GI-Fachtagung “Datenbanken in Büro, Technik und Wissenschaft” (BTW ‘97), *Informatik Aktuell*, Springer, 1997.

Section Coding:
Ein Verfahren zur Ähnlichkeitssuche in CAD-Datenbanken

Stefan Berchtold, Daniel A. Keim, Hans-Peter Kriegel

Institut für Informatik, Universität München
Oettingenstr. 67, D-80538 München
e-mail: {berchtol, keim, kriegel}@informatik.uni-muenchen.de

Zusammenfassung

Ausgehend von einer konkreten Problemstellung, die sich im Rahmen einer Kooperation mit einem industriellen Partner ergibt, untersuchen wir in diesem Artikel, was geometrische Ähnlichkeit von CAD-Objekten (in unserem Fall polygonalen Objekten) bedeutet. Aus der Untersuchung leiten wir verschiedene Eigenschaften von Ähnlichkeit ab und definieren die Begriffe Ähnlichkeitsmaß und Ähnlichkeitsmetrik sowie deren Eigenschaften. Dann stellen wir ein neues Verfahren zur Ähnlichkeitssuche vor, Section Coding genannt, das im Gegensatz zu existierenden Verfahren robust gegen Veränderungen der Polygonkontur und dennoch effizient ist. Die Grundidee unseres Verfahrens ist, Polygone als ähnlich zu bewerten, wenn sie ähnliche Flächenproportionen haben. Section Coding kodiert die Flächenproportionen der Polygone als Feature-Vektoren, die dann in einer multidimensionalen Indexstruktur gespeichert werden. Die Grundidee der Feature-Transformation von Section Coding ist, den Umkreis der zu vergleichenden Polygone zu bestimmen, diesen in k Sektoren aufzuteilen und jeweils den Flächenanteil zu berechnen, der innerhalb eines Sektors liegt. Die hierbei entstehenden k Flächenanteile stellen einen k-dimensionalen Feature-Vektor dar, der in der Indexstruktur gespeichert wird. Section Coding ist translations-, skalierungs-, spiegelungs- und bedingt auch rotationsinvariant. Anhand einer Implementierung von Section Coding im Rahmen des Datenbanksystems **S3** zeigen wir die Effektivität und Effizienz unseres Verfahrens.

Schlüsselwörter: Ähnlichkeitssuche, CAD-Datenbanken, Geometrie-basierte Ähnlichkeit

1. Einleitung

In diesem Artikel befassen wir uns mit der Suche von ähnlichen Bauteilen in einer Menge von Polygonen. Das Problem tritt im Rahmen einer Kooperation auf, die wir mit einem Zulieferer der deutschen Automobilindustrie haben. Der Betrieb produziert Teile, sogenannte ‘Clipse’, die zur Befestigung und Verbindung anderer Bauteile eingesetzt werden. Die Gesamtzahl verschiedener Teile, die dieser Zulieferer produziert, ist sehr groß, da bereits für ein einziges Fahrzeug einige Hundert verschiedene Clipse benötigt werden. Das Ziel der Kooperation ist die Reduktion der Teilevielfalt und damit verbunden eine Reduktion von Kosten. Ein wichtiges Kriterium für die Wiederverwendbarkeit eines Teils ist die geometrische Ähnlichkeit des Teils mit bereits konstruierten Teilen in der CAD-Datenbank. Werden derartige ähnliche Teile gefunden, so können Zeit und Kosten für die Konstruktion und prototypische Realisierung des Teils erheblich reduziert werden. Die Suche ähnlicher Bauteile stellt damit eine wichtige Möglichkeit zur Kostenreduktion dar. Das Problem der geometrischen Ähnlichkeitssuche ist jedoch nicht auf die beschriebene Anwendung beschränkt, sondern tritt in ähnlicher Form auch in zahlreichen anderen Anwendungsgebieten wie zum Beispiel Molekül Docking und Molekül Design, Computertomographie sowie in den verschiedensten Bereichen der Mustererkennung auf.

Die Suche ähnlicher Bauteile ist ein schwieriges Problem. Der Ansatz zur Lösung des Problems, der bisher in der Industrie verwendet wird, basiert auf sogenannten Sachmerkmalsleisten. Sachmerkmalsleisten sind Vektoren, deren Komponenten die wichtigsten Eigenschaften der Bauteile beschreiben. Beispiele für derartige Eigenschaften sind Funktionalität, Material, Hitzebeständigkeit, Kraftübertragung, etc. Die Erfahrung zeigt jedoch, daß die Suche mit Hilfe von Sachmerkmalsleisten nicht ausreichend ist, da sie auf einer beschränkten, relativ kleinen Anzahl von Eigenschaften beruht und die geometrische Form im allgemeinen nicht oder nur unzureichend berücksichtigt. Um bessere Ergebnisse zu erzielen, ist es notwendig, die Geometrie der Bauteile, die aus den CAD-Modellen verfügbar ist, vollständig zu berücksichtigen.

In diesem Artikel analysieren wir ausführlich, was geometrische Ähnlichkeit von Polygonen bedeutet. Aus der Analyse leiten wir verschiedene Eigenschaften von Ähnlichkeit ab und definieren die Begriffe Ähnlichkeitsmaß und Ähnlichkeitsmetrik sowie deren Eigenschaften (siehe Kapitel 2). Dann stellen wir ein neues Verfahren zur Ähnlichkeitssuche, genannt Section Coding vor, das im Gegensatz zu existierenden Verfahren robust gegen Veränderungen der Polygonkontur und dennoch effizient ist (siehe Kapitel 3). Unser Verfahren bewertet Polygone als ähnlich, wenn sie ähnliche Flächenproportionen haben. Das Verfahren ist translations-, skalierungs-, spiegelungs- und bedingt auch rotationsinvariant. Mit Hilfe einer Implementierung des Verfahrens im Rahmen des Datenbanksystems **S3** zeigen wir die Effektivität und Effizienz unseres Verfahrens.

2. Ähnlichkeitssuche

In diesem Kapitel wollen wir genauer betrachten, was 'geometrische Ähnlichkeit' bedeutet. Untersucht man die in verschiedenen Anwendungsgebieten auftretenden Ähnlichkeitsbegriffe, so wird schnell klar, daß diese recht unterschiedlich sind und im allgemeinen nicht durch ein einziges Ähnlichkeitsmaß dargestellt werden können. Man kann also nicht von *der* geometrischen Ähnlichkeit sprechen, sondern nur von *einer* geometrischen Ähnlichkeit. Dies gilt insbesondere für technisch naturwissenschaftliche Anwendungsgebiete wie die Suche ähnlicher Bauteile oder die Suche bösartiger Tumorzellen. Im folgenden werden ausgehend vom menschlichen Ähnlichkeitsbegriff (vgl. Abschnitt 2.1) einige wichtige Eigenschaften von Ähnlichkeitsmaßen herausgearbeitet (vgl. Abschnitt 2.2).

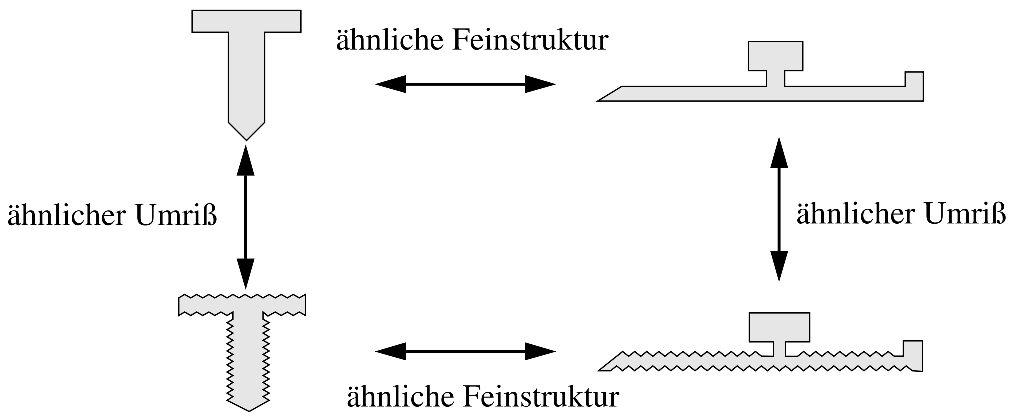


Abbildung 1: Widersprüchliche Ähnlichkeitsbegriffe

2.1 Menschliche Ähnlichkeit

Um Polygone als ähnlich zu bewerten, wenden Menschen unterschiedliche Kriterien an, wobei in vielen Fällen das Hintergrundwissen aus dem jeweiligen Anwendungsgebiet eine entscheidende Rolle spielt. Für unsere Betrachtungen beschränken wir uns auf Ähnlichkeitsbegriffe, die ausschließlich auf den geometrischen Eigenschaften der betrachteten Objekte basieren. Allerdings gibt es auch bei einer Beschränkung auf die geometrischen Eigenschaften immer noch eine Vielzahl verschiedener Ähnlichkeitsbegriffe. So können zwei Polygone als ähnlich betrachtet werden, wenn sie global ungefähr denselben Umriß haben, wenn sie die gleichen Flächenproportionen haben, wenn sie in lokalen Details gleich sind, wenn sie die gleiche Komplexität bzw. den gleichen Detailierungsgrad haben, oder wenn Teile des Umrisses die gleiche Krümmung aufweisen. Eine wesentliche Beobachtung ist, daß intuitive Ähnlichkeitsbegriffe sehr unterschiedliche Arten von Ähnlichkeit umfassen. Es kann sogar sein, daß Ähnlichkeitsbegriffe widersprüchlich sind (vgl. Abbildung 1). Daher ist es im allgemeinen unmöglich, eine formale Definition von Ähnlichkeit anzugeben, die alle in der Praxis auftretenden Ähnlichkeitsbegriffe umfaßt. Oft gibt es auch kein objektives Bewertungskriterium für die Qualität eines Ähnlichkeitsmaßes oder eines Algorithmus zur Berechnung von Ähnlichkeit. In vielen Fällen (vgl. beispielsweise [FBF+ 94] oder [Jag 91]) wird deshalb die Effektivität der Verfahren mit Hilfe von Testanwendern evaluiert, die eine Datenbank von Polygonen nach ähnlichen Polygonen durchsuchen und eine subjektive Rangfolge der Polygone erstellen, die dann mit der Rangfolge verglichen wird, die durch den Algorithmus berechnet wurde.

