r)	*objektumméret: s=0.5*	*s=4*
50 mm	123.43	141.13
20 mm	26.27	36.00
5 mm	4.84	12.96
2 mm	3.06	17.64

A keresési hatékonyság értéke k=40 mm esetén (például grafikus megjelenítésnél):

rácsméret(r)	*objektumméret: s=0.5*	*s=4*
50 mm	5.16	5.90
20 mm	2.36	3.24
5 mm	1.53	4.10
2 mm	1.72	9.92

A fenti adatok alapján 5 mm körüli grid méret tekinthető optimálisnak.

Adott s és k esetén az optimális r gridméret meghatározásához az $(r+s)^2(r+k)^2/r^2$ függvény minimumát keressük, ehhez a függvény deriváltjának zérushelyét kell meghatározni. A számításokat elvégezve erre a

$r + 2(s+k) - 2sk(s+k)/r^2 - 2s^2k^2/r^3 = 0$

egyenlet adódik, ennek megoldása adja az optimális r értéket.

Következmény. Az objektumok egyenletes térbeli eloszlása és (közel) azonos mérete esetén a grid index konstans elérési időt biztosít. A rácsméret optimális megválasztásával elérhető, hogy a konstans értéke a lehető legkisebb legyen.

A konstans elérési idő *statisztikai értelemben* érvényes. Vagyis, ha az objektumok méretére például normális eloszlást tételezünk fel, akkor az elérési idő várható értéke konstans. Mivel a grid index a dolgozatban nem központi jelentőségű, így a kérdés mélyebb vizsgálatától eltekintünk.

5.6. Grafikus megjelenítés

A képernyőn való gyors megjelenítés minden grafikus adatstruktúra kialakításánál fontos szempont. Megmutatjuk, hogy

1. A teljes DG adattömb lineáris időben kirajzolható (ez a hierarchikus PAT-struktúra miatt nem evidens).

2. *A kirajzolás ideje az aktuális kivágatba eső objektumok számával arányos* (és nem az összes objektum számával). Ez a gyakorlatban azt jelenti, hogy minél erősebben nagyítjuk ki a képet, annál gyorsabb a kirajzolás, real-time scrollozás is lehetséges – függetlenül a teljes adatállomány méretétől.

Az 1. állítás az 5.2. fejezetben tárgyalt öröklött rétegszámok alkalmazásából következik. A kirajzolás alapelve:

– NODE objektumokat nem rajzoljuk ki.

– EDGE és TEXT objektumokat csak egyszer rajzoljuk ki a (saját vagy öröklött) rétegszámnak megfelelő színnel – akkor is, ha több PAT-nak elemei.

– PAT objektumokat nem rajzolunk ki. Ezek azáltal jelennek meg, hogy rétegszámukat öröklik a komponens objektumok (lásd a SetPatLayer rekurzív eljárást, 5.2. fejezet).

Megjelenítéskor tehát csak az EDGE és TEXT objektumokon kell egyszer végigmenni.

A 2. tulajdonság a *grid index* alkalmazásának köszönhető.

5.7. Összefoglalás

A 4. fejezetben megvizsgáltuk a fontosabb adatmodelleket, és arra a következtetésre jutottunk, hogy térkép-interpretáció céljára egy direkt pointerezéssel dolgozó topológikus modell a legalkalmasabb. Ilyen a DG modell, amely mindössze *négyféle objektumtípussal* (NODE, EDGE, TEXT, PAT) gyakorlatilag tetszőlegesen komplex struktúrák leírását lehetővé teszi. Ennek igazolására megmutattuk, hogy az OpenGIS specifikációban szereplő objektumtípusok – a görbére épülő típusok kivételével – leírhatók DG-vel (5.1.1. alfejezet).

További konkrét példákkal szolgálunk a MAPINT rendszer felismerő algoritmusainak tárgyalásánál.

Megmutatható továbbá, hogy – amennyiben szükséges – az Arc/Info rendszer által használt, vonalláncokra épülő poligon-fedvénye (4.2.1. alfejezet) lineáris időben létrehozható DG-ből. (Ehhez a vonalláncok bejárása után a 6.4.6. alfejezetben ismertetett poligon struktúra bejárást alkalmazhatjuk.)

A NODE és EDGE típusok *gráfstruktúrát* adnak, amelyre a gráfalgoritmusok könnyen implementálhatók. Ez előnyt jelent a vonalláncokra épülő topológikus modellekkel szemben: erre példaként a MARIS rendszert említjük (3.1. fejezet), amelynél az önmagába záródó láncok elbonyolítják az algoritmusokat.

A fizikai modell szintjén nem erőltettük az adatokat fix szélességű (relációs) táblákba (4.3. fejezet), inkább az Obj és In tömbökkel megoldottuk a változó hosszúságú attribútumlisták hatékony kezelését. Megmutattuk, hogy

– kezdeti topológia létrehozásához N vektor esetén $N \cdot \log(N)$ idő szükséges,

– objektum felvétele, törlése, módosítása konstans időben végezhető (vagyis az időigény független a DG-objektumok számától).

Térbeli indexeléshez – az 5.5. fejezetben ismertetett indokok alapján – grid indexet alkalmazunk. Ennek köszönhetően a felismerések általában lineáris időben végezhetők, a grafikus megjelenítés ideje pedig csak az aktuális kivágatban szereplő objektumok számával lineárisan arányos.

6. A MAPINT térkép-interpretációs rendszer

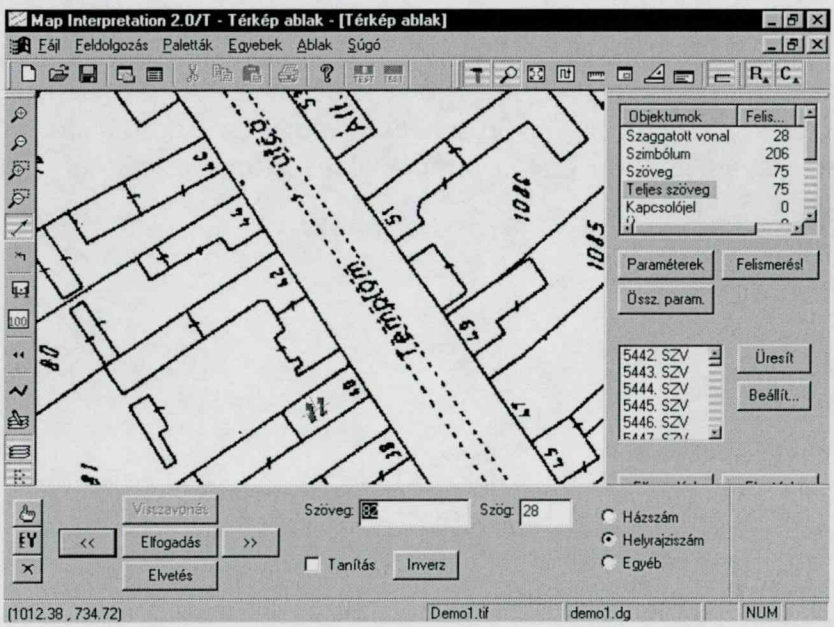
A MAPINT (MAP INTerpretation) általános célú interpretációs rendszer, amelynek jelenlegi változata elsősorban magyar kataszteri térképek feldolgozását támogatja. Fejlesztésének főbb lépései a következők voltak:

– A rendszer alapelveit és első – még DOS környezetben megvalósított – kísérleti moduljait Katona és Podolcsák (1992) mutatja be.

– A program Windows 3.1 alatti változata 1995-ben készült el (Hudra, 1995).

– Gyakorlati felhasználásra is alkalmas, jelentősen továbbfejlesztett változata már 32-bites környezetben készült el Windows 95, 98, NT platformon, a Microsoft Visual Studio 6.0 fejlesztő eszközeivel C++ nyelven (Katona, Hudra (1999a), 16. ábra).

– A rendszer beépült a Phare HU905.0203 sz. Land Consolidation Project keretében kidolgozott technológiába (Omaszta, Szabó 1999; Katona, Hudra 1999b).



16. ábra. A MAPINT rendszer képernyőképe

A rendszer az előző fejezetben ismertetett DG adatmodellre épül. Az adatmodell és a felismerő algoritmusok kidolgozását és programozását jelen dolgozat szerzője végezte, míg a rendszer felhasználói felületének kialakítása és programozása Hudra György munkája.

A fejezet további részében a MAPINT rendszert részletesen ismertetjük, majd a következő fejezetben a Phare projekt keretében történő alkalmazásról lesz szó.

6.1. A hazai földmérési alaptérképek

A magyar térképek zöme 1:1000, 1:2000, 1:4000 méretarányban készült, felépítésüket az *F7 (1983) szabályzat* határozza meg. Vetületi rendszerük *EOV (Egységes Országos Vetület)*, szelvényezésük az *EOTR (Egységes Országos Térképezési Rendszer)* szerint történik: az országot 83 db 1:100 000 méretarányú szelvény fedi le, ezek továbbosztásával adódnak a nagyobb méretarányú szelvények. Magyarország teljes területét mintegy 30 000 kataszteri szelvény fedi le.

A szokásos szelvény méret 50 x 75 cm. A térképlapon egy 10 x 10 cm-es négyzetrács metszéspontjaiban kereszteket helyeznek el, ezek az ún. *őrkeresztek* (17. ábra).

A térképállomány nem homogén, a földhivataloknál forgalomban lévő vagy rendelkezésre álló térképeket Omaszta (1998) az alábbi típusokba sorolja:

1. Régi nyilvántartási térképek. Az egész országra kiterjedő, az adózás célját szolgáló nagyméretarányú kataszteri felmérés 1856-ban kezdődött. A térképek feladata a birtokhatárok rögzítése volt. A térképezés községhatárosan történt, a külterület és belterület összefüggően. Mértékegységként a *bécsi öl*t használták, a terület mértékegysége a *kataszteri hold* volt. Az elkészült térképszelvények 1000x800 öl keretméretű, 1:2880 méretarányúak voltak. Őrkeresztek csak utólag kerültek felszerkesztésre, ezért használatuk nem követi a térképszelvényre jellemző, például papírbeszáradás okozta torzulásokat. A térképek 1884-ig vetület nélküli rendszerben készültek, ezután sztereografikus vetületi rendszerű, 1908-tól pedig ferdetengelyű hengervetületi rendszerű térképek készültek. Ez utóbbiaknál már méterrendszerű szelvényezés is megtalálható. 1927-től a felmérésben áttértek a méterrendszerre. Ettől az időszaktól a térképek méretaránya 1:1000 és 1:2000. Az egy szelvényen ábrázolt terület 800x600 méter (48 hektár), illetve 1600 x 1200 méter (192 hektár).

2. Külterületi felújított térképek. 1957-től megkezdődött a külterületi földmérési alaptérképek felújítása. A munkálatoknál a korábbi térképeket használták fel, ezért ezeknek a méretaránya megegyezik az eredetiekkel.

3. Belterületi és zártkerti ideiglenes térképek. 1962 és 1967 között egyszerűsített eljárással, csak átmeneti időre szóló belterületi és zártkerti ideiglenes nyilvántartási térképek készültek az eredeti kataszteri térképek felhasználásával. Ezek a térképek – a külterülettől függetlenül – külön szelvényeken ábrázolták a fekvéseket. Méretarányuk és vetületi rendszerük megegyezik a régi térképekével. Ezeket a térképeket csak a kataszteri térképekkel együtt lehet megbízhatóan használni.

4. Új felméréssel készült földmérési alaptérképek. 1957-től ott, ahol a régi térképek felújítása nem volt célszerű, vagy egyáltalán nem volt térkép, új felméréssel készültek a földmérési alaptérképek. Ezek méretaránya 1:1000, 1:2000 és 1:4000 volt, vetületi rendszerként az adott területen szokásos rendszert használták. 1976-tól az új felmérések és térképfelújítások egységesen EOV vetületi rendszerben készültek, felépítésüket az *F7 (1983) szabályzat* határozza meg.

A fenti heterogenitás már eleve kizárja a teljes térképállomány egységes elveken történő automatikus digitalizálását. De ha csak az állomány zömét kitevő 4. csoportot tekintjük, itt is olyan nagyszámú jelkulcsi elemmel és változatossággal találkozunk (lásd F7 (1983)), amely reménytelenné teszi a teljesen automatikus feldolgozást. (Például egy belvárosi és egy külterületi térkép méretarányban és jelölésrendszerben is alapvetően eltérő sajátosságokat mutat.)

17. ábra: 1:2000 méretarányú földmérési alaptérkép (kataszteri térkép) részlete.

A megoldás csak kompromisszum lehet: fel kell adnunk a *teljesen* automatikus feldolgozás elvét, ehelyett csak a nagy tömegben előforduló, tipikus struktúrák és jelkulcsi elemek automatikus interpretációját kell megoldanunk. Ezzel a digitalizálási munka oroszlánrészét automatizáltuk, a maradékot – az elkerülhetetlen felismerési hibákkal együtt – a manuális digitalizálásra hagyjuk.

A 17. ábra olyan alaptérkép-részletet mutat be, amely a hazai térképállomány tipikus reprezentánsának tekinthető. Ennek legfontosabb elemei a következők:

– *Földrészlet:* a földfelszín olyan összefüggő területe, amelyre a tulajdoni és kezelési viszonyok azonosak (DAT, 1996). Földrészletnek tekintendők nem csak a köznapi értelemben vett telkek, hanem a közterület különféle típusai (utca, vasút, tó, stb.) is. Alapelv, hogy földrészletek a szelvénykereten belüli teljes területet átfedés- és hézagmentesen kitöltik. A térképen a földrészletet zárt poligon határolja.

– *Épületek:* a földrészleteken belül elhelyezkedő, általában zárt poligonnal ábrázolt objektum. Az épülethez számos speciális jelkulcsi elem kapcsolódhat, főleg 1:1000 méretarányú belterületi térképeken (lépcső, árkád, lábazat, stb.).

– *Kapcsolójel:* az épületek és egyéb létesítmények területi hovatartozását fejezi ki. (Például adott épület mely földrészlethez, melléképület mely főépülethez tartozik, stb.) Jelölése: rövid hullámvonal az összekapcsolt objektumok határvonalán.

– *Helyrajzi szám:* földrészlet azonosítására szolgál, általában a megfelelő földrészlet épületen kívüli részén helyezkedik el. Megírása dőlt betűvel történik, a térkép méretarányától függő méretben.

– *Házszám:* a megfelelő főépület-poligonban helyezkedik el. Megírása álló betűvel történik, a térkép méretarányától függő méretben (általában kisebb, mint a helyrajzi szám).

– *Nullkör:* állandósított részletpontok jelölésére szolgáló, kisméretű – kb. 1 mm átmérőjű – kör (F7, 1983). Jellemzően földrészlet sarokpontján helyezkedik el.

A MAPINT rendszernél a fenti térképi objektumok felismerését tűztük ki célul.

6.2. A feldolgozás menete

A feldolgozás során mindig egy raszteres (TIFF grafikus formátumú) és vektoros (DG formátumú) állományt kezelünk együtt, amelyek a képernyőn egymásra vetítve jelennek meg. Egy szkennelt (raszteres) térképszelvényből indulunk ki, ilyenkor a DG még üres. A feldolgozás az alábbi fő lépésekből áll:

1. Koordináta-transzformáció. A raszteres állományon meghatározzuk az őrkeresztek koordinátáit, amelyek a (most még üres) DG-be kerülnek letárolásra. Ezután affin transzformációval a szelvénykereten kívüli részt levágjuk. Az így transzformált raszteres állományok hézagmentesen illeszthetők egymáshoz.

2. Nyers vektorizálás. A 2.2. alfejezetben ismertetett eljárással előállítjuk a rajz nyers vektoros képét, amely NODE–EDGE gráfként a DG-be kerül.

3. Felismerések. Az egyes térképi objektumok automatikusan felismerésre kerülnek, minden felismerési lépés utáni interaktív javításra van lehetőség. Ezen műveletek során csak a DG változik.

4. DXF export. A generált vektoros DG adatstruktúra fájlra írása DXF formátumban.

6.3. Koordináta-transzformáció

Térinformatikai felhasználáshoz a szkennelt szelvényt a megfelelő vetületi rendszerbe kell transzformálni. Ez olyan műveletet jelent, amelynek során *a szelvénykereten kívüli rész levágásra kerül*, a transzformált szelvények pedig hézag- és átfedésmentesen egymáshoz illeszthetők. Ez a művelet nem tartozik a térkép-interpretációhoz, és lényegében bármely GIS szoftverrel elgévezhető. Mégis úgy döntöttünk, hogy a MAPINT rendszerben a transzformációt is megvalósítjuk, ezzel

– önálló, más szoftvertől független technológiát biztosítva, és
– lehetővé téve a hazai térképállományhoz való maximális alkalmazkodást.

A megvalósítás során olyan megoldást sikerült kidolgozni, amely a hazai térképek kezelését hatékonyabban biztosítja más rendszereknél, ezért alább bemutatjuk azt.

A transzformációt a raszterképen végezzük *kontrollpont alapú affin transzformációval.* Az eljárás elméleti alapjait a 9.2.4. alfejezet tárgyalja, itt csak a formulákat adjuk meg. Egy

$$x' = f_x(x, y) = a_0 + a_1 x + a_2 y$$
$$y' = f_y(x, y) = b_0 + b_1 x + b_2 y$$

alakú transzformációt keresünk, amely az (x_1, y_1), ..., (x_m, y_m) kontrollpontokat az (x_1', y_1'), ..., (x_m', y_m') pontokra a legkisebb négyzetek módszere szerint képezi le, vagyis úgy, hogy

$$\Sigma_i \, (f_x(x_i, y_i) - x_i')^2 + (f_y(x_i, y_i) - y_i')^2$$

minimális legyen. A transzformáció együtthatóinak meghatározásához az

$$U = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_m & y_m \end{bmatrix}$$

mátrixot képezzük, majd ebből a

$$D = (U^T \cdot U)^{-1} \cdot U^T$$

mátrixot számítjuk. Ezután az együtthatók

$$\begin{bmatrix} a_{00} \\ a_{10} \\ a_{01} \end{bmatrix} = D \cdot \begin{bmatrix} x_1' \\ x_2' \\ \vdots \\ x_m' \end{bmatrix} \qquad \begin{bmatrix} b_{00} \\ b_{10} \\ b_{01} \end{bmatrix} = D \cdot \begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_m' \end{bmatrix}$$

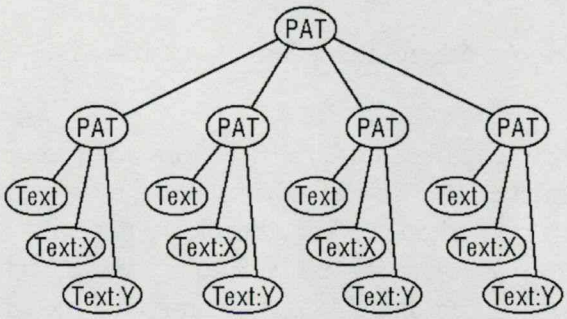
módon adódnak.

Kontrollpontként a szelvénykeret sarokpontjai és az őrkeresztek szolgálnak. Megadásuk manuálisan történik, amely művelet támogatható egy, az őrkeresztek koordinátáit tartalmazó text fájl (röviden *őrkereszt fájl*, lásd 7. fejezet) importálásával.

Természetesen a sarokpontok és őrkeresztek automatikus felismerése is lehetséges lenne. Itt azonban 100%-os felismerési biztonságra van szükség, hiszen egyetlen őrkereszt hibás felismerése is súlyos torzulást okoz a transzformációnál. Az automatikus felismerést tehát mindenképp ellenőrizni kellene, ez a művelet pedig semmivel sem jelent kevesebb manuális munkát, mint az alább leírt, a rendszer által erősen támogatott őrkereszt megadási mód.

Először a szelvénykeret *négy sarokpontját* kell a raszterképen rákattintással megjelölni, majd az egyes sarokpontokhoz geodéziai koordináták adhatók meg. EOV vetületi rendszer esetén elegendő a bal alsó és jobb felső sarok koordinátáinak megadása, őrkereszt-fájl importálása esetén pedig ez is fölösleges.

A fenti művelet eredményeként a (most még üres) DG-ben a 18. ábra szerinti struktúra kerül felvételre. Ebben minden sarokponthoz egy PAT(TEXT, $TEXT_x$, $TEXT_y$) struktúra tartozik, ahol TEXT a sarokpontot jelző kereszt szimbólum, $TEXT_x$ és $TEXT_y$ pedig a bevitt koordinátákat tárolják (height=0 attribútummal, vagyis a rajzon a koordináták nem jelennek meg).



18. ábra. A négy sarokpont adatait tartalmazó DG-adatstruktúra

A sarokpontok közötti pixel-távolságok alapján a rendszer meghatározza a leendő transzformált raszterkép s_x, s_y méreteit (pixelben) – úgy, hogy az minimális torzulást szenvedjen. Ezután meghatározzuk azon T_1 affin transzformáció együtthatóit, amely a négy sarokpontot minimális hibával a leendő kép sarkaira képezi.

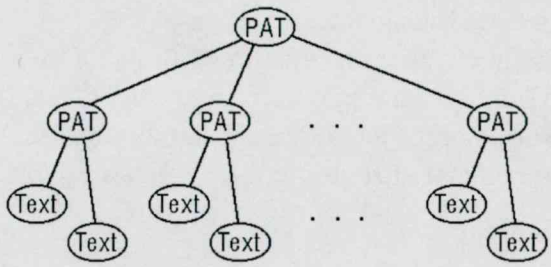
Ha az aktuális szelvényen nincsenek őrkeresztek, akkor a T_1 transzformációt kell végrehajtani. Őrkeresztek esetén egy pontosított transzformációt hajtunk végre a következőképpen:

1. Először az őrkeresztek (X_1, Y_1), ..., (X_m, Y_m) geodéziai koordinátáit határozzuk meg, amely információ vagy az őrkereszt fájlból nyerhető, vagy a sarokpont koordináták, a vetületi rendszer és a méretarány ismeretében határozható meg.
2. Számítjuk az őrkeresztek (x_1', y_1'), ..., (x_m', y_m') pixel-koordinátáit az – egyelőre még csak képzeletbeli – output képen.
3. A T_1 transzformáció inverzével meghatározzuk az őrkeresztek feltételezett (x_1'', y_1''), ..., (x_m'', y_m'') pixel-koordinátáit az eredeti képen.
4. A DG-ben létrehozzuk a 19. ábra szerinti őrkereszt struktúrát. Itt minden őrkereszthez egy PAT($TEXT_1$, $TEXT_2$) struktúra tartozik, ahol a két TEXT egymást fedő kereszt-szimbólumokként jelentik meg a rajzon, az i-edik őrkereszt esetén a fent számított (x_i'', y_i'') koordinátájú pontban.

5. A felhasználó rákattintással pontosítja az őrkeresztek helyét. Ennek során $TEXT_1$ az (x_i'', y_i'') koordinátájú pontban marad (fix őrkereszt), míg $TEXT_2$ egy a felhasználó által megadott (x, y) helyre kerül (mozgó örkereszt).

6. Meghatározzuk annak a T_2 affin transzformációnak az együtthatóit, amely a fent számított (x_1, y_1), ..., (x_m, y_m) kontrollpontokat minimális hibával az (x_1', y_1'), ..., (x_m', y_m') pontokra viszi át.

7. A T_2 transzformáció végrehajtása a raszterképen.



19. ábra. Az őrkeresztek adatait tartalmazó DG-adatstruktúra

Összefoglalva, sarokpontokra történő transzformáció esetén a T_1, őrkeresztek esetén a T_2 transzformációt alkalmazzuk. Ezeket közös néven most f transzformációnak nevezzük, amelynek végrehajtása a raszterképen a következőképp történik.

– Tulajdonképpen f *inverzével* kell számolni, hiszen minden egyes (x', y') output pixelhez az ősének (x, y) koordinátáit kell számítani. Az output pixel értékét *nearest neighbour módszer*rel határozzuk meg, vagyis azon egész koordinátájú pont értékét vesszük, amely (x, y)-hoz legközelebb esik.

– A transzformációt *partícionálva* végezzük, vagyis rögzített méretű memóriaterületen tetszőlegesen nagy mátrixok transzformálhatók. Ennek lényege, hogy f^{-1} segítségével meghatározzuk egy téglalap alakú output partíció ősének befoglaló téglalapját, és az input képnek csak ezen szegmensét töltjük a memóriába, amelyből számítható az output partíció. (Megjegyzendő, hogy magasabbfokú polinomiális transzformáció (9.2.4. alfejezet) esetén a partícionálás már komolyabb meggondolásokat igényel.)

– A partíció egyes pixeleit sorfolytonosan haladva számoljuk. Vegyük észre, hogy ha

$$x = f_x^{-1}(x', y') = a_0 + a_1 x' + a_2 y'$$
$$y = f_y^{-1}(x', y') = b_0 + b_1 x' + b_2 y'$$

akkor

$$f_x^{-1}(x'+1, y') = f_x^{-1}(x', y') + a_1$$
$$f_y^{-1}(x'+1, y') = f_y^{-1}(x', y') + b_1$$

vagyis, ha egy pixel ősének koordinátáit már meghatároztuk, a következő pixel ősének koordinátái ebből egyszerű inkrementálással adódnak. Ezzel a módszerrel a számításigény drasztikusan csökken (nem kell szorozni, csak összeadni).

– Fixpontos számolást alkalmazunk, ami különös gondot igényel a kerekítési hibák tekintetében, de lényegesen gyorsabb.

A fenti módszerek együttes alkalmazásának köszönhetően a transzformáció *lényegesen gyorsabb*, az őrkeresztek manuális pontosítása pedig hatékonyabb például a MicroStation Descartes rendszer megfelelő moduljánál.

6.4. Felismerő eljárások

A MAPINT a 6.1. fejezetben felsorolt térképi objektumokat ismeri fel (földrészletek, épületek, helyrajzi számok, házszámok, kapcsolójelek, nullkörök), ennek megfelelően a 20. ábra szerinti DG-rétegeket használjuk. Kezdetben a DG csak NODE és EDGE objektumokat tartalmaz a 0 rétegben. A felismerés során létrejövő TEXT és PAT objektumok kapják a fenti rétegjelzéseket, az EDGE-ek mindvégig a 0 rétegben maradnak.

Név	*Szám*	*Megjegyzés*
Definiálatlan	0	Fel nem ismert vektorok
Szimbólum	1	Leválogatott szimbólum
Szimbólumcsoport	2	Csoportosított szimbólum
Házszám	20	Felismert házszám
Helyrajziszám	21	Felismert helyrajziszám
Felirat	22	Felismert egyéb felirat
Szaggatott vonal	3	Leválogatott szaggatott vonal
Kapcsolójel-1	4	Felismert kapcsolójel
Kapcsolójel-2	24	Véglegesített kapcsolójel
Üreg	12	Felismert üreg
Nullkör-1	5	Felismert nullkör
Nullkör-2	25	Véglegesített nullkör
Korrekció	11	Felismert rajz korrekció
Épület	26	Épület poligon
Földrészlet	27	Földrészlet poligon
Poligon	6	Egyéb poligon
Sarokpont-1	31	Sarokpont jel
Sarokpont-2	32	Sarokpont koordináta
Kontrollpont-1	33	Fix őrkereszt
Kontrollpont-2	34	Mozgó őrkereszt

20. ábra. A felismerő algoritmusok által használt rétegszerkezet

A felismerő műveletek csak a szabad (még fel nem ismert) élekre hajtódnak végre, ezért az öröklött rétegszámok beírását végző *SetPatLayer* eljárást (5.2. fejezet) minden felismerő algoritmus előtt végre kell hajtani.

Minden egyes felismerési művelet az alábbi lépésekből áll:

1. Automatikus felismerés. Hatására a DG-ben új objektumok jönnek létre, amelyek "felismert" jelzést kapnak (lásd bitmaszk, 5.2. fejezet).

2. Manuális ellenőrzés. A "felismert" jelzésű objektumok elfogadhatók, elvethetők, vagy javíthatók. Ennek során kerülnek beállításra a "ellenőrzött" és "elfogadott" jelzések (lásd bitmaszk, 5.2. fejezet), más a DG-ben nem változik. Ez lehetővé teszi, hogy a manuális ellenőrzés során a felhasználó korlátlanul helyesbíthet ("undo" funkció).

3. Felismerések véglegesítése. Itt három lehetőségből választhatunk:

3.1. Az ellenőrizetlen objektumokat elfogadjuk. Ezt akkor alkalmazzuk, ha a felismerés nagy biztonsággal történt, és manuális javításra nincs szükség.

3.2. Az ellenőrizetlen objektumokat elvetjük. Ezt általában akkor alkalmazzuk, ha – például hibás paraméterezés miatt – a felismerés alapvetően rosszul sikerült.

3.3. Csak az ellenőrzötteket véglegesítjük.

A véglegesítés hatására a "felismert" jelzések törlődnek, és az újonnan felismert objektumok – most már visszavonhatatlanul – beépülnek a DG-be.

A következőkben a felismerő műveleteket olyan sorrendben tárgyaljuk, amilyenben azok végrehajtása célszerű. Minden műveletnél megadjuk

– a *paramétereket*, amelyek a feldolgozást vezérlik,

– az *algoritmus* vázlatát,

– a *műveletigényt N* függvényében, ahol *N* a DG-ben lévő szögpontok illetve élek száma (az esetünkben feldolgozott síkgráfoknál az élek és szögpontok száma között nagyságrendi eltérés nem lehet),

– az *ellenőrzés* módját, ha az eltér az egyszerű elfogadás/elvetéstől,

– a *véglegesítés* hatására a DG-ben történő változásokat.

6.4.1. Szaggatott vonalak felismerése

Paraméterek:

T (tolerance): a szaggatott vonal darabok egy egyenesbe esésének vizsgálatánál a megengedett legnagyobb eltérés értéke pixelben. Ha nagyobb értéket adunk meg, akkor "kanyargósabb" szaggatott vonalakat is felismer a rendszer, viszont előfordulhat, hogy más rajzi elemeket is szaggatott vonalnak tekint.

D (distance): két szaggatott vonal szakasz között megengedett maximális távolság pixelben.

Algoritmus:

Egymáshoz közeli *A*, *B* végpontokat (vagyis olyan NODE-okat, amelyekből csak egy él indul) keresünk. Jelölje A_1 és B_1 az *A*-ból ill. *B*-ből induló él másik végpontját, ekkor az élek egy egyenesbe esését a

$$Dev = \max(\, \mathrm{dist}(A, A_1B_1), \mathrm{dist}(B, A_1B_1)\,)$$

értékkel jellemezzük, ahol $\mathrm{dist}(A, A_1B_1)$ az *A* pontnak az A_1B_1 egyenesszakasztól való távolságát jelenti. Az (*A*, *B*) pontpár akkor felel meg céljainknak, ha $\mathrm{dist}(A, B) < D$ és $Dev < T$ teljesül. Ha adott A ponthoz több ilyen *B* pontot is találunk, akkor azt választjuk, amelyre a

$$V = \text{dist}(A, B) + 2 \cdot Dev$$

mérőszám minimális, vagyis ilyen esetben az egy egyenesbe esést preferáljuk a közelséggel szemben. Az (A, B) pontpárt csak akkor fogadjuk el, ha a fenti kritérium alapján nem csak A-hoz B a "legközelebbi", hanem B-hez is A a "legközelebbi".

Szaggatott vonalnak tekintünk minden olyan vektorsorozatot, ahol a végpontpárok a fenti kritériumrendszernek megfelelnek. A feldolgozás az első, még nem vizsgált végpontból indul, a már vizsgált végpontok ideiglenes "feldolgozva" jelzést kapnak. Triviális hibák elkerülése érdekében csak legalább három vonalszakaszból álló szaggatott vonalat fogadunk el. A felismert szaggatott vonal egy

$$\text{PAT}(\text{EDGE}_1, ..., \text{EDGE}_n)$$

objektum formájában kerül felvételre a DG-be ("szaggatott vonal" rétegjelzéssel).

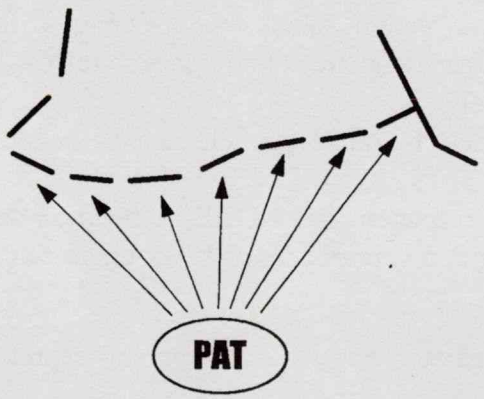
Megjegyezzük, hogy Chen és tsai (1996) részletesen dokumentált eljárást ismertet szaggatott vonalak felismerésére műszaki rajzokon. Ebből az eljárásból itt két jellemzőt emelünk ki:

– Különös gondot fordít szaggatott körök, körívek felismerésére. (Térkép-interpretációnál ennek csekély a jelentősége.)

– A T tolerancia érték helyett maximális szögeltérést vizsgál. Esetünkben azért döntöttünk az egyenestől való eltérés mérése mellett, mert a – magyar kataszteri térképeken gyakran szereplő – *rövid szaggatott vonalak* esetén már egy pixel eltérés is drasztikus szögváltozást eredményez.

Műveletigény:

Az algoritmus kritikus pontja a keresés: adott végponthoz közeli másik végpont keresése térbeli indexelés nélkül tolerálhatatlan, $O(N)$ időt igényelne, grid indexszel viszont konstans időben végezhető. Az 5.5.2. fejezet jelöléseire hivatkozva k=5 keresési környezetet, r=5 gridméretet és D=0.15 rajzsűrűséget feltételezve a vizsgálandó elemek száma $Dk^2 = 3.75$, a keresési hatékonyság 4.84, tehát a egy keresés ideje átlagosan 3.75*4.84 = 18.15 rajzelem vizsgálatát jelenti. Összességében a szaggatott vonal felismerés grid indexszel *lineáris időben* megoldható.



