Algorithmen zur Manipulation von Raster- und Hybriden Daten

Dieter Fritsch
Lehrstuhl für Photogrammetrie und Fernerkundung
Technische Universität München
Arcisstraße 21
D-8000 München 2

Zusammenfassung

Geo-Informationssysteme zeichnen sich insbesondere durch ihre Möglichkeiten der Speicherung und Analyse von raumbezogenen Daten aus. Dadurch unterscheiden sie sich grundlegend von Kartier- und CAD-Systemen. Mit der Integration von Bilddaten steigen die Ansprüche an die zuvor angesprochenen beiden dominierenden Fähigkeiten, so daß entsprechende Algorithmen zur Verarbeitung von Vektor-, Sach- und Rasterdaten bereitzustellen sind.

Der vorliegende Beitrag setzt sich mit diesen Algorithmen auseinander. Dabei sollen insbesondere Verfahren betrachtet werden, die zur Rasterwelt hinführen, dort manipulieren und verschneiden sowie dann wieder zurücktransformieren. Dies bedingt gleich zu Beginn Vektor-Rasterkonvertierungen, für die leistungsvolle Algorithmen zur Verfügung stehen. Hingegen ist die Raster-Vektorkonvertierung bisher nur unvollständig gelöst: Die Transformation von Binärbilddaten in Vektordaten ist in den meisten Fällen zufriedenstellend und bedarf nur geringer interaktiver Überarbeitung. Dagegen liefern die gleichen Verfahren innerhalb von Grauwertbildern nur unzureichende Ergebnisse – hier kann nur die Integration von a priori Wissen zu einem weiteren Fortschritt beitragen. Die Verschneidung von Sachdaten ist bereits vermittels der SQL der relationalen Datenbanksysteme zu lösen, so daß hier der geringste Aufwand erforderlich wird.

Einige Beispiele belegen den derzeitigen Stand in Forschung und Entwicklung. Diese Beipiele sind in die Ausführungen zur Algorithmik integriert, um sofort den Bezug zur Praxis herzustellen.

1 Einleitung

Seit der Einführung von Geo-Informationssystemen (GIS) in den sechziger Jahren (R.F. Tomlinson, 1972) hat sich das Umfeld, in dem sie sich befinden, enorm verändert. Mittlerweile ist ihre Technologie anerkannt worden und aus vielen Geo-Disziplinen nicht mehr wegzudenken. Bisher repräsentieren die Geo-Informationssysteme der einzelnen Fachdisziplinen in erster Linie große Datenbanken, d.h. die raumbezogenen Daten werden erfaßt und gespeichert. Jedoch steigen mit der zunehmenden Aktivierung der Daten auch die Anforderungen. Diese lassen sich im wesentlichen durch die folgenden vier *Schlagwörter* angeben:

- Integration
- Interaktivität
- Räumliche Analyse
- Objektbildung

Die Anforderungen sind etwas näher zu hinterfragen: Unter *Integration* versteht man die Verknüpfung von Vektor-, Raster- und Sachdaten sowie Daten für die graphische Ausgestaltung. Eine Auswahl von Daten aus diesen vier Datenarten zur Lösung einer bestimmten Problemstellung führt dann zu den sogenannten *Fachdaten*.

Mit der *Interaktivität* ist in erster Linie die Kommunikation zwischen dem Anwender und der Maschine bezeichnet. Dabei sollte die Zeit für Anfragen immer geringer werden, d.h. die Leistungsfähigkeit der Geo-Informationssysteme ist zwangsläufig zu steigern. Hier sind es insbesondere Simulationen, die den Einsatz von Geo-Informationssystemen erfordern, um Umweltverträglichkeit von Großprojekten abzuprüfen oder ganz einfach Planungen vorzubereiten. Dabei werden Abfragen in der Form von *Was wäre, wenn ...* immer bedeutungsvoller.

Die *räumliche Analyse* vergleicht nicht nur verschiedene Sachdatensätze miteinander, wenngleich gerade die thematischen Dimensionen ein GIS auszeichnen, sondern führt auch Auswertungen von geometrisch-nachbarschaftlichen Beziehungen durch. Dabei sind neue Datenobjekte aufzubauen sowie Abfragesprachen für die Geometrie und Sachdaten einzusetzen.

Innerhalb der *Objektbildung* sind die Aspekte des geometrischen und thematischen Modellierens zu berücksichtigen. Die Objektdefinition ist mit der Abbildung 1 wiedergegeben – diese bildet ebenso die Grundlage für eine *objektweise* Organisation der raumbezogenen Daten.

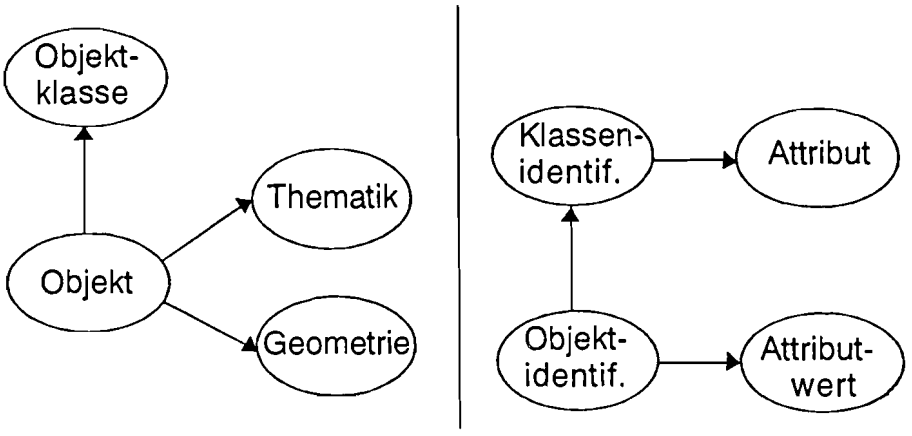

Abbildung 1: Objektdefinition in Geo-Informationssystemen

Als bereits historische Objektbildung ist die Vorgehensweise bei der *Automatisierten Liegenschaftskarte (ALK)* der BRD zu nennen, bei der seitens der Geometrie hinsichtlich der Objektfolge in *Linie-Bogen-Polygon-Spline* sowie der graphischen Ausgestaltung in der Form von *Linie-Text-Flurstücksnummer-Punkt* unterschieden worden ist.

Mit der Integration von Rasterdaten in Geo-Informationssysteme steigen nicht nur die Analysemöglichkeiten an, sondern auch der Entwicklungsaufwand. In der Abbildung 2 sind die notwendigen Verarbeitungsschritte zu einer vollständigen Integration nachgewiesen. Dabei werden verschiedene *Aktivebenen* vorgegeben, die eine unterschiedliche Intelligenz der Datenverarbeitung voraussetzen.

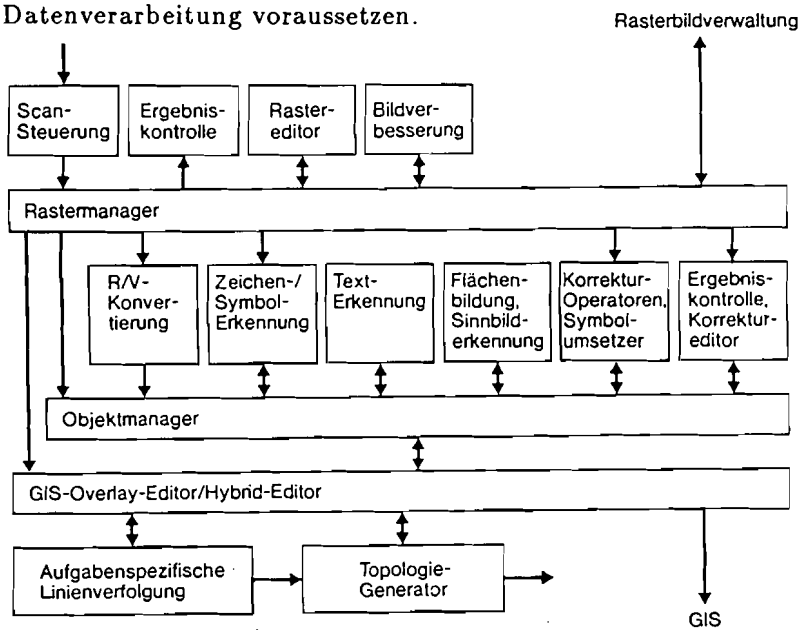

Abbildung 2: Integration von Rasterdaten in GIS

Auf der untersten Ebene erfolgt die Scan-Steuerung, die Ergebniskontrolle sowie die Bildverbesserung durch Algorithmen der digitalen Bildverarbeitung – ein Rastereditor ermöglicht die interaktive lokale Veränderung des Bildinhalts. In der nächsthöheren Ebene befindet sich der *Rastermanager*. Neben der Rasterbildverwaltung steuert der Rastermanager die Raster-/Vektorkonvertierung und die Zeichen-/Symbolerkennung; er trägt mit Korrekturoperatoren sowie Symbolumsetzern zu konsistenten graphischen Darstellungen bei. Diese kann jedoch noch in einem letzten Schritt durch einen Korrektureditor verändert werden.