2.2 Eigenschaften von Ähnlichkeitsmaßen

Im folgenden werden einige Eigenschaften praktisch relevanter Ähnlichkeitsbegriffe behandelt. Dabei gibt es eine Reihe von Eigenschaften, die für jeden praktisch relevanten Ähnlichkeitsbegriff gelten, und andere, die nur für bestimmte Ähnlichkeitsbegriffe gelten. Für die formale Beschreibung von Ähnlichkeit werden zuerst Polygone definiert:

Definition 1: *Polygon als zyklische Sequenz von Kanten*

Ein Polygon p ist definiert als eine zyklische Sequenz von m_p Kanten $\left\{ \overrightarrow{e_0^p}, \ldots, \overrightarrow{e_{m_p-1}^p} \right\}$ und m_p Stützstellen $\left\{ \overrightarrow{v_0^p}, \ldots, \overrightarrow{v_{m_{p-1}}^p} \right\}$. Sei A^p die Fläche des Polygons p.

Man beachte, daß die i-te Kante $\overrightarrow{e_i^p}$ an der i-ten Stützstelle $\overrightarrow{v_i^p}$ beginnt und an der (i+1)-ten Stützstelle $\overrightarrow{v_{i+1}^p}$ endet. Im folgenden wird Ähnlichkeit von Polygonen als Maß auf dem Raum der Polygone definiert:

Definition 2: *Ähnlichkeitsmaß*

Sei P der Raum der Polygone. Dann ist ein Ähnlichkeitsmaß definiert als eine Abbildung S: $P \times P \to \Re$, die zwei Polygone auf eine reelle Zahl abbildet. Ein Ähnlichkeitsmaß hat die folgenden Eigenschaften:

1. Positivität: $\forall(p, q \in P)$: $S(p, q) \geq 0$
2. Reflexivität: $\forall(p, q \in P)$: $(p = q) \Rightarrow S(p, q) = 0$
3. Symmetrie: $\forall(p, q \in P)$: $S(p, q) = S(q, p)$

Die Positivitäts- und Symmetrieeigenschaft gelten offensichtlich für jeden praktisch relevanten Ähnlichkeitsbegriff. Die Reflexivität muß gelten, damit identische Polygone maximal ähnlich sind. Gelten zusätzlich die Identitätseigenschaft und die Dreiecksungleichung, so sind insgesamt die Metrikeigenschaften erfüllt, so daß wir von einer *Ähnlichkeitsmetrik* sprechen können.

Definition 3: *Ähnlichkeitsmetrik*

Eine Ähnlichkeitsmetrik ist ein Ähnlichkeitsmaß S, das zusätzlich die folgenden Eigenschaften erfüllt:

1. Identität: $\forall(p, q \in P)\colon \quad S(p, q) = 0$ gdw. $p = q$
2. Dreiecksungleichung: $\forall(p, q, r \in P)\colon \quad S(p, q) \leq S(p, r) + S(r, q)$.

Eine weitere Eigenschaft aller praktisch relevanter Ähnlichkeitsbegriffe ist die Stetigkeit. Mit Stetigkeit ist die Tatsache gemeint, daß kleine Veränderungen in einem Polygon auch kleine Änderungen des Ähnlichkeitsmaßes bewirken.

Definition 4: *Stetigkeit*

Ein Ähnlichkeitsmaß S ist *stetig,* wenn für alle Polygone $p, q \in P$ gilt: Wenn p' aus p durch ‘kleine’ Veränderungen entstanden ist, dann gilt, daß p' ‘fast’ so ähnlich zu q ist wie p zu q.

Formaler: $$\forall(p', p, q \in P)\colon \quad \lim_{p' \to p} S(p', q) = S(p, q)$$

Der Grenzübergang in Definition 2 ist möglich, da Polygone topologisch äquivalent sind und daher stetig ineinander deformiert werden können. Beim Grenzübergang auf Polygonen nähern sich die Konturen der Polygone einander an. Abbildung 2 veranschaulicht die Stetigkeitseigenschaft. Jeweils benachbarte Polygone der Reihe *(p, p’, p”, ..., p””’)* in Abbildung 2 unterscheiden sich nur geringfügig. Man könnte nun noch weitere Zwischenschritte einfügen, so daß eine Reihe (p, x_1, x_2, ..., p', ..., p'', ..., x_n, *p””’)* entsteht, in der die Unterschiede zwischen benachbarten Polygonen beliebig klein sind und daher nicht mehr wahrgenommen werden können.

Aus der Stetigkeit folgt, daß Ähnlichkeitsmaße nicht transitiv sein können. Da zwei beliebige Polygone stetig ineinander deformiert werden können, würden sie, falls die Transitivität gilt, zueinander ähnlich sein, insbesondere die Polygone p und p”’’’ in Abbildung 2 wären in diesem Fall zueinander ähnlich. Da dies für beliebige Polygone gilt, wären im transitiven

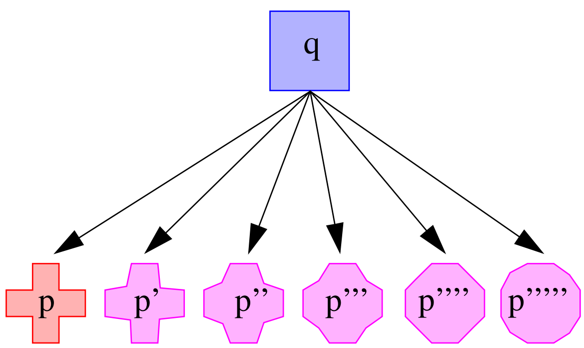


Abbildung 2: Stetigkeit von Ähnlichkeit

Abschluß alle Polygone einander ähnlich. Dies impliziert auch, daß Ähnlichkeit - im Gegensatz zum allgemeinen Sprachgebrauch - keine Äquivalenzrelation ist, woraus wiederum folgt, daß sich im allgemeinen keine Äquivalenzklassen bilden lassen.

Zusätzlich zu den bisher eingeführten Eigenschaften, können Ähnlichkeitsbegriffe einige weitere Eigenschaften haben. Hierzu gehört insbesondere die Invarianz gegenüber bestimmten affinen Abbildungen. Ist ein Ähnlichkeitsbegriff invariant gegenüber einer Abbildung, so bedeutet dies, daß der Ähnlichkeitsabstand zwischen zwei beliebigen Polygonen nicht durch die Anwendung der Abbildung auf eines der Polygone beeinflußt wird. Formaler ausgedrückt:

Definition 5: *Invarianzen*

Eine Ähnlichkeitsmetrik S heißt invariant gegenüber einer Abbildung I, wenn

$$\forall (p, q \in P):\quad S(p, q) = S(I(p), q) = S(p, I(q))$$

Folgende affine Abbildungen sind im Zusammenhang mit Ähnlichkeitsmaßen von Interesse:

1. Translation: $I_t(p) = q;\ v_i^q = v_i^p + \begin{bmatrix} u \\ w \end{bmatrix};\quad p, q \in P, u, w \in \Re, 0 \le i < m_p$
2. Rotation: $I_r(p) = q:\ v_i^q = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \cdot v_i^p;\quad p, q \in P, \alpha \in \Re, 0 \le i < m_p$
3. Skalierung: $I_s(p) = q:\ v_i^q = \begin{bmatrix} u & 0 \\ 0 & u \end{bmatrix} \cdot v_i^p;\quad p, q \in P, u \in \Re, 0 \le i < m_p$
4. Spiegelung[1]: $I_m(p) = q:\ \dot{v}_i^q = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot v_i^p;\quad p, q \in P, 0 \le i < m_p$
5. Scherung: $I_m(p) = q:\ \dot{v}_i^q = \begin{bmatrix} 1 & w \\ u & 1 \end{bmatrix} \cdot v_i^p;\quad p, q \in P, u, w \in \Re, 0 \le i < m_p$

Welche affinen Abbildungen konkret von Interesse sind, hängt vom Anwendungsbereich ab. Im Bereich der Bildverarbeitung treten beispielsweise durch die zentralperspektivische Projektion von dreidimensionalen Objekten auf die zweidimensionale Bildebene Scherungen auf. Daher wird hier ein Ähnlichkeitsbegriff benutzt, der invariant gegenüber Scherungen ist. In anderen Anwendungsfeldern sind Ähnlichkeitsbegriffe selten, die invariant gegenüber Scherungen sind. Dies liegt daran, daß die Abbildungen 1. bis 4. winkelerhaltend sind, während sich bei Scherungen die Winkel zwischen den Kanten der Polygone verändern. Wegen der geringen praktischen Relevanz von Scherungen in unserem Anwendungsbereich, werden wir im Rest dieses Artikels Invarianz gegenüber Scherungen nicht weiter betrachten.