21. ábra. Szaggatott vonal felismerés szemléltetése

Véglegesítés. A felismerés *elfogadása* esetén a szaggatott vonal PAT-ok megmaradnak, *elvetés* esetén törlődnek (a szaggatott vonal vektorai az első esetben nem vesznek részt, a második esetben részt vesznek a további felismerésben).

Az algoritmus előnyei:

– Hajlított szaggatott vonalak (például szintvonalak) felismerésére is alkalmas. Ha a szaggatott vonalon éles törés van, akkor több szaggatott vonalként (szaggatott vonallánccként) ismeri fel az algoritmus.

– T-elágazásból induló szaggatott vonalakat is felismer (21. ábra).

Az algoritmus hiányossága, hogy nem különíti el a különféle szaggatott vonaltípusokat, mivel jelenleg az algoritmus elsődleges célja a szaggatott vonalak *leválogatása* (vagyis kizárása a további felismerésekből), és nem azok pontos azonosítása. (Az egyes vonaltípusok elkülönítése egyébként a felismert szaggatott vonal szakaszhosszainak vizsgálatával viszonylag könnyen megoldható.)

6.4.2. Szimbólumok leválogatása

Szimbólumon a megírások karaktereit és olyan jelkulcsi elemeket értünk, amelyek önállóan (tehát nem vonalon, vagy más jelkulcsi elemhez kapcsolva) helyezkednek el. A művelet ezeket keresi meg, előkészítve ezzel a további feldolgozást.

Paraméterek:

S_{min}: *minimális méret.* Az ennél kisebb méretű alakzatok nem kerülnek leválogatásra.

S_{max}: *maximális méret.* Az ennél nagyobb méretű alakzatok nem kerülnek leválogatásra.

Algoritmus:

A DG-ben olyan összefüggő részgráfot keresünk, amely szeparált a gráf többi részétől. Ha a részgráf befoglaló téglalapjának hosszabbik mérete S_{min} és S_{max} közé esik, akkor szimbólumnak tekintjük, és egy, az éleit tartalmazó

$$PAT(EDGE_1, ..., EDGE_n)$$

objektumot veszünk fel a DG-be.

A részgráf bejárása verem (stack) segítségével történik. A gráf éleit járjuk be, amelyiknél már voltunk, az ideiglenes "feldolgozva" jelzést kap. A verembe viszont NODE azonosítók kerülnek. Az eljárás:

– Keresünk egy még be nem járt élt, ennek kezdőpontjának azonosítója kerül a verembe.

– Kivesszük a verem tetején lévő NODE-ot, bejárjuk az ebből induló éleket, miközben minden él másik végpontját a verembe tesszük. Az eljárást addig ismételjük, amíg a verem ki nem ürül.

Műveletigény. Belátható, hogy a fenti verem-technikával a bejárás lineáris időben elvégezhető.

Véglegesítés. A felismerés *elfogadása* esetén a szimbólum PAT-ok megmaradnak, *elvetés* esetén törlődnek. A gyakorlatban a szimbólum leválogatást ellenőrzés nélkül el lehet

fogadni, mivel a következő lépés – a megírások felismerése – mindenképp ellenőrzést igényel, és azzal a leválogatást is ellenőrizzük.

6.4.3. Megírások felismerése

A feliratok általában több karakterből állnak, ezért először meg kell keresni az egy stringet alkotó szimbólumokat, és utána következhet a felismerés.

Paraméterek:

D (distance): Két szomszédos szimbólum befoglaló téglalapjának bal felső sarka közötti maximális megengedett távolság.

T (tolerance): az egy egyenesbe esés vizsgálatánál a szimbólum-középpontoknak az egyenestől megengedett maximális eltérése.

β: preferált szög. A szkennelt rajzon a feliratok jellemző iránya. Ez általában 0 fok, ha alaphelyzetben szkennelték az anyagot (mivel a feliratok többsége talpon áll), 90°, ha az óramutató járásával ellentétesen 90°-kal elforgatva történt a szkennelés, és 270° az óramutató járása szerinti elforgatás esetén.

K: elfogadási küszöb (0 és 1 közötti érték). A felismerő eljárás minden egyes karakterhez egy 0 és 1 közötti felismerési értéket rendel, amely a felismerés biztonságát fejezi ki. Egy szimbólumot akkor tekint a rendszer felismertnek, ha felismerési értéke meghaladja *K*-t.

Az algoritmus vázlata:

1. Leválogatott szimbólumok csoportosítása. Egy csoportba kerülnek azok a szimbólumok, amelyek egymáshoz *D*-nél közelebb állnak. Három eset lehetséges:

– Több elemű csoport, amelynek elemei a *T* tolerancián belül egy egyenesbe esnek. Legyen az egyenes elforgatási szöge α. Az ilyen szimbólumcsoporthoz a DG-ben a 22. ábrán látható struktúrát hozzuk létre, vagyis egy

$$PAT(PAT_1, ..., PAT_n, TEXT_1, TEXT_2)$$

alakzat kerül felvételre, amely az egy csoportba tartozó szimbólum-PAT-okat tartalmazza, a két TEXT objektum pedig a felismerés eredményét fogja tartalmazni talpon állva illetve fejjel lefelé (vagyis α ill. $180°+\alpha$ elforgatási szöggel). A TEXT-ek most még üresek, de az elforgatási szöget már tartalmazzák.

– Több elemű csoport, amelynek elemei nem esnek egy egyenesbe (*T* figyelembevételével), az ilyen csoporthoz csak egy TEXT objektum jön létre.

– Egy elemű csoport, ehhez szintén csak egy TEXT objektum jön létre.

2. A szimbólumok felismerése egyesével, mesterséges neurális hálózattal történik (Russell, Norvig 2000). A felismerés lépései:

– Egy munkatömbbe másoljuk a felismerendő PAT_i szimbólum vektorait.

– A szimbólumot $-\alpha$ szöggel (alaphelyzetbe) forgatjuk.

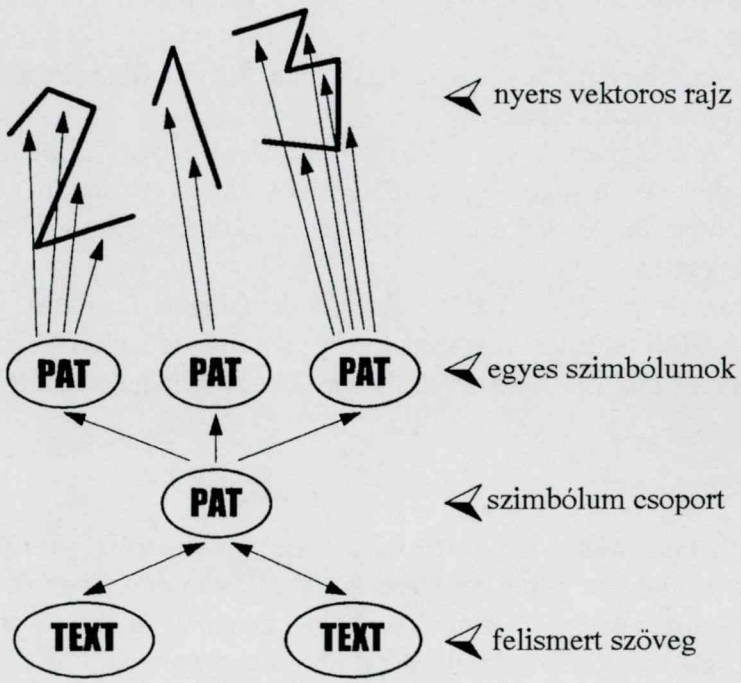
– Jellemzők kivonása (feature extraction) a neurális háló számára.

– Felismerés neurális hálóval (ennek részleteit a következő alfejezet tartalmazza). A felismerés $TEXT_1$-be kerül.

– A fentieket elvégezzük a szimbólumcsoport valamennyi elemére, az egyes elemekre kapott (0, 1 közötti) felismerési értékeket összegezzük (s_1).

– A fentieket elvégezzük $180+\alpha$ elforgatási szög esetére is. A felismerés eredménye ezúttal $TEXT_2$-be kerül, a kapott felismerési összeg s_2.

– Összehasonlítjuk az α és $180+\alpha$ elforgatással végzett felismerés eredményét. Legyen d_1 és d_2 rendre az α és $180+\alpha$ szögek abszolút értékben vett eltérése a preferált béta szögtől, ezen értékek inverzével súlyozzuk a felismerési értékeket. (A nullával osztás elkerülésére a következőt alkalmazzuk: $s_1' = s_1 d_2$, $s_2' = s_2 d_1$.) A jobb felismerési összegű TEXT-et megjelöljük.



22. ábra. Megírások felismerése

Műveletigény. Tekintettel a felismerési folyamat rendkívüli összetettségére, a műveletszám numerikus meghatározására nem vállalkozunk. A gyakorlatban a lineáris idejű algoritmusokhoz hasonló futási időket mértünk, tehát az eljárás gyors.

Ellenőrzés, javítás. A megírások felismerésének felhasználói ellenőrzése – csakúgy, mint más rendszereknél – feltétlen javasolt. Ennél az alábbi lehetőségeink vannak:

– Váltás $TEXT_1$ és $TEXT_2$ között – amennyiben a rendszer fordítva felismert változatot preferálta.

– TEXT javítása.

– Réteg megadása. Jelenleg a "házszám", "helyrajziszám" és "felirat" rétegek között választhatunk aszerint, hogy a felismert TEXT-et melyik rétegbe kell sorolni.

– Tananyagba felvétel. Akkor célszerű, ha korrekten rajzolt karaktert a neurális háló hibásan ismer fel. Ilyenkor a jellemzők vektora és az – általunk megadott – helyes karakter egy tananyag-fájlba mentésre kerül, amellyel a neurális háló tovább tanítható.

Véglegesítés. Ha egy szöveg felismerését *elfogadtuk*, akkor az elfogadott TEXT kivételével a 22. ábrán látható valamennyi objektum (beleértve a szimbólumok eredeti vektorait is) törlődik a DG-ből. Ha a szöveg felismerését elvetjük, akkor csak a szimbólumok eredeti vektorai maradnak meg, a 22. ábrán szereplő valamennyi PAT és TEXT törtődik.

6.4.4. Karakterfelismerés neurális hálóval

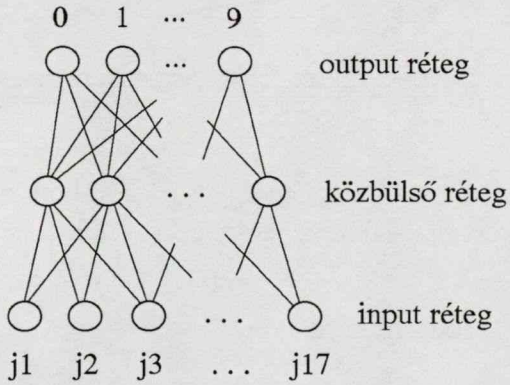
Karakterek felismerését általában raszteresen szokták végezni (Trier és tsai, 1996). Kétségtelen, hogy a vektorizálással információt vesztünk, mégis a vektoros felismerés mellett döntöttünk az alábbiak miatt:

– a vektorizálás már egyfajta lényegkiemelést jelent,

– a vektoros adat torzulásmentesen transzformálható elforgatott helyzetből alaphelyzetbe,

– így egységes adatstruktúrát (DG) használhat valamennyi felismerő algoritmus.

A felismeréshez *feedforward neurális hálózatot* alkalmazunk *back-propagation* tanuló algoritmussal (Russell, Norvig 2000). A hálózat jelenleg csak számjegyek felismerését végzi (a kataszteri térképeken ebből van a legtöbb): 3 rétegből épül fel, 17 elemű input vektort fogad és – a 0, 1, ..., 9 számjegyeknek megfelelően – 10 elemű output vektort generál (23. ábra).



23. ábra. A felismerést végző neurális háló vázlata

Neurális felismerésnél kulcsfontosságú a bemenő vektort alkotó jellemzők kiválasztása. Ehhez a felismerendő karaktert, mint DG-részgráfot vizsgáljuk. A vektor elemei a következők:

1. elem: élek (EDGE objektumok) száma.

2. elem: szögpontok (NODE objektumok) száma.

3. elem: végpontok száma (vagyis olyan NODE-ok száma, amelyekből csak egy él indul).

4. elem: töréspontok száma (vagyis olyan NODE-ok száma, amelyekből két él indul).

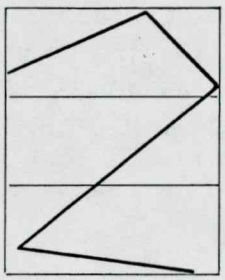
5. elem: elágazási pontok száma (vagyis olyan NODE-ok száma, amelyekből kettőnél több él indul).

A bemenő vektor további elemeinek meghatározásához osszuk 3 egyenlő szegmensre a felismerendő karakter befoglaló téglalapját (24. ábra). Adott szegmensbe eső végpontok közül válasszuk ki azt, amelyikből a leghosszabb vektor indul! Legyen ezen vektor kezdőpontja (x_1, y_1) és végpontja (x_2, y_2), ekkor az alábbi "max-vektort" határozzuk meg:

$$[\ \max(0, x_1-x_2),\ \max(0, y_1-y_2),\ \max(0, x_2-x_1),\ \max(0, y_2-y_1)\ ]$$

Ezek után a neurális háló bemenő vektorának további elemei:

6.-9. elem: a felső szegmens max-vektora.
10.-13. elem: a középső szegmens max-vektora.
14.-17. elem: az alsó szegmens max-vektora.



24. ábra. Karakter felosztása szegmensekre

A felismerés eredményének azt a számjegyet fogadjuk el, amelynek megfelelő neurális kimenet a legnagyobb értéket adja – feltéve, hogy ez az érték elérte a *K* elfogadási küszöböt.

6.4.5. Kapcsolójelek felismerése

Paraméterek:

T: tolerancia érték. A kapcsolójel hordozóél szakaszainak egy egyenesbe esési vizsgálatánál megengedett eltérés mértéke.

S_{max}*: maximális méret.* A kapcsolójel megengedett maximális mérete.

K: elfogadási küszöb (százalék). Minden egyes kapcsolójel felismeréskor a rendszer egy százalék értéket határoz meg, amely a felismerés biztonságát fejezi ki. A *K*-nál kisebb százalékban felismert kapcsolójelek automatikusan elvetésre kerülnek.

Algoritmus:

A 25. ábrán látható a kapcsolójelek három jellegzetes előfordulási típusa a vektorizált rajzon. Leggyakrabban a *b)* típus szerepel, ritkább a *c)*, az *a)* változattal pedig csak elvétve találkozunk.

b) típus felismerése: olyan rövid élt keresünk, amelynek mindkét végpontjából 3 él indul. A kiinduló vonalláncok vizsgálata alapján döntünk, hogy kapcsolójelről van-e szó. A feldolgozott éleket megjelöljük.

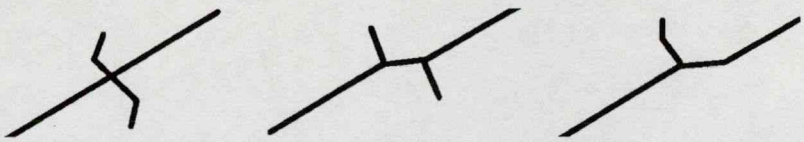
c) típus felismerése: olyan rövid élt keresünk, amelynek egyik végpontjából 3, a másikból 2 él indul. Itt is a kiinduló vonalláncok vizsgálata alapján döntünk. Mivel a b) típus felismerése és megjelölése már megtörtént, így nem fordulhat elő, hogy egy b) típust két c) típusnak ismerjünk fel.

a) típus felismerése: olyan NODE-ot keresünk, amelyből 4 él indul. A kiinduló vonalláncok vizsgálata alapján döntünk.

Minden felismert kapcsolójelhez egy

$$PAT(EDGE_1, ..., EDGE_r, EDGE_{r+1}, ..., EDGE_{r+s})$$

alakzat kerül felvételre a DG-be, ahol $EDGE_1$, ..., $EDGE_r$ a kapcsolójel szárait alkotó élek (ezeket elfogadás esetén törölni kell), $EDGE_{r+1}$, ..., $EDGE_{r+s}$ pedig a holdozóél szakaszai (ezeket elfogadás esetén egyesíteni kell). (Az utóbbiakat a PAT "obj"-listáján (12. ábra) negatív hivatkozás különbözteti meg az előbbiektől.)



25. ábra. Kapcsolójel három megjelenési formája a nyers vektoros rajzon

Műveletigény: belátható, hogy a fenti felismerés lineáris időben végezhető.

Véglegesítés. Elvetés esetén a kapcsolójel PAT törlődik, az élek változatlanok maradnak. Elfogadás esetén a kapcsolójel PAT-ban szereplő élek törlésre illetve egyesítésre kerülnek (lásd fent), ezután a PAT törlődik, majd egy speciális karakterből álló TEXT kerül felvételre, amely a kapcsolójel szimbólumot hullámvonal formájában jeleníti meg. A TEXT és a kapcsolójel hordozóélnek megfelelő EDGE között kölcsönös hivatkozási kapcsolat jön létre.

Bár a kapcsolójelek felismerési arányi igen jó, manuális ellenőrzés mégis javasolt, mert minden hiba zavart okoz az épület/földrészlet felismerés során.

6.4.6. Poligon struktúra bejárása

A későbbiekben többször szükség lesz olyan algoritmusra, amely zárt poligonokat keres a gráfban. Alább ilyet ismertetünk.

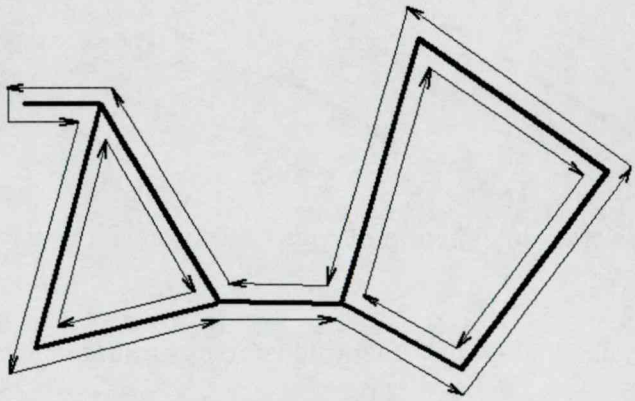
Legyen $G = (V, E)$ tetszőleges irányítatlan síkgráf, ahol V a szögpontok halmaza, $E \subseteq V \times V$ pedig az élek halmaza. Az eljárás során minden $e = (v_i, v_j)$ élet kétszer címkézünk: $v_i \rightarrow v_j$ irányú bejáráskor *f* (forward), $v_j \rightarrow v_i$ irányú bejáráskor *b* (backward) címkével.

Algoritmus poligon struktúra bejárására (26. ábra):

1. Keresünk egy élet, amely még nincs címkézve valamelyik irányban. (Ha már minden él mindkét irányban címkézve van, akkor készen vagyunk.)

2. Az élen a címkézetlen irányban indulunk, elágazásnál *jobbkéz szabály* szerint haladunk tovább. Ez azt jelenti, hogy elágazásnál a leginkább jobbra kanyarodó élt választjuk. (Végpontnál visszafordulunk.) Az érintett éleket a bejárt irányban mindig címkézzük. Addig haladunk, amíg a kiindulási élhez vissza nem érünk.

3. Folytatás 1-gyel.



26. ábra. Példa gráf bejárására

Állítás. A fenti algoritmus nem akad el, vagyis a 2. lépésében a jobbkéz szabály garantálja, hogy előrehaladásunk során mindig az adott irányban címkézetlen élt találunk, mindaddig, amíg vissza nem jutunk a kiindulási élhez.

Bizonyítás. Alakítsuk irányítottá a G gráfot úgy, hogy minden $e = (v_i, v_j)$ élből egy $e_f = (v_i, v_j)$ és $e_b = (v_j, v_i)$ élpár keletkezzen, így kapjuk az E_f és E_b élhalmazokat. A teljes gráf élhalmaza $E_f \cup E_b$ lesz. A jobbkéz szabály minden élhez egy követőt határoz meg, vagyis egy

$$T: (E_f \cup E_b) \rightarrow (E_f \cup E_b)$$

leképezést definiál. Ez egy-egyértelmű leképezés, hiszen minden élhez egyértelműen megadható annak megelőzője (ezt balkéz szabálynak lehetne nevezni). Ebből következik, hogy a T leképezés az $E_f \cup E_b$ véges halmazt diszjunkt körökre bontja. A fenti bejáró algoritmus sorra ezeket a köröket járja be. Mivel a körök diszjunktak, és új kört csak akkor kezdünk, ha a megkezdett kört már bejártuk, így bejárás közben nem juthatunk az adott irányban már címkézett (bejárt) élhez.

Következmények:

– Az algoritmus valamennyi zárt poligont körüljár az óramutató járása szerint. (Különálló poligonokat az ellenkező irányban is bejárja, lásd 29. ábra.)

– Nem csak poligonok, hanem más speciális alakzatok is bejárásra kerülnek (26. ábra).

– Az algoritmus lineáris idejű, mivel $E_f \cup E_b$ minden elemét pontosan egyszer (tehát az eredeti E minden elemét pontosan kétszer) érintjük.

6.4.7. Üregek felismerése

A nyers vektoros rajzon rövid élek által alkotott apró zárt poligonok lehetségesek, amelyek az eredeti raszteres állomány hibáiból (pixel kimaradás), illetve speciális rajzi szituációkból adódhatnak (például kis szögben találkozó vonalak esetén). Ezeket az "üregeket" meg kell szüntetni, vagyis a határoló éleiket egyetlen pontra összehúzni.

Paraméter:

S_{max}: *maximális méret.* Az üreg maximális átmérője pixelben.

Algoritmus. A poligon stuktúra bejáró algoritmust alkalmazzuk úgy, hogy minden bejárt alakzatra a következő feltételeket vizsgáljuk:

– konvexség: akkor teljesül, ha a körbejárás során a jobbkéz szabály szerint kiválasztott él ténylegesen jobbra fordul.

– méretkorlát: az alakzat befoglaló téglalapjának hosszabb oldala nem lehet nagyobb mint S_{max}.

Ha egy alakzat a fenti két feltételnek eleget tesz, akkor az éleit tartalmazó

$$\mathrm{PAT}(\mathrm{EDGE}_1, \ldots, \mathrm{EDGE}_n)$$

kerül felvételre a DG-be, egyébként a körbejárt alakzatot figyelmen kívül hagyjuk.

Műveletigény: lineáris idejű eljárás, mivel a poligon struktúra bejáró algoritmus is az.

Véglegesítés. Elfogadás esetén az EDGE_1, ..., EDGE_n élek törlődnek, szögpontjaikat pedig az üreg középpontjára húzzuk össze (ez a szögpont koordináták módosításával, majd a nulla hosszúságú élek törlésével és szögpont-összevonással történik). Elvetés esetén a PAT törlődik és az élek változatlanok maradnak. (Helyes paraméterezés esetén a felismerés általában ellenőrzés nélkül elfogadható.)

6.4.8. Nullkörök felismerése

Paraméter:

S_{max}: *maximális méret.* A nullkör megengedett maximális átmérője pixelben.

Algoritmus és műveletigény: megegyezik az üreg felismeréssel.

Véglegesítés. Annyiban tér el az üreg felismeréstől, hogy elfogadás esetén a nullkör középpontjában létrejövő új NODE "nullkör-2" rétegjelzést kap, továbbá létrejön egy hozzá

kapcsolt TEXT objektum (szintén "nullkör-2" rétegjelzéssel), amely a nullkör szimbólumát hivatott kirajzolni (27 ábra).

27. Nullkör megjelenési formája a nyers vektoros rajzon és felismerés véglegesítése után

Megjegyezzük, hogy nullkör felismerés előtt mindig üreg felismerést kell végrehajtani a minimális nullkör méretnél kisebb S_{max} paraméterrel, egyébként az üregeket is nullkörnek ismeri fel a rendszer.

6.4.9. Rajz korrekció

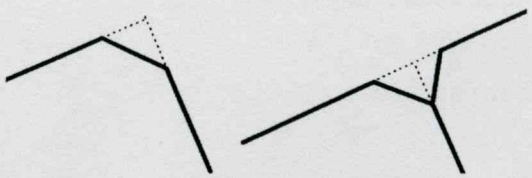
A vektorizálás jellemző torzulásokat visz a vonalrajzba (5., 28. ábra). Ezek automatikus javítását nem lehet elvégezni mindjárt a vektorizálás után, mert a korrekció eltorzítaná a szimbólumokat, kapcsolójeleket, nullköröket.

A rajz korrekciót is felismerő műveletként kezeljük: az algoritmus felismeri a javítandó helyeket és javaslatot tesz a módosításra, amelyet elfogadunk vagy elvetünk. (Helyes paraméterezés esetén a korrekció ellenőrzés nélkül elfogadható.)

Paraméterek:

T: tolerancia. A korrekció maximális mértéke pixelben, vagyis legfeljebb ennyi eltérés lehet az eredeti (nyers vektoros) és korrigált rajz vonalai között.

L_{max}*: maximális hossz.* Legfeljebb ilyen hosszúságú "rövid élek" környékén végez korrekciót a rendszer.



28. ábra. Tipikus torzulások és korrekciójuk

Algoritmus. Az eljárás arra az észrevételre alapul, hogy a korrigálandó rajzrészletek mindig tartalmaznak rövid éleket (28. ábra).

1. Megkeressük a lokálisan minimális éleket, vagyis azokat, amelyeknek egyik végpontjából sem indul rövidebb él, és amelyek hossza kisebb L_{max}-nál. Az ilyen élek ideiglenes jelzést kapnak.

2. Végigmegyünk a megjelölt éleken, mindegyiknek megvizsgáljuk a környezetét. Ha mindkét végpontjából 2 él indul ki, akkor a 28. ábra bal oldalán látható "lecsapott sarokról" lehet szó. Ha egyik végpontból 3, másikból 2 él indul, akkor a 28. ábra jobb oldalán bemutatott "T-elágazással" állhatunk szemben. Amennyiben a szögek és hosszak vizsgálata a korrekció szükségességére utal, úgy egy

$$PAT(EDGE_1, ..., EDGE_r, EDGE_{r+1}, ..., EDGE_{r+s})$$

kerül felvételre a DG-be, ahol $EDGE_1$, ..., $EDGE_r$ a korrigált élek (elfogadás esetén ezek maradnak meg), $EDGE_{r+1}$, ..., $EDGE_{r+s}$ pedig a korrigálandó élek (elvetés esetén ezek maradnak meg). Az utóbbiakat negatív hivatkozás különbözteti meg az előbbiektől.

Műveletigény: az eljárás csak lokális vizsgálatokra épül, ezért lineáris időben végrehajtható.

Véglegesítés. Amint fent írtuk, elfogadás esetén a PAT-ban szereplő egyik, elvetés esetén a másik élcsoport kerül törlésre.

6.4.10. Épületek, telkek felismerése

Itt a kataszteri térkép poligon-struktúráját vizsgálja a rendszer. Az algoritmus feltételezi, hogy a *helyrajziszámok, kapcsolójelek* és *nullkörök* felismerése, javítása és véglegesítése már megtörtént.

Paraméterek:
E_{min}*: minimális épület méret.*
F_{min}*: minimális földrészlet méret.*
A fenti értékek helyes megadásával elkerülhető, hogy különféle fel nem ismert részleteket (például vonallal összeérő betűtöredékeket, speciális jelkulcsi elemeket) épületként illetve földrészletként ismerjen fel a rendszer.

Felhasznált algoritmusok:
– *Poligon struktúra bejárása* (6.4.6. fejezet).
– *Poligon területének számítására* a jól ismert trapéz módszert alkalmazzuk (lásd pl. Márkus (1994) vagy Kollányi, Prajczer (1995)), amely az (x_1, y_1), ..., (x_n, y_n) szögpontok által meghatározott zárt poligon $(x_{n+1}=x_1, y_{n+1}=y_1)$ területét $O(n)$ időben számítja a

$$T = [\ (x_2-x_1)(y_2+y_1) + ... + (x_{n+1}-x_n)(y_{n+1}+y_n)\ ]/2$$

formulával. A terület előjele pozitív, ha az óramutató járása szerint haladunk körbe, negatív az ellenkező irányú körüljárás esetén.
– *Pont-poligon algoritmus* (Márkus, 1994): adott pont adott poligonba esésének eldöntése O(n) lépésben lehetséges n szögpontú poligon esetén. Fontos, hogy az algoritmus akkor is helyesen működik, ha a poligon sziget(ek)et is tartalmaz.

Épületek felismerése

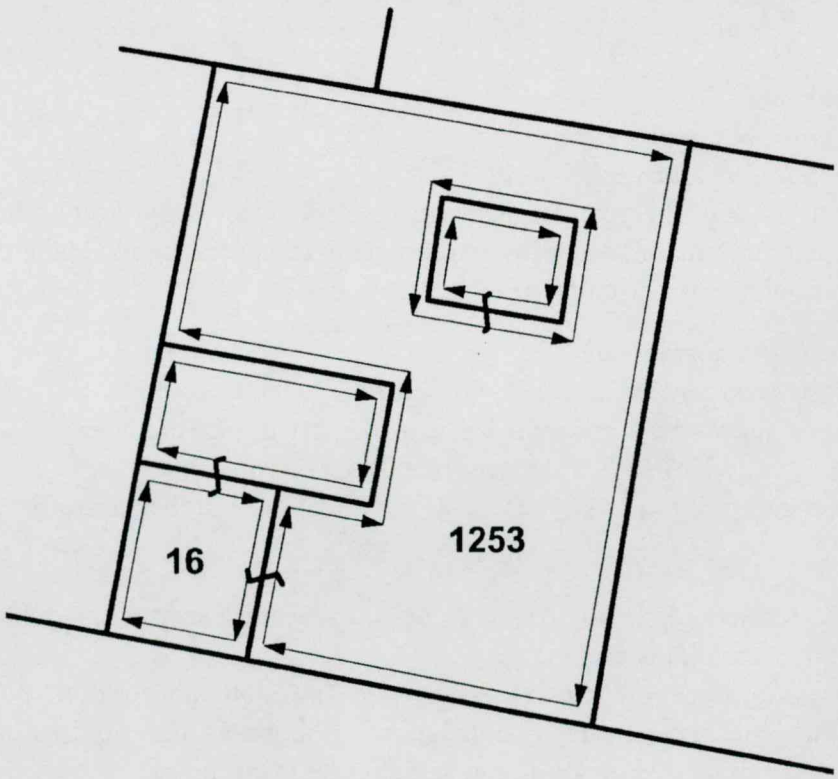
1. Végpontból induló vonallláncok ideiglenes törlése. A végpontból induló éleket addig töröljük, amíg elágazási ponthoz nem jutunk. Ezzel a vonalakra ragadt zajokat, jelkulcstöredékeket, stb. kivonjuk a felismerésből.

2. Poligon struktúra bejárása. Az E_{min}-nél nagyobb átmérőjű poligonokhoz az éleket tartalmazó PAT($EDGE_1$, ..., $EDGE_n$) felvétele az "épület" rétegbe.

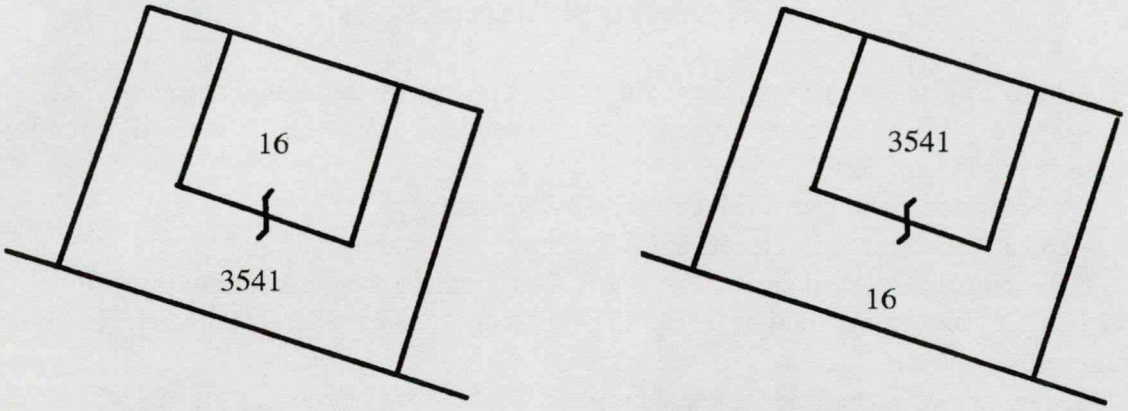
3. Sziget-poligonok kezelése. Amint a 29. ábrán látható, a nagyobb poligonban szigetként szereplő poligonokat az óramutató járásával ellentétesen is körüljárja a bejáró algoritmus. Az így keletkező poligon úgy ismerhető fel, hogy területszámítással rá negatív érték adódik. Az ilyen poligon-PAT-ot töröljük, és él-listáját a tartalmazó poligonhoz csapjuk, hiszen annak belső határát képezi (a tartalmazás ténye pont-poligon algoritmussal vizsgálható). Ha nincs tartalmazó poligon, akkor a negatív területű poligon-PAT a "poligon" rétegbe kerül át. A fentiek eredményeként az "épület" rétegben csak pozitív területű poligonok maradnak.

4. "Földrészlet-minusz-épület poligonok" törlése. Az "épület" rétegben olyan poligonok is maradnak, amelyek nem épületet, hanem egy földrészletnek az épületen kívüli részét ábrázolják. Az ilyenek onnan ismerhetők fel, hogy helyrajzi számot tartalmaznak (29., 30. ábra), amely tény pont-poligon algoritmussal detektálható. Az ilyen poligon-PAT-okat töröljük.

Megjegyezzük, hogy az épület felismerés után nem hajtjuk végre a SetPatLayer műveletet (5.2. fejezet), így az épület-poligonok élei még nem kaptak öröklött rétegjelzést, vagyis továbbra is 0 rétegbe tartoznak, és részt vesznek a további felismerésben.