Dem *Objektmanager* sind Module zur Flächenbildung, Sinnbild- sowie Texterkennung untergeordnet. Hier werden homogene Rasterflächen erkannt – beispielsweise durch interaktive oder rechnergestützte Klassifikation – und objektweise verwaltet.

Auf der obersten Ebene befindet sich der *GIS-Overlay-Editor* bzw. ein *hybrider Editor*, der Vektor- und Rasterdaten editieren kann. Aufgabenspezifische Linienverfolgungen sowie Topologie-Generatoren tragen zur vollständigen Zerlegung der aus den Bildern extrahierten Spaghetti-Graphik in die topologischen Grundprimitiven wie Knoten, Kanten und Flächen bei.

Abbildung 2 zeigt einen Blick in die Zukunft der *hybriden Geo-Informationssystemen*, die sich momentan in der Entwicklung befinden. Bisher ist lediglich die unterste Ebene vollständig realisiert – zu allen anderen Modulen ist noch sehr viel Forschungsarbeit zu leisten. Dennoch sind diese Systeme bereits jetzt schon in der Lage, anspruchsvolle Analysen und synthetische Darstellungen durchzuführen. Mit dieser Abbildung wird ebenfalls die Vielfalt der Algorithmen angedeutet, deren Anwendungsspektrum von einfachen Bildverarbeitungfunktionen über Klassifikationen mit nachgeschalteter Datenstrukturierung bis hin zur vollständigen to-

pologischen Zerlegung reicht. Hinzu kommen noch *Verschneidungen* der unterschiedlichen Datentypen, so daß der algorithmische Aspekt in GIS eine mindestens so große Bedeutung besitzt wie die Datenverwaltung.

Im nachfolgenden können wegen dieser Vielfalt exemplarisch nur einige Algorithmen vorgestellt werden. Dabei steht die Analyse von Rasterdaten im Vordergrund; jedoch ist in einigen Fällen die simultane Behandlung von Vektor-, Sach- und Rasterdaten notwendig. Eine weitere Einschränkung wird sich dadurch auferlegt, daß homogene Rasterflächen nach dem *Quadtree-Prinzip* verwaltet werden, innerhalb dessen Verschneidungen sehr elegant und einfach zu bewerkstelligen sind. Dies bedingt jedoch auch Vektor-/Rasterkonvertierungen, um die Vektordaten in Rasterdarstellungen zu überführen. Während die raumbezogenen Daten vektorseitig sehr komprimiert in umfassenden logischen Datenmodellen abgespeichert werden können, bietet die Rasterdatenverarbeitung ideale Voraussetzungen zur Datenanalyse- und synthese.

2 Operationen an Rasterdaten

Eine matrixförmige Anordnung von Daten bezeichnet man als *Rasterdaten*. Diese besitzen eine sehr einfache Struktur, da sie nach ihren Zeilen und Spalten geordnet sind. Innerhalb der Operationen an Rasterdaten ist zu unterscheiden hinsichtlich der

- Vektor-/Rasterkonvertierung, die die i.d.R. kontinuierlichen Vektordaten diskretisiert. Dabei können verschiedene Hard- und Softwarekomponenten Anwendung finden. Häufig sind bereits leistungsfähige Algorithmen wie die *digitale Differenzstufe* oder der *Bresenham-Algorithmus* hardwaremäßig realisiert.

- Speicherung bzw. Codierung in der Form von Punkt-, Linien- und Flächenorganisationen wie z.B. *Kettencodes, Runlength-Kodierung* und *Quadtrees*.

- Manipulation durch entsprechende Algorithmen, die wiederum Rasterdaten erzeugen. Hierzu gehören *geometrische* und *radiometrische* Transformation genauso wie *Analysen* in der Form von Verschneidungen.

- Raster-/Vektorkonvertierung als *inverses* Element der Rasterdatenverarbeitung. Dabei ist zu differenzieren in eine *pixelorientierte* und eine *wissensbasierte* Vorgehensweise, wobei erstere als *bottom-up*- und die letztere als *top-down*-Strategie bezeichnet wird.

Die Ein- und Ausgabe der Rasterdaten soll hier nicht weiter verfolgt werden. Hierzu wird auf die umfangreichen Betrachtungen in R. Bill/D. Fritsch (1991) verwiesen.

2.1 Vektor-/Rasterkonvertierung

Innerhalb der Vektor-/Rasterkonvertierung stehen drei Verfahren zur Verfügung: Die symmetrische digitale Differenzstufe (digital differential analyzer, DDA), der einfache DDA und der Bresenham-Algorithmus.

Ein DDA erzeugt Linien nach deren Differentialgleichung, wobei sich jedoch auf die Gleichung einer Geraden gestützt wird.

$$y = a_0 + a_1 x \Longrightarrow \frac{dy}{dx} = \Delta y / \Delta x = a_1$$

Der DDA-Liniengenerator erzeugt nun zu einer gegebenen x,y-Position ein Bildelement (Pixel) an der Stelle m, n (m=Spalten-,n=Zeilenindex) nach der Vorschrift:

$$(x,y) \rightarrow (m,n) : \begin{cases} m = Integer((x - x_{min})/\delta + 0.5) \\ n = Integer((y - y_{min})/\delta + 0.5) \end{cases}$$

Dabei beschreibt x_{min}, y_{min} eine *Umgebung* der x,y-Position - δ bezeichnet die Rasterweite. Da zur Liniendarstellung jeweils nur ganzzahlige Werte (Integer) herangezogen werden, trägt die Addition von 0.5 zu einer korrekten Rundung bei.

Der *symmetrische* DDA definiert die Aufrasterung in einer 3x3-Umgebung (4 Rasterzellen), während der einfache DDA nur eine Rasterzelle zur Verfügung stellt. Dadurch liefert der symmetrische DDA immer korrekte Liniendarstellungen, da die Abweichung des aufgerasterten Punktes von der wahren Linie nie größer als die halbe Rasterweite δ sein kann. Der Einfach-DDA erzeugt eine andere Art der Punktfolge, weil das Definitionsintervall auf eine Rasterzelle eingeschränkt ist. Er ist seitens der Software schneller als der symmetrische DDA, dafür jedoch qualitativ schlechter. In der Abbildung 3a,b sind die beiden DDA-Strategien wiedergegeben.

a) b) c)

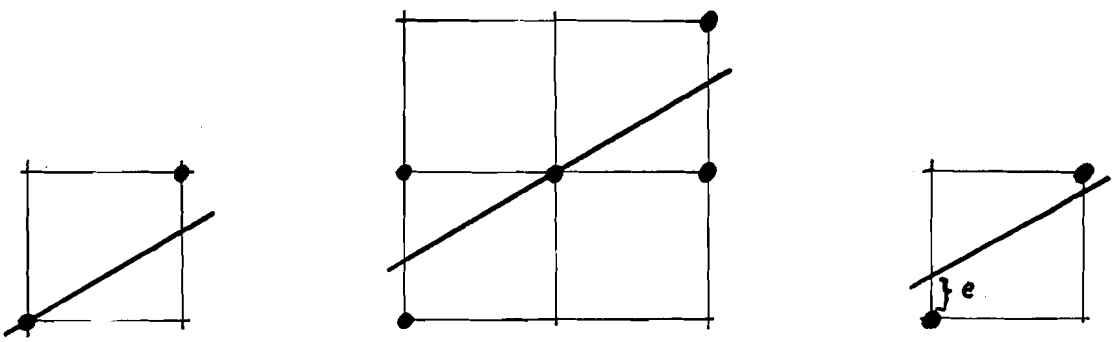

Abbildung 3: Rastermodi bei der Vektor-/Rasterkonvertierung
a) Symmetrischer DDA b) Einfacher DDA c) Bresenham-Algorithmus