Man beachte, daß Ähnlichkeitsbegriffe, die invariant gegenüber einer Abbildung I sind, keine Ähnlichkeitsmetriken nach Definition 3 sind, da sie die Identitätseigenschaft verletzen. Dies liegt daran, daß alle Polygone, die durch Anwendung der Abbildung I voneinander entstanden sind, den Abstand 0 haben. Faßt man jedoch alle diese Polygone als Repräsentanten einer Äquivalenzklasse auf, so bleiben die Eigenschaften aus Definitionen 2 bis 4 gewahrt. Die abso-

1. Dies entspricht einer Spiegelung an der positiven X-Achse.

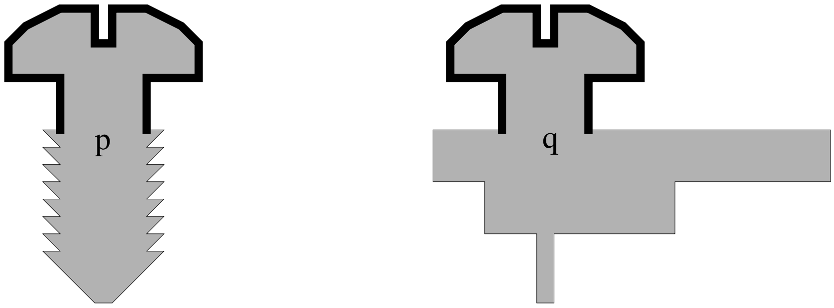


Abbildung 3: Partielle Ähnlichkeit

lute Lage eines Polygons ist allerdings nur für sehr spezielle Anwendungsgebiete wichtig. Deshalb wird im Rahmen dieses Artikels die absolute Lage eines Polygons nicht weiter betrachtet.

Ähnlichkeit in einem menschlichen Sinne ist - in gewissen Schranken - invariant gegenüber allen affinen Transformationen[1]. Da außerdem relativ leicht von invarianter Ähnlichkeit auf variante Ähnlichkeit geschlossen werden kann, ist ein Ziel des Entwurfs neuer Algorithmen, invariant gegenüber möglichst vielen affinen Abbildungen zu sein. Algorithmisch gesehen ist es jedoch schwierig, Verfahren zu entwickeln, die invariant gegenüber möglichst vielen affinen Abbildungen sind, da sich mit jeder neuen Invarianz die Dimension des zugehörigen Suchraums erhöht.

In technischen Anwendungsfeldern treten auch häufig Ähnlichkeitsbegriffe auf, die Polygone aufgrund ähnlicher Abschnitte als ähnlich bezeichnen. Man kann deshalb zwischen *globaler Ähnlichkeit*, bei der jede beliebige Veränderung an einem Polygon den Grad an Ähnlichkeit zu anderen Polygonen verändert, und *partieller Ähnlichkeit*, bei der Abschnitte der betrachteten Polygone zur Bestimmung der Ähnlichkeit ausreichen, unterscheiden (vgl. Abbildung 3). Man beachte, daß für ein partielles Ähnlichkeitsmaß nicht automatisch die oben eingeführten Eigenschaften gelten (beispielsweise gilt die Identitätseigenschaft nicht, da $S(p, q) = 0$ obwohl $p \neq q$).

3. Section Coding

Im folgenden stellen wir ein neues Verfahren zur globalen Ähnlichkeitssuche - genannt *Section Coding* - vor, das im Gegensatz zu früheren Verfahren robust und aufgrund seiner guten Laufzeitkomplexität für praktische Anwendungen geeignet ist. Bevor wir unser Verfahren beschreiben, wollen wir zunächst einen kurzen Überblick über verwandte Arbeiten auf dem Gebiet der Ähnlichkeitssuche geben.

3.1 Verwandte Arbeiten

In der Literatur werden verschiedene Verfahren zur Ähnlichkeitssuche vorgestellt. Diese entstammen den Bereichen algorithmische Geometrie, Mustererkennung und Datenbanksysteme (Multimedia- und Zeitreihen-Datenbanken). Im Bereich algorithmische Geometrie wird Ähnlichkeit durch einfache, theoretisch gut untersuchte Ähnlichkeitsmaße (bzw. Ähnlichkeitsmetriken) wie z.B. Hausdorff-Abstand oder Fréchet-Abstand definiert ([AMWW 88], [AB 92] oder

1. Trotzdem kann es für den Menschen manchmal auch schwierig sein, ein ähnliches rotiertes, skaliertes und verschobenes Polygon noch als solches zu erkennen.

[ABB 92]). Aus Sicht der algorithmischen Geometrie sind zwei Polygone ähnlich, wenn ihr Abstand bezüglich dieser Metriken klein ist. Hierdurch hat man eine saubere theoretische Grundlage für die Untersuchung von Algorithmen. Die von diesen Metriken implizierten Ähnlichkeitsbegriffe sind für unsere Anwendungen jedoch ungeeignet, da die vorgeschlagenen Ähnlichkeitsmaße sehr weit von der menschlichen Vorstellung von Ähnlichkeit entfernt sind und die praktische Einsetzbarkeit derartiger Algorithmen in den meisten Fällen nicht im Vordergrund der Überlegungen steht. Außerdem geht es in der algorithmischen Geometrie hauptsächlich um den 1:1 Vergleich von Polygonen, wohingegen wir uns mit der 1:n Ähnlichkeitssuche in sehr großen Datenbanken beschäftigen. Im Bereich der Mustererkennung geht es darum, Objekte in einer Szene zu erkennen [MG 93, MG 95]. Da die Menge der möglichen Objekte dabei in der Regel a priori eingeschränkt ist, kann man Modelle für jedes dieser Objekte generieren und diese dann mit den Objekten in der Szene vergleichen. Beim Vergleich ist es in den meisten Fällen ausreichend, ein signifikantes Detail zu erkennen, um auf Gleichheit der betrachteten Objekte zu schließen. In der Mustererkennung gibt es zahlreiche, relativ spezialisierte Ansätze, die zum Teil sogar invariant gegenüber Scherungen sind. Ein Beispiel für ein derartiges Verfahren ist die von Wallace und Wintz vorgestellte Fourier-basierte Ähnlichkeitssuche [WW 80]. Auch in anderen Anwendungen gibt es Problemstellungen, die mit der Mustererkennung verwandt sind. Ein Beispiel ist das von Helmer-Citterich und Tramontano entwickelte Verfahren zum Docking von Proteinen [HT 94].

Auch im Bereich der Datenbanksysteme gibt es eine Reihe von Ansätzen zur Ähnlichkeitssuche. Ein Beispiel ist die Ähnlichkeitssuche in Multimedia-Datenbanken, wo es unter anderem auch um die Suche ähnlicher Objekte in Bildern geht. Die Problemstellung ist dabei jedoch anders als in unserer Anwendung, da die Beziehungen zwischen den Objekten eines Bildes im allgemeinen wichtiger sind als Ähnlichkeit zwischen den Objekten [FBF+ 94] und Invarianzen in der Regel nicht betrachtet werden [PF 94]. Ein weiteres verwandtes Problem ist die Ähnlichkeitssuche in Zeitreihen-Datenbanken. In [AFS 93] wird eine effiziente Methode zur Ähnlichkeitssuche in eindimensionalen Sequenzdaten vorgeschlagen. Der Ansatz verwendet eine Fourierkodierung der Daten und bestimmt die Ähnlichkeit zweier Sequenzen als ihren euklidischen Abstand im Fourierraum. [FRM+ 94] erweitert diese Idee im Hinblick auf partielle Ähnlichkeit von Teilsequenzen und [ALSS 95] berücksichtigt zusätzlich auch Rauschen und Skalierungsinvarianz. Die bisher beschriebenen Ansätze betrachten jedoch nur eindimensionale Zeitreihendaten.

Ein Verfahren, das die Ähnlichkeitssuche auf zweidimensionalen Polygonen unterstützt, wurde von Jagadish vorgeschlagen [Jag 91]. Das Verfahren beruht auf der Beobachtung, daß Menschen Polygone als ähnlich empfinden, wenn diese ähnliche Flächenproportionen haben. Jagadish geht von achsenparallelen Polygonen aus. Diese können durch wiederholte Addition (Vereinigung)

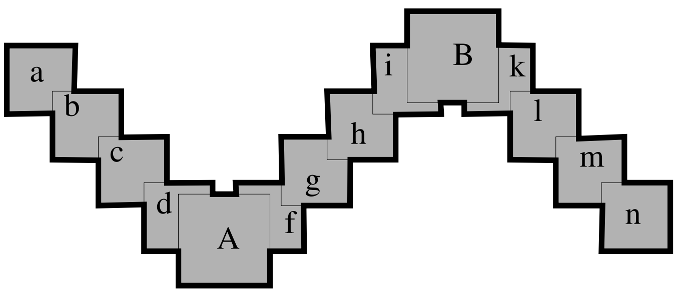


Abbildung 4: Probleme des Ansatzes nach Jagadish

von Rechtecken verschiedener Größe dargestellt werden. Jagadish kodiert die Menge dieser Rechtecke in Vektoren, die in einer multidimensionalen Indexstruktur gespeichert werden. Die Ähnlichkeit zweier Objekte ist als euklidischer Abstand der Vektoren definiert. Hauptproblem des Ansatzes nach Jagadish ist, daß die Darstellung mit achsenparallelen Rechtecken nicht robust ist. Das heißt, die Darstellung ist nicht eindeutig und kleine Veränderungen am Umriß des Objekts können große Veränderungen des Vektors verursachen. Außerdem funktioniert das Verfahren nur für achsenparallel-rechtwinklig-begrenzte Objekte und ist nicht rotationsinvariant. Ein Beispiel für die Probleme des Jagadish-Ansatzes ist in Abbildung 4 dargestellt: da sowohl die beiden Rechtecke 'A' und 'B' als auch die Recktecke 'a' bis 'n' jeweils gleich groß sind, existieren $2 \cdot 12!$ Möglichkeiten, die Recktecke nach den Kriterien von Jagadish anzuordnen. Das Verfahren wählt aus diesen Möglichkeiten einige aus, die dann im Index gespeichert werden. Verändert sich das Polygon minimal, so werden einzelne Rechtecke größer als andere. Daher verändert sich die Reihenfolge der Kodierung und damit der zugehörige Vektor entscheidend.