29. ábra. Épületek és földrészletek felismerése



30. Kétértelmű helyzet: kis épület nagy telken, illetve nagy épület kis udvarral. Ilyen esetben a helyrajzi szám dönt, amely mindig a épületen kívüli telekrészen van

Földrészletek felismerése

5. Kapcsolójel hordozóélek ideiglenes törlése a belőlük kiinduló vonalláncokkal együtt. A kataszteri térkép logikája szerint a kapcsolójellel összekötött objektumokat területileg egyesítve kapjuk a megfelelő földrészlet területét, erre épül az alábbi eljárás. Két, kapcsolójellel összekötött poligon egyesítése a közös határvonal törlésével lehetséges. Ezért törlünk minden kapcsolójel-hordozóélt, és mindkét irányban a belőle kiinduló vonalláncot addig, amíg elágazási ponthoz nen jutunk.

6. Poligon struktúra bejárása. Az F_{min}-nél nagyobb átmérőjű poligonokhoz az éleket tartalmazó PAT($EDGE_1$, ..., $EDGE_n$) felvétele a "földrészlet" rétegbe.

7. Sziget-földrészletek kezelése. Minden negatív területű földrészlet-poligont a tartalmazó poligon-PAT-hoz csapunk. Ha ilyen nincs, akkor a "poligon" rétegbe kerül át.

8. Ideiglenesen törölt élek visszaállítása.

Műveletigény:

Futási idő tekintetében az algoritmus 3. (tartalmazó poligon keresése) és 4. (tartalmazott helyrajziszám keresése) pontjai tekinthetők kritikusnak. (Bár a 7. pontban is szükség van tartalmazó poligon keresésére, de sziget-földrészlet igen ritkán fordul elő, ezért ennek műveletigénye elhanyagolható.)

Figyelembe véve, hogy a fenti műveletekben csak poligonok és helyrajzi számok vesznek részt, célszerű *csak ezekre* egy grid indexet létrehozni. Az 5.5.2. fejezet jelöléseire hivatkozva r=20 gridméretet, s=20 átlagos objektum méretet, k=40 keresési környezetet és D=0.0075 rajzsűrűséget feltételezve (ugyanis a poligonok és helyrajzi számok darabszáma kb. 1/20-a az összes rajzelem számának) a vizsgálandó elemek száma Dk^2 = 12, a keresési hatékonyság 9, tehát egy keresés ideje átlagosan 12*9 = 108 rajzelem vizsgálatát jelenti. Ezt figyelembe véve az épület/földrészlet felismerés grid indexszel *lineáris időben* megoldható.

Véglegesítés. Elvetés esetén a megfelelő poligon-PAT törlődik.

6.5. Összefoglalás, értékelés

A MAPINT rendszer főbb jellemzőit az alábbiakban foglalhatjuk össze:

– Gyors affin transzformáció a szkennelt szelvény vetületi rendszerbe transzformálásához (6.3. fejezet).

– Vékonyítás alapú nyers vektorizálás (2.2. fejezet).

– DG adatmodellre (5. fejezet) épülő felismerő algoritmusok (6.4. fejezet).

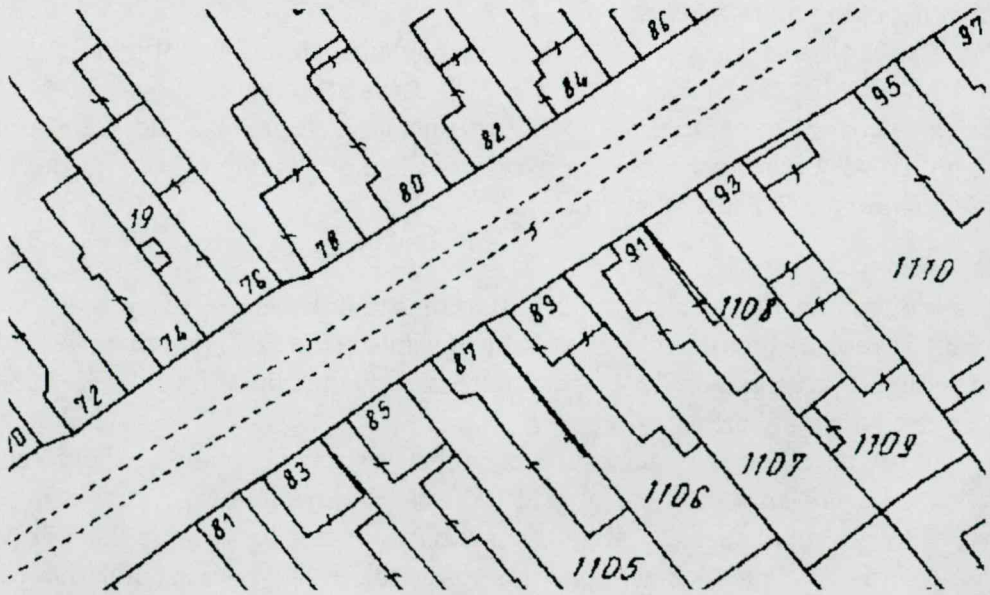
– Neurális hálózat alkalmazása, amely nem szokásos a térkép-interpretációnál (a 3. fejezetben tárgyalt egyik alkalmazásnál sem használják, a karakterek felismerését többnyire raszteresen végzik).

– Magyar kataszteri térképek komplex feldolgozása egészen a földrészlet/épület poligonstruktúra topológikus felismeréséig.

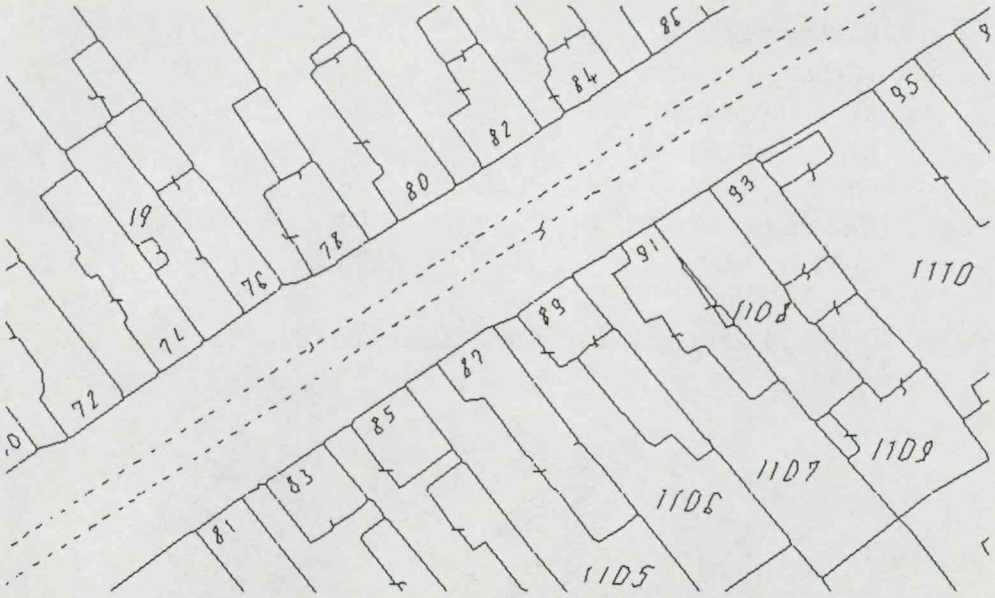
A következőkben a rendszert részleteiben és – amennyire lehetséges – számszerűen értékeljük.

6.5.1. Felismerési eredmények

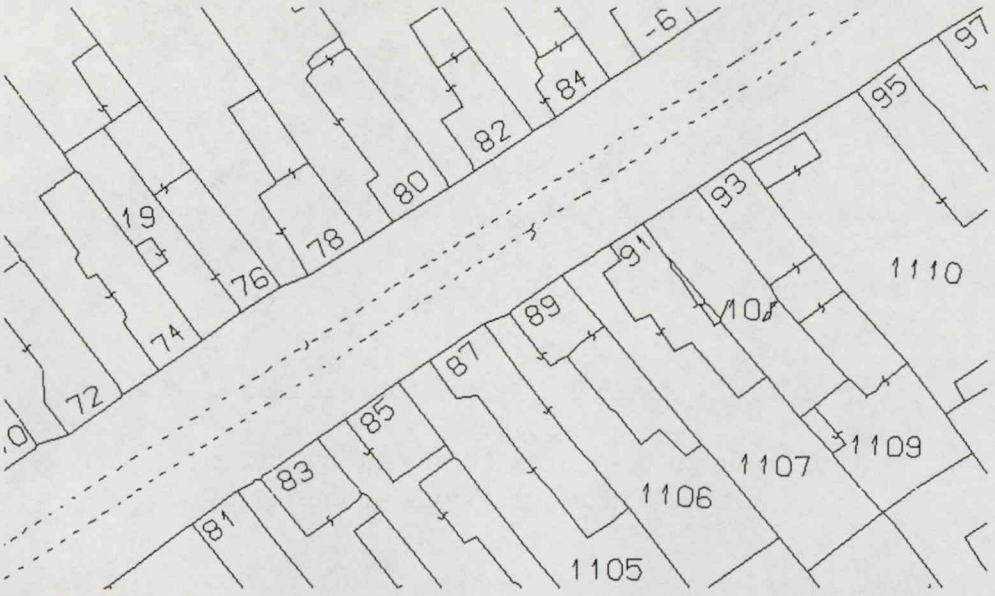
A teljes feldolgozás eredményét a 31., 32. és 33. ábrák szemléltetik. A felismerés számszerű értékelését a 34. ábra tartalmazza. Tesztanyagként jó átlagminőségű szkennelt térképanyagot vettünk alapul (17. ábra). Kimondottan rossz minőségű anyagnál (35. ábra) automatikus interpretáció csak korlátozottan, vagy egyáltalán nem alkalmazható.



31. ábra. 1:2000 méretarányú kataszteri szelvény részlete (300 dpi-vel szkennelt raszterkép)



32. ábra. Nyers vektorizálás eredménye a 31. ábra szerinti térképrészleten

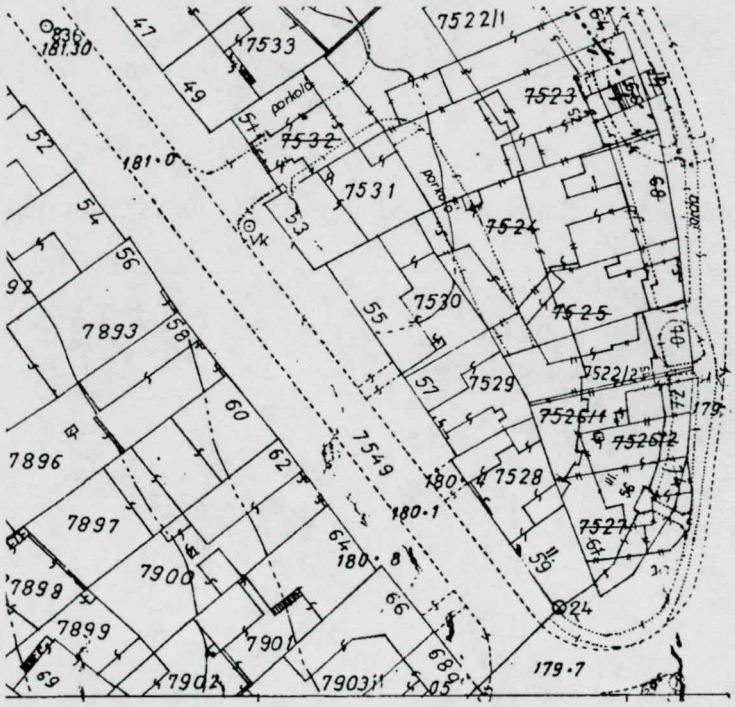


33. ábra. Automatikus felismerés eredménye (manuális korrekció nélkül) a 31. ábra szerinti térképrészleten

Az értékelés során egy hibát csak egy helyen veszünk figyelembe. Például, egy hibás kapcsolójel felismerés következményeként a rendszer nem ismeri fel a megfelelő épületet és/vagy földrészletet, de ezt a kapcsolójel felismerés hibájának tekintjük, és nem az épület/földrészlet felismerés hibájának. Vagyis, az értékelésben szereplő hibaszázalékok feltételezik, hogy minden felismerési művelet után interaktív javítás történt.

objektum típus	*min.*	*max.*	*átlag*
szaggatott vonal	84%	99%	94%
numerikus karakter	70%	97%	85%
kapcsolójel	87%	97%	92%
épület	79%	98%	93%
földrészlet	83%	97%	91%
rajz korrekció	91%	96%	93%

34. ábra. A felismerések százalékos értékelése



35. ábra. Javításokkal, szintvonalakkal zsúfolt kataszteri szelvény részlete. Itt nem várható jó felismerési arány

6.5.2. Futási idők

A 36. ábrán egy átlagos és egy nagy rajzsűrűségű EOV szelvény feldolgozási lépéseinek futási ideje látható (100 MHz Pentium processor). Látható, hogy a teljes futási idő is 5 perc alatt marad, a rendszer tehát igen gyorsnak tekinthető.

Megjegyzések a 36. ábrához:

– A jelenlegi programváltozatban a szaggatott vonalak felismerése grid index helyett a *Sort* tömböt (5.2. fejezet) használja, amely $N\sqrt{N}$ feldolgozási időt biztosít.

– Az épület/földrészlet felismerés jelenleg egyáltalán nem használ térbeli indexelést. Az ennek ellenére tolerálható időadatok annak köszönhetők, hogy a poligonok száma viszonylag kevés az összes rajzelem számához képest.

	Átlagos szelvény	*Sűrű szelvény*
Koordináta-transzformáció	38 sec	38 sec
Nyers vektorizálás	93 sec	114 sec
DG létrehozás (kezdeti topológia)	9 sec	13 sec
Szaggatott vonalak felismerése	5 sec	19 sec
Szimbólumok leválogatása	4 sec	5 sec
Megírások felismerése	6 sec	8 sec
Kapcsolójelek felismerése	4 sec	6 sec
Üregek felismerése	5 sec	7 sec
Nullkörök felismerése	5 sec	7 sec
Rajz korrekció	4 sec	6 sec
Épület, földrészlet felismerés	22 sec	39 sec
Összesen	*195 sec*	*262 sec*

36. ábra. Futási idők egy kataszteri szelvény feldolgozásánál

6.5.3. Összehasonlítás

Különféle interpretáló rendszerek számszerű összehasonlítása meglehetősen nehéz feladat: a legtöbb rendszert speciális térképtípusra fejlesztik (lásd 3. fejezet), ezért más térképanyagon nem hatékony.

> Phillips és Chhabra (1999) kísérletet tesz olyan objektív kritériumrendszer kialakítására, amely segítségével a műszaki rajzokat feldolgozó rendszerek számszerűen összehasonlíthatók. A tanulmány három rendszer teljesítményét vizsgálja folytonos és szaggatott vonalak, körök, körívek és szöveg felismerésén. Természetesen csak olyan rajzi objektumok felismerése vethető össze, amelyeket mindegyik vizsgált rendszer tud, például az említett tanulmány eltekint a műszaki rajzokon kulcsfontosságú méretvonalak felismerésétől.
>
> Az összehasonlítást tovább nehezíti a megfelelő tesztanyag kiválasztása (országonként különböző térképszabványok), a szoftver beszerzés (piaci rendszereknél ez "csak" pénz kérdése, kutatás-fejlesztési prototípusoknál további nehézségek merülnek fel), a szoftverek gyakran speciális hardver igényei (pl. gyorsítóprocesszor).

Utalunk viszont Hudra (2000) vizsgálataira, amelyek a MAPINT-et a M.O.S.S. cég RoSy rendszerével (3.5. fejezet) hasonlítják össze. A szerző leprogramozta a MAPINT algoritmusok egy részét a RoSy fejlesztő környezetében, így a két rendszer hasonló felismerési eredményeket produkált. A MAPINT viszont lényegesen kedvezőbb futási eredményeket adott, amely feltehetően a DG adatmodellnek és – szemben a RoSy interpreteres megoldásával – a közvetlen C nyelvű megvalósításnak köszönhető.

6.5.4. Továbbfejlesztés lehetőségei

A rendszer már jelen állapotában is alkalmazható a gyakorlatban (7. fejezet), de szélesebbkörű felhasználásához továbbfejlesztése célszerű. A főbb fejlesztési irányok:

– Szövegek felismerése (ezt magyar nyelvű szövegek esetén az ékezetek nehezítik), vonalon átírt feliratok felismerése.

– Neurális hálók szerepének növelése. Arra törekszünk, hogy karakterfelismerés mellett minél több rajzi objektumtípust neurális hálóval ismerjünk fel, mert ennek taníthatósága növeli a rendszer univerzalitását. Példaként említjük, hogy Katona és tsai (1995) szkennelt, vektorizált aláírások azonosítására adott megoldást neurális háló segítségével.

– Gyenge minőségű szkennelt anyag feldolgozása. Halvány, töredezett, "szalmiákos" fénymásolatok gyakran feldolgozhatatlanok automatikus vektorizáló/interpretáló rendszerekkel. Egy lehetséges megoldás az anyag szürkeárnyalatosan szkennelése, és képminőség javító eljárások alkalmazása (lásd pl. Musavi és tsai (1988)).

– Manuális rajzszerkesztés támogatása (mivel felismerési hibák minden erőfeszítésünk ellenére a jövőben is lesznek).

7. Alkalmazás: A PHARE Land Consolidation projekt

7.1. A projekt célja

A MAPINT elsőként a Phare HU905.0203 sz. Land Consolidation Project keretében került alkalmazásra. A Földművelési és Vidékfejlesztési Minisztérium felügyeletével folytatott projekt célja (Omaszta 1998): *eljárás kidolgozása a termőföld-privatizáció során előállított digitális térképi állományok betöltésének támogatására a TAKAROS rendszerbe.* Ennek lényegét alább fejtjük ki.

A TAKAROS (= Térképen Alapuló KAtaszteri Rendszer Országos Számítógépesítése) a földhivatalok új, integrált információs rendszere, amely *MicroStation* térinformatikai szoftverre és *Oracle* adatbázis-kezelőre épül. Segítségével a digitális földmérési alaptérképek és a tulajdoni lap adatbázis együtt, integráltan kezelhető, és a TAKARNET hálózaton, valamint a Földmérési és Távérzékelési Intézet FISH szolgáltatásán keresztül az adatok hálózati elérése is lehetséges (Márkus B., Mihály Sz. 1999).

Amíg azonban a tulajdoni lapok adatai digitális formában már rendelkezésre állnak a földhivataloknál, addig a térképekkel nem ilyen jó a helyzet: az egész országot lefedő, viszonylag egységes térképrendszer jelenleg csak papírtérképek formájában létezik. A digitális térképi adatok előállítására a Phare projekt kettős megoldást javasol:

1. Meglévő digitális térképfoltok betöltése a TAKAROS rendszerbe. A földprivatizációval kapcsolatban szükségessé váló új felmérések során jelentős mennyiségű digitális térkép(részlet) is készült, különféle vektorgrafikus formátumokban. Ezen állományok ingatlan-nyilvántartási átvezetése a földhivatalokban megtörtént, tehát hitelesek, formátum konverzió után a TAKAROS rendszerbe betölthetők. Az így előálló digitális térképfoltok azonban az ország területének csak töredékét fedik le. Teljes, hiteles digitális térkép előállítása igen hosszú időt venne igénybe, ezért javasolt a következő:

2. Szkennelt nyilvántartási térképek betöltése. Ez a földhivatalok által használt, az egész országot lefedő papírtérkép állomány felvitelét jelenti *raszteres formátumban*, erre – mint háttérképre – rávetítve jelennének meg a vektoros digitális foltok. Az eredmény egy viszonylag gyorsan előállítható hibrid (raszter+vektor) térképi adatbázis, amely az ország teljes területét lefedi, és amelynek raszteres részei – új felméréssel vagy vektorizálással – fokozatosan lecserélhetők vektorosra. Az elgondolás nem új, Ausztriában már sikerrel alkalmaznak hasonló megoldást.

7.2. A javasolt technológia

A feladatra kidolgozott technológiát Omaszta és Szabó (1999) részletesen ismerteti. Itt csak a főbb lépéseket végrehajtó programmodulokat foglaljuk össze.

ITR2DGN program: konverzió ITR vektorgrafikus formátumról a MicroStation rendszer DGN formátumára.

CleanUp program: DGN állományok topológiai ellenőrzése és javítása.

FindLotNo program: a tulajdonilap-adatok és digitális foltok összehasonlítása.
DcRast program: raszteres állományok katalogizálása.
LotNumCheck program: DAT állományok konverziója Oracle adatbázisba.
MAPINT program: raszter állományok transzformációja, és automatikus helyrajziszám felismerés. A továbbiakban ezt részletezzük.

7.3. A MAPINT szerepe a projektben

A szkennelt papírtérképeket először koordináta-transzformációnak kell alávetni, ez a MAPINT rendszerrel elvégezhető (6.3. fejezet). A transzformáció támogatható *őrkereszt-fájl* segítségével, amely az adott szelvény őrkeresztjeinek EOV-koordinátáit tartalmazza, és a fentiekben említett katalógusból nyerhető.

A transzformációval kapott raszteres állományok már hézag- és átfedésmentesen egymás mellé illeszthetők. Továbbá, ha a képernyőn rámutatunk a raszterkép egy pontjára, a rendszer számítani tudja annak EOV koordinátáit.

A cél azonban az, hogy a raszteres állományok ne csupán háttérképként szolgáljanak, hanem összekapcsolhatók legyenek a tulajdonilap-adatbázissal. Ez is megoldható a földrészletek *geokódolás*ával, vagyis azzal, hogy minden földrészlethez az adatbázisban felvesszük egy referenciapontjának (x,y) koordinátáit. Ezzel kétirányú kapcsolat jön létre a raszterkép és az adatbázis között:

– Ha rámutatunk a raszterkép megfelelő pontjára, a rendszer számítja annak EOV koordinátáit, és ki tudja keresni az adatbázisból azt a földrészletet, amelynek referenciapontja ehhez legközelebb van.

– Ha kiválasztunk egy földrészletet az adatbázisból, a képernyőn megjeleníthető a neki megfelelő térképrészlet.

A földrészletek a tulajdonilap-adatbázisban azonban nincsenek geokódolva. A probléma megoldása: a MAPINT segítségével *automatikusan felismerjük a helyrajzi számokat*, ezzel megkapjuk azok EOV koordinátáit is. Mivel a helyrajzi szám a térképen mindig a földrészleten belül helyezkedik el, így a megírás helye referenciapontként is szolgálhat. A felismerés eredményeként a MAPINT egy szövegfájlt szolgáltat, amelynek minden sora egy

helyrajzi szám, X-koordináta, Y-koordináta

hármast tartalmaz. Ezen szövegfájl segítségével a tulajdonilap-adatbázisba felvihetők a koordináták.

Fel kell készülni olyan térképszelvényekre is, ahol az automatikus felismerés rossz hatékonysággal használható (35. ábra). Ezért a MAPINT rendszert olyan eszközökkel bővítettük, amelyek a helyrajzi számok manuális digitalizálását támogatják (raszterkép pásztázása, TEXT objektum felvitele automatikus szám inkrementálással).

8. A domborzati réteg interpretációja

A domborzat a térképeken általában külön rétegben (azaz eltérő színnel), szintvonalrajz formájában jelenik meg, amelyet speciális jelkulcsi elemek egészítenek ki. Ezen réteg interpretációs sajátosságait vizsgáljuk ebben a fejezetben.

8.1. Szintvonalas térképek jelkulcsai

A domborzatábrázolás alapelvei – szemben a síkrajzzal – viszonylag egységesek a különböző térképtípusokon, sőt még különböző országok térképein is. Alábbiakban ezeket a közös sajátosságokat tárgyaljuk, természetesen utalva a specialitásokra is.

A domborzatábrázolás általános jelkulcsrendszerét Lerner (1989) alapján a 37. ábra tekinti át. A jelkulcsokat két fő csoportra oszthatjuk:

– szintvonalak a hozzájuk kapcsolódó jelölésekkel,

– szintvonaltól független jelölések.

A szintvonalakkal kapcsolatos jelkulcsok és fogalmak a következők:

– *Alapszintvonal* (normál szintvonal): folyamatos vékony vonallal jelölik.

– *Alapszintköz:* két szomszédos alapszintvonal magasságkülönbsége. Értéke a térkép méretarányától függ, egy térképszelvényen belül állandó.

– *Főszintvonal:* folyamatos vastag vonallal jelölik. Általában minden ötödik alapszintvonalat tekintenek főszintvonalnak, de például 2.5 méter alapszintköz esetén minden negyediket.

– *Felező szintvonal:* hosszú szaggatott vonallal jelölik. Két alapszintvonal között alkalmazzák az alapszintköz felének megfelelő magasságkülönbség kifejezésére, olyan helyeken, ahol a felszín megfelelő leírására az alapszintvonalak nem elegendők.

– *Negyedelő szintvonal:* rövid szaggatott vonallal jelölik, jelentése a felező szintvonalhoz hasonló.

– *Szintvonalszám:* a szintvonalra – azt megszakítva – írt szám, amely a szintvonal magasságértékét adja. Általában megkövetelik, hogy a szintvonalszám talpa a lejtés irányába nézzen. Nem minden szintvonalra kerül szintvonalszám, gyakran csak a főszintvonalakon alkalmazzák.

– *Eséstüske:* a szintvonalra merőleges rövid vonal, amely a lejtés irányát mutatja. Félreérthető terepviszonyok esetén alkalmazzák.

Szintvonaltól független jelölések (a 37. ábra és T3 (1981) alapján):

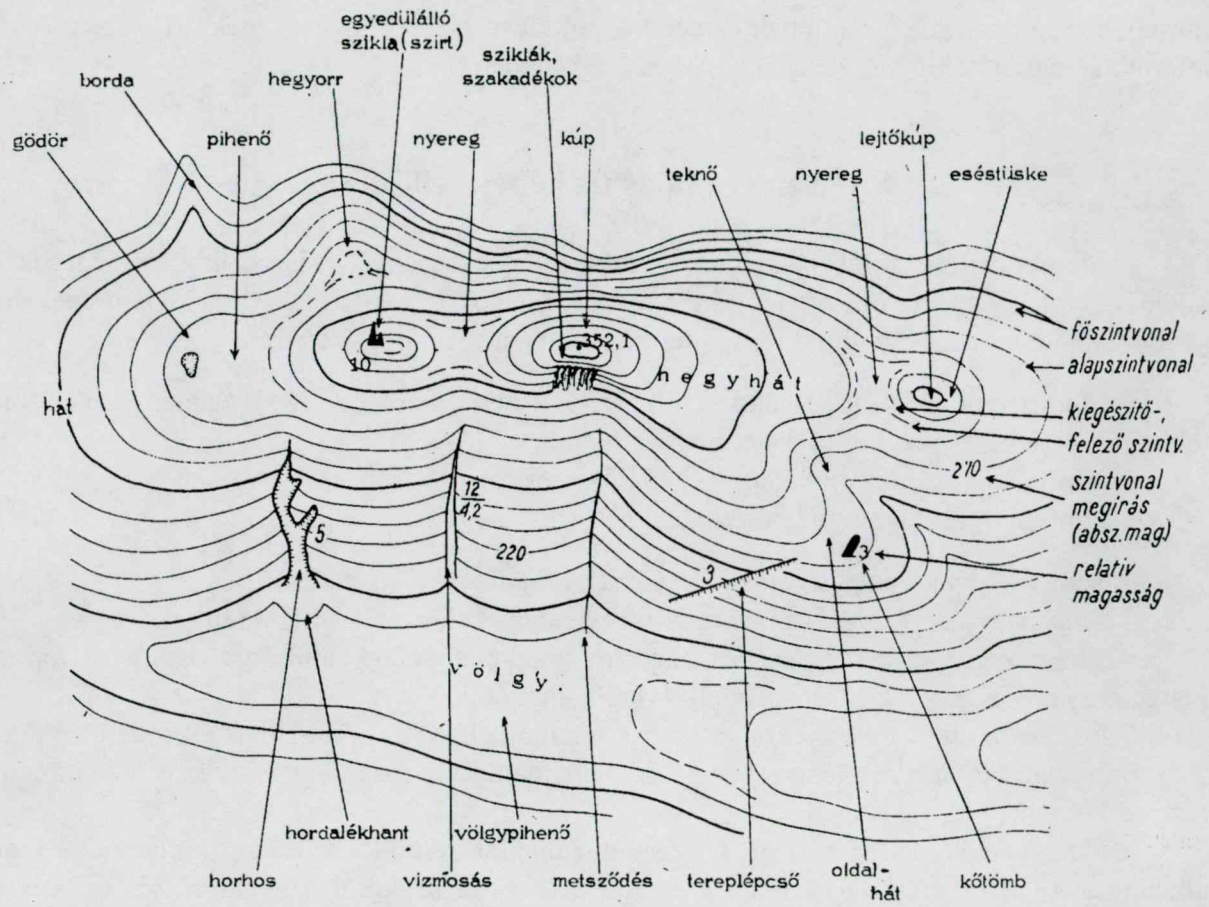
– *Magassági pont:* kereszttel, ponttal, vagy más módon azonosított pont, amelyhez magasságértéket megadó szám tartozik.

– *Tereplépcső, terasz, szakadás:* a terepmagasság ugrásszerű változása egy vonal mentén, általában a relatív magasság megírásával.

– *Vízmosás, horhos*: sajátos jelölése mellett gyakran az átlagos relatív mélység számértékét is feltüntetik.

– *Metsződés:* vízmosásként nem ábrázolható, élestalpú teknő.

– *További domborzati idomok:* horpadás, karsztlyuk (töbör), suvadás, omladék, sziklafal, sziklaszirt, stb.



37. ábra. A domborzatábrázolás jellegzetes jelkulcsi elemei (Lerner 1989)

Külön érdemes kitérni a *magyar kataszteri térképek* domborzati rétegére, amelynek felépítését F7 (1983) szabályozza. Néhány jellemző, illetve számunkra érdekes részlet a szabályzatból:

– A domborzatábrázolás készülhet a földmérési alaptérképpel azonos lapon, barna színnel kirajzolva és megírva, vagy külön lapon, fekete tussal. A topográfiai térkép domborzatát, valamint térképátalakítás esetén az 1:1000 és 1:2000 méretarányú térképek meglévő domborzatát minden esetben külön műanyaglapra kell átszerkeszteni.

– Magassági pontok (úgynevezett kótált pontok) megadása méterben, két tizedesjegy pontossággal történik, ahol a tizedespont jelöli a magassági pont helyét.

– A magassági alappontok magasságát a földmérési alaptérképen nem kell megírni (ugyanis ezeket a pont azonosítási száma határozza meg)

– Az 1:1000 és 1:2000 méretarányú földmérési alaptérképeken alkalmazandó alapszintköz egységesen 1 méter. Kisebb méretarányú térképeknél a terepviszonyoktól függően 1, 2, 2.5, 5 méter lehet.

– Állóvizeknél a meder mélységi adatait mélységvonalakkal (izobátokkal) kell ábrázolni, amennyiben ezek az adatok rendelkezésre állnak. Egyébként a medervonalon belül semmilyen mélységi adatot nem kell ábrázolni.
– Vízmosások (horhosok) méretét mélység/szélesség tört alakban kell megírni.
– A szintvonalak folyamatos kirajzolását az alábbiak megszakítják: másik domborzati elem jele, vasút, árok, csatorna, továbbá 1:1000 és 1:2000 méretarány esetén utca, épített közút, és kótált magasságokkal ellátott egyéb út.
– Speciális jelkulccsal ábrázolt, (a fenti felsorolásban nem szereplő) domborzati formák: védtöltés, töltésformájú parti lerakódás, vízzel érintkező szakadó part, magasvezetésű csatorna, agyaggödör, mesterséges halom, stb.

Összefoglalva megállapítható, hogy a szintvonalas térképek – viszonylagos egységességük és homogenitásuk mellett – rendkívül sokféle speciális jelkulcsot tartalmaznak, így teljesen automatikus interpretációjuk – csakúgy, mint a síkrajznál – gyakorlatilag megoldhatatlan. Az interpretációnak elsősorban a domborzati ábrázolás zömét kitevő szintvonalrajzra és a hozzá közvetlenül kapcsolódó jelkulcsok felismerésére kell koncentrálnia.

8.2. Szintvonalrajz interpretációs eljárások

A szintvonalrajz interpretációjának célja kétféle lehet:
– digitális szintvonalas térkép előállítása,
– digitális terepmodell előállítása.

Mind a felszín leírásának pontossága, mind az alkalmazási lehetőségek az utóbbi mellett szólnak (a digitális alaptérképek előállítására vonatkozó DAT1 (1996) szabvány is a terepmodellt preferálja), ezért a továbbiakban az interpretáció elsődleges céljának terepmodell előállítását tekintjük.

Mivel a terepmodellező rendszerek (lásd 9. fejezet) általában csak szintvonalakat, törésvonalakat és egyedi magasságpontokat fogadnak, így az interpretáció során az a cél, hogy valamennyi domborzati jelkulcsot ezekre az objektumokra képezzünk le.

Színes térképek feldolgozása problematikus, ezért lehetőleg csak a domborzati réteget tartalmazó fóliák szkennelésével célszerű dolgozni. Gondot jelent, hogy az ilyen fóliák általában *nem tartalmaznak őrkereszteket*, azokat a színes térképről kell átrajzolni a fóliára.

A domborzati réteg vonalait a legtöbb térképtípusnál gyakran megszakítják más tereptárgyak (utak, épületek, stb.) jelkulcsai miatt. Mivel ezek a tereptárgyak a színes nyomásnál általában más réteghez tartoznak, így a domborzati rétegen értelmetlen (és sokszor értelmezhetetlen) megszakításokat okoznak, amelyek helyes korrekciója – a színes nyomat figyelembe vételével – többnyire csak manuálisan lehetséges.

Szintvonalas fólia híján alkalmazható módszer, hogy a színes térképről kézzel pauszra átrajzolják a szintvonalakat, munka közben egyben javítják is azokat (vonalhiányok pótlása, stb.), és természetesen átrajzolják az őrkereszteket is. Az így előállt rajzot szkennelik és transzformálják az őrkeresztek alapján.