Beide Strategien benötigen eine Divisionslogik zur Berechnung der Pixelposition m, n. Da eine Hardwarerealisierung Multiplikationen und Divisionen möglichst ausschließen sollte, um schnell die Vektor-/Rasterkonvertierung durchführen zu können, kommen beide Verfahren hierfür nur bedingt zur Anwendung. Eine divisionsfreie Strategie verfolgt der *Bresenham-Algorithmus*. Dieser ist ähnlich wie der Einfach-DDA aufgebaut - dabei wird am Anfang ein Fehlermaß e:=$2\Delta y$-Δx definiert, welches im Verlauf der Iterationen zu verändern ist. Ein postives e (vgl. Abbildung 3c) bedeutet, daß sich der Linienverlauf über dem jeweiligen Rasterpunkt befindet; der entsprechende Zeilenindex n wird um 1 erniedrigt. Ist e negativ, so bleibt n unverändert. Die Umsetzung des Bresenham-Algorithmus in eine Pascal-Codierung ist in W.M. Newman/R.F. Sproull (1984) angegeben. Da dieser Algorithmus sowohl Multiplikationen als Divisionen umgeht, findet man ihn häufig in einfachen Mikroprozessoren als *Firmware* wieder.

3 Speicherung von Rasterdaten

Die Transformation der Vektordaten in Rasterdaten ist der erste Schritt, um die Methoden der Rasterdatenverarbeitung anzuwenden. Das Resultat der Vektor-/Rasterkonvertierung muß für die Analyse vielfach zwischengespeichert werden. Da die Rasterdaten stets *punktbezogene* Daten darstellen, ist ihre Topologie ebenfalls punktbezogen. Jedoch können Nachbarschaftsrichtungen direkt Berücksichtigung finden, wenn die Verzweigungsrichtungen angegeben sind. Die Abspeicherung dieser Richtungen kann zu linienhaften Rasterdaten führen, da hiermit quasi Linienstrukturen oder auch *Spaghettis* erzeugt werden können.

Eine flächenhafte bzw. hierarchische Vorgehensweise ist mit dem Quadtree-Verfahren möglich – dieses wird ebenso im folgenden kurz vorgestellt.

3.1 Kettencode

Ein sehr einfaches Prinzip zur Ableitung von flächenhaften Strukturen für quadratische Raster verfolgt das *Kettencode-Prinzip*, welches auch als *Freeman-chain* bezeichnet ist. Dabei bedient man sich der Abspeicherung der *acht* möglichen Verzweigungsrichtungen, wie sie in der Abbildung 4 mit einem kleinen Beispiel wiedergegeben sind. Die Linienstruktur wird erhalten, wenn ihre Orientierung in Form einer zahlenmäßig codierten Verzweigungsrichtung abgespeichert ist. Dieser Code heißt *Freeman-Zahl*.

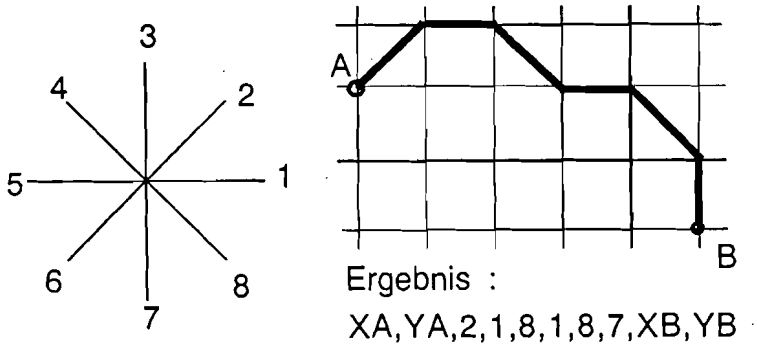

Abbildung 4: Verzweigungsrichtungen von Rasterdaten und Anwendung des Kettencodes

Das Ergebnis einer Linienverfolgung mittels der Kettencode-Struktur können die Anfangsposition im Raster sowie die Freeman-Zahlen sein. Ferner kann auch der jeweilige Zeilen- und Spaltenindex mit abgespeichert werden. Betrachtet man die Freeman-Zahlen bzw. den Zeilen- und Spaltenindex als *Koordinaten*, so ergeben sich lange, dünne Linienlisten – diese werden als *Spaghettis* bezeichnet. Als Nachteil des Kettencodes gilt seine ungünstige Weiterverarbeitung; des weiteren fällt ein unnötig hoher Speicherbedarf an insbesondere für lange, gerade Linien. Er eignet sich aber für die Ausgabe auf Stift- und Rasterplottern.

Der Kettencode wird häufig in der Raster-/Vektorkonvertierung eingesetzt. Dabei ist nicht unbedingt die Freeman-Zahl abzuspeichern – es genügt die Ermittlung der Zeilen- und Spaltenindizes. Hierauf wird an späterer Stelle wieder eingegangen.

3.2 Runlength-Codierung

Die *Runlength-Codierung* dient der Zusammenfassung von homogenen Zeilenabschnitten der Rasterdaten. Insbesondere binäre Rasterdaten lassen sich mit ihr sehr komprimiert und dadurch speicherplatzsparend verwalten - jedoch findet man die Runlength-Codierung auch häufig als Komprimierungsmethode für Grauwertbilder eingesetzt. Hier gibt es verschiedene Ausprägungen: Die einfachste Form speichert die Anzahl der gleichartigen Grauwerte (vgl. Abbildung 5).

```
0 0 0 0 0 1 1 1 1 0 0 0 1 1 1        5*0,4*1,3*0,3*1
0 1 1 1 1 1 0 0 0 1 1 1 0 0 0        1*0,5*1,3*0,3*1,3*0
0 0 0 1 1 1 1 1 0 0 0 1 1 1 1        3*0,5*1,3*0,4*1
```

Abbildung 5: Runlength-Codierung in einfacher Form

Eine der Vektor-/Rasterkonvertierung adequate Form ist mit der Unterscheidung in Knoten (Kn), Kanten (Ka) und Flächen (Fl) gegeben. Dadurch können Informationen über die topologischen Grundelemente der Vektordaten erhalten bleiben, was in vielen Fällen durchaus gewünscht sein kann. Mit der Abbildung 6 ist ein kleines Beispiel hierzu demonstriert.

```
x x x x x x x x x x x x x o o o     (13,3,Fl)
x x x o x x x x o x x x o o o o     (4,1,Kn)  (9,1,Kn) (13,4,Fl)
x x x x x x x x x x o o o o o o     (11,6,Fl)
o x x x x x x x x x o o o o o o     (1,1,Ka) (11,6,Fl)
x o x x x x x x x o o o o o o o     (2,1,Ka) (10,7,Fl)
x o x x x x x x x x x x x x x x     (2,1,Ka)
x x o x x x x x x x x o x x x x     (3,1,Ka) (12,1,Kn)
```

Abbildung 6: Topologisch orientierte Runlength-Codierung

3.3 Quadtree-Zerlegung

Das *Quadtree-Prinzip* eignet sich besonders als flächenhafte Datenstruktur sowie zur Verschneidung von thematischen Rasterdaten. Dies impliziert eine Vorverarbeitung insofern, daß Grauwertbilder zu *homogenen* Einheiten zusammengefäßt werden müssen. Die Hilfsmittel hierzu sind Klassifikationen, die automatisiert oder auch interaktiv ablaufen können.

Das Quadtree-Prinzip kann ebenso als hierarchische Struktur gesehen werden. Somit läßt sich auf lokale Rasterzellen durch die vordefinierten Wege des Quadtrees sehr schnell zugreifen.