3.2 Das Prinzip Index-basierter Ähnlichkeitssuche

Das von uns neu entwickelte Verfahren ist eine Weiterentwicklung des Jagadish-Ansatzes. Es funktioniert aber für beliebige Polygone und, wie wir später sehen werden, ist es auch zu einem bestimmten Grad rotationsinvariant. Wie beim Ansatz von Jagadish verwendet unsere Section Coding Technik einen Index-basierten Ansatz, das heißt, aus der Polygon-Datenbank wird eine Menge von Eigenschaften extrahiert, die dann in einen sogenannten Feature-Vektor transformiert werden. Die Feature-Vektoren werden zum Zeitpunkt des Indexaufbaus in eine multidi-

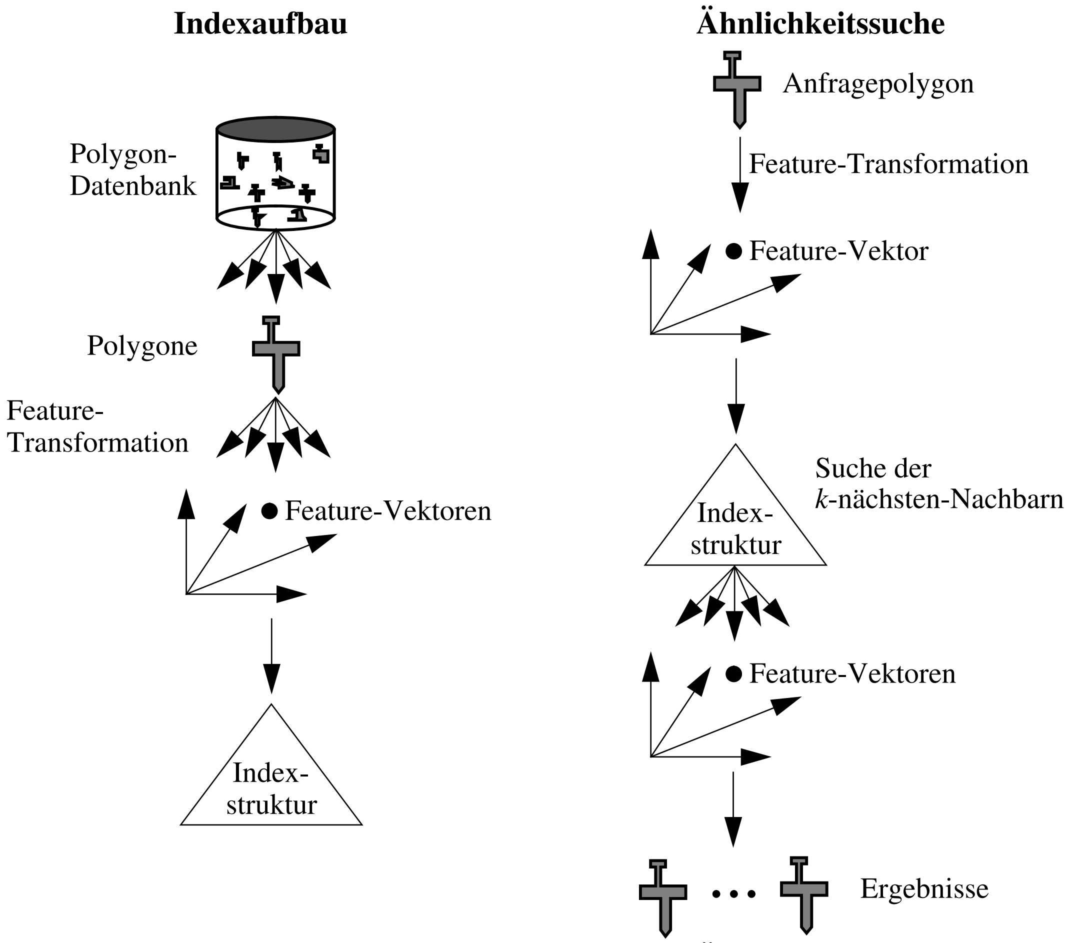


Abbildung 5: Das Prinzip Index-basierter Ähnlichkeitssuche

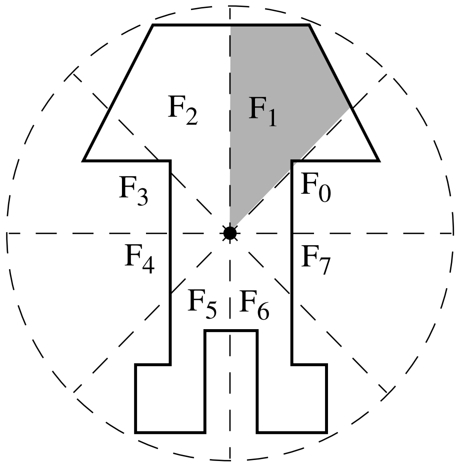


Abbildung 6: Section Coding (*k*=8)

mensionale Indexstruktur, wie zum Beispiel den R*-Baum [BKSS 90], BANGFile [Fre 87], KDB-Baum [Rob 81] oder X-Tree [BKK 96], eingefügt. Um eine Ähnlichkeitsanfrage zu bearbeiten, wird das Anfragepolygon ebenfalls in einen Feature-Vektor transformiert. Mit diesem Vektor wird eine Suche der *k* nächsten Nachbarn durchgeführt. Als Ergebnis der Suche erhält man eine Menge von Feature-Vektoren, deren euklidischer Abstand zum Feature-Vektor des Anfragepolygons klein ist. Diese gehören zu Polygonen aus der Datenbank, die ähnliche Eigenschaften wie das Anfragepolygon besitzen und daher ähnlich zum Anfragepolygon sind. Abbildung 5 zeigt einen Überblick über den Ablauf der Index-basierten Ähnlichkeitssuche.

3.3 Die ‘Section Coding’ Technik

Die Section Coding Technik basiert auf dem Ähnlichkeitsbegriff: ‘Polygone sind ähnlich, wenn sie ähnliche Flächenproportionen haben’. Die Grundidee von Section Coding ist, den Umkreis um den Schwerpunkt der zu vergleichenden Polygone zu bestimmen, diesen in *k* Sektoren aufzuteilen und jeweils den Flächenanteil zu berechnen, der innerhalb eines Sektors liegt (vgl. Abbildung 6). Die hierbei entstehenden *k* Flächenanteile stellen einen *k*-dimensionalen Feature-Vektor dar. Man beachte, daß es mehrere Möglichkeiten gibt, den Schwerpunkt eines Polygons zu definieren. Für unser Verfahren ist es günstig, den Schwerpunkt des Polygons als Umkreismittelpunkt zu verwenden. Um die Section Coding Technik genauer zu erläutern, benötigen wir einige Definitionen. Um die Flächenanteile eines Polygons definieren zu können, müssen wir zunächst den Umkreismittelpunkt bestimmen. Dieser entspricht dem Schwerpunkt des Polygons.

Definition 6: *Umkreismittelpunkt*

Sei $\Delta_p = \{\delta_0, \ldots, \delta_{D-1}\}$ die Menge der Dreiecke der Triangulierung von *p*. Dann ist der Umkreismittelpunkt von *p* definiert als

$$\overrightarrow{um_p} = \sum_{i=0}^{D-1} \frac{\overrightarrow{sp(\delta_i)} \cdot |\delta_i|}{A^p},$$

wobei $|\delta_i|$ die Fläche und $\overrightarrow{sp(\delta_i)}$ den Schwerpunkt des Dreiecks δ_i bezeichnen.

Für die Berechnung der Flächeninhalte benötigen wir als nächstes den Winkel zwischen der positiven X-Achse und dem Vektor vom Umkreismittelpunkt $\overrightarrow{um_p}$ zu einer Stützstelle $\overrightarrow{v_i^p}$.

Definition 7: *Winkel einer Stützstelle*

Der Winkel einer Stützstelle $\overrightarrow{v_i^p}$ eines Polygons p ist definiert als

$$winkel(\overrightarrow{v_i^p}) = \begin{cases} \operatorname{acos}\left(\left\langle \dfrac{\overrightarrow{v_i^p} - \overrightarrow{um_p}}{\left\| \overrightarrow{v_i^p} - \overrightarrow{um_p} \right\|}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\rangle\right) & falls \;\; det\left(\begin{bmatrix} \overrightarrow{v_i^p} & \overrightarrow{um_p} \end{bmatrix}\right) \geq 0 \\ 2\pi - \operatorname{acos}\left(\left\langle \dfrac{\overrightarrow{v_i^p} - \overrightarrow{um_p}}{\left\| \overrightarrow{v_i^p} - \overrightarrow{um_p} \right\|}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\rangle\right) & sonst \end{cases},$$

wobei $\langle a, b \rangle$ das kanonische Skalarprodukt zwischen den Vektoren a und b ist.

Bevor wir nun den Flächeninhalt innerhalb eines bestimmten Umkreissektors beschreiben können, müssen wir zunächst den Umkreissektor definieren.