8.2.1. Feldolgozás a MAPINT rendszerrel

Az alább leírt domborzat interpretáló eljárásokat a MAPINT program még csak részben tartalmazza. Ezek bemutatásának célja elsősorban a vektoros interpretáció lehetőségének megmutatása, amely így összehasonlíthatóvá válik a 11. fejezetben tárgyalt és gyakorlatban is alkalmazott raszteres feldolgozási technikával.

Vektorizálás után a DG adatmodellben dolgozunk. A felismerés során minden szintvonalhoz egy

$$PAT(TEXT_1, TEXT_2, EDGE_1, ..., EDGE_n)$$

úgynevezett szintvonal-PAT kerül felvételre, ahol

– $TEXT_1$ a szintvonal *magasságértéke* (a szintvonal PAT létrehozásakor még üres string).

– $TEXT_2$ a szintvonal orientációja, amely azt adja meg, hogy a szintvonal melyik oldalán van a lejtő: 1 = bal oldalon, 2 = jobboldalon, 0 = definiálatlan. Akkor kerül beállításra, ha a szintvonalon van felismert eséstüske.

– $EDGE_1, ..., EDGE_n$ a szintvonalat alkotó egyenesszakaszok.

Alább az egyes jelkulcsi elemek felismerésének folyamatát vázoljuk a megfelelő végrehajtási sorrendben. Kezdetben minden EDGE objektum a "folytonos vonal" rétegben van.

Szaggatott vonalak felismerése. A 6.4.1. alfejezetben ismertetett szaggatott vonal felismerési eljárás görbe vonalakat is kezelni tud, így változtatás nélkül alkalmas felező és negyedelő szintvonalak felismerésére. Minden felismert és elfogadott szaggatott vonalhoz egy szintvonal-PAT (lásd fent) kerül felvételre.

Megírások felismerése. Itt is a 6.4.2. és 6.4.3. alfejezetben ismertetett felismerő algoritmusok használhatók. Itt a felismert TEXT objektumok "magassági pont" vagy "felirat" rétegbe sorolhatók: az elsőbe a kótált pontok magasságértékei, az utóbbiba a szintvonalszámok és minden egyéb esetleges felirat kerül.

Amennyiben tisztán kótált pontokból álló fedvénnyel van dolgunk (magyar kataszteri térképeknél ez előfordul), úgy a feldolgozás lényegében csak a megírások felismeréséből és interaktív ellenőrzéséből áll. Ügyelni kell viszont arra, hogy a magassági pont helyét a tizedespont adja meg (lásd 8.1. fejezet), a felismert TEXT koordinátáit ennek megfelelően kell hogy beállítsa az algoritmus.

Szintvonalrajz esetén a megírások jelentősége csekély, mivel a vonalak zömén nincs szintvonalszám, és – amint látni fogjuk – a magasságérték hozzárendelése mindenképp manuálisan célszerű. Ilyenkor nem a megírások felismerése, hanem csupán a megírásokat alkotó vektorok "szimbólum" rétegbe helyezése a fontos (hogy ne zavarják a további feldolgozást), ezért a felismerés interaktív javítása nem szükséges.

Eséstüskék felismerése. Itt 6.4. fejezetben tárgyalt kapcsolójel-felismerő algoritmust kell kis mértékben módosítani, amennyiben itt csak a 25. ábra jobb szélső változatát kell felismerni. A felismerés elfogadása esetén a megfelelő szintvonal-PAT $TEXT_2$ komponense beállításra kerül.. Megjegyzések:

– A felismerő algoritmus az eséstüskét könnyen összetéveszthetí a vonalakon gyakran előforduló, szkennelési hibából eredő szálkákkal, ezért a felismerés mindenképp ellenőrzendő.

– Ha a további feldolgozás során nem hasznosítjuk az eséstüskék hordozta információt, akkor nem az eséstüskék felismerésére, hanem eltávolítására van szükség – a szintvonalon lévő szálkákkal együtt. Ebben az esetben a felismerés interaktív javítása nem szükséges.

Szintvonal összeérések megszüntetése. Az algoritmus rákeres az elágazási pontokra, minden egyes pontnál interaktív javítási lehetőséget kínál.

Szakadások összekötése. Az algoritmus egymáshoz közeli végpontokat keres a "folytonos vonal" rétegben, és felajánlja összekötésüket.

Szintvonalak felismerése. Egyszerű vonalkövetéssel történik, minden vonalhoz egy szintvonal-PAT (lásd fent) kerül felvételre.

Főszintvonalak felismerése. A szintvonal minden egyes vektorának középpontjára merőleges egyenes mentén megszámoljuk a raszterkép pixeleit. Ha a kapott értékek átlaga meghalad egy küszöbértéket, akkor főszintvonalról van szó. A főszintvonalak PAT-jai külön rétegbe kerülnek.

Magasságérték hozzárendelés. Minden egyes szintvonal-PAT-hoz egy magasságértéket rendelünk, amely $TEXT_1$-be íródik. A hozzárendelés manuálisan történik, mivel a helyes értékadás a problematikus helyeken komoly térképolvasási rutint követel, és egyetlen hibás értékadás is durva torzulásokat eredményezhet a terepmodellben. Az értékadás az alábbiakkal támogatható:

– Automatikus inkrementálás. A magasságértéket nem kell begépelni, hanem a rendszer felajánlja az alapszintközzel növelt/csökkentett értéket, ez az érték szükség esetén felülírható.

– Csoportos értékadás. Kijelölt szintvonalköteghez alapszintközzel növelt értéksorozat hozzárendelése – feltéve, hogy a köteg egy eleme már rendelkezett értékkel.

– Ellenőrzés. A értékadás helyességének ellenőrzését a felismert eséstüskék és főszintvonalak is támogatják.

Manuális kiegészítések. Amint már említettük, a speciális jelkulcsok felismerésére nem törekszünk. Ezeket manuálisan kell helyettesíteni szintvonallal, törés- vagy szakadásvonallal illetve magassági ponttal.

A feldolgozás eredményeként egy DXF formátumú fájl állítható elő, az alábbi tartalommal:

– szintvonalak: 3D vonallánc, ahol a szögpontok z koordinátája a szintvonalhoz rendelt magasságérték.

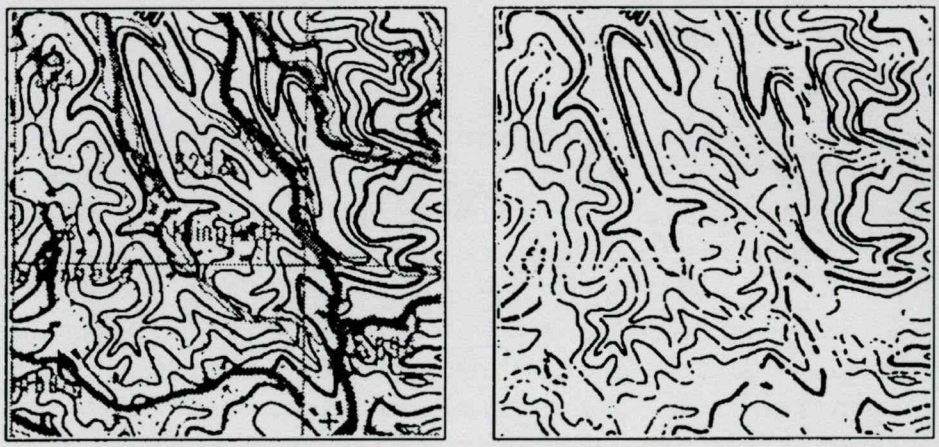
– törés- és szakadásvonalak: 2D vonallánc formátumban.

– magassági pont: 3D pont formátumban.

8.2.2. További szintvonal interpretáló megoldások

Shimada és tsai (1995) érdekes technikát alkalmaz a szintvonalak összeéréseinek és szakadásainak automatikus kezelésére. A felhasználónak egy *n* szintvonalból álló, közel párhuzamosan futó vonalköteget kell kijelölnie, ezután a szintvonalakon *n* db úgynevezett VTA (Vector Trace Agent) fut végig, párhuzamosan vektorizálva azokat. A VTA-k munkáját egy SA (Supervisor Agent) felügyeli és koordinálja. Ha valamelyik VTA elakad összeérés vagy szakadás miatt, a szomszédok segítségét kérve oldja meg a feladatot. A szellemes módszer valóban jól működhet a cikk által bemutatott speciális szituációkban, de véleményünk szerint ezek az esetek a szintvonalrajzok interpretációs problémáinak csak töredékét fedik le.

Dupont és Gondran (1999) komplex rendszert ismertet színes topográfiai térképek domborzati rétegének feldolgozására és terepmodell előállítására. Az eljárás vázlata a következő:



38. ábra. Dupont és Gondran (1999) eljárása: bal oldalon a színes térkép, jobb oldalon a belőle kivont szintvonalrajz

1. Színszétválasztás RGB hisztogram alapján. Eredményül erősen szakadozott és zajos szintvonalrajz keletkezik (38. ábra).

2. Vektorizálás: vázképzés (vékonyítás) vonalvastagság meghatározással, utána vonalkövetés.

3. Karakter felismerés: magassági pontok és szintvonalszámok felismerését végzi interaktív kontroll mellett.

4. Főszintvonalak meghatározása. A vonalak átlagos vastagságának meghatározása után úgynevezett kétszínezéses módszert alkalmaz. Alapvető állítás: egy gráf különálló körökből áll akkor és csak akkor, ha a duális gráf két színnel színezhető. Az eljárás során minden élhez súlyt rendel, amely az él hosszával és a lokális görbülettel arányos. Ezután relaxációval keresi az optimális kétszínezést: maximum a kezdő éleken, minimum a többi élen. A kétszinezés eredményeként rekonstruálhatók a főszintvonalak. Megjegyzendő, hogy az esetleges vonalvastagság felismerési hibák miatt a normál szintvonalakat nem-kezdő élekként kezeli.

5. Magasságérték adás a főszintvonalaknak diszkrét relaxációval. A felismert szintvonalszámokból indul ki. Interaktív támogatást igényelhet, főleg sík területeken.

6. Közbülső DTM előállítása interpolációval (elastique grid technique). Értékadott főszintvonalak és egyedi magasságpontok képezik a magassági feltételt, az érték nélküli szintvonalakat csak "érintési feltételként" veszi figyelembe.

7. Normál szintvonalak orientációjának meghatározása (a vonal melyik oldalán csökken a terepmagasság). Módszer: közbülső DTM alapján digitális gradiens számítása minden szintvonalszakaszhoz.

8. Értékadás a normál szintvonalaknak:

– Szakadások felismerése: lokális geometriai és szomszédsági viszonyok és Voronoi váz alapján.

– Értékadás 5-színezéssel: súlyok meghatározása a vonalak szomszédsági viszonyai alapján.

9. Uj DTM előállítása (a 6. pont eljárásával). A 7. és 8. pont után megmaradt ismeretlen magasságértékű vonalakat "érintési feltételként" veszi figyelembe. Ha túl sok ismeretlen vonal volt, akkor a 7.-9. eljárást többször megismétli (összesen legfeljebb 4-szer).

Bár a fenti eljárás figyelemre méltó eredményeket ér el színes térképek feldolgozásában, a generált terepmodell pontossága és megbízhatósága megkérdőjelezhető (erre vonatkozó adatot a cikk nem közöl, csupán a 38. ábrából következtetünk), ezért gyakorlati alkalmazhatósága korlátozott.

9. Digitális terepmodell előállítása (áttekintés)

A Föld felszínének leírására szolgáló számítógépes modelleket *digitális terepmodell*nek (*DTM = Digital Terrain Model),* vagy *digitális domborzatmodell*nek (DDM) nevezik.

Általában feltételezik, hogy a felszín egy kétváltozós $f(x,y)$ függvénnyel leírható, ahol x, y a felszín egy adott pontjának koordinátái, $f(x,y)$ pedig az adott pontban mért (tengerszint feletti) magasság. Ezzel az ún. 2.5 dimenziós modellezési technikával bizonyos felszíni képződményeket (pl. kihajló sziklákat) csak közelítően lehet leírni, könnyű kezelhetősége folytán mégis ezen modell alkalmazása vált általánossá.

Az $f(x, y)$ függvényt "majdnem mindenütt" folytonosan differenciálhatónak tételezzük fel. A kivételes helyeket a DTM előállításakor külön jelölni kell:

– szakadásvonal: $f(x, y)$ nem folytonos. Ilyennel találkozunk például tereplépcsők esetén.

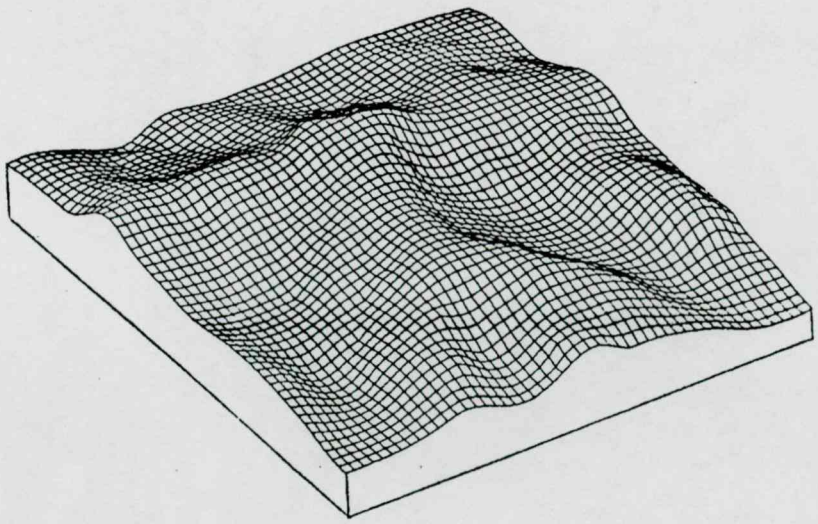
– törésvonal: $f(x, y)$ deriváltja nem folytonos. Törésvonal léphet fel például utak mentén, vízfelületek (tó, folyó) partvonalán, feltéve, hogy a felszínt – és nem a medret – kívánjuk modellezni.

Alapvetően raszteres és vektoros DTM-et különböztethetünk meg.

a). DEM = Digital Elevation Model: raszteres DTM, ahol az $f(x,y)$ függvényt egy $Z[i,j]$ mártixszal közelítjük (39. ábra). (A DEM elnevezés eredetileg a U. S. Geological Survey raszteres terepmodell adatformátumát jelölte.) A DEM jellemző paraméterei:

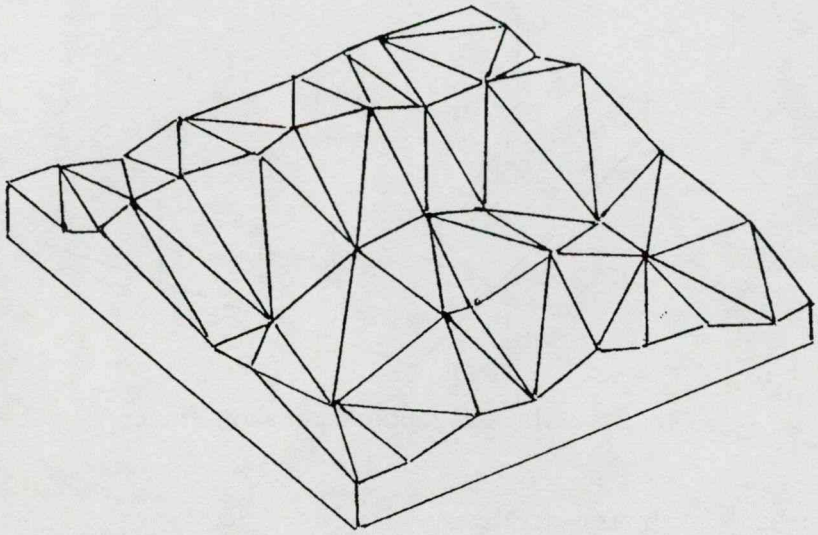
– felbontás: egy raszterpontnak megfelelő négyzet alakú terület oldalhossza. Típikus értéke 10 méter.

– pontosság: a magasságérték legkisebb egysége, kisméretarányú modelleknél általában 1 méter.

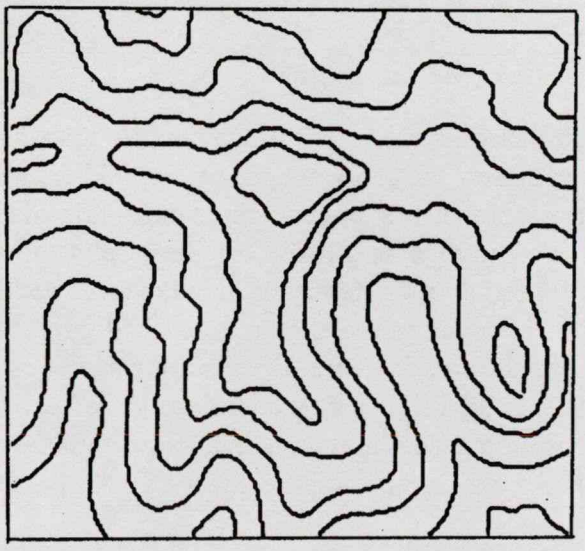


39. ábra: Raszteres terepmodell (DEM) 3D ábrázolása.

b). TIN = Triangulated Irregular Network: vektoros DTM, ahol a felszínt szabálytalanul elhelyezett háromszöglapokkal közelítjük (40. ábra). Itt a szögpontok magasságértéke és összekapcsolási struktúrája kerül tárolásra. Előnye, hogy síkvidéken nagyméretű, hegyvidéken a domborzatot követő, kisebb háromszögek alkalmazhatók.

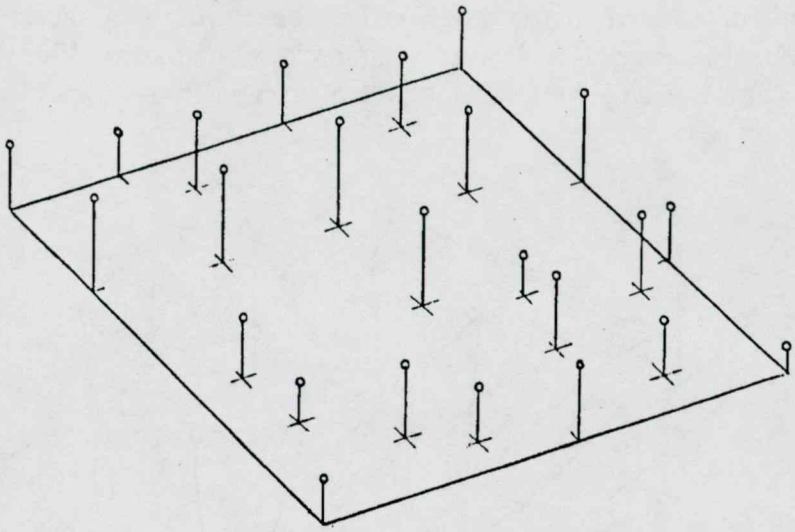


40. ábra: Vektoros terepmodell (TIN) 3D ábrázolása.



41. ábra: Digitalizált, vékonyított szintvonalrajz.

DTM előállítására kiindulási adatként szintvonalrajzot (41. ábra) vagy magassági pontokat (42. ábra) használnak. Gyakran a kettő kombinációjára van szükség, amely kiegészíthető még törésvonalak és szakadásvonalak megadásával.



42. ábra: Magassági pontokkal adott fedvény

A DTM néhány alkalmazási területe:
– digitális ortofotó előállítása (Kraus, 1998),
– távközlési modellezés,
– vízgyűjtőanalízis (Kertész, 1972), vízlefolyás modellezés (Márkus, 1994),
– különféle derivátumok előállítása (lejtőkategória térkép, lejtőkitettség térkép, keresztszelvények, stb.),
– tájtani alkalmazások (Mezősi, 1991).

DTM előállítására számos módszer használatos. A többségükre jellemző, hogy jobban működnek közel egyenletesen szétszórt magassági pontokból, mint szintvonalrajzból. Ennek oka, hogy szintvonalrajz esetén a szintvonalak mentén túl sok információval rendelkezünk (oversampling), míg a szintvonalak között túl kevéssel (undersampling) (Weibel, Heller 1991), a problémával később még többször találkozunk. Eklundh és Martensson (1995) például olyan módszert javasol, amellyel szintvonalak digitalizálását megkerülve már eleve magassági pontokra lehet áttérni, de eljárásuk erős információvesztéssel jár.

A fontosabb DTM előállító módszereket a következőkben tekintjük át. A TIN-generátorokkal csak érintőlegesen foglalkozunk, figyelmünket inkább a DEM-et előállító módszerekre fordítjuk. Az általunk preferált eljárást a 10. és 11. fejezet részletesen ismerteti.

9.1. TIN előállítása szintvonalrajzból

Kiindulásként vektorizált szintvonalrajzot alkalmaznak, amely az alábbi komponenseket tartalmazhatja:

– szintvonal, amely lehet 2D vonal (ebben az esetben attribútumként tartalmazza a magasságértéket), vagy 3D vonal (ekkor *z*-koordinátaként jelenik meg a magasságérték).

– törésvonal, amely a szintvonalaktól független tereptöréseket ír le.

– egyedi magasságpont.

Más speciális jelkulcsi elem fogadására a TIN-generátorok általában nem képesek.

Az egyszerűbb eljárások a szintvonalak töréspontjaira építve Delaunay-triangularizációval háromszögrácsot képeznek. (Egy ponthalmazra illesztett háromszögrácsot Delaunay-triangularizációnak nevezünk, ha bármely háromszög köré írt kör belseje nem tartalmaz további szögpontot (Weibel, Heller 1991).) A fejlettebb módszerek már a szintvonalak között is meghatározzák egyes pontok magasságértékét térbeli interpolációval.

Carrara és tsai (1997) érdekes összehasonlító vizsgálatokat végez, többek között az Arc/Info rendszer *ArcTin* terepmodellezőjét és az Intergraph *Terrain Modeller*ét vizsgálja. Az eredmények:

Az *ArcTin* magasságpontokra és szintvonalak kiválasztott pontjaira épít Delaunay-triangularizációval. A felhasználó törésvonalakat adhat meg, ezzel finomítva a modellt. Carrara és tsai (1997) szerint a rendszer nem kezeli korrekten a völgytalpakat, a kúpokat (hegycsúcsokat) és általában domborzat finom részleteit. Jellemző hiba a *lapos háromszög*, amelynek mindhárom csúcspontja azonos magasságú. Ilyen például a 41. ábra közepén lévő hegyhátnál léphet fel, vagy kúpok esetén, ha a csúcspont magassága nem adott.

Hasonló elveken működik a *Terrain Modeller* terepmodellezője. A fő eltérés: ahol erős szintvonal görbület van (hegygerinc, vízfolyás), ott a rendszer automatikusan törésvonalat generál. Carrara és tsai (1997) vizsgálatai alapján az ArcTin-nél említett anomáliák itt ritkábban lépnek fel.

Külön figyelmet érdemel a Chai és tsai (1998) által publikált eljárás, amely végeselemes módszerrel állít elő TIN-t. A felszínt lényegében membrán-modell (lásd a 10. fejezetben) alapján határozza meg. A terep símaságát úgy biztosítja, hogy a szintvonalak mentén a lejtő meredekségét (a terepfüggvény gradiensét) becsüli, ennek figyelembe vételével interpolál a szintvonalak között. A következmény: a szintvonalaknál nem lesz törés, és a kúpok is felmagasodnak. A teljes eljárás a következő lépésekből áll:

1. Szintvonalpárok közötti területekre TIN előállítása.
2. Szintvonalak mentén gradiens becslés a csatlakozó háromszögek gradienseinek átlagolásával.
3. TIN újragenerálása a gradiensek figyelembe vételével.

Az eljárás – a szerzők saját bevallása szerint – a kúpok kezelésében még javításra szorul.

9.2. DEM előállítása szintvonalrajzból

Kiindulásként kétféle adatstruktúra jöhet szóba:

– Vektorizált szintvonalrajz (hasonlóan, mint a TIN-generátoroknál), amelyet a DEM-generáló algoritmusoknál általában raszterizálni kell. Carrara és tsai (1997) vizsgálatai szerint a raszterméret mindenképp kisebb kell hogy legyen az alapszintköznél, annak 1/10-e már jó minőségű terepmodellt ad.

– Szkennelt raszteres szintvonalrajz. Ebben az esetben elhagyjuk a vektorizálást, hiszen a végső adatstruktúra is raszteres kell hogy legyen. Ezzel a megközelítéssel a 11. fejezetben foglalkozunk részletesen.

A DEM-generátorok szükségképpen térbeli interpolációt használnak, mivel itt egy előre adott raszter pontjaiban kell magasságértékeket számolni. A következőkben röviden áttekintjük a fontosabb térbeli interpolációs eljárásokat. Az általunk preferált eljárást részletesen a 10. fejezet ismerteti.

9.2.1. Távolság inverzével súlyozott mozgóátlag

A módszer lényege, hogy a meghatározandó P pont környezetében kiválasztanak n ismert magasságú pontot (P_1, ..., P_n), és ezek d_1,...,d_n magasságértékéből átlagolják a P pont h magasságértékét. Jellemzően a

$$h = (d_1/s_1 + ... + d_n/s_n) / (1/s_1 + ... + 1/s_n)$$

formula használatos, ahol s_i a P_i pontnak P-től való távolságát jelenti. A távolság inverzével való súlyozás nyilván a közelebbi pontoknak ad nagyobb szerepet, az $(1/s_1 + ... + 1/s_n)$ tényezőnek pedig normáló szerepe van (ha $d_1 = ... = d_n$, akkor h a közös értéket kell hogy adja).

Előfordul, hogy a távolság négyzetével súlyoznak (Eklundh, Martensson 1995). Általában 4...12 pontot átlagolnak, a pontok kiválasztására a módszer érzékeny.

Szintvonalrajz esetén a módszer a következőképp alkalmazható. Indítsunk n egyenest a P pontból $360/n$ fokos szöginkrementummal, és legyenek P_1 ,..., P_n azon szintvonalpontok, amelyeket ezen egyenesek a P pontból indulva elsőként metszenek! Nevezzük ezeket a *P-ből látható szintvonalpontok*nak. Az így kiválasztott pontokra végezzük az interpolációt.

Az eljárás hátrányai:

– Lokális minimumok, maximumok csak az adott pontokban (szintvonalakon) léphetnek fel, ami általában nem felel meg a valóságnak.

– Az eljárás "nem lát át" a szintvonalakon, ezért a szintvonalaknál a terepen törés keletkezhet.

– Zárt görbét alkotó szintvonalak (kúpok, mélyedések) esetén a görbén belül konstans (lapos) felületet ad, mivel minden irányban azonos magasságértékeket "lát".

– Ha n kicsi, akkor nagy a hibalehetőség, ha viszont nagy, akkor a látható szintvonalpontok megkeresése számításigényessé válik.

– A módszer érzékeny a szintvonalak szakadására.

9.2.2. IDRISI módszer

Az IDRISI raszteres térinformatikai szoftver Intercon rutinja (Eastman, 1992) alapján az alábbi egyszerű interpolációs módszert alkalmazza, kimondottan szintvonalas fedvényekre:

– Vízszintes, függőleges és átlós irányú metszeteket készít a generálandó DEM felbontásának megfelelően.

– Minden metszet mentén lineáris interpolációval határozza meg az ismeretlen pontok magasságát és minden ponthoz a – metszet mentén való – lejtés értékét.

– A fentiek szerint a DEM minden pontjához több magasság- és lejtésérték keletkezik. Ezek közül azt a magasságértéket választja az algoritmus, amelyhez a legnagyobb lejtés tartozik.

Az eljárás nem biztosít sima felületet, és lokális maximumok/minimumok itt is csak az adott pontokban léphetnek fel (a kúpok tehát laposak maradnak). További hátrány, hogy szintvonalak megszakadása esetén erős torzulások léphetnek fel, márpedig a gyakorlatban nehéz garantálni a szakadásmentes szintvonalrajzot.

9.2.3. GRASS módszer

A GRASS térinformatikai rendszer vektorizált szintvonalakat raszterizál a leendő DEM felbontásában. Egy ún. *flood fill* algoritmussal lényegében távolság-fedvényt képez (lásd 10.2.5. fejezet), amelyben minden egyes mátrixelem a legközelebbi szintvonalponttól való távolságot tartalmazza. Ezután az egyes pontok értékét lineáris interpolációval számítja a két közrefogó szintvonal magasságából.

A kúpokat ez az eljárás is laposan hagyja, ráadásul Carrara és tsai (1997) szerint a GRASS rendszerben rossz hatásfokkal van leprogramozva, mert Sun Sparc 20 gépen a flood fill algoritmusra 100 óra futási idő becsülhető.

9.2.4. Polinomiális interpoláció

A módszer elméleti alapjait Maling (1991) alapján írjuk le.

Legyen adott az $f(x, y)$ függvény értéke az (x_1, y_1), ..., (x_m, y_m) pontokban, legyenek ezek az értékek d_1, ..., d_m. Az f függvényt a legkisebb négyzetek módszerével szeretnénk közelíteni, vagyis olyan $p(x, y)$ r-edfokú polinomot találni, amelyre a

$$E = \Sigma_i (p(x_i, y_i) - d_i)^2$$

összeg minimális. A könnyebb írásmód kedvéért legyen $r = 3$, ekkor

$$p(x, y) = a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2 + a_{30}x^3 + a_{21}x^2y + a_{12}xy^2 + a_{03}y^3$$

Az együtthatók számát jelöljük m_r-rel. Általában $m_r = 1 + 2 + ... + (r+1) = (r+1)(r+2)/2$, például r = 1, 2, 3, 4, 5 esetén rendre m_r = 3, 6, 10, 15, 21. Vezessük be a következő jelöléseket:

$$\underline{u} = (1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3)$$
$$\underline{a} = (a_{00}, a_{10}, a_{01}, a_{20}, a_{11}, a_{02}, a_{30}, a_{21}, a_{12}, a_{03})^T$$

Ekkor

$p(x, y) = \underline{u} \cdot \underline{a}$.

Legyen $\underline{u}_i$ az $\underline{u}$ értéke az i-edik pontra, vagyis

$\underline{u}_i = (1, x_i, y_i, x_i^2, x_i y_i, y_i^2, x_i^3, x_i^2 y_i, x_i y_i^2, y_i^3)$

és képezzük az

$$U = \begin{bmatrix} \underline{u}_1 \\ \vdots \\ \underline{u}_m \end{bmatrix}$$

m x m_r-es mátrixot. Legyen továbbá az adott pontok vektora $\underline{d} = (d_1,...,d_m)^T$. Ekkor a legkisebb négyzetek feltétele a következőképp írható:

$E = (\underline{u}_1\underline{a} - d_1)^2 + ... + (\underline{u}_m\underline{a} - d_m)^2 = (U \cdot \underline{a} - \underline{d})^T \cdot (U \cdot \underline{a} - \underline{d})$

Mivel a polinomot keressük, így ebben a formulában most $\underline{a}$ elemei az ismeretlenek, és olyan értéküket keressük, amelyre E minimális. Ez ott fog teljesülni, ahol az $E(\underline{a})$ függvény parciális deriváltjai nullák, vagyis

$\partial E/\partial a_i = 2u_{1,i}(\underline{u}_1\underline{a} - d_1) + ... + 2u_{m,i}(\underline{u}_m\underline{a} - d_m) = 2(u_{1,i}, ..., u_{m,i})(U \cdot \underline{a} - \underline{d}) = 0$

Az összes parciális deriváltra együtt a következő adódik:

$U^T(U \cdot \underline{a} - \underline{d}) = 0$

Innen

$U^T \cdot U \cdot \underline{a} = U^T \underline{d}$

Ha $m \geq m_r$, akkor az m_r x m_r-es $U^T \cdot U$ mátrix invertálható, innen

$\underline{a} = (U^T \cdot U)^{-1} \cdot U^T \underline{d}$

Következmények:

– Ha $m < m_r$, akkor több megoldás is lehetséges, amely az adott pontokra illeszkedik.

– Ha $m = m_r$, akkor a megoldás egyértelmű, és pontosan illeszkedik az adott pontokra ($E = 0$). Ezt az $U \cdot \underline{a} = \underline{d}$ egyenletrendszer megoldásával nyerjük.

– Ha $m > m_r$, akkor pontos illeszkedés általában nem teljesül, a legkisebb négyzetek szerinti optimális megoldást $\underline{a} = (U^T \cdot U)^{-1} \cdot U^T \underline{d}$ adja.

Értékelés:

– A mozgóátlagos megközelítéssel szemben itt lokális minimumok, maximumok nem csak az adott pontokban léphetnek fel.

– Raszteres szintvonalrajz esetén minden szintvonalpont adott pontnak tekintendő. Ez azt jelenti, hogy például egy 2000 x 2000 pixeles fedvényen a szintvonalpontok becsült száma 50*2000 = 10^5 (40 pixel átlagos szintvonaltávolságot feltételezve). Ha tehát pontos illeszkedést szeretnénk, akkor az együtthatók meghatározásához egy 10^5 x 10^5 méretű mátrixot kell invertálni, a transzformációnál pedig pontonként több mint 10^5 műveletet kellene végezni.

– A számításigény természetesen csökkenthető a szintvonalpontok ritkításával, vagy vektoros szintvonalrajznál azzal, hogy csak a töréspontokat tekintjük adott pontnak. Ezzel viszont lemondunk a szintvonalakra való pontos illeszkedésről.

– Tartományonként polinomiális függvények alkalmazása esetén is csökken a számításigény, de ekkor gondot okoz a szegmensek kijelölése és csatlakoztatása.

– Az eljárás nem veszi figyelembe a terep fizikai jellemzőit. Ebben a tekintetben a vékonylemez modell (lásd 10 fejezet) jelent majd előrelépést.

A polinomiális interpoláció tehát szintvonalrajz esetén csak erős kompromisszumokkal alkalmazható.

9.2.5. Végeselemes módszer

A 9.1. fejezetben mutattuk be Chai és tsai (1998) végeselemes módszerét TIN előállítására. A végeselemes technika azonban természetes módon alkalmas DEM előállítására is, amennyiben a végeselem felbontás négyzetrács szerint történik. A legjobb minőségű terepmodellt a *vékonylemez modellre épülő multigrid eljárások* szolgáltatják, ilyen elven működik az ANUDEM program (Hutchinson 1989, 1996) és az Arc/Info rendszer TOPOGRID modulja (ESRI 1994b).