Die Definition des Quadtree-Prinzips ist sehr einfach: Ein Quadratsegment wird durch sukzessive Viertelung unterteilt, wobei die Auflösung - die kleinste Einheit des Quadtrees - durch den Anwender vorgegeben werden kann. Der Baum repräsentiert dann den Weg zum Zugriff auf das entsprechende Element (siehe Abbildung 7). Durch die sukzessive Viertelung werden einem Vater vier Söhne zugewiesen - jeder Sohn kann wiederum vier Kinder haben.

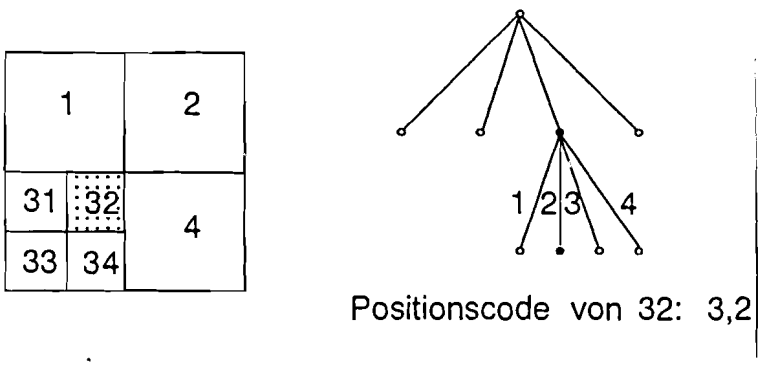

Abbildung 7: Definition des Quadtrees

Mit den Abildungen 8a,b sind ein synthetisches und ein reales Beispiel zur Quadtree-Zerlegung wiedergegeben. In 8a ist eine Ellipse entsprechend codiert worden – diese Abbildung will gleichfalls die Zerlegung eines raumbezogenen Gebiets anzeigen. Homogene Flächen können mit großen Elementen ausgefüllt werden, wohingegen Randbereiche stets eine große Verdichtung des Quadtrees erfordern. In der Abbildung 8b ist eine Landsat MSS-Aufnahme vermittels einer geologischen Bodenkarte klassifiziert und danach Quadtree-codiert worden. Die unterschiedlichen Flächen treten durch eine unterschiedliche Grauwertintensität in Erscheinung.

a) b.)

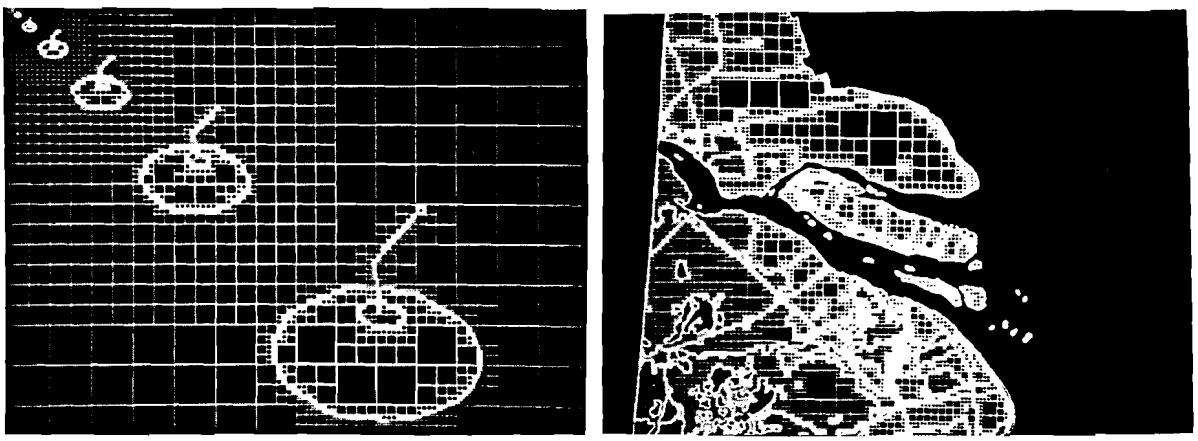

Abbildung 8: Anwendung der Quadtree-Zerlegung

Werden nun die größeren Elemente kontinuierlich ausgeblendet – auch als clipping bezeichnet – so erhält man eine *Randdarstellung* der Quadtree-codierten Rasterflächen. Je stärker die Ausblendung, desto mehr nähert sich diese Randdarstellung einer Quasi-Vektorisierung, was mit der Abbildung 9a,b belegt ist. Dieser Effekt trägt nun auf einfache Weise zur Raster-/Vektorkonvertierung bei (A. Kern, 1990).

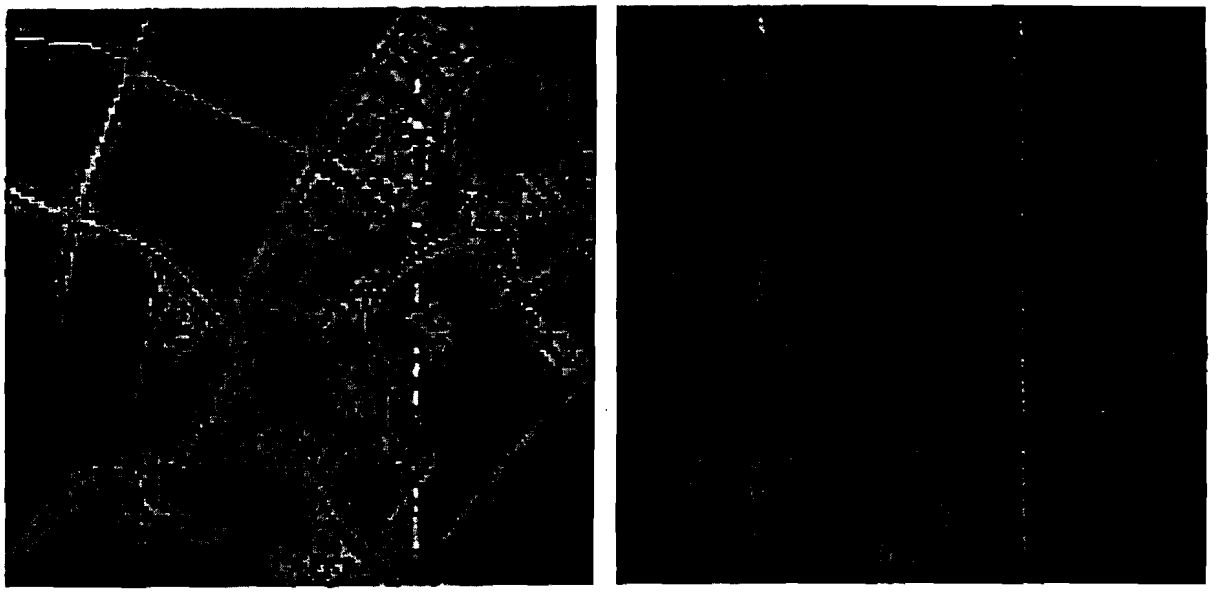

Abbildung 9: Randeffekt bei der Quadtree-Zerlegung

Dabei brauchen lediglich die einzelnen Randelemente auf der untersten Stufe des Quadtrees geordnet zu werden, d.h. je nach farblicher Codierung sind relationale Tabellen der Randelemente aufzustellen. Diese Kanten- und Knotentabellen ergeben schon die gewünschte Ordnung hinsichtlich topologischer Grundprimitive; jedoch entspricht der Kantenzug einer *Treppenfunktion* und ist daher noch durch approximierende Polynome (Geraden, Splines, Kreisbögen) nachzubearbeiten.

4 Manipulationen

Wie bereits vorher angedeutet, sind Manipulationen an Rasterdaten nicht nur innerhalb von Analysen, sondern auch zur Nachbearbeitung verschiedener Speichermodi notwendig. Dies impliziert an breites Spektrum an Algorithmen, die von einfacher Bildverarbeitung über Approximationen bis hin zu Verschneidungen reichen. Rasterdaten stehen nie isoliert, sondern müssen meistens in Verbindung mit Vektor- und Sachdaten gesehen werden.