Definition 8: *Umkreissektor*

Der i-te Umkreissektor $S^p[i]$ eines Polygons p ist definiert durch seine rechte und linke Begrenzungslinie $\overrightarrow{S_r^p[i]}$ und $\overrightarrow{S_l^p[i]}$:

$$\overrightarrow{S_r^p[i]} = \lambda \cdot \left(\begin{bmatrix} \cos\left(i \cdot \frac{2\pi}{k}\right) & -\sin\left(i \cdot \frac{2\pi}{k}\right) \\ \sin\left(i \cdot \frac{2\pi}{k}\right) & \cos\left(i \cdot \frac{2\pi}{k}\right) \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) + \overrightarrow{um_p}, \text{ wobei } 0 \leq i < k,\ \lambda \geq 0 \quad \text{und}$$

$$\overrightarrow{S_l^p[i]} = \mu \cdot \left(\begin{bmatrix} \cos\left((i+1) \cdot \frac{2\pi}{k}\right) & -\sin\left((i+1) \cdot \frac{2\pi}{k}\right) \\ \sin\left((i+1) \cdot \frac{2\pi}{k}\right) & \cos\left((i+1) \cdot \frac{2\pi}{k}\right) \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) + \overrightarrow{um_p}, \text{ wobei } 0 \leq i < k,\ \mu \geq 0.$$

$\overrightarrow{S_r^p[i]}$ und $\overrightarrow{S_l^p[i]}$ sind demnach die beiden Halbgeraden, die vom Umkreismittelpunkt $\overrightarrow{um_p}$ des Polygons ausgehen und den i-ten Sektor des Polygons beschränken. Die Richtungsvektoren entstehen durch Rotation des Einheitsvektors $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ um den Winkel $i \cdot \frac{2\pi}{k}$ bzw. $(i+1) \cdot \frac{2\pi}{k}$. Abbildung 7 veranschaulicht Definitionen 7 und 8.

Zur Kodierung des Polygons in einen Feature-Vektor muß nun für jeden Umkreissektor $S^p[i]$ der Flächenanteil des Polygons berechnet werden, der innerhalb dieses Sektors liegt. Mit Hilfe der obigen Formeln sind wir in der Lage, diese Flächenanteile $F^p[i]$ eines Polygons p zu bestimmen. Sie ergeben sich durch Aufsummieren der Flächen, die von den einzelnen Kanten beigetragen werden.

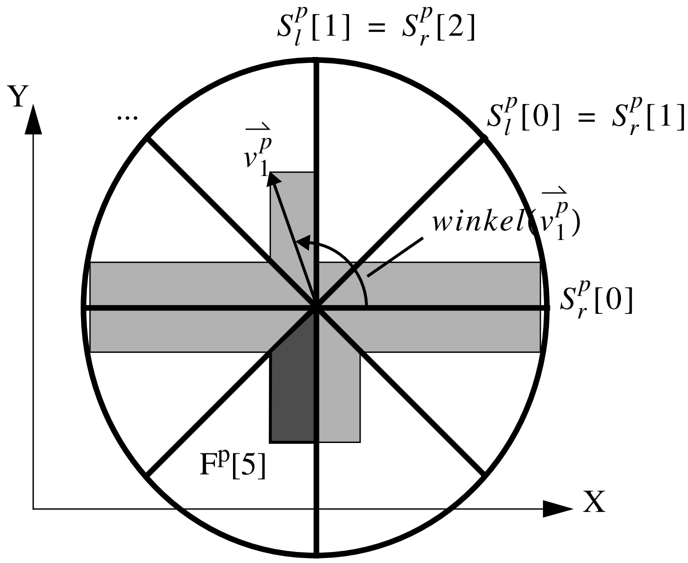


Abbildung 7: Winkel und Umkreissektoren

Definition 9: *Flächenanteil*

Der i-te Flächenanteil ist definiert als

$$F^p[i] = \left| \sum_{j=0}^{m^p-1} vorzeichen(\vec{e}_j^p) \cdot flaeche(\overrightarrow{um_p}, \vec{u}, \vec{w}) \right|$$

$$= \left| \sum_{j=0}^{m^p-1} vorzeichen((\vec{e}_j^p) \cdot \sqrt{s(s - \|\vec{u} - \overrightarrow{um_p}\|)(s - \|\vec{w} - \vec{u}\|)(s - \|\overrightarrow{um_p} - \vec{w}\|))} \right|,$$

$$\text{mit } s = \frac{\|\vec{u} - \overrightarrow{um_p}\| + \|\vec{w} - \vec{u}\| + \|\overrightarrow{um_p} - \vec{w}\|}{2}.$$

Bei dieser Vorgehensweise wird das Polygon in Dreiecke zerlegt, die vom Umkreismittelpunkt ausgehen und zu zwei Punkten $\vec{u}$ und $\vec{w}$ laufen. Die Dreiecke können sich jedoch überlappen, so daß bei einer einfachen Aufsummierung einerseits einzelne Flächen des Polygons mehrfach gezählt würden, andererseits Flächenteile, die eigentlich außerhalb des Polygons liegen, mit berücksichtigt würden. Um dies zu vermeiden, wird die Funktion $vorzeichen(\vec{e}_j^p)$ eingeführt. Sie nimmt die Werte 0, -1 oder 1 an und beschreibt, in welcher Richtung - aus Sicht des Umkreismittelpunkts - die Kante $\vec{e}_j^p$ verläuft:

$$vorzeichen(\vec{e}_j^p, j) = \begin{cases} 1 & \text{falls } winkel(\vec{v}_j^p, \overrightarrow{um_p}) > winkel\left(\overrightarrow{v^p_{(j+1)\ mod\ m_p}}, \overrightarrow{um_p}\right) \\ 0 & \text{falls } winkel(\vec{v}_j^p, \overrightarrow{um_p}) = winkel\left(\overrightarrow{v^p_{(j+1)\ mod\ m_p}}, \overrightarrow{um_p}\right) \\ -1 & \text{falls } winkel(\vec{v}_j^p, \overrightarrow{um_p}) < winkel\left(\overrightarrow{v^p_{(j+1)\ mod\ m_p}}, \overrightarrow{um_p}\right) \end{cases}.$$

Abbildung 8 verdeutlicht anhand eines Beispiels, wie die Flächenanteile des Polygons gezählt werden. Verläuft eine Kante aus Sicht des Umkreismittelpunkts von links nach rechts (Kanten

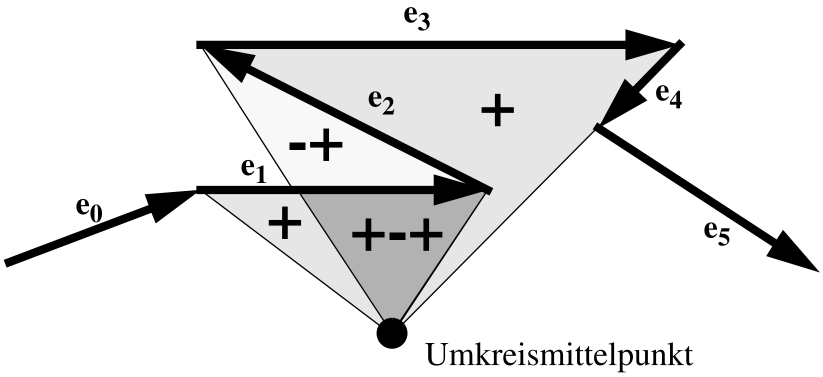


Abbildung 8: Positive und negative Kanten

e_0, e_1, e_3 und e_5 in Abbildung 8), so wird diese Kante als positive Kante bezeichnet, verläuft sie von rechts nach links, so ist sie eine negative Kante (z.B. Kante e_2). Verläuft eine Kante exakt vom Mittelpunkt weg oder auf ihn zu, so kann sie ignoriert werden (Kante e_4).

Die Punkte $\vec{u}$ und $\vec{w}$ begrenzen den Anteil der Kante $\overrightarrow{e_j^p}$, der sich innerhalb des Sektors $S^p[i]$ befindet. Betrachtet man die beiden Stützstellen $\overrightarrow{v_j^p}$ und $\overrightarrow{v_{j+1}^p}$ der Kante, so können fünf Fälle zur Bestimmung der Punkte $\vec{u}$ und $\vec{w}$ unterschieden werden:

1. $\vec{u} = \overrightarrow{v_j^p}$ und $\vec{w} = \overrightarrow{v_{j+1}^p}$, — falls die Kante vollständig innerhalb des Sektors liegt.
2. $\vec{u} = \vec{w} = \overrightarrow{um_p}$, — falls die Kante vollständig außerhalb des Sektors liegt.
3. $\vec{u} = \overrightarrow{v_j^p}$ und $\vec{w} = \overrightarrow{e_j^p} \cap S^p[i]$, — falls $\overrightarrow{v_j^p}$ innerhalb und $\overrightarrow{v_{j+1}^p}$ außerhalb des Sektors liegen.
4. $\vec{u} = \overrightarrow{v_{j+1}^p}$ und $\vec{w} = \overrightarrow{e_j^p} \cap S^p[i]$, — falls $\overrightarrow{v_{j+1}^p}$ innerhalb und $\overrightarrow{v_j^p}$ außerhalb des Sektors liegen.
5. $\vec{u} = \overrightarrow{e_j^p} \cap S_l^p[i]$ und $\vec{w} = \overrightarrow{e_j^p} \cap S_r^p[i]$, — falls $\overrightarrow{v_{j+1}^p}$ und $\overrightarrow{v_j^p}$ außerhalb des Sektors liegen, jedoch mindestens ein Punkt der Kante innerhalb liegt.

Die Funktion $\overrightarrow{e_j^p} \cap S^p[i]$ bezeichnet dabei den Schnittpunkt der Kante $\overrightarrow{e_j^p}$ mit den Sektorgrenzen des Sektors $S^p[i]$.