Az eljárás matematikai alapjait a 10. fejezetben ismertetjük saját vizsgálatainkkal kiegészítve. Az alkalmazott vékonylemez modell segítségével *minimális görbületi energiájú felület* állítható elő, amely megfelelően síma, és a kúpok is felmagasodnak.

A 11. fejezetben tisztán raszteres feldolgozási technológiát adunk a módszer alkalmazására. A 11.3. alfejezetben saját *multigrid relaxációs algoritmusunkat* ismertetjük, amely szintvonalrajz és magassági pontok feldolgozására egyaránt alkalmas. Megmutatjuk, hogy az eljárás futási ideje jelentősen leszorítható, így ez nem jelent akadályt a gyakorlati alkalmazásnál. A 11.5. fejezet az eljárás felhasználását tárgyalja DDM-10 projektben.

Értékelésünk szerint a végeselemes módszer a legmegfelelőbb DEM előállítására, és a legpontosabb terepmodell a 11. fejezetben leírt eljárással nyerhető.

10. Variációs spline illesztés

Olyan $f(x, y)$ függvényt keresünk, amely az adott (x_1, y_1), ..., (x_m, y_m) pontokban a megfelelő d_1, ..., d_m értékeket veszi fel, és "megfelelően síma" felületet alkot (spline illesztés). A megoldás egy "energia mérőszám" bevezetése lehet: a felszín energiája annál nagyobb, minél kevésbé síma a felület. Tehát olyan függvényt keresünk, amely illeszkedik az adott pontokra, és energiája minimális.

Terzopoulos (1986) az alábbi általános r-edfokú spline energiaformulát adja:

$$E_f^r = \iint \left[\sum_{i=0}^{r} \binom{r}{i} \left(\frac{\partial^r f}{\partial x^i \partial y^{r-i}} \right)^2 \right] dx\, dy$$

Ez egy funkcionál, amely minden $f(x, y)$ függvényhez egy skalár E_f^r energiaértéket rendel. Ezen funkcionál szélsőértékét keressük, tehát variációszámítási feladattal állunk szemben. A formula két legfontosabb speciális esete:

r=1: membrán modell. Ekkor az integrál jó közelítéssel egy membrán energiáját adja:

$$E_f^1 = \iint \left(f_x^2 + f_y^2 \right) dx\, dy \qquad (1)$$

ahol f_x az x szerinti parciális deriváltfüggvényt jelöli, hasonlóan f_y. A membrán lényegében minimális felszínű felület felvételére törekszik. (A tényleges felszín minimalizálás az úgynevezett Plateau-problémához vezet.). Terepmodell esetén a membrán modell megengedi, hogy a szintvonalaknál törések lépjenek fel, a zárt szintvonallal határolt kúpok pedig laposak lesznek, amennyiben magassági pont nincs hozzájuk megadva.

r=2: vékonylemez (thin plate) modell. Ekkor az integrál jó közelítéssel egy idealizált vékony acéllemez görbületi energiáját adja:

$$E_f^2 = \iint \left(f_{xx}^2 + 2 f_{xy}^2 + f_{yy}^2 \right) dx\, dy \qquad (2)$$

ahol f_{xx} az x szerinti második parciális deriváltfüggvényt jelöli, hasonlóan a többi. A vékonylemez modell nem viseli el az éles töréseket, ezért a szintvonalaknál sima átmenetet alkot, a csúcsok feldomborodnak és a mélyedések besüllyednek.

Magasabbfokú modellek már nem eredményeznek javulást a terep minőségében, ugyanakkor jelentősen növelik a számításigényt. Ezért egyértelmű, hogy *terepmodellezésre elsősorban a vékonylemez modell alkalmas*, de – amint alább látni fogjuk – bizonyos esetekben a membrán modellre is szükség van.

Törésvonalak és szakadásvonalak kezelése

Terzopoulos (1988) szerint a törésvonalakat egy $t(x, y)$ függvény segítségével lehet leírni, amelynek értéke 0 a törésvonalak mentén, 1 egyébként. Finomított modell esetén 0 és 1 közötti értékek is alkalmazhatók. Mivel a törésvonalak mentén a parciális deriváltak hirtelen változása megengedett, de a felszín folytonosságát megköveteljük, így ezen vonalak mentén membrán modellt, míg a többi helyen vékonylemez modellt alkalmazunk, vagyis az alábbi energiafüggvényt használjuk:

$$E_f = \iint \left\{ [1 - t(x,y)] \left(f_x^2 + f_y^2 \right) + t(x,y) \left(f_{xx}^2 + 2 f_{xy}^2 + f_{yy}^2 \right) \right\} dx\, dy$$

A szakadásvonalak egy $r(x, y)$ függvénnyel írhatók le, amelynek értéke 0 a vonalak mentén, 1 egyébként. Finomított modell esetén itt is alkalmazhatók 0 és 1 közötti értékek. Szakadásvonal mentén a felszín folytonossága megszűnik, ezért itt a teljes energiamodellt ki kell kapcsolni (Szeliski, 1990):

$$E_f = \iint r(x,y) \left\{ [1 - t(x,y)] \left(f_x^2 + f_y^2 \right) + t(x,y) \left(f_{xx}^2 + 2 f_{xy}^2 + f_{yy}^2 \right) \right\} dx\, dy$$

Jelen dolgozat további részében a törésvonalak és szakadásvonalak modellezésétől eltekintünk.

10.1. A variációs feladat analítikus megoldása

Kérdés, hogy felírható-e explicit formában olyan függvény, amely az adott (x_1, y_1), ..., (x_m, y_m) pontokban a megfelelő d_1, ..., d_m értékeket veszi fel, és eleget tesz a vékonylemez feltételnek, vagyis a (2) integrál értéke minimális? A választ Bookstein (1989) adja meg az alábbiak szerint.

Tekintsük az

$U(x, y) = r^2 \cdot \log r^2$

függvényt, ahol $r = \sqrt{x^2 + y^2}$. Vezessük be az

$u_{i,j} = U(x_i - x_j, y_i - y_j)$

jelölést, és oldjuk meg az alábbi egyenletrendszert az a_0, a_1, a_2, b_1, ..., b_m ismeretlenekre:

$$\begin{bmatrix} 0 & u_{12} & \cdots & u_{1m} & 1 & x_1 & y_1 \\ u_{21} & 0 & \cdots & u_{2m} & 1 & x_2 & y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ u_{m1} & u_{m2} & \cdots & 0 & 1 & x_m & y_m \\ 1 & 1 & \cdots & 1 & 0 & 0 & 0 \\ x_1 & x_2 & \cdots & x_m & 0 & 0 & 0 \\ y_1 & y_2 & \cdots & y_m & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \\ a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Bookstein (1989) kimutatja, hogy a megoldásul kapott a_0, a_1, a_2, b_1, ..., b_m együtthatókkal felírt

$$f(x, y) = a_0 + a_1x + a_2y + \Sigma_i\, b_i \cdot U(x_i - x, y_i - y) \qquad (3)$$

függvény illeszkedik az adott pontokra, és eleget tesz a vékonylemez feltételnek. Ez utóbbi úgy értendő, hogy a (3) típusú formulával felírható függvények osztályán a (2) integrál értéke minimális.

Mivel szintvonalrajz esetén minden szintvonalpont adott pontnak tekintendő, ezért – csakúgy, mint a polinomiális interpolációnál – itt is az adott pontok nagy száma okoz gondot. A számításigényt itt még egy logaritmusfüggvény kiértékelése is növeli, ezért szintvonalas térképből terepmodell előállítására a gyakorlatban ez a módszer nem alkalmas.

10.2. A variációs feladat végeselemes megoldása

10.2.1. A végeselem-egyenletrendszer levezetése

Ebben a fejezetben egy Grimson (1983) és Terzopoulos (1983) által egyaránt publikált megközelítést ismertetünk kisebb módosításokkal és kiegészítésekkel. Megjegyezzük, hogy a szerzők eljárásukat elsősorban robotlátás (computer vision) alkalmazásokra, és nem terepmodellezésre dolgozták ki.

Végeselemes módszerrel dolgozunk, éspedig négyzetrács szerint felosztjuk a síkot: az $f(x, y)$ függvény helyett egy $Z[i,j]$ mátrixot veszünk, ahol $z_{i,j}$ az f függvény átlagos értéke az (i, j) négyzeten. A továbbiakban az egyes $z_{i,j}$ értékeket tekintjük változóknak, és ezek függvényeként írjuk fel az E energiaértéket.

Membrán modell esetén a (1) integrál értékének a következő felel meg:

$$E_Z^{\,1} = \Sigma_i \Sigma_j \left[(z_{i+1,j} - z_{i,j})^2 + (z_{i,j+1} - z_{i,j})^2 \right] =$$

$$= \Sigma_i \Sigma_j \left[z_{i+1,j}^{\,2} - 2z_{i+1,j}z_{i,j} + z_{i,j}^{\,2} + z_{i,j+1}^{\,2} - 2z_{i,j+1}z_{i,j} + z_{i,j}^{\,2} \right]$$

Keressük azon Z mátrixot, amelyre az $E_Z^{\,1}$ függvény értéke minimális. Ez ott teljesül, ahol a függvény valamennyi $z_{i,j}$ szerinti parciális deriváltja nulla. Határozzuk meg tehát a $z_{i,j}$-re vonatkozó parciális deriváltat:

$$\partial E_Z^{\,1}/\partial z_{i,j} = -2z_{i+1,j} + 2z_{i,j} - 2z_{i,j+1} + 2z_{i,j} + 2z_{i,j} - 2z_{i-1,j} + 2z_{i,j} - 2z_{i,j-1} =$$

$$= 8z_{i,j} - 2z_{i+1,j} - 2z_{i-1,j} - 2z_{i,j+1} - 2z_{i,j-1}$$

A mátrix szélein a hiányzó szomszédok miatt a fenti formula módosítandó, ezzel a kérdéssel a 10.2.4. alfejezetben foglalkozunk.

Tehát olyan Z mátrixot keresünk, amelynek minden $z_{i,j}$ elemére $\partial E_Z^{\,1}/\partial z_{i,j} = 0$, vagyis a fenti formulát 2-vel leosztva

$$4z_{i,j} - z_{i+1,j} - z_{i-1,j} - z_{i,j+1} - z_{i,j-1} = 0$$

Így egy lineáris egyenletrendszert kapunk, amely a Z mátrix minden eleméhez egy egyenletet tartalmaz, a megoldás adja a keresett Z mátrixot. Szemléletesen ez azt jelenti, hogy az egyes mátrixelemekre a

```
    -1
-1   4  -1
    -1
```

maszkot illesztve a szomszédok súlyozott átlaga nullát kell hogy adjon.

Alábbiakban *vékonylemez modell* esetén is elvégezzük a fenti levezetést. Itt a (2) integrál értékének a következő végeselemes összeg felel meg:

$$E_z^2 = \Sigma_i \Sigma_j [(z_{i+1,j} - 2z_{i,j} + z_{i-1,j})^2 + 2(z_{i+1,j+1} - z_{i,j+1} - z_{i+1,j} + z_{i,j})^2 + (z_{i,j+1} - 2z_{i,j} + z_{i,j-1})^2] =$$

$$= \Sigma_i \Sigma_j [\, z_{i+1,j}^2 + 4z_{i,j}^2 + z_{i-1,j}^2 - 4z_{i+1,j}z_{i,j} - 4z_{i,j}z_{i-1,j} + 2z_{i+1,j}z_{i-1,j} +$$

$$+ 2z_{i+1,j+1}^2 + 2z_{i,j+1}^2 + 2z_{i+1,j}^2 + 2z_{i,j}^2 - 4z_{i+1,j+1}z_{i,j+1} - 4z_{i+1,j+1}z_{i+1,j} +$$

$$+ 4z_{i+1,j+1}z_{i,j} + 4z_{i,j+1}z_{i+1,j} - 4z_{i,j+1}z_{i,j} - 4z_{i+1,j}z_{i,j} +$$

$$+ z_{i,j+1}^2 + 4z_{i,j}^2 + z_{i,j-1}^2 - 4z_{i,j+1}z_{i,j} - 4z_{i,j}z_{i,j-1} + 2z_{i,j+1}z_{i,j-1} \,]$$

A fenti formula csak a $z_{i,j}$ elemet és nyolc szomszédját tartalmazza, ezért vezessük be az $N_{i,j}$ jelölést $z_{i,j}$ szomszédságára, vagyis

$$N_{i,j} = (z_{i+1,j+1}, z_{i+1,j}, z_{i+1,j-1}, z_{i,j+1}, z_{i,j}, z_{i,j-1}, z_{i-1,j+1}, z_{i-1,j}, z_{i-1,j-1})$$

Ekkor a fenti összeg

$$E_z^2 = \Sigma_i \Sigma_j S(N_{i,j})$$

alakban írható fel, ahol S az alábbi 9-változós függvény:

$$S(N_{i,j}) = 2z_{i+1,j+1}^2 + 3z_{i+1,j}^2 + 3z_{i,j+1}^2 + 10z_{i,j}^2 + z_{i,j-1}^2 + z_{i-1,j}^2 +$$

$$+ 4z_{i+1,j+1}z_{i,j} - 4z_{i+1,j+1}z_{i,j+1} - 4z_{i+1,j+1}z_{i+1,j}$$

$$+ 2z_{i+1,j}z_{i-1,j} - 8z_{i+1,j}z_{i,j} + 4z_{i+1,j}z_{i,j+1}$$

$$- 8z_{i,j+1}z_{i,j} + 2z_{i,j+1}z_{i,j-1} - 4z_{i,j}z_{i,j-1} - 4z_{i,j}z_{i-1,j}$$

Keressük azon Z mátrixot, amelyre az E_z^2 függvény értéke minimális. Ez ott teljesül, ahol a függvény valamennyi $z_{i,j}$ szerinti parciális deriváltja nulla. A $z_{i,j}$ szerinti parciális derivált értékének meghatározásához a fenti összegből csak $z_{i,j}$-t tartalmazó tagokat kell figyelembe venni. Belátható, hogy ilyenek csak az

$S(N_{i+1,j+1})$, $S(N_{i+1,j})$, $S(N_{i+1,j-1})$,

$S(N_{i,j+1})$, $S(N_{i,j})$, $S(N_{i,j-1})$,

$S(N_{i-1,j+1})$, $S(N_{i-1,j})$, $S(N_{i-1,j-1})$

összegekben vannak, ezért

$$\partial E_z^2/\partial z_{ij} = \partial S(N_{i+1,j+1})/\partial z_{i,j} + \partial S(N_{i+1,j})/\partial z_{i,j} + \partial S(N_{i+1,j-1})/\partial z_{i,j} +$$
$$+ \partial S(N_{i,j+1})/\partial z_{i,j} + \partial S(N_{i,j})/\partial z_{i,j} + \partial S(N_{i,j-1})/\partial z_{i,j} +$$
$$+ \partial S(N_{i-1,j+1})/\partial z_{i,j} + \partial S(N_{i-1,j})/\partial z_{i,j} + \partial S(N_{i-1,j-1})/\partial z_{i,j}$$

Részletszámítások mellőzésével az egyes komponensek értéke:

$$\partial S(N_{i+1,j+1})/\partial z_{i,j} = 0$$

$$\partial S(N_{i+1,j})/\partial z_{i,j} = 2z_{i,j} + 2z_{i+2,j} - 4z_{i+1,j}$$

$$\partial S(N_{i+1,j-1})/\partial z_{i,j} = 0$$

$$\partial S(N_{i,j+1})/\partial z_{i,j} = 2z_{i,j} + 2z_{i,j+2} - 4z_{i,j+1}$$

$$\partial S(N_{i,j})/\partial z_{i,j} = 20z_{i,j} + 4z_{i+1,j+1} - 8z_{i+1,j} - 8z_{i,j+1} - 4z_{i,j-1} - 4z_{i-1,j}$$

$$\partial S(N_{i,j-1})/\partial z_{i,j} = 6z_{i,j} - 4z_{i+1,j} + 4z_{i+1,j-1} - 8z_{i,j-1} + 2z_{i,j-2}$$

$$\partial S(N_{i-1,j+1})/\partial z_{i,j} = 0$$

$$\partial S(N_{i-1,j})/\partial z_{i,j} = 6z_{i,j} - 4z_{i,j+1} + 2z_{i-2,j} - 8z_{i-1,j} + 4z_{i-1,j+1}$$

$$\partial (N_{i-1,j-1})/\partial z_{i,j} = 4z_{i,j} + 4z_{i-1,j-1} - 4z_{i-1,j} - 4z_{i,j-1}$$

Innen a parciális derivált:

$$\partial E_z^2/\partial z_{ij} = +2z_{i+2,j} + 4z_{i+1,j+1} - 16z_{i+1,j} + 4z_{i+1,j-1} +$$
$$+ 2z_{i,j+2} - 16z_{i,j+1} + 40z_{i,j} - 16z_{i,j-1} + 2z_{i,j-2} +$$
$$+ 4z_{i-1,j-1} - 16z_{i-1,j} + 4z_{i-1,j+1} + 2z_{i-2,j}$$

2-vel leosztva az alábbi maszk adódik:

$$\begin{array}{ccccc} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & 20 & -8 & 1 \\ & 2 & -8 & 2 & \\ & & 1 & & \end{array} \qquad (4)$$

Vagyis olyan Z mátrixot keresünk, amelynél minden egyes elemre a fenti maszkot illesztve a környező elemek súlyozott átlaga nullát ad.

A fenti levezetések a variációszámításban használatos Euler-Lagrange differenciálegyenlet segítségével is elvégezhetőek lettek volna. Eszerint az E_f funkcionál szélsőértékére teljesül

– membrán esetén: $f_{xx} + f_{yy} = 0$ (Laplace-féle differenciálegyenlet),

– vékonylemez esetén: $f_{xxxx} + 2f_{xxyy} + f_{yyyy} = 0$ (biharmonikus differenciálegyenlet).

Ha a fenti differenciálegyenletekre végezzük el a végeselemes átalakítást, vagyis a parciális deriváltakat véges differenciákkal helyettesítjük, akkor a fentiekkel megegyező eredményhez jutunk.

Az eddigiek során mind a membrán, mind a vékonylemez modell esetén egy-egy lineáris egyenletrendszer megoldására vezettük vissza a feladatot. A ritka mátrixú egyenletrendszerek szerkezetét szemléltető maszkok sugallják, hogy megoldásuk *iterációs módszer*rel célszerű. Jacobi-iterációt alkalmazva membrán esetén a

$$z_{i,j}' = (z_{i+1,j} + z_{i-1,j} + z_{i,j+1} + z_{i,j-1})/4 \qquad (5)$$

iterációs formula használható, amely annyit jelent, hogy a Z mátrixra ismételten *konvolúciót* alkalmazunk az

$$\begin{matrix} & 1/4 & \\ 1/4 & 0 & 1/4 \\ & 1/4 & \end{matrix} \qquad (6)$$

maszkkal mindaddig, amíg a mátrix be nem konvergál. A konvergencia biztosításához azonban különös gonddal kell eljárni a konvolúciós maszk megválasztásánál (elsősorban vékonylemez modell esetén), ezt tárgyaljuk a következő pontban.

10.2.2. Konvolúciós maszk meghatározása Fourier-analízissel

A digitális képfeldolgozásból jól ismert konvolúciós tétel szerint (lásd például Gonzalez, Wintz (1987)) két függvény konvolúciójának Fourier-transzformáltja megegyezik a függvények Fourier-transzformáltjai szorzatával, vagyis valamely f, g kétváltozós függvényekre

$$F(f*g) = F(f)\cdot F(g)$$

ahol F a Fourier-transzformációt, $*$ a konvolúciót jelöli.

Esetünkben a Z mátrixra és az M konvolúciós maszkra a következőt mondhatjuk:

$$F(Z*M) = F(Z)\cdot F(M)$$

ahol F diszkrét Fourier-transzformációt jelöl, az egyenlet jobb oldalán pedig mátrixok elemenkénti szorzása értendő. A konvolúciót iteratívan ismételve alkalmazzuk, így a következő áll fenn:

$$F(Z*M*\ldots*M) = F(Z)\cdot F(M)\cdot\ldots\cdot F(M) \qquad (7)$$

A fenti egyenlet jobb oldalán mátrixok elemenkénti szorzása történik, ezért az iteráció konvergenciája jól vizsgálható az $F(M)$ mátrix elemein keresztül.

Egy $g(x, y)$ függvény $G(u, v)$ Fourier-transzformáltja

$$G(u,v) = \iint g(x,y) \exp(-j2\pi(ux+vy))\, dx\, dy$$

módon definiált (j a képzetes egység), az inverz transzformáció hasonlóan:

$$g(u,v) = \iint G(x,y) \exp(j2\pi(ux+vy))\, du\, dv$$

A diszkrét transzformáció n x m méretű mátrix esetén

$$G(u,v) = \sum_{x=0}^{n-1} \sum_{y=0}^{m-1} g(x,y) \cdot \exp[-j2\pi(ux/n + vy/m)] \tag{8}$$

az inverz transzformáció pedig

$$g(x,y) = \frac{1}{nm} \sum_{u=0}^{n-1} \sum_{v=0}^{m-1} G(u,v) \cdot \exp[j2\pi(ux/n + vy/m)] \tag{9}$$

ahol j a képzetes egység. Megjegyzendő, hogy diszkrét esetben mind g, mind G (n x m)-re periodikus függvényeknek tekintendők.

> A diszkrét Fourier-transzformációt gyakran a fentitől kissé eltérő módon definiálják. Például Gonzalez, Wintz (1987) $n = m$ feltételezéssel él, és a (9) formulában szereplő $1/nm$ helyett mind a transzformációnál, mind az inverznél $1/n$ korrekciós tényezőt alkalmaz. Ennek az a magyarázata, hogy (8) és (9) egymás utáni alkalmazása az eredeti függvényt kell hogy visszaadja, de valójában mindegy, hogy az $1/nm$ korrekciót mikor végezzük el.
>
> Esetünkben a (7) összefüggést vizsgáljuk, amelynek bal oldalán egy, jobb oldalán $r+1$ (r az iterációszám) transzformáció történik, ezért a korrekciót az inverz transzformációra tettük át, egyébként a korrekciós tényező a (7) egyenlőséget felborítaná.

Először vizsgáljuk meg a membrán modellre levezetett (5) iterációs formulának megfelelő (6) iterációs maszk konvergenciáját! Ezen maszk Fourier-transzformáltja (a komplex számok abszolút értékét tekintve) a következő:

```
1.000  0.250  0.250
0.250  0.500  0.500
0.250  0.500  0.500
```

A (8) formulából jól látszik, hogy $G(0,0)$ értéke az összes $g(x, y)$ összegével egyenlő. Mivel a maszkelemek összege 1, a transzformált mátrix bal felső sarka szükségképpen 1-et tartalmaz. A többi elem 1-nél kisebb, vagyis a (7) formula jobb oldala olyan mátrixhoz tart, amelynek bal felső sarkában Z elemeinek s összege szerepel, a többi elem nulla. Erre az inverz transzformációt végrehajtva kapjuk a $Z*M*...*M$ mátrixot, amelynek minden eleme s/nm-hez tart.

Következmény: a membrán modell tetszőleges kezdőértékből indítva olyan konstans függvényhez konvergál, amelynek értéke a kezdőelemek átlaga. (Ez a modell elvi viselkedését mutatja abban az esetben, ha nincs illeszkedési feltétel, vagyis nem követeljük meg a felület illeszkedését adott magassági pontokra és szintvonalakra.)

Általában, egy iterációs maszkot akkor tekinthetünk konvergensnek, ha a Fourier-transzformáltjának a bal felső eleme 1, a többi elem abszolút értéke pedig kisebb mint 1.

Bár a fenti Fourier-analízis jól jellemzi az iterációs maszk viselkedését, a konvergencia egzakt bizonyításához még további tényezőket is figyelembe kellene venni:

1. A konvolúciós tétel alkalmazásához az M mátrixot nullákkal ki kell egészíteni a Z mátrix méretére, vagyis a vizsgálandó M mátrix mérete függ a mindenkori Z mátrix méretétől. Az eredeti és a kiegészített mátrix Fourier-transzformáltja nem azonos, de a konvergenciára vonatkozó jellemzőik alapvetően megegyeznek.
2. A mátrix szélein az iterációs maszk szükségképpen módosul (lásd később), ennek vizsgálatától itt eltekintünk.
3. Az adott pontok (szintvonalak pontjai és magassági pontok) esetében a számítás szintén módosul (lásd később).

Hangsúlyozzuk, hogy a gyakorlati futási tapasztalapok megerősítik a Fourier-analízis eredményét, vagyis a fent divergensnek mutatkozó maszkok valóban divergálnak (47. ábra), a konvergensnek mutatkozók valóban konvergálnak (45. és 46. ábra).

Vékonylemez modell esetén a (4) feltételt

$$20z_{i,j} + s_{i,j} = 0 \tag{10}$$

alakban írjuk, ahol

$$s_{i,j} = z_{i+2,j} + 2z_{i+1,j+1} - 8z_{i+1,j} + 2z_{i+1,j-1} +$$
$$+ z_{i,j+2} - 8z_{i,j+1} - 8z_{i,j-1} + z_{i,j-2} +$$
$$+ 2z_{i-1,j-1} - 8z_{i-1,j} + 2z_{i-1,j+1} + z_{i-2,j}$$

A (10) egyenlet legegyszerűbben a következőképp alakítható iterációs formulává:

$$z_{i,j}' = -s_{i,j}/20$$

amely az alábbi konvolúciós maszknak felel meg:

$$\frac{1}{20}\begin{bmatrix} & & -1 & & \\ & -2 & 8 & -2 & \\ -1 & 8 & 0 & 8 & -1 \\ & -2 & 8 & -2 & \\ & & -1 & & \end{bmatrix} \tag{11}$$

Ennek Fourier-transzformáltja (a komplex számok abszolút értékét tekintve) az alábbi:

```
1.000  0.905  0.345  0.345  0.905
0.905  0.618  0.250  0.250  0.618
0.345  0.250  1.618  1.618  0.250
0.345  0.250  1.618  1.618  0.250
0.905  0.618  0.250  0.250  0.618
```

Mivel a mátrix 1-nél nagyobb elemeket is tartalmaz, így nem konvergens. Tekintsük azonban a

$$z_{ij}' = (12z_{ij} - s_{ij})/32$$

iterációs formulát, ahol a gépi számítás gyorsítása érdekében választottunk a 32-es osztót (ekkor ugyanis osztás helyett csak 5 bináris helyértéket kell jobbra léptetni). A fenti formula a

$$\frac{1}{32}\begin{bmatrix} & & -1 & & \\ & -2 & 8 & -2 & \\ -1 & 8 & 12 & 8 & -1 \\ & -2 & 8 & -2 & \\ & & -1 & & \end{bmatrix} \qquad (12)$$

iterációs maszknak felel meg, ennek Fourier-traszformáltja

```
1.000  0.940  0.591  0.591  0.940
0.940  0.761  0.219  0.219  0.761
0.591  0.219  0.636  0.636  0.219
0.591  0.219  0.636  0.636  0.219
0.940  0.761  0.219  0.219  0.761
```

Mivel a bal felső sarok kivételével minden elem kisebb 1-nél, így az iterációs maszk konvergens.

A továbbiakban membrán modell esetén a (6), vékonylemez modell esetén a (12) iterációs maszkokat alkalmazzuk.

10.2.3. Illeszkedési feltétel

Ha pontos illeszkedést követelünk meg, akkor az $f(x, y)$ függvény az adott (x_1, y_1), ..., (x_m, y_m) pontokban a pontosan az előre adott d_1, ..., d_m értékeket kell hogy felvegye. Ebben az esetben az iteráció kezdetén az adott pontok értékét a d_1, ..., d_m értékekre állítjuk be, és az iteráció során csak a többi pont értékét változtatjuk.

Bizonytalan mérési adatok esetén nem célszerű pontos illeszkedést megkövetelni, ilyenkor az eltérést a hibák négyzetösszegével szokás mérni:

$$E_d(f) = \Sigma_i (f(x_i, y_i) - d_i)^2$$

Az optimális felület leírására Szeliski (1990) szabályozás-elméleti alapon az

$$E(f) = E_s(f) + E_d(f)$$

egyesített energiafüggvényt használja, ahol E_s fejezi ki a *símasági feltételt* (smoothness constraint, amelyet az eddigiekben vizsgáltunk), E_d pedig az *illeszkedési feltételt* (data compatibility constraint). Vagyis olyan $f(x, y)$ függvényt keresünk, amelyre az egyesített energiafüggvény értéke minimális, azaz a lehető legsímább és illeszkedik adott pontokra.

Ha az egyes pontokban a mérés pontossága eltérő, ezt w_i súlyok segítségével fejezhetjük ki:

$E_d(f) = \Sigma_i w_i[f(x_1,y_1) - d_i]^2$

Fizikai modell szintjén a fentiek úgy képzelhetők el, mint ha az adott pontokhoz rugók rögzítenék a felszínt, ahol w_i a i-edik rugó erősségét fejezi ki.

> Megjegyzendő, hogy Szeliski (1990) az egyesített energiafüggvénynél $E_s(f)$-hez még egy súlyozó konstanst is használ, ennek hatását azonban a gyakorlatban a fenti w_i súlyok vehetik át, ezért azt elhagytuk.

Végeselemes módszer esetén az $f(x, y)$ függvény helyett a $Z[i,j]$ mátrixot vesszük. A továbbiakban az egyes $z_{i,j}$ értékeket tekintjük változóknak, és az illeszkedési energiafüggvényt a következőképp írjuk fel:

$E_d(Z) = \Sigma_{i,j} w_{i,j}(z_{i,j} - d_{i,j})^2 = \Sigma_{i,j} w_{i,j}(z_{i,j}^2 - 2z_{i,j}d_{i,j} + d_{i,j}^2)$

ahol $w_{i,j}$ értéke csak ott nemnulla, ahol adott pont van, és értéke annál nagyobb, minél pontosabb a mérés. Keressük azon Z mátrixot, amelyre az

$E(Z) = E_s(Z) + E_d(Z)$

függvény értéke minimális. Ez ott teljesül, ahol a függvény valamennyi $z_{i,j}$ szerinti parciális deriváltja nulla:

$\partial E/\partial z_{i,j} = \partial E_s/\partial z_{i,j} + w_{i,j}(2z_{i,j} - 2d_{i,j}) = 0$

A (10) formula felhasználásával

$\partial E/\partial z_{i,j} = 20z_{i,j} + s_{i,j} + w_{i,j}(2z_{i,j} - 2d_{i,j}) = 0$

Ezt iterációs formulává alakítva:

$32z_{i,j} = 12z_{i,j} - s_{i,j} - 2w_{i,j}z_{i,j} + 2w_{i,j}d_{i,j}$

$(32 + 2w_{i,j})z_{i,j} = 12z_{i,j} - s_{i,j} + 2w_{i,j}d_{i,j}$

$z_{i,j}' = (12z_{i,j} - s_{i,j})/(32 + 2w_{i,j}) + w_{i,j}d_{i,j}/(16 + w_{i,j})$

Más felírásban:

$z_{i,j}' = a\cdot(12z_{i,j} - s_{i,j})/32 + b\cdot d_{i,j}$

ahol

$a = 16/(16 + w_{i,j}), \quad b = w_{i,j}/(16 + w_{i,j})$

és

$a + b = 1$

teljesül. Vagyis, a $z_{i,j}$ pont új értékét az iterációs maszkkal számolt érték és $d_{i,j}$ súlyozott átlagaként képezzük.

10.2.4. Szélek kezelése

A Z mátrix szélein a hiányzó szomszédok miatt az iterációs maszkot módosítani kell. Erre Terzopoulos (1988) ad javaslatot, amelynek lényege: az iterációs maszkot komponensekből építi fel, és a széleken csak azokat a komponenseket veszi figyelembe, amelyek teljes egészében a mátrixba esnek.

Saját alkalmazásunk esetén a fenti helyett egyszerűen körülkereteztük a mátrixot a szélső pixel értékek kiterjesztésével, így a szélső elemek a belsőkkel azonos módon kezelhetőkké váltak (43. ábra). Ez a megoldás programozástechnikailag egyszerűbb és gyorsabb. Membrán modell esetén a keretezéses megoldás egybeesik a komponensekből építkezővel, vékonylemez modell esetén a kettő némileg eltér, de tapasztalataink szerint ez érdemben nem befolyásolja a terepmodellt.

$$
\begin{array}{cccccccc}
z_{11} & z_{11} & z_{11} & \cdots & z_{1m} & z_{1m} & z_{1m} \\
z_{11} & z_{11} & z_{11} & \cdots & z_{1m} & z_{1m} & z_{1m} \\
z_{11} & z_{11} & z_{11} & \cdots & z_{1m} & z_{1m} & z_{1m} \\
 & & & & & & \\
\cdot & \cdot & \cdot & & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & & \cdot & \cdot & \cdot \\
 & & & & & & \\
z_{n1} & z_{n1} & z_{n1} & \cdots & z_{nm} & z_{nm} & z_{nm} \\
z_{n1} & z_{n1} & z_{n1} & \cdots & z_{nm} & z_{nm} & z_{nm} \\
z_{n1} & z_{n1} & z_{n1} & \cdots & z_{nm} & z_{nm} & z_{nm}
\end{array}
$$

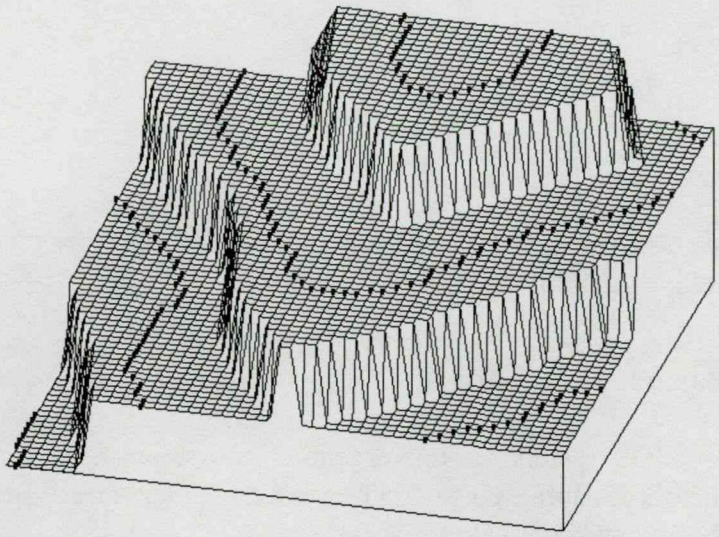
43. ábra. A szélek kezelése keretezéssel

10.2.5. Kezdőértékadás

Iterációs eljárások konvergenciáját jelentősen gyorsíthatja a jó kezdőértékadás. Szintvonalrajz esetén kézenfekvő gondolat, hogy minden pont a hozzá legközelebbi szintvonal magasságértékével induljon. Erre adunk lineáris idejű megoldást az alábbiakban.