4.1 Geometrie

Hinsichtlich der geometrischen Manipulation an Rasterdaten können verschiedene Einteilungen vorgenommen werden. Diese können sich an der Integration von externem Wissen orientieren: Eine Entzerrung (global oder differentiell) ist rein pixelorientiert: die bessere Güte in Form der differentiellen Orthoprojektion benötigt jedoch schon a priori Wissen in der Form eines digitalen Geländemodells. Die Extraktion von anthropogenen Objekten wie z.B. Häuser setzt umfassende Wissensbasen voraus, so daß mit den steigenden Ansprüchen der Komplexitätsgrad ebenso ansteigt. In der Raster-/Vektorkonvertierung wird nochmals hierauf zurückgekommen.

Eine weitere Strategie verfolgt die automatische Extraktion der Geländehöhen von zueinander homologen digitalen Bildern. Hier ist in den letzten Jahren seitens der Photogrammetrie umfangreiche Forschungsarbeit geleistet worden – eine Zusammenstellung der Methoden zur digitalen Bildzuordnung befindet sich in E.P. Baltsavias et al. (1990), C. Heipke (1991).

4.2 Radiometrie

Die *Radiometrie* wird vielfach mit den Methoden der digitalen Bildverarbeitung verändert. Im einfachsten Fall stellt sich eine radiometrische Korrektur als Kombination einer additiven und multiplikativen Komponente c und f dar:

$$x'(m,n) = x(m,n) + [x(m,n) - \bar{x}]f + c$$

Diese Veränderung ist jedoch in den meisten Fällen durch Hardware in der Form von Look-up-Tabellen realisiert, so daß der Anwender beliebige Farbkompositionen der Rasterdaten selbst zusammenstellen kann.

Weitere Operationen sind Filterverfahren sowie Gradientenberechnungen – auf die Ableitung von Gradienten wird in der Raster-/Vektorkonvertierung näher eingegangen.

4.3 Verschneidungen

Innerhalb der Analysen von raumbezogenen Daten sind häufig *Verschneidungen* durchzuführen. Entsprechend der Datentypen wird unterschieden in Verschneidungen zwischen

- Vektor-/Vektordaten
- Vektor-/Rasterdaten
- Vektor-/Sachdaten
- Raster-/Rasterdaten
- Raster-/Sachdaten
- Sach-/Sachdaten

Um nun die einzelnen toplogischen Grundprimitive bei Verschneidungen unterscheiden zu können, müssen Identifikatoren (ID) eingeführt werden. Somit ergeben sich ein Knoten_ID, Kanten_ID und Flächen_ID für die Gemetriedaten – der Sachdaten_ID kann der jeweilige Zugriffsschlüssel (Flustücks-Nr., Postleitzahl etc.) sein. Die Grundoperationen der Verschneidung sind durch die Mengenalgebra gegeben; hier unterscheidet man zwischen *Vereinigung*, *Differnz* und *Durchschnitt*. In der Abbildung 10 sind diese Operationen für zwei Flächen A und B angegeben.

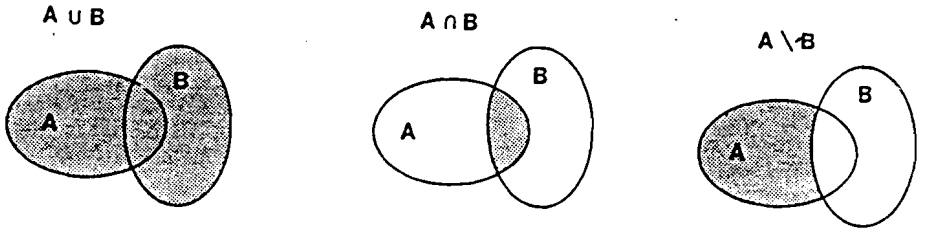

Abbildung 10: Grundelemente der Verschneidung

Die algorithmischen Aspekte insbesondere bei der Verschneidung von Flächen liegen in der Ausarbeitung einer geeigneten Strategie. Dabei kommt häufig die *Divide-and-Conquer-Methode* zur Anwendung, bei der die Flächen erst in einfachere Elemente zerlegt, dann verschnitten und hinterher wieder zusammengesetzt werden. Explizit versteht man unter

Divide: *Jede komplexe geometrische Fläche wird in viele einfache Flächen zerlegt.*

Verschneidung: *Die Verschneidung wird mit vielen einfachen Flächen vorgenommen*

Conquer: *Die Schnittmengen vieler einfacher Flächen werden wieder zu einer komplexen Schnitttmenge zusammengefaßt.*

Vektor-/Vektordaten

Die Verschneidung von Vektor-/Vektordaten ist wegen der umfangreichen topolgischen Modellbildung häufig sehr kompliziert und vielfach unvollständig gelöst. Die zuvor beschriebene Divide-and-Conquer-Methode kann hier in die einzelnen Elementmengen aufteilen, die Elementmengen miteinander verschneiden und die Schnittmengen der Grundelemente wieder zusammenfügen.

In R. Bill/D. Fritsch (1992) wird die Verschneidung von Vektordaten ausführlich dargestellt. Da die vektoriellen Geometriedaten jedoch in Rasterdaten transformiert werden können, findet man zunehmend rein rasterorientierte Verschneidungen.

Raster-/Rasterdaten

Die Verschneidung innerhalb von Rasterdaten kann sehr effizient auf der Basis der Quadtree-Zerlegung ablaufen. Im übertragenen Sinne ist diese Zerlegung auch als eine Divide-and-Conquer-Methode aufzufassen, da eine komplexe Fläche in Basisquadrate unterschiedlicher Größe zerlegt wird, die dann verschnitten und hinterher wieder vereinigt werden können.

Aufgrund der Vektor-/Rastertransformation bietet sich nun eine durchgängige Codierung der Knoten, Kanten und Flächen an – diese kann durchaus auf Quadtree-Basis erfolgen. Das Verschneidungsproblem ist dann besonders einfach: Eine separierte Verschneidung der drei quadtree-codierten Elementarmengen führt zu dem gewünschten Ergebnis.

Sach-/Sachdaten

Die Verschneidung verschiedener Sachdatensätze kann vielfach schon vermittels der Abfragesprache SQL (structured query language) der relationalen Datenbanksysteme bewerkstelligt werden. Ein kleines Beispiel möchte dies erläutern:

Beispiel: *Aus den gegebenen Sachdaten-Tabellen soll der Bauplatz herausgefunden werden, dessen Fläche* $\geq 600m^2$ *beträgt, und dessen Eigentümer nicht in x-Stadt wohnt.*

Eigentümer	-Tabelle		
ID	Name	geb. am	Wohnort
I	Hans Meier	1.1.1921	x-Stadt
II	Fritz Müller	6.5.1921	x-Stadt
III	Bertha Schmidt	3.3.1930	y-Stadt

Nutzungs	-Tabelle		
ID	Nutzung	Größe	Bodenschätzung
I	Bauplatz	650 qm	
II	Wiese	480 qm	sLIII-48/52
III	Bauplatz	780 qm	

```
SELECT Eigentuemer.NAME,
       Eigentuemer.WOHNORT,
       Nutzung.NUTZUNG,
       Nutzung.GROESSE
FROM   Eigentuemer-Tabelle, Nutzungs-Tabelle
WHERE  Eigentuemer-Tabelle_ID = Nutzungs-Tabelle_ID
AND    NOT Eigentuemer.WOHNORT = x-Stadt
AND    Nutzung.NUTZUNG = BAUPLATZ
AND    Nutzung.GROESSE > 600
```

Als Ergebnis wird erhalten: Bertha Schmidt, y-Stadt, 780 m^2.

Vermittels des Einsatzes der SQL können neue Tabellen aufgebaut werden, die dann zur Lösung des jeweiligen Verschneidungsproblems beitragen. Diese Tabellen treten nicht nur bei der Sach-/Sachdaten-Verschneidung auf, sondern fallen ebenso bei der Verschneidung anderer Datentypen an.