Definition 10: *Section Coding*

Section Coding definiert die Ähnlichkeit zweier Polygone p und q als den euklischen Abstand der durch die Flächenanteile von p und q bestimmten Feature-Vektoren, wobei die Flächenanteile normiert sind. Formaler:

$$S(p, q) = \sqrt[k]{\sum_{i=0}^{k-1} \left(\frac{F^p[i]}{A^p} - \frac{F^q[i]}{A^p} \right)^2} \quad .$$

Betrachten wir nun, ob Section Coding ein Ähnlichkeitsmaß bzw. eine Ähnlichkeitsmetrik nach Definitionen 2 und 3 ist, und welche der in den Definitionen 4 und 5 beschriebenen Eigenschaften es besitzt. Daß Section Coding ein Ähnlichkeitsmaß ist, das heißt, daß die Positivitäts-, Reflexivitäts- und Symmetrieeigenschaften gelten, ergibt sich direkt aus der obigen Definition von Section Coding. Die Gültigkeit der Dreiecksungleichung folgt ebenfalls direkt aus der obigen Definition. Die Identitätseigenschaft gilt nur eingeschränkt, nämlich für konvexe Polygone und für großes *k*. Dies liegt daran, daß es bei kleinem *k* vorkommen kann, daß verschiedene Polygone auf den gleichen Feature-Vektor abgebildet werden. Die Konfliktwahrscheinlichkeit geht jedoch für große *k* gegen 0. Section Coding ist folglich nur eingeschränkt als Ähnlichkeitsmetrik zu sehen.

Im folgenden betrachten wir die weiteren Eigenschaften, nämlich die Stetigkeitseigenschaft nach Definition 4 sowie die Invarianzen nach Definition 5. Die Stetigkeitseigenschaft gilt, da die Flächenanteile eines variierten Polygons *p'* bei stetiger Transformation in das Ursprungspolygon *p* gegen die Flächenanteile von *p* konvergieren. Section Coding ist ferner invariant gegenüber einigen der affinen Abbildungen aus Definition 5. Invarianz gegenüber Translation gilt wegen der Invarianz der Flächenanteileberechnung gegenüber der Translation. Skalierungsinvarianz gilt wegen der Normierung der Flächeninhalte (siehe Definition 10). Von den verbleibenden affinen Abbildungen sind für unseren Anwendungsbereich nur noch die Rotation und Spiegelung interessant. Invarianz gegenüber Rotationen um die Winkel $i \cdot (2\pi)/k$ $(0 \le i < k)$ kann sehr einfach durch das Stellen von *k* Anfragen erreicht werden. Hierzu folgende Überlegung: Rotiert man ein Polygon *p* um den Winkel $(2\pi)/k$, erhält man ein Polygon *p'*. Vergleicht man die Feature-Vektoren von *p* und *p'*, so stehen die zugehörigen Feature-Vektoren F^p und $F^{p'}$ in der einfachen Beziehung: $F^p[i] = F^{p'}[(i+1) mod\ k]$. Stellt man nun *k* Anfragen mit dem *k*-mal rotierten Feature-Vektor des Anfragepolygons, so findet man alle Polygone, die ähnlich zum Anfragepolygon bzw. den betrachteten Rotationen sind. Ist die Suchzeit wichtiger als der Speicherplatzaufwand, so kann alternativ auch jedes Polygon der Datenbank jeweils mit allen *k* Rotationen gespeichert werden. Dabei braucht die Kodierung des Polygons nur ein einziges Mal in einer beliebigen Rotation berechnet zu werden. Die *k* Rotationen können dann durch Verschieben der *k* Flächenanteile des Feature-Vektors erzeugt werden. Ein ähnliches Vorgehen ist zur Erreichung von Spiegelungsinvarianz möglich. Dabei wird als Anfrage neben dem Feature-Vektor des eigentlichen Anfragepolygons auch der Feature-Vektor des gespiegelten Anfragepolygons verwendet. Steht genügend Speicher zur Verfügung, können alternativ die Datenbankpolygone samt ihrer Spiegelungen im Index gespeichert werden, wodurch die Anfragezeit bei der Suche reduziert wird.

Bei der Implementierung von Section Coding sind zwei Aspekte zu berücksichtigen: Zum einen muß eine Indexstruktur verwendet werden, die auch für Anfragen auf hochdimensionalen Feature-Vektoren geeignet ist und zum anderen ist eine effiziente Implementierung der Flächenberechnung erforderlich. Als Indexstruktur verwenden wir bei unserer Implementierung den X-Tree [BKK 96], eine multidimensionale Indexstruktur, die speziell für hochdimensionale Feature-Vektoren entwickelt wurde. Bei der Flächenberechnung sind nach Definition 9 insgesamt *k* Flächenanteile zu berechnen. Deshalb ist die Gesamtlaufzeit der Feature-Kodierung eines Polygons nach Definition 9 maximal in der Größenordnung $O(k \cdot m_p)$. Eine effizientere Implementierung der Flächenanteilsberechnung ist jedoch möglich, indem statt der

Sektoren die Kanten des Polygons durchlaufen werden. Liegt eine Kante vollständig in einem Sektor, so muß zur Bestimmung der Flächenanteile genau dasjenige Dreieck betrachtet werden, das vom Umkreismittelpunkt mit den beiden Stützstellen der Kante gebildet wird. Ist die

```
Feature SectionCoding::transform(Polygon p, int k)
{
   int i, j, z, start, end, factor;
   Vector section_old, last_point, section_new;
   Feature f;

   // initialize feature-vector
   for (i = 0; i < k; i++)
      f[i] = 0.0;

   // for all edges
   for (i = 0; i < p.get_num(); i++)
   {
      // ignore edge[i], if it's line goes through center
      if (p.edge[i].inside(p.center))
         continue;

      // determine starting and ending sectors of edge[i]
      start  = determine_sector(p.edge[i].start);
      end    = determine_sector(p.edge[i].end);
      factor = determine_direction(p.edge[i]);

      // if edge[i] starts and ends in the same sector,
      // determine the area in this sector
      if (s == e)
         f[s] += factor * area(p.edge[i].start, p.edge[i].end, p.center);
      else
      {
         // edge[i] crosses more than one sector,
         // walk through sectors
         if (factor == 1)
         {
            // swap start and end
            start = z; start = end; end = z;
            section_old = p.edge[i].end;
            last_point = p.edge[i].start;
         }
         else
         {
            section_old = p.edge[i].start;
            last_point = p.edge[i].end;
         }

         // for all sectors crossed by edge[i]
         for (j = start; j < end; j++)
         {
            // if there is an intersection between the
            // edge and a sector border, than continue with sector (j+1)
            section_new = section(edge[i], sector[j+1]);
            f[j] += factor * area(section_new, section_old, p.center);
            section_old = section_new;
         }
         f[j] += factor * area(section_old, last_point, p.center);
      }
   }

   // normalize feature
   total_area = f.sum();
   for (i = 0; i < p.get_num(); i++)
      f[i] = fabs(f[i] / total_area;
}
```

Abbildung 9: Algorithmus zur Feature-Transformation (Section Coding)