Kezdetben a *Z* mátrix háromféle pontot tartalmaz:
– szintvonalpont,
– egyedi magasságpont,
– definiálatlan pont.

Ebből a mátrixból egy *D* távolság-fedvényt készíthetünk, amely minden definiálatlan ponthoz a legközelebbi értékes ponttól való távolságát rendeli. Most nem valós távolságra, hanem az úgynevezett "city block distance"-re gondolunk, amely 4-szomszédság szerint lépkedve definiálja a távolságot. A távolság-fedvény lineáris időben számítható (Borgefors, 1984) a következőképpen:



44. ábra. Kezdőértékadás után előálló terep

1. D kezdetben legyen egyenlő Z-vel úgy, hogy a definiált pontok D-ben 0 értéket, a definiálatlanok plusz végtelen értéket kapnak.

2. D-t körülkeretezzük plusz végtelen értékű pontokkal 1 pixel szélességben.

3. Előre haladó fázis:

```
for i=1 to n
for j=1 to m
  d_i,j := minimum(d_i-1,j+1, d_i,j-1+1, d_i,j)
```

4. Visszafelé haladó fázis:

```
for i=n to 1
for j=m to 1
  d_i,j := minimum(d_i+1,j+1, d_i,j+1+1, d_i,j)
```

A fenti algoritmus 3., 4. lépését úgy módosítjuk, hogy közben a Z mátrix definiálatlan értékeit feltöltjük: $z_{i,j}$ mindig annak a szomszédnak az értékét veszi át, amelyikre a távolságfedvényben a minimum-feltétel teljesül:

3. Előre haladó fázis:

```
for i=1 to n
for j=1 to m
{ if (d_i,j > d_i-1,j+1)
    then { d_i,j := d_i-1,j+1,   z_i,j := z_i-1,j }
  if (d_i,j > d_i,j-1+1)
    then { d_i,j := d_i,j-1+1,   z_i,j := z_i,j-1 }
}
```

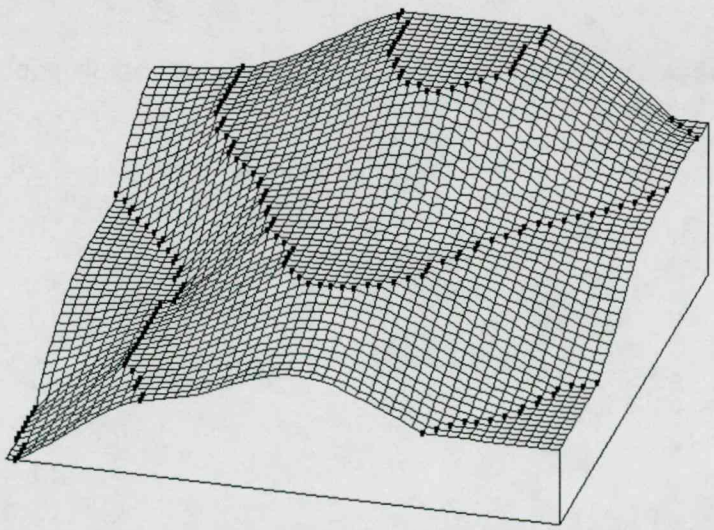
4. Visszafelé haladó fázis:

```
for i=n to 1
for j=m to 1
{ if (d_i,j > d_i+1,j+1)
    then { d_i,j := d_i+1,j+1,  z_i,j := z_i+1,j }
  if (d_i,j > d_i,j+1+1)
    then { d_i,j := d_i,j+1+1,  z_i,j := z_i,j+1 }
}
```

A leírt algoritmussal a Z mátrixból egy D távolság-fedvényt és egy Z' kezdőérték-mátrixot állítunk elő, ez utóbbit a 44. ábra szemlélteti.

10.2.6. Relaxáció (iteráció)

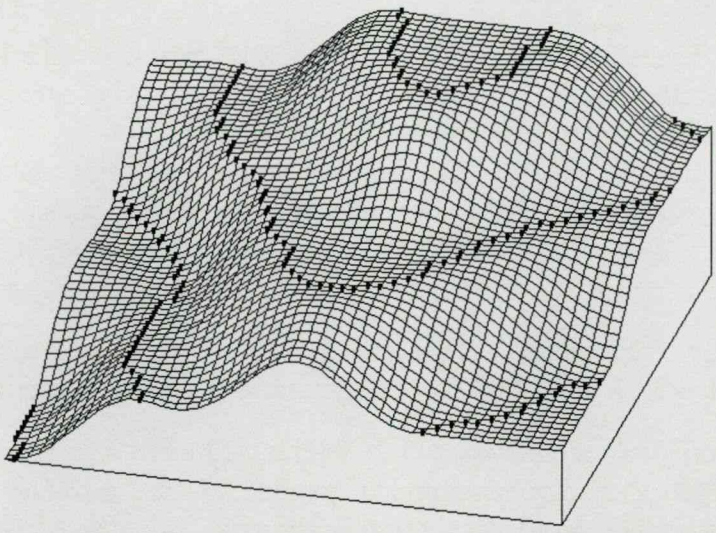
Membrán modell esetén a konvergencia gyors, a terep már 40 iteráció után hozzávetőleg kialakul (45. ábra). Ugyanakkor a vékonylemez modell még 500 iteráció után is használhatatlan eredményt produkál (46. ábra).



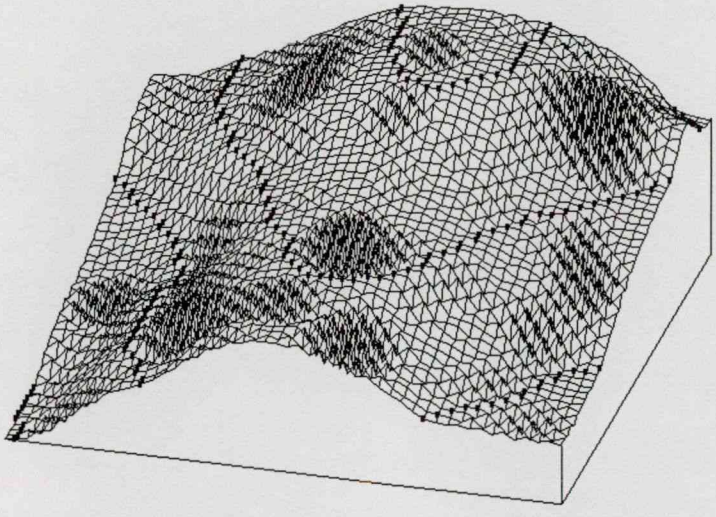
45. ábra. Membrán modell 40 iteráció után

Terzopoulos (1983) megállapítása szerint n x n-es mátrix esetén elfogadható eredmény eléréséhez n^r iteráció szükséges, ahol r a parciális deriváltak fokszáma (vagyis membrán esetén r=1, vékonylemez esetén r=2). A hiba Fourier-analízise azt mutatja, hogy a nagyfrekvenciás komponensek gyorsan eltűnnek, míg a kisfrekvenciások sokáig megmaradnak. Vagyis, az iteráció a hibák gyors símítására alkalmas, de a teljes bekonvergálás igen sokáig tarthat.

Ilyen lassú konvergencia mellett a vékonylemez modell a gyakorlatban használhatatlan. A megoldást a multigrid technika jelenti, amely már 50 iteráció után jó eredményt ad (lásd az 55. ábrát a 11.3. fejezetben).



46. ábra. Vékonylemez modell 500 iteráció után. A felszín természetellenesen hullámos, még láthatóan nem konvergált be



47. ábra. Divergens maszk hatása közel bekonvergált állapotból indítva, 7 iteráció után. (10 iteráció után a terep már szétesik.)

Megjegyzések a 45., 46. és 47. ábrákhoz:

- A kivágat mátrix mérete mindegyik esetben 50 x 50 pixel.
- A 45. és 46. ábra esetében a 44. ábra szerinti kezdőállapotból, a 47. ábra esetén a 55. ábra szerinti kezdőállapotból indultunk.
- A 45. ábra szerinti terepmodellt a (6) maszkkal, a 46. ábráét a (12) maszkkal, végül a 47. ábráét a (11) maszkkal állítottuk elő.
- Mindegyik esetben pontos illeszkedéssel számoltunk szintvonalaknál.

11. DEM előállítása szintvonalrajzból

Ebben a fejezetben részletesen ismertetjük azt az eljárást, amelynek tömör összefoglalását Katona (1995) tartalmazza, és amely – kis eltérésekkel – a DDM-10 projektben gyakorlati alkalmazásra is került (11.5. fejezet).

Az eljárás lényege: mindvégig raszteresen dolgozunk, vagyis elkerüljük a szintvonalak szokásos vektorizálását, majd DEM generáláskor visszaraszterizálását. Amint látni fogjuk, ez a *tisztán raszteres megoldás* biztosítja a legpontosabb terepmodellt. Az eljárás három fő lépésből áll:

1. Vékonyított szintvonalrajz előállítása (11.1. fejezet). A szkennelt raszterkép manuális javítása és a szintvonalak algoritmikus vékonyítása után a 48. ábra szerinti mátrixhoz jutunk.

2. Szintvonalmátrix előállítása (11.2. fejezet). Minden szintvonalhoz magasságértéket rendelünk, amelyet a szintvonal minden pontja felvesz. Ezután felvisszük az egyedi magasságpontokat is (49. ábra). Itt a szintvonalmátrix felbontása megegyezik a szkennelési felbontással (így kapjuk a legnagyobb pontosságot).

3. Szintvonalmátrixból DEM generálása (11.3. fejezet). Multigrid relaxációs módszerrel határozzuk meg a definiálatlan pontok értékét a 10. fejezetben tárgyalt variációs spline illesztés alapján, ezzel áll elő a végső DEM, amelynek felbontása már kisebb is lehet a szintvonalmátrixénál.

```
0 0 1 0 0 1 0 0 0 0
0 1 0 0 0 1 0 0 0 0
0 1 0 0 0 1 0 0 0 1
1 0 0 0 1 0 0 0 1 0
0 0 0 1 0 0 0 1 0 0
0 1 1 0 0 0 1 0 0 0
1 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 1
```

48. ábra. Szkennelt,vékonyított szintvonalrajz részlete.

```
x x 200 x x 240 x x x x
x 200 x x x 240 x x x x
x 200 x x x 240 x x x 280
200 x x x 240 x x x 280 x
x x x 240 x x x 280 x x
x 240 240 x x x 280 x x x
240 x x x x x 280 x x x
x x x 253 x x x 280 x x
x x x x x x x x 280 280
```

49. ábra. Szintvonalmátrix részlete. X definiálatlan pontot, a 253 érték egyedi magasságpontot jelöl.

11.1. Vékonyított szintvonalrajz előállítása

A feldolgozás ezen fázisa sok manuális munkát igényel, amelyhez részben szoftver támogatás is nyújtható. Az elvégzendő feladatok:

1. Szintvonalrajz szkennelése. Nyersanyagként az adott térképtípus szintvonalas fóliái a legalkalmasabbak, amelyek *csak* domborzati információt tartalmaznak. A szkennelt raszterképet B_0-lal jelöljük.

2. Szintvonalrajz transzformációja a szelvénykeret négy sarokpontjának megfelelően. A szelvénykereten kívüli rész levágásra kerül (6.3. fejezet). A transzformált raszteres állományt B_1-gyel jelöljük (ezt megőrizzük).

3. Szintvonalrajz tisztítása valamely raszter-editor programmal. El kell távolítani a rajzról a zajokat és a szintvonalakon kívül minden jelkulcsi elemet.

4. Összeérések megszüntetése. Minden szintvonal-összeérés hibát okoz a továbbiakban, így ezeket kivétel nélkül meg kell szüntetni. Ha a szintvonalak annyira sűrűn helyezkednek el, hogy az adott szkennelési felbontás mellett nem választhatók szét, akkor minden másodikat meg kell szakítani.

5. Szintvonalrajz algoritmikus vékonyítása. A képfeldolgozásban használatos valamely vékonyító (vázképző) algoritmus segítségével minden szintvonalat egy pixel vastagságúra húzunk össze.

6. Szakadások megszüntetése. Szintvonal szakadások az alábbi okok miatt léphetnek fel:
– Szkennelési hiba.
– Nyomdatechnikai okok. Mivel általában a szintvonalas réteg nyomdai fóliáját dolgozzuk fel, így ezen megszakadnak a szintvonalak minden olyan helyen, ahol azokat valamely más rétegbeli elem – például út – kitakarja.
– Felező és negyedelő szintvonalak. Ezek a jelkulcs szerint szaggatott vonalak, amelyeket folytonos vonallá kell összekötni.

A szakadások összekötése szoftverrel támogatható: egy algoritmus megkeresi az egymáshoz közeli végpontokat, és felajánlja azok összekötését. (Vékonyítás után a végpontok már könnyen felismerhetők, mert 8-szomszédság mellett csak egy szintvonalpont-szomszédjuk van.)

A szakadások csak csekély hibát okoznak a további feldolgozásban, ezért e műveletnél nem szükséges teljességre törekedni.

7. Összeérés ellenőrzés. A program rákeres minden megmaradt elágazási pontra (amelynek 8-szomszédság mellett 2-nél több szintvonalpont-szomszédja van). Az így kimutatott összeéréseket meg kell szüntetni. Az eredményül kapott raszteres állományt B-vel jelöljük.

11.2. Szintvonalmátrix előállítása

E művelet célja: a szintvonalakhoz történő magasságérték hozzárendeléssel és magassági pontok megadásával a 48. ábra szerinti *B* mátrixból a 49. ábra szerinti *Z* mátrix előállítása. Tekintettel a gyakran rendhagyó domborzati viszonyokra a művelet automatikusan nem oldható meg (lásd 8.2.2. alfejezet), viszont gépi támogatással jelentősen gyorsítható.

Ezen támogatás alapja, hogy ha a felhasználó rákattintással értéket ad egy szintvonalnak, akkor azt az értéket a vonal minden pontjához hozzá kell rendelni. Ha a Z mátrix nem fér be egyben a memóriába (ami könnyen előfordulhat, hiszen a *B* mátrix minden bitjét itt egy 16- vagy 32-bites szóval helyettesítjük), akkor a magasságérték végigterjesztése igen lassúvá válik, ami az interaktív munkát gyakorlatilag lehetetlenné teszi.

Erre a problémára olyan megoldást alkalmazunk, amely a mátrix méretétől függetlenül, *gyakorlatilag konstans időben* oldja meg a magasságérték hozzárendelést a szintvonal valamennyi pontjához. A memóriában csak akkora képrészlet kell hogy legyen, amelyet a felhasználó a képernyőn lát. Ehhez előbb egy *W* munkamátrixot kell létrehozni.

11.2.1. A bináris *B* mátrix konverziója a *W* szavas mátrixra

B-ben megkeressük a 8-szomszédság szerint összefüggő alakzatokat (szintvonalakat), mindegyikhez egyedi azonosítószámot (sorszámot) rendelünk, és *W*-ben az alakzat minden pontját ezzel az azonosítószámmal helyettesítjük. Erre a klasszikus feladatra (connected component labelling) számos *lineáris idejű algoritmus* ismert (lásd például Rosenfeld és Pfaltz (1966), Stefano és Bulgarelli (1999)) ezeknek egy kissé módosított változatát alkalmazzuk.

Az eljárás során az azonosítási számok 1-től kezdve folyamatosan kerülnek kiosztásra legfeljebb *maxid* értékig. A cél az, hogy különböző alakzatokhoz (szintvonalakhoz) mindenképp különböző azonosítási számok tartozzanak, ha pedig egy alakzathoz több azonosítási szám is kiosztásra kerül, akkor ezek egy speciális NUMLIST adatstruktúrában összekapcsolódjanak.

A NUMLIST adatszerkezet a NUM és LIST tömbökből áll. NUM fastruktúrákat, LIST pedig az egyes fák gyökerénél elhelyezett információkat tartalmazza:

– A NUM tömb *maxid*+1 elemű, *i*-edik eleme az *i* azonosítási számhoz tartozó pointer. Ha a pointer pozitív, akkor NUM-beli címet, ha negatív, akkor LIST-beli címet jelent. Egy *maxnum* változó a legnagyobb kiosztott azonosítási szám értékét tartalmazza (induláskor *maxnum* = 0).

– A LIST tömb 3-szavas listaelemek sorozatából áll, egy elem felépítése (x, y, z), ahol (x, y) az objektum (szintvonal) egy tetszőleges referenciapontja (az objektum megtalálásának megkönnyítésére), z pedig az adott objektumhoz rendelt számérték (szintvonal magasság). Kezdetben valamennyi listaelem egy szabad láncra van fűzve, így LIST kezdőértéke: (–1, –1, 1), (–1, –1, 2), (–1, –1, 3), ..., (–1, –1, –1), ahol (–1, –1) = definiálatlan koordináták, z pedig most pointerként szolgál, amely a következő listaelem címét mutatja (–1 a lista vége). Egy *kezdlist* változó a szabad lánc első elemére mutat (kezdetben *kezdlist* = 0).

Az eljárás a *B* mátrix egyszeri sorfolytonos végigjárásával végezhető. Egyszerre mindig két sort tartunk a tárban: az aktuális sort és a megelőző sort. A 0 értékű pixelek helyett *W*-be definiálatlan (X) értéket, az 1 értékűek helyett azonosítási számot írunk (negatív előjellel, hogy megkülönböztessük a későbbiekben beírandó magasságértékektől). Az azonosítási számok kiosztása a *B* mátrix aktuális során balról jobbra haladva a következőképp történik:

– Az aktuális sorban rendre megkeressük az összefüggő 1-es pixelsorozatokat.

– Ha olyan 1-es pixelsorozatot találtunk, amelynek a megelőző sorban nincs 1-es szomszédja, akkor új azonosítási szám kerül kiosztásra (valamennyi pixelhez ugyanaz). Az első pixel (x, y) koordinátáit elhelyezzük a szabad lánc elejéről leválasztott elembe, amelynek z komponensét nullázzuk. NUM-ba pointert helyezünk el, amely erre a listaelemre mutat:

```
maxnum := maxnum+1              // maxnum-ot osztjuk ki a pixelekhez
u := kezdlist
kezdlist := LIST[kezdlist].z    // Ha kezdlist=-1, akkor LIST betelt
NUM(maxnum) := -u;
LIST[u].x := x
LIST[u+1].y := y
LIST[u+2].z := 0
```

– Ha az előző sorban volt egy vagy több 1-es szomszéd, és mindegyiknek ugyanaz volt az azonosítási száma, akkor a közös azonosítási számot osztjuk ki.

– Ha az előző sorban több 1-es szomszéd is volt, és ezek különböző azonosítási számokkal rendelkeztek (legyenek ezek pl. num_1 és num_2), akkor az egyik (mondjuk az első) azonosítási számot osztjuk ki, és a közössé vált azonosítási számokhoz tartozó fákat NUMLIST-ben összekapcsoljuk a következőképp. Megkeressük a num_1-hez és num_2-höz tartozó fák gyökerét, vagyis NUM-ban addig keresünk visszafelé, amig negatív pointert nem találunk:

```
while NUM(num1) ≥ 0 do num1:=NUM(num1)
while NUM(num2) ≥ 0 do num2:=NUM(num2)
```

Most num1 fájához kapcsoljuk num2 fáját: LIST-ben töröljük a NUM(num2) című listaelemet és a szabad lánchoz kapcsoljuk, majd

```
NUM(num2) := num1
```

A létrejött *W* mátrix tulajdonsága: ha kiválasztunk egy szintvonalpontot, az ott lévő azonosító alapján NUM-ban megkeressük a megfelelő fa gyökerét, majd az ott található, LIST-re mutató pointerrel megkapjuk a szintvonal (x, y) referenciapontját és a szintvonalhoz rendelt z magasságértéket lekérdezhetjük vagy módosíthatjuk. A NUM-ban szereplő fák mélysége a szintvonalak kanyargósságától függ, a gyakorlatban 1 és 10 között mozog. Erre mondtuk fentebb, hogy a magasságérték hozzárendelés/lekérdezés "gyakorlatilag konstans időben" végezhető.

> Amennyiben a modellezendő területen negatív magasságértékellel is számolni kell, úgy az objektum-azonosítók nem 1-től *maxid*-ig, hanem z_0+1-től z_0+*maxid*-ig osztandók ki – ugyanis a *W* mátrixba az objektum-azonosítókat negatív előjellel írtuk be, és csak így garantálható, hogy azok ne ütközzenek a magasságértékekkel.

11.2.2. Interaktív értékadás a szintvonalaknak

Az eljárás során háttérképként látjuk a B_1 raszterképet, amely még valamennyi jelkulcsot, megírást tartalmazza. Az eljárás lépései:

– Meg kell adni a program számára az alapszintközt.

– Be kell gépelni egy kezdő magasságértéket.

– Egérrel rákattintunk a megfelelő szintvonalra, ekkor a LIST-be beíródik a megfelelő magasságérték.

– Rákattintunk a szomszédos szintvonalra, amelynek pontjaihoz – beállítástól függően – alapszintközzel növelt vagy csökkentett érték rendelődik, és így tovább.

– Rendhagyó esetekben a magasságértéket egyedileg kell begépelni.

– Magassági pont értékét szintén begépeléssel lehet megadni. Amennyiben a magassági pontok (x, y, z) koordinátái egy szövegfájlon rendelkezésre állnak, úgy azokat egy segédprogram automatikusan beemeli a W mátrixba. Ha a szkennelt szelvényre automatikus interpretációt alkalmazunk, ennek egyik outputja lehet egy ilyen szövegfájl.

Amikor a felhasználó értéket ad egy szintvonalnak, az rögtön átszineződik – mivel a megjelenítés NUMLIST alapján történik. Ez azt az illúziót kelti, hogy valamennyi szintvonalpont értéket kapott. A LIST-ben tárolt referenciapontok segítségével könnyen megkereshetők a még értékkel nem rendelkező szintvonalak.

Ha a felhasználó már valamennyi szintvonalnak értéket adott, akkor a W mátrix újabb sorfolytonos végigjárásával az azonosítási számokat lecseréljük a LIST-beli magasságértékekre, és egy Z mátrixot állítunk elő, amelynek minden egyes eleme egy *magasságértéket* és egy kétbites *jelzőkódot* tartalmaz. A mátrixban háromféle adat szerepelhet:

– Magassági pont: a mátrixelem a magasságértéket és 11 jelzőkódot tartalmaz.

– Szintvonalpont: a mátrixelem a szintvonal magasságértékét és 01 jelzőkódot tartalmaz.

– Definiálatlan pont: a mátrixelem X-szel jelölt definiálatlan magasságértéket és 00 jelzőkódot tartalmaz. (A definiálatlan pontokra 16-bites implementációban –32768 értéket alkalmaztunk.)

A további feldolgozás során a definiálatlan pontok magasságértéket kapnak, de megtartják 00 jelzőkódjukat (ez lesz a negyedik adattípus a mátrixban).

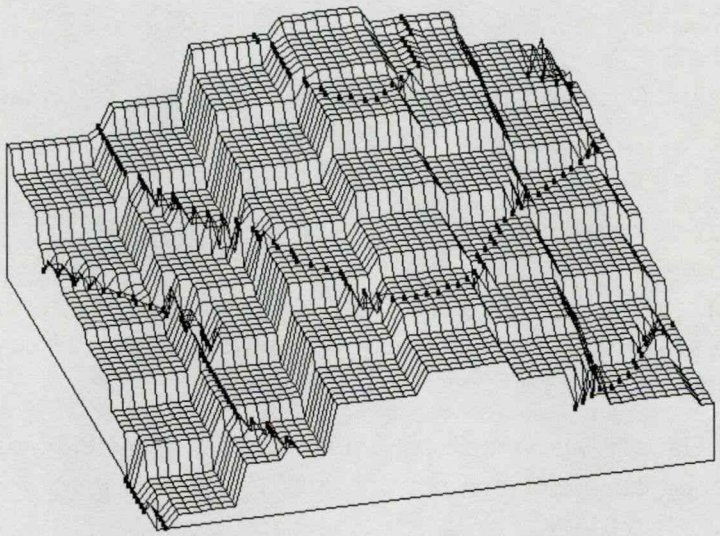
11.3. DEM generálása multigrid relaxációval

A digitális terepmodell előállításához a 10.2. fejezetben bemutatott végeselemes módszert alkalmazzuk vékonylemez modell szerint. Amint ott rámutattunk, az eljárás konvergenciája igen lassú: a hatások pixelről pixelre terjednek, így a távoli szintvonalak közötti kölcsönhatás csak lassan jelentkezik, az egész fedvényre érvényes globális hatások pedig még nehezebben érvényesülhetnek.

A megoldást a következő ötlet kínálja: készítsünk egy kicsinyített (durva felbontású) fedvényt a szintvonalpontok átlagolásával, és előbb erre végezzünk iterációt! A fent említett

távoli szintvonalak most közel kerülnek egymáshoz, a konvergencia gyors lesz, és a globális hatások is jól érvényesülhetnek a kicsinyített fedvényen. Ugyanakkor a számításigény drasztikusan csökken a mátrix kisebb méretéből adódóan is. Az így kapott kicsinyített terepmodellt használjuk fel kezdőértékként az eredeti szintvonalmátrix definiálatlan pontjaiban! Ekkor az iteráció már a végeredményhez közeli kezdőértékkel indul (50. ábra), tehát az iterációs lépések száma drasztikusan csökken.

Végül alkalmazzuk a fenti technikát több szinten, így jutunk a *multigrid relaxáció* elvéhez, amelyet számos területen alkalmaznak, például a digitális képfeldolgozásban, differenciálegyenletek megoldására, stb. (Brand, 1982).



50. ábra. Kezdőértékadás 8-szoros kicsinyítésű fedvényből

11.3.1. A multigrid relaxáció módszere

Alábbiakban saját multigrid eljárásunkat mutatjuk be, amelyet speciálisan szintvonalas fedvényekhez dolgoztunk ki.

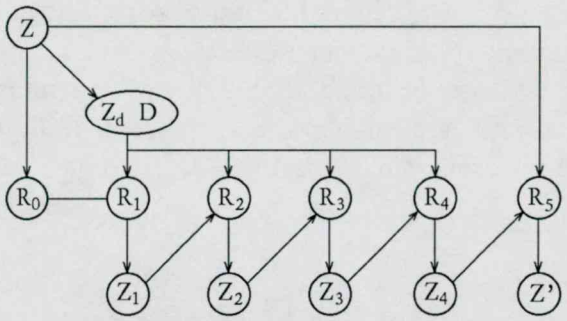
Az egyszerűség kedvéért tegyük fel, hogy a Z szintvonalmátrix 2^n x 2^n méretű. Ekkor n felbontási fokozatban végzünk iterációt, egyre finomodó rasztereken. Az i-edik menetben egy 2^i x 2^i méretű redukált (kicsinyített) Z_i mátrixszal dolgozunk (51. ábra):

– Először a Z mátrixból egy R_0 1 x 1-es (azaz egyelemű) "mátrixot" képezünk az összes Z-beli szintvonalpont átlagolásával. (Ez a lépés megspórolható becsült átlagérték megadásával.)

– Ezután a Z mátrixból egy 2 x 2-es kicsinyített R_1 mátrixot képezünk, amelynek definiálatlan elemei – ha vannak ilyenek – R_0-ból kapnak kezdőértéket. Erre az R_1-re végzünk iterációt, ennek eredménye a Z_1 mátrix.

– Ezután egy 4 x 4-es R_2 redukált mátrixot képezünk, amelynek definiálatlan elemei a 2 x 2-es Z_1 mátrix megfelelő elemeinek értékét kapják kezdőértékként. R_2-re is iterálunk, így áll elő Z_2.

– Az eljárást folytatva eljutunk a R_n=Z mátrixhoz, amelynek definiálatlan pontjai Z_{n-1}-ből kapnak kezdőértéket, erre iterálva a Z' eredménymátrixot nyerjük.

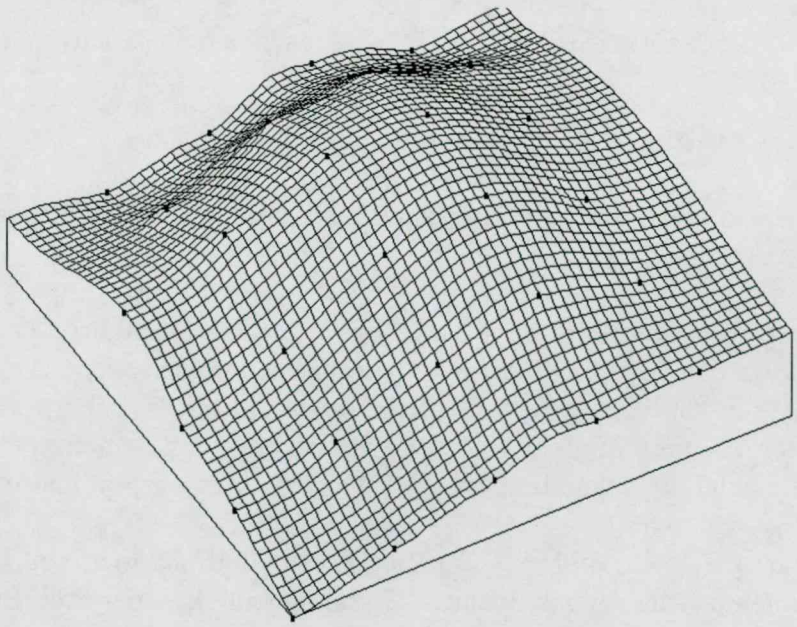


51. ábra. Multigrid eljárás vázlata $n = 5$ esetén. Z_d és D magyarázatát lásd később

11.3.2. A redukció kérdése

A redukált mátrixok előállítási módja kulcsfontosságú a módszer hatékonysága szempontjából: a szükséges iterációs lépések száma jelentősen csökkenthető, ha megfelelően érzékeny eljárást alkalmazunk a redukciónál.

Elnevezés. Ha egy n x m méretű Z mátrixot q x q méretű négyzetekre osztunk, és minden négyzetet egyetlen elemmel helyettesítünk, akkor egy n/q x m/q méretű *q-redukált R mátrix*ot kapunk.

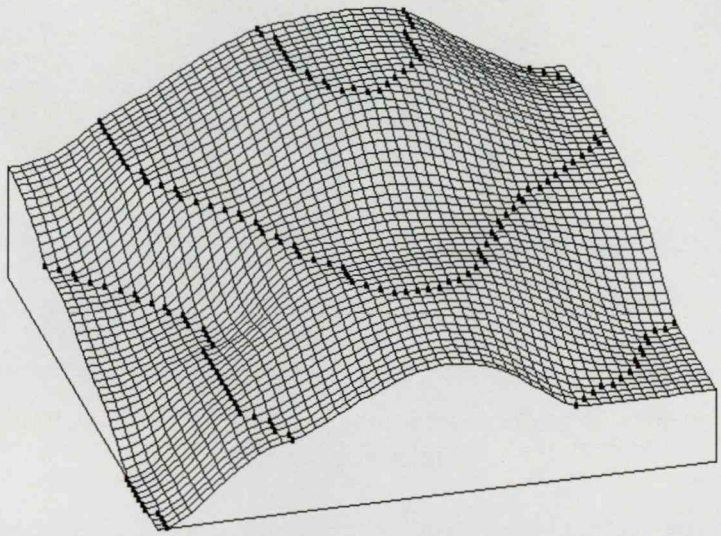


52. ábra. Egyszerű redukciós eljárással nyert DTM magassági pontok esetén

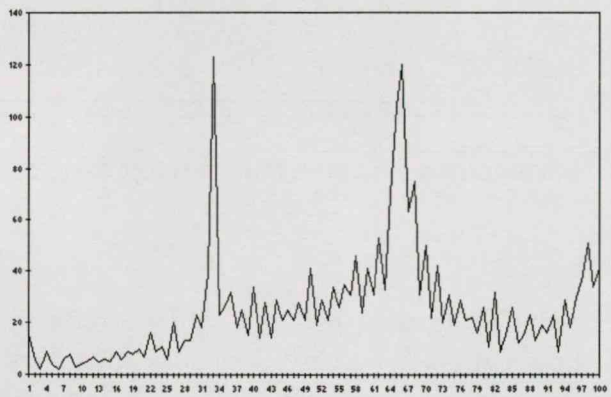
Egyszerű redukciós eljárás. Ha egy q x q méretű négyzet tartalmazott szintvonalponto(ka)t, akkor azok átlaga lesz a redukált pixel értéke, és ezt adott pontnak tekintjük az iteráció során. Ha a q x q méretű négyzet nem tartalmazott szintvonalpontot, akkor a redukált pixel értéke definiálatlan lesz.

Ezen kicsinyítési eljárás jó eredményt ad egyedi magasságpontok esetén (52. ábra). Szintvonalrajznál azonban alapvető hibája, hogy többnyire egy pixelnél vastagabb szintvonalak keletkeznek az R mátrixban, ami teraszhatást eredményez az iteráció során (folyamatos lejtő helyett a szintvonal mentén enyhébb lejtésű sáv keletkezik, 53. ábra). A teraszhatást jól mutatja a domborzatmátrix hisztogramja is (54. ábra).