5 Raster-/Vektorkonvertierung

Die *Raster-/Vektorkonvertierung* ist dann notwendig, wenn Strichkarten gescannt und anschließend automatisch wieder in Vektordarstellungen überführt werden müssen, oder aber Vektorinformation aus Luftbilder und Fernerkundungsaufnahmen zu extrahieren ist. Eine

ausführliche Arbeit zur Binärbildkonvertierung hat A. Illert (1990) zusammengestellt – heutige kommerzielle Systeme zur Raster-/Vektorkonvertierung können 70-90 % der ursprünglichen Vektordaten aus binären Rasterdateien extrahieren. Jedoch benötigen die restlichen 20-30 % eine aufwendige Nachbearbeitung, so daß der hohe Wirkungsgrad der automatischen Erfassung schnell an Attraktivität verliert. Dennoch ist damit ein Weg eingeschlagen worden, der bei mittlerem Anspruchsniveau durchaus zu einer schnellen Datenerfassung führt.

Im folgenden sollen exemplarisch für den weit schwierigeren Weg – nämlich die Extraktion von Vektordaten aus Grauwertbildern – einige Agorithmen angegeben werden. Diese befinden sich jedoch auch teilweise in der Anwendung der Binärbildverarbeitung.

5.1 Binärisierung

Da gescannte Karten i.d.R. große Grauwertsprünge aufweisen, ist ihre Binärisierung nicht schwierig. Die geometrischen Ausgangsprimitive sind bereits Knoten und Kanten, aus denen Flächen definiert werden können. Falls Grauwertbilder einer Vektorisierung zu unterwerfen sind, ist der Binärisierung eine *Kantenextraktion* vorzuschalten. Diese Kantenextraktion wird in *Gradientenbildern* offensichtlich, so daß der erste Schritt in der Berechnung von Bildgradienten besteht.

Der Bildgradient visualisiert die Differenzen in den Grauwerten, d.h. kleine Grauwertdifferenzen bilden sich dabei als mehr oder weniger homogene Flächen und Grauwertsprünge als gerasterte Linien ab.

Es sei

$$x(m,n) \qquad \forall m = 0,1,2,....,M-1, \quad und \quad n = 0,1,2,....,N-1$$

das Grauwertbild. Sein Gradient $g(m,n)$ wird erhalten aus

$$g(m,n) = \nabla x(m,n) = grad[x(m,n)]$$

Zur Berechnung des Bildgradienten stehen verschiedene lineare und nichtlineare Operatoren zur Verfügung. Die Vorschrift in der Anwendung solcher Operatoren ist gegeben durch

$$\Delta x_m = h_m * x(m,n) \qquad \Delta x_n(m,n) = h_n m,n * x(m,n)$$
$$\Longrightarrow g(m,n) = \sqrt{(\Delta x_m(m,n)^2 + \Delta x_n(m,n)^2}$$

Ein Vergleich verschiedener Operatoren ist mit der Abbildung 11 wiedergegeben. Dabei wurden die Operatoren nach *Sobel* und *Roberts* einem *3x3 Differentiator*, ein 2x2 und dem *Maximum-Operator* gegenübergestellt. Es zeigen sich keine Unterschiede in der Güte des Bildgradienten, wobei jedoch ein gut definiertes Bildsignal vorgegeben worden ist.

Die *Impulsantwort* $h_m(m,n)$ eines einfachen 3x3 Differentiators ist nachfolgend vorgestellt; auf die Definition der weiteren Operatoren wird auf E.P. Baltsavias et al. (1990) verwiesen. Zur Anwendung in Spaltenrichtung ist dieser Operator zu transponieren, also $h_n(m,n) = h_m(m,n)^T$

$$h_m(m,n) = \begin{pmatrix} 0 & 0 & 0 \\ -0.5 & 0 & 0.5 \\ 0 & 0 & 0 \end{pmatrix}$$

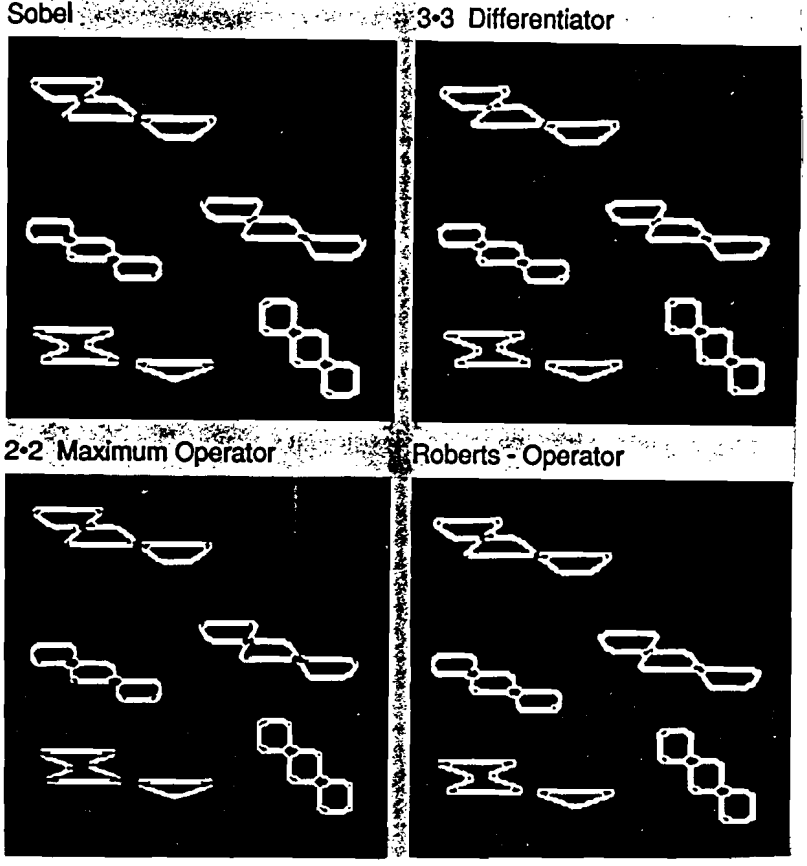

Abbildung 11: Vergleich verschiedener Gradientenoperatoren

Nach der Gradientberechnung kann dann die endgültige Binärisierung erfolgen, indem den Kanten der höchste (tiefste) und den restlichen Flächen die tiefsten (höchsten) Grauwerte zugewiesen werden.

5.2 Linienverfolgung

Das Ziel einer *Linienverfolgung* besteht in der Aufindung der Mittellinien – auch als Mittelachsen bezeichnet – von Kanten in binären Bildern. Hierzu gibt es grundsätzlich zwei voneinander differierende Methoden:

- Parametrische Verfahren (Kalman-Filter, Polynome etc.)
- Nichtparametrische Verfahren

Dabei erweisen sich die nichtparametrischen Verfahren als sehr schnell und robust – im Gegensatz zu den parametrischen Linienverfolgern, die häufig instabil werden können und zeitkritisch sind.

Ein Algorithmus, der sich innerhalb der Binärbildverarbeitung bewährt hat, ist nachfolgend vorgestellt. Es handelt sich dabei um eine abgewandelte Form des Aufsuchens des Kettencodes, wobei nicht die Verzweigungsrichtungen, sondern lediglich die jeweiligen Koordinaten der Mittelachse abgespeichert werden. Die folgenden Schritte sind hierzu notwendig:

```
0. Definiere ein Fenster der Groesse k*l (Beispiel: 5*5 oder 9*9)
1. Finde Anfangswerte oberhalb eines Schwellenwertes und zentriere das Fenster
2. Fuelle das Fenster mit Grauwerten
3. Berechne den geometrischen Schwerpunkt des Fensters
4. Speichere den geometrischen Schwerpunkt (evt. auch den Kettencode)
5. Loesche das Fenster durch Nullzuweisung
6. Bewege das Fenster
7. Gehe nach 2.
8. Bei Luecken gehe nach 1.
```

Als Beispiel hierzu diene eine Fenster $f(m,n)$ der Größe 5*5, dessen Grauwerte gegen einen Schwellenwert von 20 abgeprüft werden sollen. Gemäß der Fenstergröße ergibt sich als erster Fensterschwerpunkt $\tilde{m} = \tilde{n} = 3$.

$$f(m,n) = \begin{pmatrix} 11 & 9 & 15 & 25 & 36 \\ 10 & 8 & 21 & 25 & 42 \\ 11 & 7 & 36 & 45 & 42 \\ 9 & 42 & 41 & 40 & 40 \\ 10 & 42 & 40 & 36 & 36 \end{pmatrix}$$