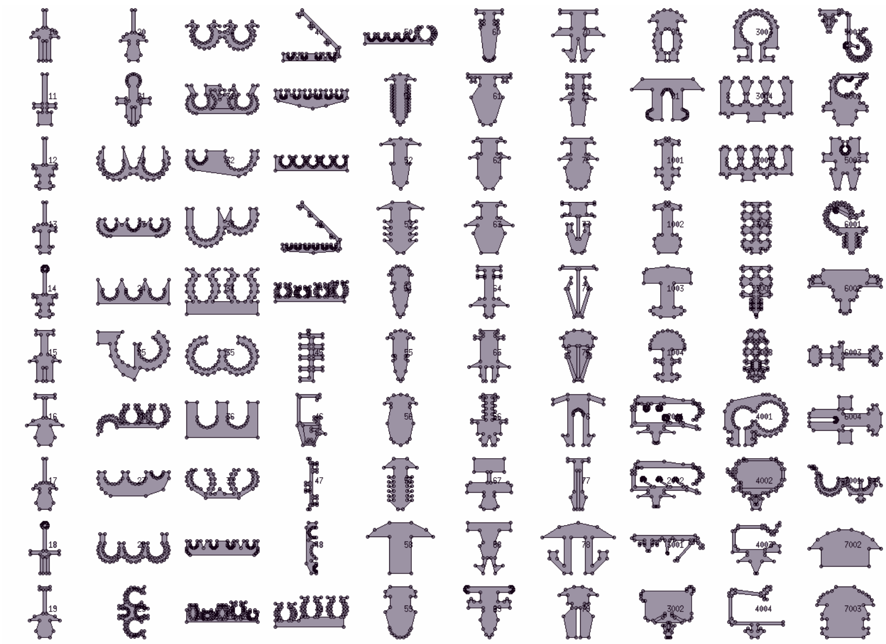


Abbildung 10: Ausschnitt aus dem Datenbestand von Clipsen

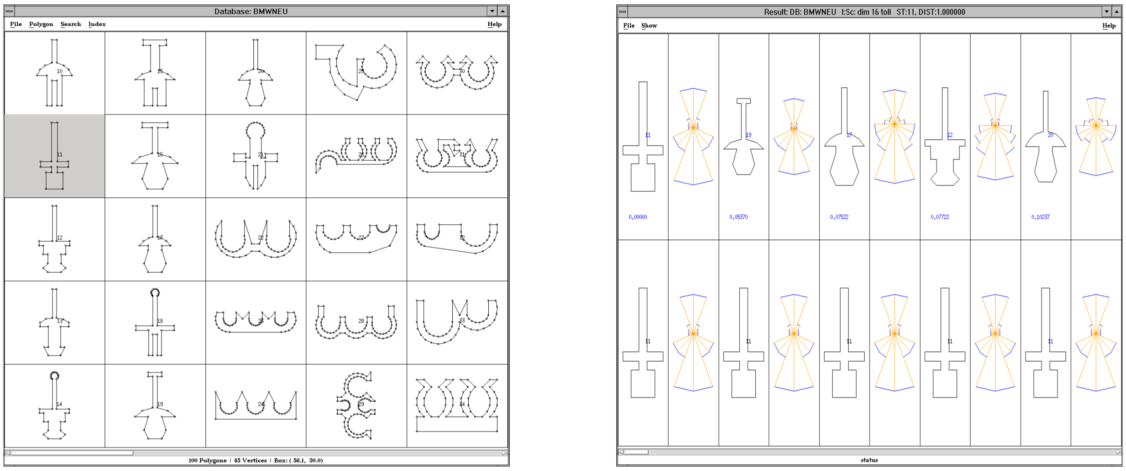
Kante positiv, so wird die Fläche des Dreiecks zur Gesamtfläche des Sektors addiert, ansonsten wird sie subtrahiert. Wie in Abbildung 8 zu sehen ist, können sich die einzelnen Dreiecke überlagern. Durch die Aufteilung in positive und negative Kanten ist jedoch sichergestellt, daß alle Bereiche des Polygons in der Summe genau einmal positiv gezählt werden. Durchläuft eine Kante mehrere Sektoren, so wird sie mit den betroffenen Sektorgrenzen geschnitten und die dabei entstehenden einzelnen Kantensegmente wie oben beschrieben gewertet. Um Invarianz gegenüber Skalierungen zu erreichen, werden die Flächenanteile in einem nachfolgenden Schritt derart normiert, daß die Summe alle Flächenanteile genau 1 ist. In Abbildung 9 ist der vollständige Algorithmus dargestellt. Die Laufzeitkomplexität des modifizierten Algorithmus ist $O(\bar{k} \cdot m_p)$, da für alle Kanten jeweils nur diejenigen $\bar{k}$ Sektoren betrachtet werden, die von der jeweiligen Kante geschnitten werden. Eine Kante kann dabei höchstens $\frac{k}{2} + 1$ Sektoren betreffen. Für praktisch relevante Polygone gilt jedoch $\bar{k} \ll \frac{k}{2} < k$.

4. Implementierung und Analyse

In diesem Kapitel geben wir einen Überblick über die Implementierung und die experimentelle Evaluierung unseres Verfahrens. Wir wenden unser Verfahren auf eine große Menge realer Daten an und zeigen sowohl die Effektivität als auch die Effizienz des Verfahrens.

4.1 Implementierung

Um Section Coding unter realen Bedingungen testen zu können, wurden die Algorithmen in unser Prototyp-Ähnlichkeitssystem ***S3*** (**S**imilarity **S**earch **S**ystem) integriert. **S3** ist ein Datenbanksystem für die Verwaltung industrieller Bauteile, die durch ein zweidimensionales polygonales CAD-Modell beschrieben sind. Das System speichert die Kontur eines jeden Bauteils der Datenbank als Polygon und erlaubt dem Anwender, mit verschiedenen Verfahren und Parametern Indexstrukturen zu erzeugen. **S3** ist daher eine Testumgebung für Algorithmen zur Ähnlichkeitssuche. **S3** wurde in C++ unter X11/OSF Motif entwickelt und läuft sowohl unter HP/UX



a) Anfragespezifikation b) Ergebnis der Suche

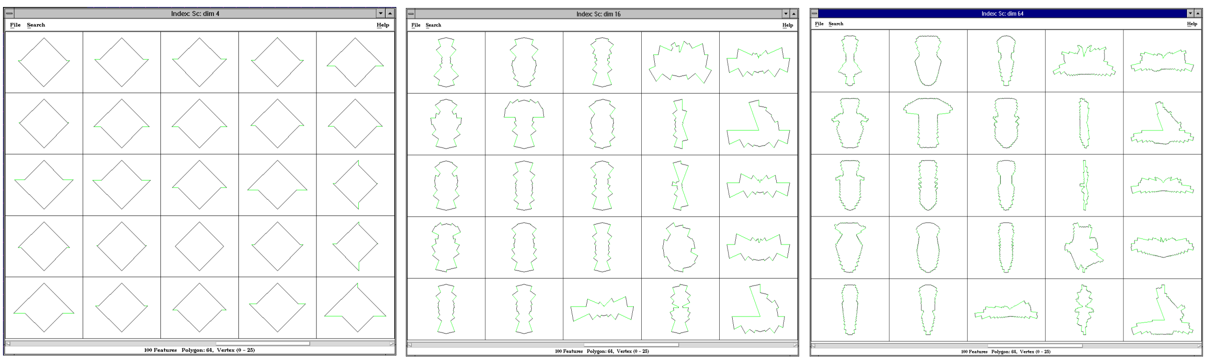
Abbildung 12: Beispiel für eine Anfrage in S3 und deren Ergebnis (Query by Example)

als auch unter Linux. Sämtliche Experimente, die in den nächsten Abschnitten präsentiert werden, wurden auf einer HP 715/64 Workstation mit 64MB Hauptspeicher und einigen Gigabytes Sekundärspeicher durchgeführt. Als Indexstruktur für die Feature-Vektoren wurde die objektorientierte Implementierung des X-Tree benutzt [BKK 96]. Der X-Tree unterstützt verschiedene Anfragetypen, unter anderem auch Bereichsanfragen und *k*-nächste-Nachbarn Anfragen.

4.2 Effektivität

Um die Qualität unseres Algorithmus zu zeigen, haben wir diverse Experimente durchgeführt. Die verwendete Datenbasis besteht aus einer großen Datenbank (ca. 10 Megabytes) realer Bauteile, die durch zweidimensionale Polygone beschrieben sind und von einem industriellen Partner, einem Zulieferer der Automobil-Industrie, stammen. Die betrachteten Bauteile, sog. 'Clipse', sind Verbindungselemente, die im Fahrzeugbau in großer Anzahl auftreten (siehe Abbildung 10). Clipse werden im allgemeinen im Spritzgußverfahren aus Kunststoff hergestellt und können daher in beinahe jeder beliebigen Form erzeugt werden.

Um die Wirkungsweise des Verfahrens zu demonstrieren, erlaubt **S3** eine Visualisierung nicht nur der Daten selbst sondern auch der berechneten Feature-Vektoren. Im Fall des Section Coding-Verfahrens werden die Flächenanteile wieder als Sektoren in einem Kreis aufgetragen. Abbildung 11 zeigt eine derartige Menge von visualisierten Feature-Vektoren. Vergleicht man Abbildung 11 b und c, so erkennt man, daß mit steigender Dimension der Feature-Vektoren (k) die



a) Dimension 4 b) Dimension 16 c) Dimension 64

Abbildung 11: Feature-Vektoren unterschiedlicher Dimension

Ausgangspolygone immer besser approximiert werden. Für unsere Datenmenge erwies sich die Dimension 16 als optimaler Kompromiß zwischen Detailgenauigkeit und Effizienz.

Das **S3**-System unterstützt zwei Arten von Anfragen:

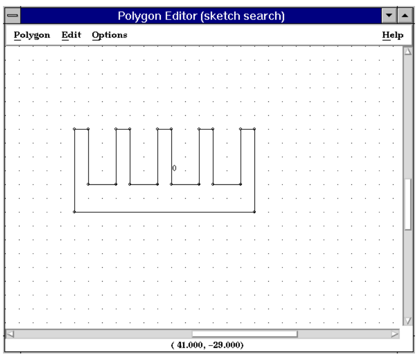
- Query by Example: Der Anwender markiert ein Bauteil aus der Datenbank und S3 sucht alle dazu ähnlichen Bauteile.
- Query by Sketch: Der Anwender zeichnet eine Skizze des gesuchten Teils und S3 sucht alle dazu ähnlichen Bauteile.

Abbildung 12 zeigt an ein Beispiel für eine ‘Query by Example’-Anfrage. Der Anwender hat im linken Teil der Abbildung ein Bauteil markiert. Im rechten Teil der Abbildung ist das Ergebnis der Suche abgebildet. In der unteren Hälfte des Fensters stellt **S3** dabei jeweils die Anfragespezifikation und deren Feature-Kodierung dar, im oberen Teil des Fensters ist das gefundene Teil und dessen Kodierung dargestellt. Dadurch erhält der Anwender zusätzlich zu den Ergebnisbauteilen eine Begründung, warum die Bauteile als ähnlich erkannt wurden.