A hibát nem oldhatjuk meg a redukált szintvonal algoritmikus vékonyításával, mivel redukált mátrixon a szomszédos szintvonalak összeérhetnek (sőt egybeolvadhatnak, ha egy q x q méretű négyzetbe több szintvonal esett).

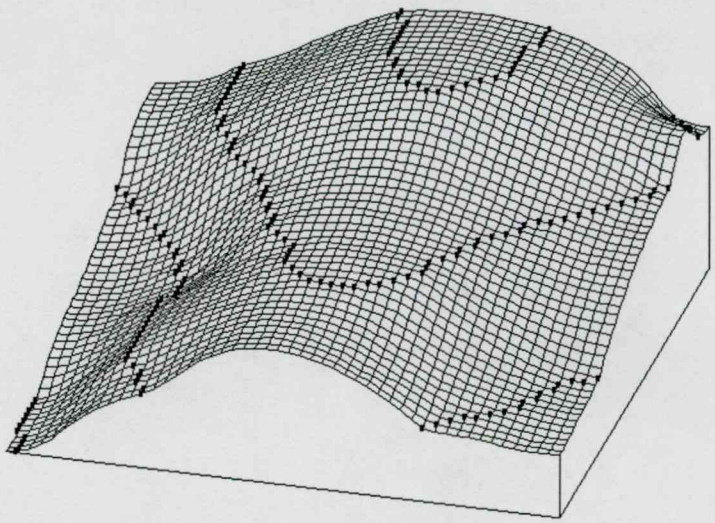


53. ábra. Egyszerű redukciós eljárással nyert DTM. A teraszhatás érzékelhető

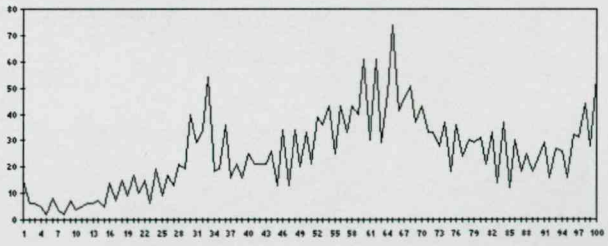


54. ábra. Hisztogram a 53. ábrán látható tereprészletről. A szintvonalmagasságoknál fellépő csúcsok jelzik a teraszhatást

Küszöböléses redukció. A fenti eljárást annyiban módosítjuk, hogy a redukált mátrix egy pontját csak akkor tekintjük adott pontnak, ha a megfelelő q x q méretű négyzetben a szintvonalpontok száma meghalad egy adott K küszöböt. A küszöb értéke redukciós fokozatonként változik. A küszöbölés eredményeként a redukált mátrixban a szintvonalak elvékonyodnak, és a teraszhatás minimálisra csökken (55. ábra). Ha azonban a szintvonalrajz egyedi magasságpontot is tartalmaz, az eredmény csúnya torzulással jelenik meg (57. ábra). Mindebből az következik, hogy az egyedi magasságpontok más algoritmikus kezelést igényelnek, mint a szintvonalpontok.



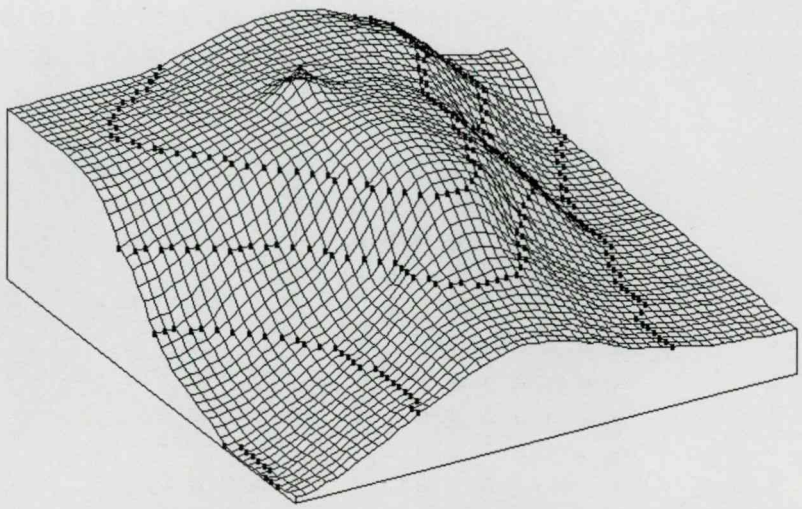
55. ábra. Küszöböléses redukcióval nyert DTM. A teraszhatás jelentősen csökken



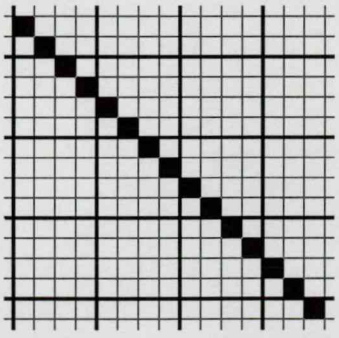
56. ábra. Hisztogram a 55. ábrán látható tereprészletről. A teraszhatás csökken

A küszöböléses redukció további hátránya, hogy a redukált mátrixban a szintvonalak helyenként megszakadnak. Egy-két pixelnyi szakadás még nem okoz gondot a relaxáció során, de ha olyan nagy küszöböt választunk, amely a szintvonalak vékonyságát garantálja, akkor szintvonalak akár hosszabb szakaszon is eltűnhetnek a redukciónál, ami már a terep torzulásához vezethet. Például a 58. ábrán látható esetben ha K=1, akkor 2 pixel vastag szintvonal keletkezik, ha viszont K=2, akkor a szintvonal eltűnik.

A fentiek arra utalnak, hogy a küszöböléses redukciót erősen finomítani kell a jó minőségű terepmodell előállítása érdekében. Az alábbiakban leírjuk azt az eljárást, amely tapasztalataink szerint a legjobb eredményt adja szintvonalak és egyedi magasságpontok együttes alkalmazása esetén is.



57. ábra. Torzulás a magasságpontnál küszöböléses redukció esetén



58. ábra. Kritikus szintvonalszakasz 4-es redukció esetén. A vastag vonalak a redukciós négyzetek határát jelölik

11.3.3. Szintvonalritkító redukció

Az eredeti Z mátrixból egy q-szorosan kicsinyített R mátrixot kívánunk előállítani. Ez azt jelenti, hogy Z-t $q \times q$ méretű négyzetekre osztjuk, és minden négyzetből egy pixel keletkezik R-ben. Az eljárás lépései a következők:

1. Övezetképzés: $Z \rightarrow Z'$. A redukált pixelek értékének pontosabb meghatározása érdekében Z-ből egy Z' mátrixot állítunk elő $q/2$ pixeles övezetképzéssel a szintvonalak és magasságpontok körül, 4-szomszédság szerint (59. ábra). Ez lineáris időben megoldható:

1.1. A 10.2.5. fejezetben ismertetett kezdőértékadó algoritmussal létrehozzuk a Z' kezdőérték-mátrixot és a D távolság fedvényt.

1.2. A Z' mátrixot küszöböljük D szerint:

```
if (d_i,j > q/2) then  z_i,j' = definiálatlan
```

Az övezetképzést nem kell minden egyes redukciónál elvégezni, hanem csak egyszer, a teljes multigrid eljárás kezdetén. Ekkor a kezdőértékadó algoritmussal létrehozunk egy Z_d kezdőérték-mátrixot és egy D távolság-fedvényt. Minden egyes redukciónál evvel a Z_d-vel és D-vel dolgozunk: a q/2-küszöbölés redukció közben végezhető, a Z' mátrix ténylegesen nem jön létre (51. ábra).

```
. . . . . . . b . . . . . . . c c
. . . . . . b b b . . . . . c c C
a a . . . b b B b b . . . c c C c
a a a . . . b b b . . . . c c C c
A A a a . . . b . . . . c c C c c
a a A a a . . . . . c c C c c . .
. a a A a a . . . . . c c C c c . .
. . a a A a a . . . c c C c c . . .
. . a a A a a . . . c c C c c . . .
. . a a A a a . . . c c C c c . . .
. . . a a A a a . . c c C c c . . .
. . . . a a A a a c c C c c . . . .
. . . . a a A a c c C c c . . . . .
. . . . a a A a c C c c . . . . . .
. . . . a a A a c C c c . . . . . .
. . . a a A a a c c C c c . . . . .
. . a a A a a . c c C c c . . . . .
```

59. ábra. Övezetképzés 2 lépésben

2. Küszöbértékek meghatározása ($0 < k_1 < k_2 \leq 1$):

– $k_1 = (q+1)/2q$. Ez a *minimális metszési arányszámot* adja: ha egy q x q négyzet egy átlós szintvonalnak csak egy pixelét tartalmazza (a sarokban), akkor az övezetképzés után a négyzet $q(q+1)/2$ értékes pixelt fog tartalmazni. Ezt leosztva a négyzet területével (q^2) kapjuk k_1-et.

– $k_2 = (3q+2)/4q$. Ez a *maximális metszési arányszámot* adja: ha egy átlós szintvonal egy q x q négyzetnek éppen az átlója mentén halad keresztül, akkor az övezetképzés után a négyzet $q(3q+2)/4$ értékes pixelt fog tartalmazni. Ezt leosztva a négyzet területével (q^2) kapjuk k_2-et.

3. Kezdeti redukált mátrix előállítása: $Z' \rightarrow S$. Amint a 11.2.2. alfejezetben írtuk, a Z szintvonalmátrix elemei magasságértékből és jelzőkódból állnak, az S mátrix is ilyen felépítésű lesz.

3.1. Jelzőkód meghatározása S-ben. Legyen n egy q x q méretű négyzetbe eső definiált pontok darabszáma, és j a meghatározandó jelzőkód.

– ha $n/q^2 < k_1$, akkor $j := 00$,

– ha $k_1 \le n/q^2 < k_2$, akkor $j := 01$,
– ha $k_2 \le n/q^2$, akkor $j := 10$,
– ha a q x q méretű négyzet magassági pontot tartalmaz, akkor $j := 11$.

3.2. Magasságértékek meghatározása S-ben. A q x q méretű négyzetbe eső értékes pontok átlagát vesszük. Ha nincs értékes pont, a megfelelő S-beli elem definiálatlan lesz.

Megjegyzendő, hogy amennyiben a Z mátrix mérete nem osztható q-val, úgy a széleken csonka (q x q-nál kisebb) négyzetek keletkeznek, ezekre a fentiek értelemszerűen módosítandók.

4. Szintvonalpontok ritkítása: $S \rightarrow R$. Az eljárás célja: a fentiekben beállított nemnulla jelzőkódok számának csökkentése úgy, hogy ne legyenek egy pixelnél vastagabb szintvonalak, ugyanakkor szintvonal ne szakadjon meg két pixelnél hosszabb szakaszon. Ezen "ritkítás" során ügyelni kell arra, hogy a fontosabb szintvonalpontok maradjanak meg, és a kevésbé fontosak törlődjenek. Az eljárás:

4.1. R kezdőértéke úgy képezendő S-ből, hogy a magasságértékeket átemeljük, de csak az "11" jelzőkódokat tartjuk meg (egyedi magasságpontok), a többit kinullázzuk.

4.2. Visszatörlés. S-ben törlünk minden olyan jelzőkódot, amely R-ben nemnulla jelzőkóddal szomszédos. Vagyis, ha $r_{i,j}$ és $s_{i,j}$ jelöli az R illetve S-beli jelzőkódokat:

```
if (r_{i-1,j} > 0 or r_{i+1,j} > 0 or r_{i,j-1} > 0 or r_{i,j+1} > 0) then s_{i,j} := 0
```

4.3. "10" jelzőkódok átírása S-ből R-be, páros pozíciókon. Vagyis, ha egy sakktáblát képzelünk az S mátrixra, csak a fekete kockákra eső "10" jelzőkódokat írjuk át:

```
if (s_{i,j} = 10 and i+j páros) then r_{i,j} := s_{i,j}
```

4.4. Visszatörlés (lásd 4.2. pont).

4.5. "10" jelzőkódok átírása S-ből R-be, páratlan pozíciókon. Vagyis, ha egy sakktáblát képzelünk az S mátrixra, csak a fehér kockákra eső "10" jelzőkódokat írjuk át:

```
if (s_{i,j} = 10 and i+j páratlan) then r_{i,j} := s_{i,j}
```

4.6. Visszatörlés (lásd 4.2. pont).

4.7. "01" jelzőkódok átírása S-ből R-be, páros pozíciókon:

```
if (s_{i,j} = 01 and i+j páros) then r_{i,j} := s_{i,j}
```

4.8. Visszatörlés (lásd 4.2. pont).

4.9. "01" jelzőkódok átírása S-ből R-be, páratlan pozíciókon:

```
if (s_{i,j} = 01 and i+j páratlan) then r_{i,j} := s_{i,j}
```

Alább megmutatjuk, hogy a fent ismertetett eljárás megfelel várakozásainknak, vagyis a redukált mátrixban nem hoz létre egy pixelnél vastagabb szintvonalat, és a redukált szintvonalakon csak egy vagy két pixelnyi szakadás keletkezhet. Mivel a redukció során szintvonalak összeolvadhatnak egymással vagy – kanyargó vonal esetén – önmagukkal, így állításainkat csak ún. *homogén szintvonalszakaszokra* tudjuk megfogalmazni.

Definíció. Egy vonalszakaszt *homogén*nek nevezünk, ha lánckódja csak két szomszédos iránykódot tartalmaz. Ez a definíció a digitális egyenes általánosításának tekinthető (v.ö. 2.2.4. alfejezet).

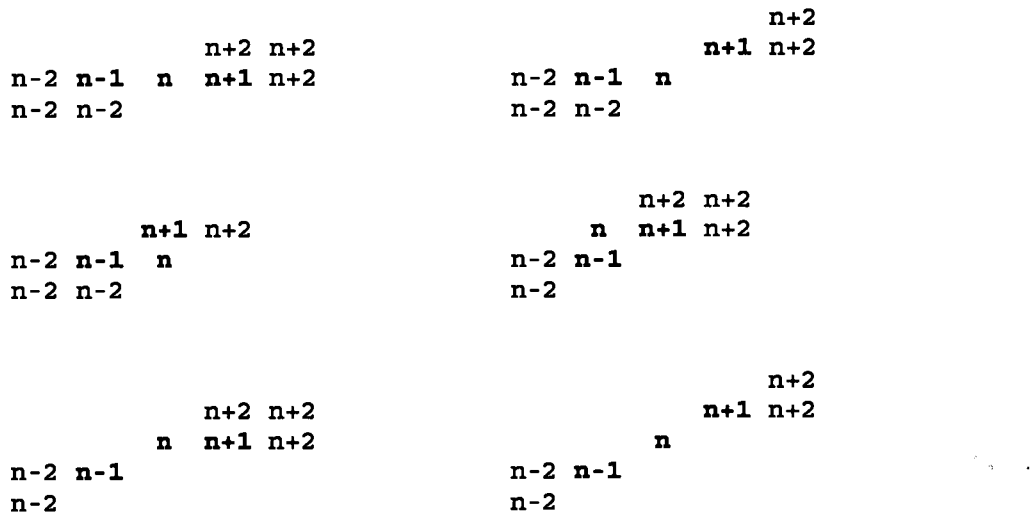
Az általánosság megszorítása nélkül a továbbiakban feltételezhetjük, hogy a vizsgált homogén vonalszakasz csak 1 és 2 iránykódot tartalmaz (ÉK és K irány).

1. Állítás. Homogén vonalszakasz redukált képe nem lehet egy pixelnél vastagabb.

Bizonyítás. Homogén vonal redukált képe legfeljebb 2 pixel vastag lehet. Ugyanakkor a szintvonal ritkítás módjából következik, hogy a redukált vonal nem tartalmazhat vízszintesen vagy függőlegesen szomszédos pixeleket. Ebből következik az állítás.

2. Állítás. Homogén vonalszakasz redukált képe legfeljebb 2 pixelnyi szakaszon szakadhat meg.

Bizonyítás. Tegyük fel, hogy 3 pixelnyi szakaszon történik szakadás, és vizsgáljuk meg a lehetséges eseteket! Ehhez sorszámozzuk be a vonal pontjait balról jobbra haladva, és sorszámozzuk be a redukált pixeleket is a leképezés sorrendjének megfelelően. Jelölje n–1, n, n+1 a feltételezett szakadás pontjainak sorszámát a redukált vonalon, ekkor a 60. ábrán látható hat eset lehetséges. Az ábrán n–2 és n+2 lehetséges pozícióit is feltüntettük. Jól látható, hogy n–2 és n+2 bárhová esik, az n sorszámú pixel négy szomszédja közül egyik sem lesz vonalpont a hat eset egyikében sem. Ekkor pedig – a szintvonalritkító algoritmus szabályai szerint – az n sorszámú pixel vonalpont kell hogy legyen, vagyis ellentmondunk a kiindulási feltételnek.



60. ábra. Lehetséges esetek a 2. állítás bizonyításához.

Műveletigény. A szintvonalritkító redukció valamennyi lépése lineáris idejű, így az egész eljárás is az.

Az eljárást a 58. ábrára alkalmazva egy pixel szélességű, szakadásmentes átlós vonalat generál a redukált mátrixon. Az, hogy a két lehetséges átlós vonal közül melyik marad meg, a véletlentől (a vonalnak a képzeletbeli sakktáblán való elhelyezkedésétől) függ.

11.3.4. Időigény

A teljes futási idő meghatározásához összegezzük valamennyi redukciós fokozat számításának időigényét! Ha az utolsó Z_n fokozatban a relaxáció időigénye t, akkor a Z_{n-1}-es redukált fokozatnál $t/4$, a Z_{n-2}-es redukált fokozatnál $t/16$, és így tovább. Innen a teljes időre a

$$t + t/4 + t/16 + ... + t/4n < 4t/3$$

becslés adódik, tehát a multigrid technikánál az utolsó redukciós fokozat időigénye a meghatározó. Ugyanez vonatkozik a tárigényre is.

Noha a multigrid módszer drasztikusan csökkenti a szükséges iterációszámot, a tárgyalt végeselemes eljárás még így is a számításigényes interpolációs technikák közé tartozik, más DTM generáló eljárásokhoz képest. A futási idő csökkentésének lehetőségeit vizsgáljuk alább.

Fixpontos aritmetika. Tapasztalataink szerint a 32-bites fixpontos számolás pontosságban és konvergencia sebességben is lényegében egyenértékű a 64-bites lebegőpontos aritmetikával. Fixpontos esetben kihasználható, hogy a 10.2.2. alfejezetben levezetett (12) vékonylemez maszk elemei egy kivétellel kettőhatványok, vagyis szorzás és osztás helyett a lényegesen gyorsabb léptetés művelet alkalmazható. Összességében *a fixpontos számolás 7-szer gyorsabb* a lebegőpontosnál!

Optimális iterációszám meghatározása. Vizsgálataink szerint 40-50 iteráció már jó eredményt ad, a relaxáció folytatása érdemben már nem javítja a terepmodell minőségét. Kérdés viszont, hogy csökkenthető-e az iterációszám? A redukált mátrixok esetében nem, mivel ezek szerepe kulcsfontosságú a terep kialakításában, ugyanakkor a számításigény csekély a kisebb mátrixméretek miatt. Az utolsó fokozatban viszont már 10-20 iteráció is jó eredményt adhat, ha a szintvonalak távolsága legalább 10 pixel.

Kérdés azonban, hogy egyáltalán szükség van-e az utolsó redukciós fokozatra, amely a szkennelési felbontásban állít elő DEM-et. A forrásadatok pontosságát figyelembe véve rendszerint elegendő az utolsó előtti (esetleg még korábbi) redukciós fokozatig elmenni, ezzel a futási idő jelentősen csökken.

Kettős multigrid eljárás. Amint a 51. ábrán látható, a Z_d és D mátrixokat n redukciós fokozat esetén $(n-1)$-szer kell beolvasni és feldolgozni. Igaz, hogy a feldolgozás lineáris idejű, de winchesteren tárolt nagy mátrixok esetén ez is lassú lehet. Ezért nagyméretű Z mátrix esetén először egy kicsinyített – például 8-as redukált – Z_x mátrixot állíthatunk elő, erre hajtunk végre egy "belső multigrid eljárást", majd az eredményül kapott Z_x' kicsinyített terepmodellt kezdőértékként használva egy "külső multigrid eljárás" adja a Z' eredménymátrixot.

Méréseink szerint 50 iterációs multigrid algoritmus időigénye (ha minden redukciós fokozatban 50-et iterálunk) fixpontos számolásnál *1000 x 1000 pixel méretű mátrixon 2.5*

*perc*nek adódott (400 MHz Celeron processzor). Az időigény a mátrix elemszámával és az iterációszámmal lineárisan arányos, így ebből könnyen számítható a futási idő más mátrix méretre és iterációszámra is.

Az iteráció lokális, párhuzamos jellege miatt hatékonyan lehet tömbprocesszorral gyorsítani a számítást (Katona 1985, Terzopoulos 1988), bár a szekvenciális gépek sebességének növekedésével ennek jelentősége egyre inkább csökken. Erre a kérdésre a 11.5. fejezetben még visszatérünk.

> Ismert tény, hogy a konvolúciós tétel (lásd 10.2.2. alfejezet) és a gyors Fourier-transzformáció (FFT) felhasználásával a konvolúció számítása gyorsítható. A lényeg: mind a kép, mind a maszk Fourier-transzformáltját képezzük, majd ezek szorzatát vissza-transzformáljuk. Ez különösen előnyös lenne relaxáció esetében, hiszen több konvolúciót kell egymás után végezni. Tekintettel azonban az illeszkedési feltételre, minden egyes konvolúció után az adott pontok értékét korrigálni kell, emiatt minden iterációs lépés oda-vissza transzformációt igényelne. Ráadásul a fent említett fixpontos számolás előnyeit is elveszítenénk, ezért esetünkben az FFT-módszertől nem remélhetünk gyorsítást.

11.4. Összefoglalás, értékelés

Digitális terepmodell előállítására egy tisztán raszteres eljárást ismertettünk, amely szkennelt szintvonalrajzból és/vagy magassági pontokból raszteres terepmodellt (DEM) állít elő.

Az eljárás első részében szintvonalmátrixot állítunk elő több kevesebb manuális munkával és gépi támogatással, amely a szintvonalak mentén és magassági pontoknál ismert magasságértéket, azokon kívül definiálatlan értékeket tartalmaz.

Az eljárás második része teljesen automatikus: vékonylemez modellen alapuló multigrid relaxációs módszerrel értéket adunk valamennyi definiálatlan pontnak, így áll elő a DEM mátrix.

Az eljárás korrekten működik csak szintvonalrajzot (55. ábra), csak magassági pontokat (52. ábra), és a kettőt vegyesen tartalmazó fedvényekre (61. ábra)), és például nyeregfelületet is jól képez le (62. ábra).

A tisztán raszteres feldolgozás *mellett* az alábbi érvek szólnak:

– A forrásadatok (szkennelt szintvonalrajz) és a generált eredmény (DEM) egyaránt raszteres, ezért természetesebb végig raszteresen dolgozni.

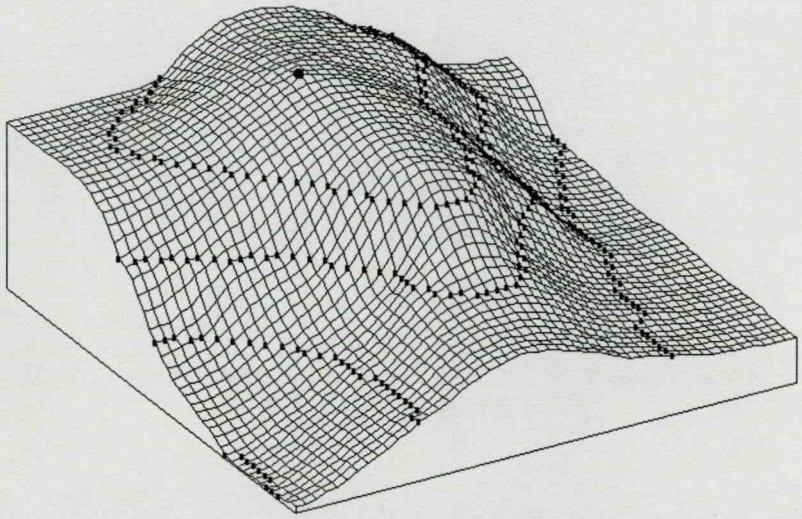
– Az eljárás a lehető legpontosabb terepmodellt állítja elő, hiszen elkerüljük a vektorizálás és raszterizálás okozta kisebb-nagyobb torzulásokat (63. ábra).

A tisztán raszteres feldolgozással *szemben* az alábbi ellenérveket szokták felhozni:

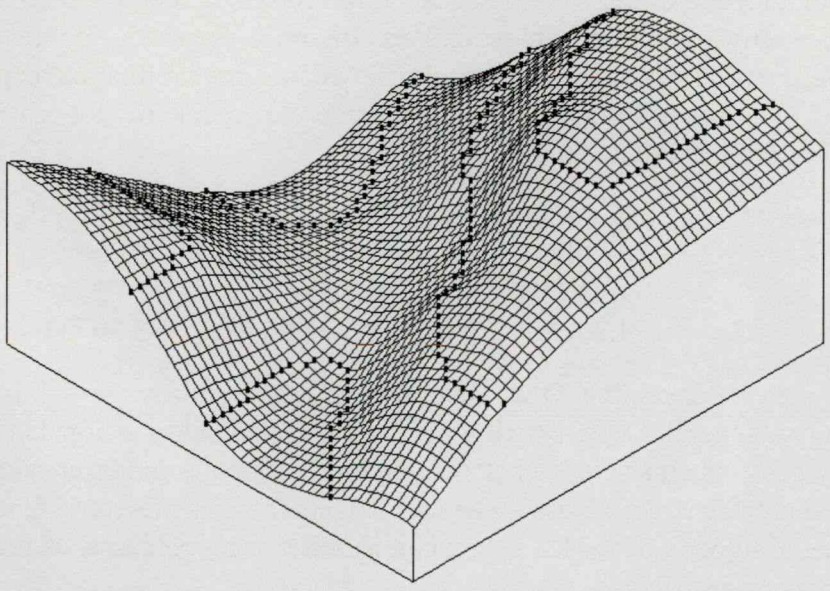
1. A vektoros adatok hatékonyabban kezelhetők.
2. A raszteres feldolgozás hosszú futási időt és nagy tárkapacitást igényel.

Ami az 1. ellenérvet illeti, a széles körben elterjedt vektoros szoftverek sokrétű szolgáltatásai valóban előnyt jelentenek. Ugyanakkor a 11.1. és 11.2. fejezetekben megmutattuk, hogy a raszteres feldolgozás is számos módon támogatható, és a manuális

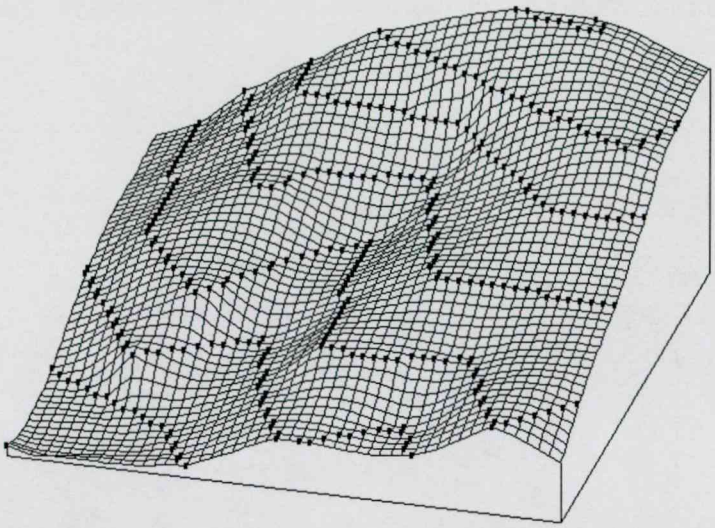
munka mennyisége nem sokkal több, mint a 8.2.1. alfejezetben vizsgált vektoros technológia esetén. Ha ehhez még figyelembe vesszük a 11.4.1. alfejezet vizsgált hibrid továbbfejlesztési lehetőséget, akkor a vektoros adatkezelés valamennyi előnye átmenthető a raszteres technológiába.



61. ábra. Az ismertetett multigrid eljárással generált terepmodell. A 57. ábrán látható torzulás megszűnik



62. ábra. A multigrid eljárás eredménye nyeregfelületen (kivágat egy 512 x 512-es DTM-ből)



63. ábra. Vektorizált (és visszaraszterizált) szintvonalrajzból generált, kissé "szögletes" DEM

A 2. ellenérvet egy példán vizsgáljuk meg. Ha egy 60 x 40 cm-es keretméretű, 1:10 000 méretarányú magyar topográfiai térképszelvény domborzati rétegét 300 dpi-vel szkenneljük, akkor 7086 x 4725 pixeles fedvényt kell feldolgozni. Ennek mérete, egy pixelt 16 biten tárolva, 67 MB. A feldolgozási idő – a 11.3.4. alfejezetben szereplő időadat alapján – 88 percnek adódik, a generált DEM felbontása – a földfelszínen mérve – 0.85 méter.

Ez a fedvény nyilvánvalóan redundáns, hiszen a forrásadat pontossága nem éri el a szkennelés felbontását. Ilyen esetben a multigrid eljárást megállíthatjuk az utolsó előtti (2 x 2-es rácsméretű) vagy még korábbi (4 x 4-es rácsméretű) redukciós fokozat kigenerálása után, ekkor az alábbi értékek adódnak:

Multigrid fokozat	*Helyigény*	*Futási idő*	*DEM felbontás*
1 x 1	67 MB	88 perc	0.85 m
2 x 2	16.8 MB	22 perc	1.69 m
4 x 4	4.2 MB	5.5 perc	3.39 m

Egész méterre kerekített DEM felbontás előállításához vagy a kiindulási (szkennelt) fedvényt, vagy a generált DEM-et kell megfelelő affin transzformációnak alávetni.

A helyigény és futási idő tehát nem jelent hátrányt a módszer alkalmazásánál. Igaz, hogy a szintvonalmátrix előállítása során mindenképp a teljes méretű (a fenti példánál 67 MB) mátrixszal kell dolgozni, de a 11.2.1. pontban ismertetett megoldással ez mérettől függetlenül, real-time módon megoldható.

11.4.1. Továbbfejlesztési lehetőségek

1. Törésvonalak és szakadásvonalak kezelése. Ennek elméleti alapjait a 10. fejezet elején tárgyaltuk, implementációjához még számos részletkérdést meg kell oldani. Megjegyzendő, hogy a jelenlegi rendszerben szorosan egymás mellé rajzolt, kiegészítő szintvonalakkal helyettesíthető a törésvonalak és szakadásvonalak megadása.

2. Vízlefolyás modellezés. A kérdést Hutchinson (1989, 1996) vizsgálja, módszereinek felhasználásával az eróziós hatásokat is figyelembe vevő, a valóságnak jobban megfelelő terepmodell hozható létre.

3. Hibrid raszter-vektor technológia. Az alapgondolat: a vékonyított szintvonalrajzot vektorizáljuk úgy, hogy a raszterképet a vektorokkal egy speciális adatstruktúrában összekapcsoljuk (hasonlóan, mint a 3.2. fejezetben bemutatott képgráfnál). Így a vektoros interpretáció (8.2.1. alfejezet) és a raszteres feldolgozás (11. fejezet) együtt, egymást kiegészítve alkalmazható.

11.5. Alkalmazás: a DDM-10 projekt

A 11. fejezetben ismertetett módszer – pontosabban, annak egy korábbi változata – valós gyakorlati alkalmazásra került 1991-92-ben, a *Magyar Honvédség Tóth Ágoston Térképészeti Intézete* (MH-TÁTI) által vezetett *DDM-10 projekt*ben (Katona 1993, Szabó B. 1994), amelynek során Magyarország egész területét lefedő, akkor legnagyobb felbontású domborzatmodelljét állították elő.

A projekthez a technológia kidolgozását és a programozási munkákat a *Cellware Kft* végezte, jelen dolgozat szerzőjének szakmai irányításával (Katona, 1995). A technológiát a megrendelő MH-TÁTI kis mértékben kiegészítette és módosította (Bakó, 1994). Az elkészült adatállomány pontossági vizsgálatait a Geomatic Kft végezte. A munka első megrendelője és anyagi támogatója a Frekvenciagazdálkodási Intézet volt.

Adatforrásként 1:50 000 méretarányú, 1985-91 évi kiadású katonai topográfiai térképek szintvonalas domborzati fóliái álltak rendelkezésre Grauss-Krüger vetületi rendszerben, amelyek 300 dpi felbontással kerültek szkennelésre.

Az akkori számítástechnikai eszközök színvonala miatt számos problémával kellett megküzdeni:

– Egyetlen szelvény feldolgozási ideje mintegy 100 óra (!) lett volna (386-os processzor, 33 MHz), ezért a Cellware Kft. által kifejlesztett gyorsítóprocesszor (úgynevezett sejtprocesszor, lásd például Legendi és tsai (1988), Katona (1992)) alkalmazására került sor. A konvolúció-típusú műveletek jól párhuzamosíthatók, így sejtprocesszorral hatékonyan gyorsíthatók (Katona, 1985), ennek köszönhetően egy szelvény feldolgozási idejét 7 órára sikerült csökkenteni.

– Mivel egy szelvény szintvonalmátrixa 50 MB tárolóhelyet igényelt, az akkor elérhető 80 MB-os winchesteren nem lehetett két példányban tárolni, ezért a szintvonalmátrixot sorkötegekre tördelve kellett feldolgozni.

A generált szelvények átfedésmentesek voltak, így azok csatlakoztatására a következő módszert alkalmazták. Két szomszédos szintvonalas szelvény széléből egy keskeny (10 cm széles) átfedő szelvényt készítettek, amelyre újra kigenerálták a DEM-et. A tapasztalat azt mutatta, hogy ha az átfedő szelvényt a két eredetire illesztjük, csak a szelvényhatár legfeljebb 2 cm-es sávjában van eltérés. Ebben a sávban az eredeti szelvényből generált DEM-et lecserélték az átfedő szelvényből generált értékekre, így teljesen folytonos domborzatmodellt kaptak.