Der neue Fensterschwerpunkt wird als gewogenes Mittel erhalten, also

$$\begin{aligned} \tilde{m} &= [2*2+4*3+5*4+5*5]/16 = \quad \text{Int}(61/16) = 4 \\ \tilde{n} &= [2*1+3*2+3*3+4*4+4*5]/16 = \quad \text{Int}(53/16) = 3 \end{aligned}$$

Die Fensterbewegung erfolgt nach $\tilde{m}$=4 und $\tilde{n}$=3, so daß vor der Bewegung das bisherige Fenster gelöscht und nun mit neuen Grauwerten aufgefüllt wird. Das neue Fenster sei

$$f(m,n) = \begin{pmatrix} 0 & 0 & 0 & 0 & 36 \\ 0 & 0 & 0 & 0 & 25 \\ 0 & 0 & 0 & 0 & 28 \\ 0 & 0 & 0 & 0 & 29 \\ 0 & 0 & 0 & 0 & 37 \end{pmatrix}$$

Wie leicht einzusehen ist, liegt der neue Schwerpunkt bei $\tilde{m}$=5, $\tilde{n}$=3, so daß das Fenster um Δm=2 verschoben werden muß. Die Vorteile dieser Vorgehensweise liegen in:

- Linienverdünnung sehr schnell
- Festerschwerpunkte liefern bereits eine erste Vektorisierung (punktuelle Darstellung)
- Lücken sind durch Fangkreisstrategien und größere Fenster zu überbrücken

5.3 Vektorisierung

Für die *Vektorisierung* in Form einer funktionalen Beschreibung können verschiedene Verfahren zur Anwendung kommen. Sehr einfache Strategien wie z.B. das *Split-and-Merge* (B. Moser, 1990) kommen bereits mit Geradenstücken aus und erzielen schon eine recht gute

Vektorisierung. Die Güte der Approximation ist dabei direkt an die einzelnen Iterationen gekoppelt. Sehr gute Ergebnisse werden i.d.R. durch *Splineapproximationen* erzielt, von denen ein spezieller Spline nachfolgend vorgestellt wird.

Splines sind lokale Modellierfunktionen, die an ihren Intervallgrenzen nahtlos zusammengesetzt werden müssen. Je nach dem Grad der Kontinuität unterscheidet man zwischen C_0-Kontinuität, C_1-Kontinuität und C_2-Kontinuität, abhängig davon, ob die Funktionswerte, ihre 1. Ableitungen oder gar auch ihre 2. Ableitungen übereinstimmen sollen.

Da die Vektorisierung von ausgedünnten Rasterpolygonen i.d.R. ein *zweidimensionales* Approximationsproblem darstellt, ist ferner eine geeignete Parametrisierung für die einzusetzende Splinefunktion zur Verfügung zu stellen. Eine häufig verwendete Parametrisierung benutzt die Bogenlänge t.

Es sei

$$z = z(x, y) = \text{const}$$

eine ebene Kurve im zweidimensionalen euklidischen Raum, für die die Wertepaare (x_0, y_0),(x_1, y_1), ...,(x_{n-1}, y_{n-1}) vorliegen, die die Lage der Mittelachsen der Rasterpolygone repräsentieren. Mit Hilfe der Sehnenabschnitte

$$t_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \qquad \forall i = 1, 2, 3, ..., n-1$$

und

$$t_k = \sum_{i=1}^{k} t_i \quad , \quad t_0 = 0, \qquad 1 \leq k \leq n-1$$

erhält man eine Aproximation für die Bogenlänge t, so daß das zweidimensionale Approximationsproblem in *zwei* eindimensionale Problemstellungen zerfällt.

$$z = z(x, y) = \text{const} \begin{cases} x(t) \\ y(t) \end{cases}$$

Die diskreten Wertepaare hierzu lauten: (t_0, x_0),(t_0, y_0), (t_1, x_1),(t_1, y_1), ... Wird nun jede Stützstelle (t_i, x_i) bzw. (t_i, y_i) als Knoten eines approximierenden Splines betrachtet, so läßt sich eine Splinefunktion definieren, die einerseits die Krümmung minimiert und andererseits der Unschärfe der *Beobachtungen* Rechnung tragen kann (D. Fritsch, 1991).

$$\int_t (g''(t))^2 dt = min \quad \text{unter} \quad \sum_{i=0}^{n-1} \frac{g(t_i) - x_i}{\sigma_{xi}} \leq S$$

Die erste Bedingung beinhaltet automatisch C_2-Kontinuität, wohingegen die letzte Bedingung die Glattheit der Splinefunktion $g(t)$ steuern kann. Dadurch können Unschärferelationen σ_{xi} - d.h. Abweichungen der geglätteten Funktion von den vorgegebenen Werten - eingehalten werden. Die Auflösung dieser Spline-Definiton unter der Nebenbedingung einer endlichen Genauigkeitsschranke führt zu einem *Variationsproblem*. Der resultierende Spline lautet

$$s(t) = p_{i,i+1}(t)$$

und wird erhalten aus den kubischen Polynomen

$$p_i(t) = a_i + b_i(t - t_i) + c_i(t - t_i)^2 + d_i(t - t_i)^3$$

Zur Berechnung der Koeffizienten a_i, b_i, ci, d_i können die in D. Fritsch (1991) angegebenen Formeln verwendet werden.

5.4 Beispiele zur Raster-/Vektorkonvertierung

Die hier angegebene Vorgehensweise zur Raster-/Vektorkonvertierung bestehend aus Binärisierung, Linienverfolgung und Vektorisierung soll an zwei Beispielen demonstriert werden. Es handelt sich dabei um einen Test der OEEPE – einer europäischen photogrammetrischen Vereinigung, die Forschungsergebnisse der Praxis zuführen möchte. Das erste Beispiel bietet ein sehr gutes synthetisches Rasterbild, welches aus Rauten verschiedener Größe besteht. Die Abbildung 12 gibt von rechts nach links die durchgeführten Verarbeitungsschritte wieder.

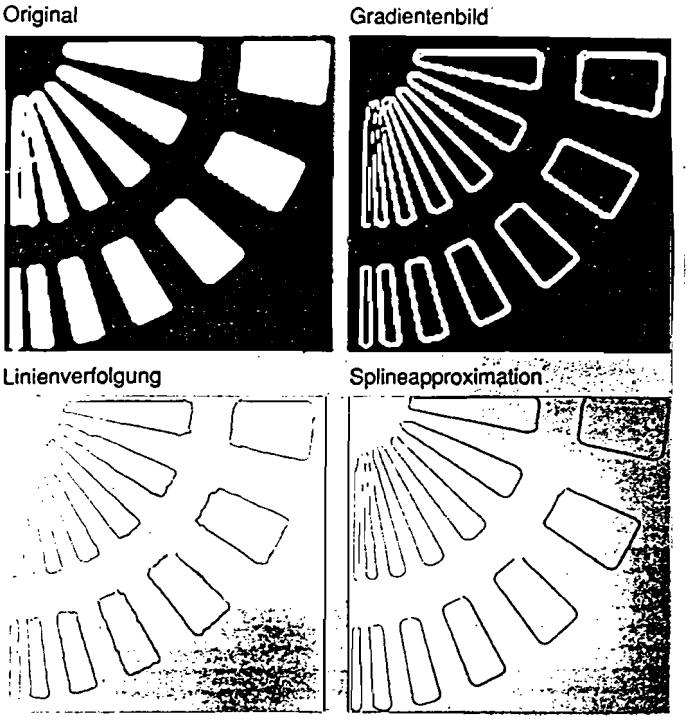

Abbildung 12: RVK am Beispiel eines synthetischen Bildes

Mit der Abbildung 13 ist eine Realszene wiedergegeben. Es handelt sich dabei um ein Ziegelsteinmauerwerk, bei dem die einzelnen Ziegel die zu vektorisierenden Elemente darstellen. Bereits die visuelle Interpretation der Realszene bereitet Schwierigkeiten, da die Fugen tlw. *ausgewaschen* erscheinen. Dies wird in dem Gradientenbild offensichtlich. Das Ergebnis der Splineapproximation unterscheidet sich nicht sehr von der Linienverfolgung, da weder extrapoliert noch stark geglättet worden ist. Ein weiterer Verarbeitungsschritt könnte nun diese beiden Randbedingungen berücksichtigen – dabei sollte jedoch gleichzeitig eine vollständige topologische Separation von Knoten und Kanten angestrebt werden.