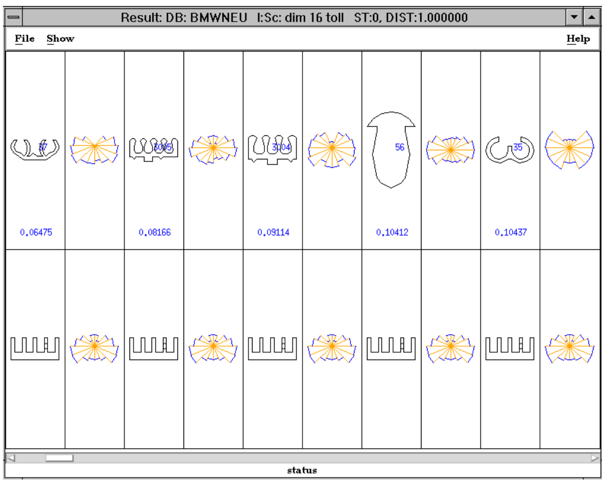
Hat der Benutzer keine detaillierten Kenntnisse vom Inhalt der Datenbank, so kann er die geometrische Anfrage auch in Form einer Skizze spezifizieren. Abbildung 13 zeigt ein Beispiel für eine ‘Query by Sketch’-Anfrage. Der Benutzer sucht in dem Beispiel nach einem sogenannten Kabelbinder und skizziert ungefähr dessen Form, ohne einen speziellen Kabelbinder aus der Datenbank zu kennen. Wie in der Abbildung zu sehen ist, findet **S3** in der Mehrzahl Kabelbinder, es treten jedoch auch sog. ‘False Hits’ auf (zum Beispiel in Abbildung 13 das vierte Polygon). Diese können jedoch leicht vom Anwender aussortiert werden, da die Gesamtzahl der Treffer nur relativ gering ist.

4.3 Effizienz

Um die Effizienz von Section Coding zu zeigen, haben wir unser Verfahren mit großen Datenmengen getestet. Abbildung 14 zeigt das Ergebnis dieser Experimente. Bei den Experimenten wurde nach den 20 ähnlichsten Bauteilen gesucht, wobei die Datenbankgröße von 50 bis zu 6.400 Bauteilen gesteigert wurde. Wie zu erwarten war, sind die Suchzeiten logarithmisch in der Anzahl der Polygone. Es ist damit gezeigt, daß sich das Verfahren auch für sehr große Men-



a) Anfragespezifikation



b) Ergebnis der Suche

Abbildung 13: Beispiel für eine Anfrage in *S3* und deren Ergebnis (Query by Sketch)

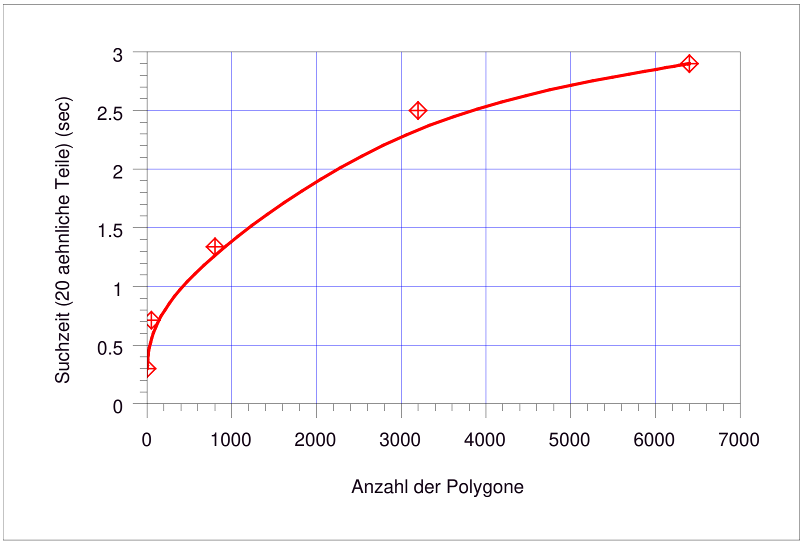


Abbildung 14: Effizienz von Section Coding

gen von Polygonen eignet. Weiterführende Tests in Zusammenarbeit mit dem Anwender des Verfahrens sind geplant.

5. Zusammenfassung und Ausblick

Ausgangspunkt dieses Artikels ist eine Problemstellung, die häufig in CAD-Anwendungen auftritt: die Suche ähnlicher CAD-Bauteile in einer CAD-Datenbank. Eine Voraussetzung zur Behandlung dieser Problemstellung ist eine genaue Definition von Ähnlichkeit. Deshalb haben wir zunächst die wichtigsten Eigenschaften menschlicher Ähnlichkeitsbegriffe analysiert und daraus die Begriffe Ähnlichkeitsmaß und Ähnlichkeitsmetrik sowie Eigenschaften von Ähnlichkeitsmaßen und -metriken abgeleitet. Als nächstes haben wir ein neues Ähnlichkeitsmaß für die globale Ähnlichkeitssuche, genannt Section Coding, vorgestellt und einen effizienten Algorithmus zu seiner Berechnung angegeben. Im Gegensatz zu bisherigen Verfahren ist Section Coding speicherplatz- und suchzeiteffizient, erlaubt einen hohen Grad an Invarianzen und ist robust gegenüber kleinen Veränderungen der Kontur. Wir haben unser Verfahren in das CAD-Datenbanksystem **S3** integriert und umfangreiche Experimente durchgeführt, die sowohl die Effektivität als auch die Effizienz von Section Coding demonstrieren.

In der Zukunft werden wir uns mit alternativen Verfahren zur Ähnlichkeitssuche befassen. Ein Möglichkeit wäre beispielsweise, die Sektorengrenzen jeweils auf Stützstellen zu legen. Dadurch entstünde eine variable Anzahl von Sektoren und die Berechnung der Flächenanteile würde vereinfacht. Da jedoch keine Indexstrukturen für Feature-Vektoren unterschiedlicher Dimension bekannt sind, wurde diese Alternative bisher nicht weiter verfolgt. Ferner wollen wir uns mit der Erweiterung des Verfahrens für die partielle Ähnlichkeitssuche beschäftigen. So ist eine detaillierte Untersuchung des Komplements von Ähnlichkeit, der Unähnlichkeit, geplant. Die bisher bekannten Verfahren zur Berechnung von Ähnlichkeit eignen sich dazu im allgemeinen nicht, da sie nur qualitative Aussagen über die Unähnlichkeit liefern.

Danksagung

Wir danken den Gutachtern dieses Artikels für ihre ausführlichen Anmerkungen und wertvollen Korrekturen, die zur Verbesserung des Artikels beigetragen haben.

Referenzen

[AB 92] Alt H., Blömer J.: *'Resemblance and Symmetries of Geometric Patterns',* Data Structures and Efficient Algorithms, in: LNCS, Vol. 594, Springer, 1992, pp. 1-24.

[ABB 92] Alt H., Behrends M., Blömer J.: *'Approximate Matching of Polygonal Shapes',* Proc. of the 7th Annual ACM Symp. on Computational Geometry, 1992, pp. 186-193.

[AFS 93] Agrawal R., Faloutsos C., Swami A.: *'Efficient similarity search in sequence databases',* Proc. 4th Int. Conf. on Foundations of Data Organization and Algorithms, 1993, LNCS 730, pp. 69-84.

[ALSS 95] Agrawal R., Lin K., Sawhney H., Shim K.: *'Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases',* Proc. of the 21st Conf. on Very Large Databases, 1995, pp. 490-501.

[AMWW 88] Alt H., Mehlhorn K., Wagener H., Welzl E.: *'Congruence, Similarity and Symmetries of Geometric Objects',* Discrete Computational Geometry 3, 1988, pp. 237-256.

[BKK 96] Berchtold S., Keim D., Kriegel H.-P.: *'The X-tree: An Index Structure for High-Dimensional Data',* 22nd Conf. on Very Large Databases, 1996, Bombay, India.

[BKSS 90] Beckmann N., Kriegel H.-P., Schneider R., Seeger B.: *'The R*-tree: An Efficient and Robust Access Method for Points and Rectangles',* Proc. ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ, 1990, pp. 322-331.

[FBF+ 94] Faloutsos C., Barber R., Flickner M., Hafner J., et al.: *'Efficient and Effective Querying by Image Content',* Journal of Intelligent Information Systems, 1994, Vol. 3, pp. 231-262.

[Fre 87] Freeston M.: *'The BANG file: A new kind of grid file',* Proc. ACM SIGMOD Int. Conf. on Management of Data, San Francisco, CA, 1987, pp. 260-269.

[FRM+ 94] Faloutsos C., Ranganathan M., Manolopoulos Y.: *'Fast Subsequence Matching in Time-Series Databases',* Proc. ACM SIGMOD Int. Conf. on Management of Data, 1994, pp. 419-429.

[HT 94] Helmer-Citterich M., Tramontano A.: *'PUZZLE: A New Method for Automated Protein Docking Based on Surface Shape Complementarity',* Journal of Molecular Biology, Vol. 235, pp.1021-1031, 1994.

[Jag 91] Jagadish H. V.: *'A Retrieval Technique for Similar Shapes',* Proc. ACM SIGMOD Int. Conf. on Management of Data, 1991, pp. 208-217.

[MG 93] Mehrotra R., Gary J. E.: *'Feature-Based Retrieval of Similar Shapes',* Proc. 9th Int. Conf. on Data Engineering, Vienna, Austria, 1993, pp. 108-115.

[MG 95] Mehrotra R., Gary J. E.: *'Feature-Index-Based Similar Shape retrieval',* Proc. of the 3rd Working Conf. on Visual Database Systems, March 1995.

[PF 94] Petrakis E., Faloutsos C.: *'Similarity Searching in Large Image DataBases',* Technical Report CS-TR-3388, University of Maryland, 1994.

[PS 85] Preparata F.P., Shamos M.I.: *‘Computational Geometry: An Introduction’,* Springer-Verlag, 1985.

[Rob 81] Robinson J. T.: *‘The K-D-B-tree: A Search Structure for Large Multidimensional Dynamic Indexes’,* Proc. ACM SIGMOD Int. Conf. on Management of Data, 1981, pp. 10-18.

[WW 80] Wallace T., Wintz P.: *‘An Efficient Three-Dimensional Aircraft Recognition Algorithm Using Normalized Fourier Descriptors’,* Computer Graphics and Image Processing, Vol. 13, pp. 99-126, 1980.