A projekt eredményeként elkészült terepmodellt az MH-TÁTI az alábbiak szerint ismerteti a Térinformatika 1996/5 számában (27.oldal):

DDM-50, DDM-10: Az ország területére tartalmazza a felszín tengerszint feletti magasságát (Balti alapszint) egy 50 x 50 ill. 10 x 10 méteres rács pontjaiban. Az adatállomány EOV vetületi rendszerű raszter-adatstruktúrában, 84 db 1:100 000 méretarányú EOTR szelvényre bontva áll rendelkezésre, de lehetőség van a fent említettnél ritkább rácssűrűség leválogatására is. Hozzáférhető Grauss-Krüger rendszerben is. A teljes állomány mérete 10 x 10 méteres ráccsal 2.5 GB, 50x50 méteres ráccsal 100 MB.

A domborzatmodell pontossága felszín típusonként:

Síkvidék (átlagos tereplejtés)	< 2%,	középhiba ± 0.8 m
Dombvidék (átlagos tereplejtés)	2-6%,	középhiba ± 2.5 m
Hegyvidék (átlagos tereplejtés)	> 6%,	középhiba ± 5.0 m

Végül megjegyezzük, hogy a DDM-10 projektben nem a 10.2.2. alfejezetben levezetett (12) iterációs maszkot, hanem az

$$E_f^2 = \iint \left(f_{xx}^2 + f_{yy}^2\right) dx\, dy$$

energiafüggvényből levezethető, gyorsabb számolást lehetővé tevő alábbi maszkot használtuk:

$$\frac{1}{16}\begin{bmatrix} & & -1 & & \\ & & 4 & & \\ -1 & 4 & 4 & 4 & -1 \\ & & 4 & & \\ & & -1 & & \end{bmatrix}$$

A generált terepmodell gyakorlatilag nem különbözik a (12) maszkkal generálttól, amiből azt az érdekes következtetést lehet levonni, hogy az eljárás többi eleme (multigrid technika, szintvonalritkító redukció) sokkal erőteljesebben befolyásolja az eredményt, mint maga a konvolúciós maszk.

12. Eredmények összefoglalása

Az alábbiakban röviden, tézisszerűen összefoglaljuk és értékeljük eredményeinket.

1. Raszter-vektor konverziós algoritmus (2.2. fejezet).

Összehasonlítva a 2.1. fejezetben áttekintett vektorizáló eljárásokkal megállapíthatjuk, hogy saját eljárásunk a vékonyításon alapuló megoldások csoportjába tartozik, amelyben a digitális egyenes fogalmára épülő, lineáris idejű, automatával való vonalkövetés módja jelent újdonságot.

Az eljárást leprogramoztuk és sikerrel alkalmaztuk a MAPINT rendszerben. Futási ideje a 6.5.2. alfejezetben található.

2. A DG adatmodell (5. fejezet).

A különféle adatmodelleket a 4. fejezetben vizsgáltuk, és megállapítottuk, hogy a topológikus modellek a legalkalmasabbak térkép-interpretáció céljára.

Saját interpretáló rendszerünkhöz egy új, DG-nek nevezett adatmodellt dolgoztunk ki. Ez egy gráf alapú topológikus modell, amely mindössze 4 adattípussal komplex, hierarchikus struktúrák leírására is alkalmas.

A fizikai adatmodell (5.2. fejezet) alapján megmutattuk, hogy N vektor esetén a kezdeti topológia létrehozása $O(N \cdot \log(N))$ időt, az adatstruktúra tömörítése $O(N)$ időt, az aktualizálási műveletek pedig konstans időt igényelnek, ami összességében azt jelenti, hogy az adatmodell hatékonyan kezelhető.

A keresések gyorsítására az adatmodellt térbeli indexeléssel egészítettük ki (grid index, lásd 4.6.2. és 5.5. alfejezetek). Ennek köszönhetően gyakorlatilag minden felismerési művelet lineáris időben végezhető (6.4. fejezet).

Az adatmodellt kezelő algoritmusokat leprogramoztuk és beépítettük a MAPINT rendszerbe.

3. Magyar kataszteri térképek interpretációja a MAPINT rendszerrel (6. fejezet).

Saját fejlesztésű rendszerünk a DG adatmodellre épül. Segítségével a szkennelt, vektorizált térképszelvényen a kataszteri térképek legfontosabb és legnagyobb számban előforduló objektumai ismerhetők fel (szaggatott vonalak, házszámok, helyrajzi számok, kapcsolójelek, nullkörök, épületek és földrészletek).

Összevetve a MAPINT-et más, a 3. fejezetben bemutatott interpretáló rendszerekkel az alábbiakat állapíthatjuk meg:

– A MAPINT *hasonlít* más rendszerekhez a nyers vektorizálással induló feldolgozásban, a szimbólumok elkülönítésének módjában és egyes gráf alapú algoritmusokban.

– A MAPINT *eltér* más rendszerektől a koordináta-transzformáció megoldásában (6.3. fejezet), a DG adatmodell használatában, a magyar kataszteri térképekre vonatkozó specialitásokban és neurális hálóval történő vektoros karakterfelismerésben.

A Phare Land Consolidation projekt keretében (7. fejezet) a MAPINT beépült a földhivatali TAKAROS rendszer adatfeltöltési technológiájába.

4. Szintvonalrajz interpretációja MAPINT-tel (8.2.1. alfejezet).

Megmutattuk, hogy a DG adatmodell és a MAPINT alkalmas szintvonalrajzok interpretációjára: felismerhetők a fő- és felező (negyedelő) szintvonalak, magassági megírások és eséstüskék, (fél)automatiksan javíthatók a szakadások és összeérések.

5. DEM terepmodell előállítása tisztán raszteres technológiával (11. fejezet).

A szintvonalak vektorizálásra helyett mindvégig raszteres műveletek során jutunk el a szkennelt térképtől a terepmodellig. Rámutattunk, az eljárás jobb minőségű terepmodellt állít elő, mint a vektorizálásra épülő módszer (63. ábra). Megmutattuk, hogy a raszteres eljárással szemben felhozott ellenérvek inkább gyakorlati, mint elvi jellegűek (11.4. fejezet), ugyanis a feldolgozás kritikus lépései (topológia ellenőrzés, szintvonal értékadás) raszteresen is hatákonyan végezhetők.

Eljárásunk gyakorlati alkalmazásra került a Magyar Honvédség DDM-10 projektjében (11.5. fejezet).

6. Multigrid relaxáció szintvonalritkító redukcióval (11.3. fejezet).

A 9.1. és 9.2. alfejezetekben áttekintettük a fontosabb terepmodell előállító eljárásokat. Megmutattuk, hogy a vékonylemez modellre épülő végeselemes multigrid cljárásoktól várható a legjobb minőségű terepmodell. A 10. fejezetben részletesen ismertettük a – más rendszerek által is használt – vékonylemez modellre épülő végeselemes terepmodellezés elméleti alapjait.

Erre építve dolgoztuk ki saját szintvonalritkító redukciós eljárásunkat (11.3. fejezet), amely az előbb említett tisztán raszteres technológiába épülve magassági pontokat és/vagy szintvonalakat tartalmazó fedvényekből jó minőségű terepmodellt állít elő.

Irodalom

Aho A. V., Hopcroft J. E., Ullman J. D. (1982): Számítógép-algoritmusok tervezése és analízise. Műszaki Könyvkiadó, Budapest.

Álló G., Hegedűs Gy. Cs., Kelemen D., Szabó J. (1989): A digitális képfeldolgozás alapproblémái. Akadémiai Kiadó, Budapest.

Atwood T., Duhl J., Ferran G., Loomis M., Wade D.: The Object Database Standard ODMG-93. Morgan Kaufmann Publishers, San Francisco, California, 1994.

Bakó Z. (1994): Digitális térképészeti adatbázis fejlesztések a Magyar Honvédség Kartográfiai Üzemében. Térinformatika Magyarországon. NCGIA Core Curriculum melléklete, szerk.: Márkus Béla, EFE FFFK, Székesfehérvár.

Bentley (1998): MicroStation GeoGraphics Version 5.7, User's Guide. Bentley Systems, Inc.

Boatto L., Consorti V., Buono M., Zenzo S., Eramo V., Esposito A., Melcarne F., Meucci M., Morelli A., Mosciatti M., Scarci S., Tucci M. (1992): An Interpretation System for Land Register Maps. Computer, Vol. 25, No. 7, pp. 25-33.

Bookstein F. L. (1989): Principal Warps: Thin-Plate Splines and the Decomposition of Deformations. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 11, No. 6, pp. 567-584.

Borgefors G. (1984): Distance Transformations in Arbitrary Dimensions. Computer Vision, Graphics and Image Processing, Vol. 27, pp. 321-345.

Brand K. (1982): Multigrid bibliography. In: Multigrid Methods, eds: W. Hackbusch and U. Trottenberg, Lecture Notes in Mathematics Vol. 960., Springer-Verlag, pp. 631-650.

Carrara A., Bitelli G., Carla R. (1997): Comparison of techniques for generation digital terrain models from contour lines. Int. Journal on Geogr. Inform. Science, Vol. 11, No. 5, 451-473.

Chai J., Miyoshi T., Nakamae E. (1998): Contour Interpolation and Surface Reconstruction of Smooth Terrain Models. Proc. of conf. on Visualization '98, ACM, pp. 27-33.

Chen L.-H., Liao H.-Y., Wang J.-Y., Fan K.-C., Hsieh C.-C. (1996): An Interpretation System for Cadastral Maps. Proc. of 13th Internat. Conf. on Pattern Recognition, IEEE Press, Los Alamitos, California, pp. 711-715.

Chen Y., Langrana N. A., Das A. K. (1996): Perfecting Vectorized Mechanical Drawings. Computer Vision and Image Understanding, Vol. 63, No. 2, pp. 273-286.

DAT (1996): Digitális térképek. MSZ-7772-1 szabvány, Magyar Szabványügyi Testület.

DAT1 (1996): DAT1 szabályzat digitális alaptérképek előállítására. Földművelésügyi Minisztérium, Földügyi és Térképészeti Főosztály, Budapest.

Detrekői Á., Szabó Gy. (1995): Bevezetés a térinformatikába. Nemzeti Tankönyvkiadó, Budapest.

Di Zenzo S., Morelli A. (1989): A useful image representation. Proc. of 5th Internat. Conf. on Image Analysis and Processing, World Scientific Publishing, Singapore, pp. 170-178.

Dori D. (1997): Orthogonal Zig-Zag: an Algorithm for Vectorizing Engineering Drawings Compared with Hough Transform. Advances in Engineering Software 28 (1), pp. 11-24.

Douglas D. H., Peucker T. K. (1973): Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Can. Cartographer Vol. 10, No. 2, pp. 112-122.

Dupont F., Gondran M. (1999): DTM Extraction from Topographic Maps. Internat. Conf. on Document Analysis and Recognition, IEEE Press, Los Alamitos, California, pp. 475-478.

Eastman J. R. (1992): IDRISI Version 4.0, Technical Reference. The Clark Labs for Cartographic Technology and Geographic Analysis (Worcester, Mass.: Clark University, Graduate Scool of Geography)

Ebi N. B. (1995): Image Interpretation of Topographic Maps on a Medium Scale Via Frame-based modelling. International Conference on Image Processing, IEEE Press, California, Vol. I., pp. 250-253.

Eklundh L., Martensson U. (1995): Rapid generation of Digital Elevation Models from topographic maps. Int. Journal on Geogr. Inform. Systems, Vol. 9, No. 3, 329-340.

ESRI (1994a): Arc/Info Data Management. Environmental System Research Institute, Inc.

ESRI (1994b): Arc/Info Version 7 - Arc Commands. Environmental System Research Institute, Inc.

ESRI (1998): Spatial Database Engine. A white paper, Environmental System Research Institute, 18 pages, http://www.esri.com

F7 (1983): F7 szabályzat az egységes országos térképrendszer földmérési alaptérképeinek készítésére. MÉM Országos Földügyi és Térképészeti Hivatal, Földmérési és Térképészeti Főosztály, Budapest.

Falk Gy., Voloncs Gy. (1992): Térkép-digitalizálás. ComputerWorld-Számítástechnika, 1992/30, pp. 15.

Freeman H. (1974): Computer processing of line-drawing images. Computer Survey 6, pp. 57-97.

Gonzalez R. C., Wintz P. (1987): Digital Image Processing. Addison-Wesley, Reading, Mass.

GRASS (1996): User manual, Version 4.1. US Army Corps of Engineers, CERL, Champaign, Illinois, USA.

Grimson W. E. L. (1983): An Implementation of a Computational Theory of Visual Surface Interpolation. Computer Vision, Graphics and Image Processing, Vol. 22, pp. 39-69.

Hori O., Shimotsuji S., Hoshino F., Ishii T. (1992): Probabilistic relaxation methods for line-drawing interpretation. Proc. of 11th IAPR Internat. Conf. on Pattern Recognition, IEEE Press, Vol. 2, pp. 158-161.

Hudra Gy. (1995): Digitális térképek előállítása automatikus interpretációval. Vizsgadolgozat, JATE, Szeged.

Hudra Gy., Kern H. (1999): Automatische Erkennung von linienhaften Strukturen auf gescannten Karten. Horizonte, Fachhochschule Mannheim, Nr. 15, pp. 68.

Hudra Gy. (2000): Interpretation of Hungarian Cadastral Maps. Karlsruher Geowissenschaftliche Schriften, Fachhochschule Karlsruhe, 31 pages (megjelenés alatt).

Hutchinson M. F. (1989): A new procedure for gridding elevation and stream-line data with automatic removal of spurious pits. Journal of Hydrology, 106, pp. 211-232.

Hutchinson M. F. (1996): A locally adaptive approach to the interpolation of digital elevation models. Third International Conference on Integrating GIS and Environmental Modeling, NCGIA, University of California, Santa Barbara.

Janssen R. D. T., Vossepoel A. M. (1997): Adaptive Vectorization of Line Drawing Images. Computer Vision and Image Understanding, Vol. 65, No. 1, pp. 38-56.

Katona E. (1985): Cellular Algorithms for Image Convolution. Proceedings of the Conference "Automatische Bildverarbeitung", Berlin, Oct., Kammer der Technik of GDR, pp. C3/1-C3/4

Katona E. (1992): Cellular Processing. In: Fuzzy, Holographic and Parallel Intelligence. By Branco Soucek and the IRIS Group, John Wiley & Sons, Inc., New York, pp.215-230.

Katona E., Podolcsák Á. (1992): Az automatikus térképdigitalizálás lehetőségei. Geodézia és Kartográfia, 1992/6, pp. 424-427.

Katona E. (1993): Gyorsítóprocesszorok a digitális térképészetben. *Térinformatika* 1993/2, pp. 8-9.

Katona E., Palágyi K., Tóth N. (1995): Signature verification using neural nets. *Proceedings of 9th Scandinavian Conference on Image Analysis*, pp. 1115-1122.

Katona E. (1995): Digitális terepmodell számítása multigrid relaxációs eljárással. *Geodézia és Kartográfia* 1995/5, pp. 20-25.

Katona E., Hudra Gy. (1999a): An Interpretation System for Cadastral Maps. Proceedings of 10th International Conference on Image Analysis and Processing (ICIAP 99), IEEE Press, pp. 792-797.

Katona E., Hudra Gy. (1999b): Kataszteri adatfeltöltés automatikus térkép felismeréssel. *Térinformatika*, 1999/8, pp. 12-13.

Kertész Á. (1972): Matematikai-statisztikai módszerek alkalmazási lehetőségei a geomorfológiában. 1972/4, pp. 487-502.

Kern H. (1996): OGS-Referenzhandbuch. (A RoSy rendszer OGS nyelvének referencia kézikönyve.)

Kern H., Mezősi G., Garay G. (1997): Korszerű eszköz segíti a térképek automatikus vektorizálását. Térinformatika, 1997/4, 23. old.

Klauer R. (1993): Untersuchungen zur Optimierung von Verfahren der automatisierten Digitialisierung von Flurkarten. Diplomarbeit, Hannover.

Kollányi L., Prajczer T. (1995): Térinformatika a gyakorlatban. GeoGroup Bt, Budapest.

Kraus K. (1998): Fotogrammetria. Alapok és általános módszerek. Tertia Kiadó, Budapest.

Lam L., Lee S. W., Suen C. Y. (1992): Thinning methodologies - a comprehensive survey. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 14, No. 9, pp. 869-887.

Legendi T., Katona E., Tóth J., Zsótér A. (1988): Megacell Machine. Proc. of the Conference "VAPP III.", Liverpool, Aug., 1987, Parallel Computing 8, pp. 195-199.

Lerner J. (1989): Térképészeti alapismeretek. ELTE jegyzet.

Li L., Nagy G., Samal A., Seth Sh., Xu Y. (1999): Cooperative Text and Line-art Extraction from a Topographic Map. Internat. Conf. on Document Analysis and Recognition, IEEE Press, Los Alamitos, California, pp. 467-470.

Lin X., Shimotsuji S., Minoh M., Sakai T. (1985): Efficient diagram understanding with characteristic pattern detection. Computer Vision, Graphics and Image Processing, Vol. 30, pp. 84-106.

Lladós J., Martí E., Lopez-Krahe J. (1999): A Hough-based method for hatched pattern detection in maps and diagrams. Internat. Conf. on Document Analysis and Recognition, IEEE Press, Los Alamitos, California, pp. 479-482.

Maderlechner G., Mayer H. (1994): Automatic Acquisition of Geometric Information from Scanned Maps for GIS using Frames and Semantic Networks. Proc. of 12th IAPR Internat. Conf. on Pattern Recognition, IEEE Press, Los Alamitos, California, Vol. 2, pp. 361-363.

Maling D. H. (1991): Coordinate Systems and Map Projections for GIS. In: D. J. Maguire, M. F. Goodchild, D. W. Rhind (eds): Geographic Information Systems. Principles and Applications, John Wiley and Sons, pp. 135-146.

Márkus B., Mihály Sz. (1999): FISH – Földügyi információs szolgáltatások. Geodézia és Kartográfia, 1999/6, pp. 14-20.

Márkus B. (1994) (szerk.): NCGIA Core Curriculum, magyar fordítás (az eredeti kiadás szerkesztői M. F. Goodchild és K. K. Kemp). EFE FFFK, Székesfehérvár.

Mezősi G. (1991) (szerk.): A mikroszámítógépes módszerek használata a természet-földrajzban. Egyetemi jegyzet, József Attila Tudományegyetem, Szeged.

Moore L. R. (1992): Software for cartographic raster-to-vector conversion. International archives of photogrammetry and remote sensing. Volume XXIX. Part B4. Comission IV. Washington, D.C.

MOSS (1996): RoSy-GEO Rasterorientietres System. (Kézikönyv a RoSy rendszer Hedi szerkesztőjének használatához.) M.O.S.S. Ltd.

MOSS (1998): EASI - Erweiterbarer Applikations-Sprach-Interpreter. (A RoSy rendszer EASI nyelvének referencia kézikönyve.) M.O.S.S. Ltd.

Musavi M. T., Shirvaikar M. V., Ramanathan E., Nekovei A. R. (1988): A Vision Based Method to Automate Map Processing. Pattern Recognition, Vol. 21, No. 4, pp. 319-326.

Nagasamy V., Langrana N. A. (1990): Engineering Drawing Processing and Vectorization System. Computer Vision, Graphics and Image Processing, Vol. 49, pp. 379-397.

Niemann H., Sagerer G. F., Schröder S., Kummert F. (1990): ERNEST: A Semantic Network System for Pattern Understanding. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 12, No. 9, pp. 883-905.

Omaszta S. (1998): Eljárás a termőföld privatizáció során keletkezett digitális térképi állományok betöltésére a TAKAROS rendszerbe. Tanulmány, FM Földügyi és Térképészeti Főosztály, Phare Land Consolidation Project.

Omaszta S., Szabó J. (1999): TAKAROS lehetőségek az EU csatlakozás tükrében. Geodézia és Kartográfia 1999/6 szám.

OpenGIS (1997): Simple Features Specification for SQL. OpenGIS Consortium Inc., http://www.opengis.org.

OpenGIS (1998): Simple Features Specification for OLE/COM, Revision 1.0. OpenGIS Consortium Inc., http://www.opengis.org.

Oracle (1997/a): Oracle8 Spatial Cartridge. A technical white paper, Oracle Corporation, http://www.oracle.com.

Oracle (1997/b): Oracle8 Server Concepts. Chapter 10: User-Defined Data Types (Object Option). Oracle Corporation, http://www.oracle.com.

Pratt W. K., Kabir I. (1985): Morphological Binary Image Processing with a Local Neighborhood Pipeline Processor. Frontiers in Computer Graphics (Proc. of Computer Graphics Tokyo '84), Springer-Verlag.

Phillips I. T., Chhabra A. K. (1999): Empirical Performance Evaluation of Graphics Recognition Systems. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 21, No. 9, pp. 849-870.

Ray B. K., Ray K. S. (1992): Detection of significant points and polygonal approximation of digitized curves. Pattern Recognition Letters Vol. 13, No. 6, pp. 443-452.

Rosenfeld A., Pfaltz J. (1966): Sequential Operations in Digital Picture Processing. Journal of the ACM, 13 (4) pp. 471-494.

Russell S. J., Norvig P. (2000): Mesterséges intelligencia modern megközelítésben. Panem - Prentice Hall, Budapest.

Samet H. (1989): Design and Analysis of Spatial Data Structures. Addison Wesley.

Samet H., Soffer A. (1998): MAGELLAN: Map Acquisiton of GEographic Labels by Legend ANalysis. International Journal on Document Analysis and Recognition, Vol. 1, pp. 89-101.

Sárközi F. (1999): Térinformatikai elméleti oktató anyag. http://bme-geod.agt.bme.hu/tutor_h/

Shimada S., Maruyama K., Matsumoto A., Hiraki K. (1995): Agent-based Parallel Recognition Method of Contour Lines. Internat. Conf. on Document Analysis and Recognition, IEEE Press, Los Alamitos, California, pp. 154-157.

Shimotsuji S., Hori O., Asano M., Suzuki K. (1992): A Robust Recognition System for a Drawing Superimposed on a Map. Computer, Vol. 25, No. 7, pp. 56-63.

Stefano L., Bulgarelli A. (1999): A Simple and Efficient Connected Components Labeling Algorithm. Proc. of 10th Internat. Conf. on Image Anal. and Processing, IEEE Press, pp. 322-327.

Suzuki S., Ueda N. (1991): Robust vectorization using graph-based thinning and reliability-based line approximation. IEEE Conf. on Computer Vision and Pattern Recognition, IEEE Press, Los Alamitos, California, pp. 494-500.

Suzuki S., Yamada T. (1990): MARIS: Map Recognition Input System. Pattern Recognition, Vol. 23, No. 8, pp. 919-933.

Szabó B. (1994): A katonai térképészet térképművei. Geodézia és Kartográfia, 1994/3, pp. 138-141.

Szeliski R. (1990): Fast Surface Interpolation Using Hierarchical Basis Functions. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 12, No. 6, pp. 513-528.

T3 (1981): T3 szabályzat - az egységes országos térképrendszer 1:10 000, 1:25 000 és 1:100 000 méretarányú topográfiai térképeinek jelkulcsa. MÉM Országos Földügyi és Térképészeti Hivatal, Földmérési és Térképészeti Főosztály, Budapest.

Tan C. L., Ng P. O. (1998): Text Extraction Using Pyramid. Pattern Recognition, Vol. 31, No. 1, pp. 63-72.

Terzopoulos D. (1983): Multilevel Computational Processes for Visual Surface Reconstraction. Computer Vision, Graphics and Image Processing, Vol. 24, pp. 52-96.

Terzopoulos D. (1986): Regularization of Inverse Visual Problems Involving Discontinuities. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 8, No. 4, pp. 413-423.

Terzopoulos D. (1988): The Computation of Visible-Surface Representations. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 10, No. 4, pp. 417-438.

Trier O. D., Jain A.K., Taxt T. (1996): Feature Extraction Methods for Character Recognition - A survey. Pattern Recognition, Vol. 29, No. 4, pp. 641-662.

Yamada H., Yamamoto K., Saito T., Hosokawa K. (1994): Recognition of Elevation Value in Topographic Maps by Multi-Angled Parallelism. Int. Journal of Pattern Recogn. and Artif. Intell., pp. 1149-1170.

Vaxiviere P., Tombre K. (1992): Celesstin: CAD Conversion of Mechanical Drawings. Computer, Vol. 25, No. 7, pp. 46-54.

Wall K., Danielsson P.-E. (1984): A fast sequential method for polygonal approximation of digitized curves. Computer Vision, Graphics and Image Processing, Vol. 28, pp. 220-227.

Wenyin L., Dori D. (1996): Sparse Pixel Tracking: A Fast Vectorization Algorithm Applied to Engineering Drawings. Proc. of 13th Internat. Conf. on Pattern Recognition, IEEE Press, Los Alamitos, California, pp. 808-812.

Wenyin L., Dori D. (1999): From Raster to Vectors: Extracting Visual Information from Line Drawings. Pattern Analysis and Applications, Springer-Verlag, 1999/2, pp. 10-21.

Weibel R., Heller M. (1991): Digital Terrain Modelling. In: Geographic Information Systems - Principles and Applications, John Wiley and Sons, pp. 269-297.

Köszönetnyilvánítás

Elsőként köszönetet mondok *Mezősi Gábor* egyetemi tanárnak, a Szegedi Tudományegyetem Természeti Földrajzi Tanszék vezetőjének, hogy támogatta munkámat és részvételemet a doktori programban.

Köszönettel tartozom *Hudra Györgynek*, aki előbb egyetemi hallgatóként, majd PhD ösztöndíjasként professzionális programozási munkájával támogatta a kutatási eredmények tesztelését és korszerű felhasználói felület kialakításával azok gyakorlati alkalmazását.

Köszönöm *Hans Kern* professzornak, a Karlsruhei Műszaki Főiskola tanárának a segítségét, amelyet kutatómunkájuk és a RoSy rendszer részletes bemutatásával nyújtott.

Köszönet illeti *Kiss Richárdot* és *Bódis Katalint*, a Természeti Földrajzi Tanszék munkatársait, akik elsősorban a terepmodellezés terén adtak értékes szakmai segítséget.

Köszönöm továbbá *Palágyi Kálmán* kollégám (Szegedi Tudományegyetem, Alkalmazott Informatikai Tanszék) segítségét a témához kapcsolódó szakirodalom felkutatásában.

Nem hagyhatom ki *Podolcsák Ádámot* sem, aki felkeltette érdeklődésem a kataszteri térképek digitalizálása és általában a térinformatika iránt.

Végül köszönettel tartozom családomnak, hogy elviselték a kutatómunka és a dolgozat írása során szükségszerű távollétemet.

Automatic Map Interpretation – Summary

To create a spatial database for some GIS application, it is a big challenge to recognize all the simple and complex map objects automatically on scanned maps. This research field is referred generally as *map interpretation.*

First part of the present study (Chapters 1 to 7) offers an overview of the topic and gives a detailed discussion of our own interpretation system called *MAPINT* which has special support for Hungarian cadastral maps.

Second part of the study (Chapters 8 to 11) takes special attention to the interpretation of contour line maps. Techniques of generating terrain models (DTM) from contour lines are also discussed. It is shown, that scanned contour line maps can be processed efficiently also without vectorization.

Chapter 1 (Introduction) shows the place of map interpretation concerning related topics *document analysis*, *OCR* (Optical Character Recognition), *form analysis* and *engineering drawing interpretation.*

Chapter 2 (Raster-to-Vector Conversion) gives an overview of techniques used to convert a scanned raster image into a set of vectors. This *raw vectorization* is usually the first step of map interpretation. The following approaches are discussed:

- thinning based vectorization (Lam et al. 1992),
- the rungraph method (Di Zenzo and Morelli (1989),
- mesh pattern technique (Lin et al. 1985),
- OZZ (Orthogonal Zig-Zag) vectorization (Wenyin and Dori 1996),
- Hough-transform based vectorization (Dori 1997).

Next, our own vectorization algorithm – applied in MAPINT system – is presented. This method differs from the usual thinning based solutions in a special automaton model to detect straight line segments. This detection is based on the notion of digital straight line and is performed in linear time.

Chapter 3 (Overview of Map Interpretation Systems) discusses interpretation systems which represent main approaches in the field:

– The *MARIS system* (Suzuki, Yamada 1990), developed to process large scale maps of Japan.

– A robust system, based on rungraph vectorization, to interpret *Italian cadastral maps* (Boatto et al. 1992).

– A Japanese system to recognize underground electric cable maps (Shimotsuji et al. 1992), which applies a probabilistic relaxation method to distinguish between lines of different meanings.

– The FRIMAP system to interpret German topographic maps (Ebi 1995) using the semantic network model of Niemann et al. (1990) which has special advantages in recognition of filling patterns.

– A commercial system, called RoSy (http://www.moss.de), which is a general purpose software package for vectorization, recognition and manual raster/vector editing. RoSy has its own development languages OGS and EASI to program recognition algorithms for given map types.

– The MAGELLAN system (Samet és Soffer 1998) recognizes and learns the legend of map, and after that interprets symbols on the map.

– Yamada et al. (1994) and Tan és Ng (1998) use pure raster techniques (without vectorization) to recognize map objects.

Chapter 4 (Data Modelling Aspects) gives an overview and evaluation of data models applied (or can be applied) in map interpretation. Following models are discussed:

– spaghetti model,
– topological models: the Arc/Info model and the SGD model of RoSy,
– relational model,
– object oriented model,
– semantic networks.

Investigations conclude to prefer *topological models* for map interpretation. Finally, spatial indexing techniques are discussed (quadtree index and grid index).

Chapter 5 (The DG Data Model) describes our own topological data model, called DG (Drawing Graph), constructed for map interpretation and applied in MAPINT. This model involves only four object types:

– NODE is a point with coordinates (x, y),
– EDGE is a straight line segment between two nodes,
– TEXT is an inscription on the map,
– PAT is an arbitrary set of other objects.

We show that these four types are sufficient to describe all complex structures required in map interpretation. The DG model is discussed both on logical (abstract) level and on physical (machine data structure) level. We show that for N vectors initial topology generation takes $O(N*\log(N))$ time, and all update operations take constant time.

Chapter 6 (The Interpretation System MAPINT) gives detailed description of our interpretation system developed in C++ for Windows environment. Although the system is basically universal, it has special support for Hungarian cadastral maps (Katona, Hudra 1999a).

Processing starts with an affine coordinate transform followed by raw vectorization (Chapter 2), the result of which is converted into DG format (Chapter 5). At this moment DG contains only NODE and EDGE objects, while TEXT and PAT objects are generated during recognition. The final result can be exported in DXF format. The following map objects are interpreted:

– dashed lines,
– inscriptions (house numbers and parcel numbers),
– connection signs (a special notation of Hungarian maps expressing the relationship between a building and a parcel),

– null-circles (denoting measured points),
– building and parcel polygons.

Character recognition is performed by a feed-forward back-propagation neural network, to ensure learning abilities for the system.

We show that practically all recognition operations can be performed in linear time when grid indexing is used for the DG.

Chapter 7 (Application: The Phare Land Consolidation Project) shows, how can MAPINT support the creation of a Hungarian cadastral map database. In the presented project (Omaszta, Szabó 1999) MAPINT serves for recognizing parcel numbers and connecting them with parcel records stored in an Oracle database (Katona, Hudra 1999b).

Chapter 8 (Interpretation of Contour Line Maps) investigates the possibilities of terrain interpretation. As a first step, recognition of contour lines and other terrain features of the legend is studied, based on the DG data model. For a full interpretation of terrain a Digital Terrain Model (DTM) should be generated, this topic is investigated in the next chapters.

Chapter 9 (Digital Terrain Modelling) gives an overview of techniques generating raster (DEM) or vector (TIN) models from a contour line map. We prefer the finite element DEM modelling applied by the ANUDEM program (Hutchinson 1989, 1996) and the TOPOGRID module of Arc/Info (ESRI 1994b). This method is detailed in the next chapter.

Chapter 10 (Variational Spline Interpolation) discusses the theoretical basis of finite element DEM modelling with thin plate splines (Terzopoulos 1983). The problem is reduced to a linear equation system, which can be solved by Jacobi iteration. Considering the locality of the problem, the iteration can be considered as repeated convolution on the DEM matrix. As a consequence, we use the convolution theorem and Fourier analysis (Gonzalez, Wintz 1987) to investigate the convergence of the procedure. Finally we point out the slow convergence of thin plate relaxation, this problem can be solved by multigrid relaxation discussed in next chapter.

Chapter 11 (Generating DEM from Contour Lines) gives a full technology starting from paper-based contour line maps and producing a high resolution DEM model. The presented technology does not use vectorization, all processing stages are performed on raster data. Main steps are:

– Scanning the paper map, manual cleaning and thinning of contour lines.
– A height value is given to each contour line. As a result, a so-called contour line matrix is gained containing elevation values at contour line points (and at given elevation points) while undefined values at other points.
– Spatial interpolation for undefined points. We use the method discussed in the previous chapter, using multigrid relaxation to speed up convergence.

We point out that the quality of generated DEM is drastically influenced by the reduction method use in the multigrid algorithm. Our new method, called *sparse contour reduction*, is introduced and its main features are proved.

An earlier version of the above technology has been applied in the DDM–10 project (Katona 1995). A 10 meter resolution DEM has been generated covering the full area of Hungary, based on the 1:50 000 contour line maps of Hungarian army.

Chapter 12 (Summary of Results) highlights the new results of the dissertation:

1. Line tracing method of the vectorization algorithm in Chapter 2.
2. The DG data model (Chapter 5).
3. The MAPINT interpretation system (Chapter 6).
4. Contour line interpretation by MAPINT (Chapter 8).
5. DEM generation using raster technology (Chapter 11).
6. Multigrid relaxation using sparse contour reduction (Chapter 11).