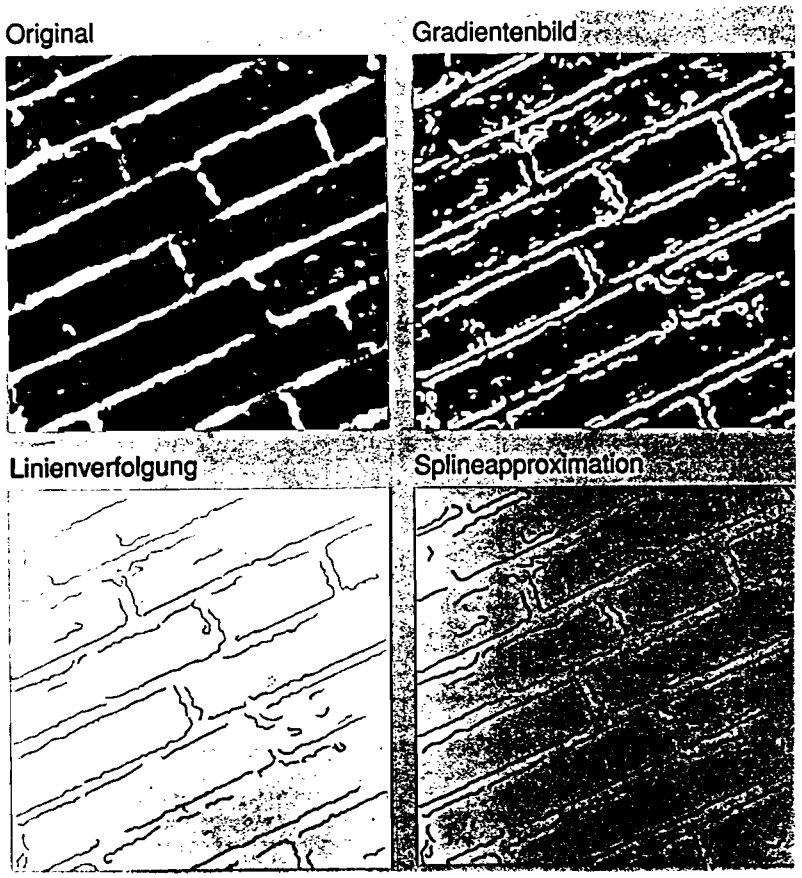

Abbildung 13: RVK am Beispiel einer Realszene

Diese beiden Beispiele belegen die pixelorientierte Vorgehensweise, die jedoch nie bestimmte Phänomene identifizieren kann. Somit tragen alle *bottom-up*- Prozesse lediglich zu einer mehr oder weniger zufriedenstellenden geometrischen Randwiedergabe bei – die in Geo-Informationssystemen so bedeutende Semantik (Thematik) kann auf diese Weise nicht extrahiert werden.

Abhilfe versprechen hier die *top-down*-Strategien, bei denen a priori Wissen in die Bildanalyse integriert wird. Jedoch ist der Entwicklungsstand noch nicht so weit fortgeschritten, um diese Verfahren in heutige GIS zu integrieren.

6 Schluß und Ausblick

Der vorliegende Beitrag zeigt algorithmische Aspekte in Geo-Informationssystemen auf, wie sie bei der Speicherung und Verarbeitung von Rasterdaten anfallen. Die zunehmende Integration der Rasterdaten in GIS hat zu den hybriden Geo-Informatiossystemen geführt, die eine neue, noch leistungsfähigere Generation darstellen. Jedoch können diese Systeme erst dann als ausgereift angesehen werden, wenn nicht nur die Vektor- und Sachdaten aktiv in die Analysen sondern auch die bisher überwiegend passiv genutzten Rasterdaten denselben Aktivierungsgrad erfahren.

Die vorgestellten Algorithmen konnten nur eine kleine Auswahl aus dem breiten Spektrum, wie es in der Geo-Informationsverarbeitung benötigt wird, repräsentieren. Ein schon seit langem bekanntes Problem – nämlich die kartographische *Generalisierung* von Rasterdaten – wurde überhaupt nicht angesprochen. Der Grund hierfür liegt nicht in der Minderung

der Aktualität, sondern in der Schwierigkeit, hierfür geeignete Regeln aufzustellen, die der Computer abarbeiten kann. Eine neue Arbeit ist mit E. Jäger (1990) gegeben, der sich speziell der Verdrängung und kartographischen Symbolisierung gewidmet hat. Das Problem der Generalisierung ist nicht mehr nur ein kartographisches Problem; vielmehr ist es mittlerweile zu einem generellen Problem in der raumbezogenen Datenverarbeitung geworden. Der Anwender wünscht sich eine Verfügbarkeit seiner teuren und kostbaren Daten zu verschieden *Aggregationsstufen*, d.h. daß neben der untersten Ebene mit der 1:1 Abbildung nach andere Ebenen mit Abildungsmaßstäben $1 : m_i$ ($m_i > 1$) automatisch generiert werden können. Die Algorithmik hierzu ist noch unterentwickelt – ein pragmatischer Weg besteht im Aufbau von verschiedenen Geo-Informationssystemen, die diesen Aggregationen Rechnung tragen.

Ebenso konnte in diesem Beitrag der *Unschärfeaspekt* der raumbezogenen Daten nicht behandelt werden, da es auch hier an geigneten Strategien fehlt. Die dem Vermessungsingenieur so vertraute Fehlerrechnung kann nur einen geringen Beitrag leisten, da die Aussage schnell verfügbar sein und viele Belange berücksichtigen sollte. Aus diesen Gründen muß künftig der Algorithmik in Geo-Informationssystemen mindestens der gleiche Entwicklungsaufwand zukommen, wie dies bei den Datenbanken der Fall ist.

7 Literatur

Baltsavias, E.P., Beyer, H.A., Fritsch, D., Lenz, R. (1990): Fundamentals of Real-Time Photogrammetry. Lecture Notes, Inst. Geod., Photogr., ETH Zurich, Zurich.

Bill, R., Fritsch, D. (1991): Grundlagen der Geo-Informationssysteme, Band 1. Wichmann, Karlsruhe.

Bill, R., Fritsch, D. (1992): Grundlagen der Geo-Informationssysteme, Band 2. Wichmann, Karlsruhe.

Fritsch, D. (1991): Raumbezogene Informationssysteme und Digitale Geländemodelle. Deutsche Geod. Kommission, Reihe C, Nr. 369, München.

Heipke, C. (1991): Integration von Bildzuordnung, Punktbestimmung, Oberflächenrekonstruktion und Orthprojektion innerhalb der digitalen Photogrammetrie. Deutsche Geod. Kommission, Reihe C, Nr. 366, München.

Illert, A. (1990): Automatische Erfassung von Kartenschrift, Symbolen und Grundrißobjekten aus der Deutschen Grundkarte 1:5000. Wiss. Arb. Fachricht. Verm. Wesen, Univ. Hannover, Nr. 166, Hannover.

Jäger, E. (1990): Untersuchungen zur kartographischen Symbolisierung und Verdrängung im Rasterdatenformat. Wiss. Arb. Fachricht. Verm. Wesen, Univ. Hannover, Nr. 167, Hannover.

Kern, A. (1990): Untersuchungen zur Raster-/Vektorkonvertierung innerhalb der Quadtree-Organisation von thematischen Rasterdaten. Diplomarbeit Techn. Univ. München (nicht

veröffentlicht).

Moser, B. (1990): Untersuchungen zur Segmentierung von digitalen Bildern. Diplomarbeit Techn. Univ. München (nicht veröffentlicht).

Newman, W.M., R.F. Sproull (1984): Grundzüge der interaktiven Computergraphik. McGraw-Hill, Hamburg.

Tomlinson, R.F. (1972): 1972 Geographical Data Handling (Symposium Edition). UNESCO/IGU Second Symposium Geographical Information Systems. Ottawa.

Yang, H. (1991): Zur Integration von Vektor- und Rasterdaten in Geo-Informationssystemen. Deutsche Geod. Kommission, Reihe C, München.