

Abbildung 3.15: Komplexität des geometrischen in Prädikats

2. Probleme treten nur in Zusammenhang mit Formprimitiven auf, die nicht zu Verkehrszeichen beitragen. Damit wird in diesem Fall die Aufgabe den Lösungsraum einzuschränken nicht erfüllt, was jedoch nicht zu Fehlern, sondern zur Erhöhung des Aufwands der Klassifikation beiträgt.
3. Es können bei der Klassifikation keine neuen Fehler durch fehlerhafte Segmentierung entstehen, da die Vollständigkeit des Prädikats *enthält* gegeben ist.

Einige dieser Punkte bedürfen einer näheren Erläuterung. Zur Plausibilisierung von Punkt 2 betrachten wir die Formen von Verkehrszeichen. Ein Kreis, Rechteck oder Dreieck stellt eine recht einfache geometrische Form dar. Angenommen, es tritt nun die Situation ein, daß eine solche geometrische Form A eine Form B enthält. Wenn es sich bei B wiederum um die Kontur eines Verkehrszeichens handeln soll, so wird dieses sicherlich so klein, so weit entfernt, sein, daß es zum aktuellen Zeitpunkt noch nicht klassifiziert werden kann. Es geht also keine Information verloren, wenn B vernachlässigt wird. Genau dies geschieht mit solchen Objekten während der Klassifikationsphase.

Betrachtet man andererseits die Kontur eines Objektes, das kein Verkehrszeichen darstellt, so kann es durchaus vorkommen, daß diese Kontur ein Verkehrszeichen fälschlicherweise *enthält* (siehe Abbildung 3.15). Auch dieser Fehler resultiert nicht in einer Fehlklassifikation. Es wird lediglich von der Klassifikationskomponente festgestellt, daß es sich bei der äußeren Kontur des betrachteten Segments *nicht* um eine Verkehrszeichenkontur handelt und die Kontur wird aus dem Segment entfernt. Es ist demnach nicht möglich, daß neue Fehler auftreten, weil eine Kontur zuviel aufgenommen wurde. Zu wenig Konturen können nicht aufgenommen werden, denn das würde bedeuten, daß ein Objekt A ein Objekt B fälschlicherweise *nicht enthält*, obwohl A geometrisch *in* B liegt. Genau das verhindert jedoch die Vollständigkeits-eigenschaft des Prädikats.

**Der Segmentierungsvorgang** Das zu untersuchende Bild wird in Form einer Menge von Formprimitiven übergeben. Diese Menge wird einmal durchlaufen, mit dem Ziel die enthaltenen Formprimitive in eine Struktur zu bringen, die zum einen syntaktisch auf die Wissensbasis abbildbar ist, und zum anderen Formprimitive zu Gruppen zusammenfaßt, die jeweils einen Szenenbereich beschreiben, in dem ein Verkehrszeichen zu finden sein könnte.

Der folgende Algorithmus beschreibt den Ablauf der Segmentierung:

Sei  $B = (f_1, \dots, f_n)$ ,  $f_i \in \Gamma$  eine Liste von Formprimitiven, die eine Verkehrsszene beschreiben und  $S$  die Ausgabe des Segmentierungsalgorithmus.  $S = (b_1, \dots, b_m)$  bezeichnet die Menge der Blöcke, die Formprimitive aus  $B$  zusammenfassen. Ein Block wird in Form eines Baumes dargestellt. Dabei bilden die Formprimitive  $f_i$  die Knoten und die geltende *enthält* Beziehung die Kanten des Baumes. Die Funktion *insert* fügt ein Formprimitiv anhand der geltenden *enthält*-Beziehungen in einen Baum ein.

```

Segmentierung(B)
BEGIN
  n := Anzahl der Formprimitive in B
  S := ()
  FOR i = 1, ..., n:
    IF S = ()
      THEN S := S + fi
    ELSE
      j := 1
      inserted := FALSE
      WHILE not(inserted) ∧ j ≤ length(S) DO
        IF ∃ a ∈ bj : fi enthält a ∨ a enthält fi
          THEN insert(fi, bj); inserted := TRUE.
        ELSE j := j + 1
        IF j > length(S)
          THEN S := S + fi
      END WHILE
    END FOR
  END FOR

```

Aus der Liste  $S$  der gefundenen Blöcke, werden diejenigen aussortiert, und in einer zweiten Liste zusammengefaßt, die nur aus einem einzigen

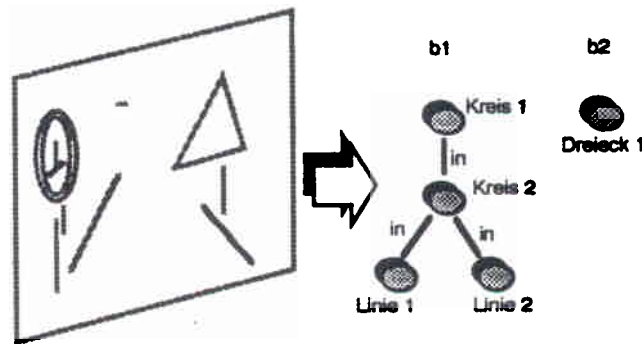


Abbildung 3.16: Beispiel für die Segmentierung

offenen Formprimitiv bestehen. Die beiden so erhaltenen Listen werden an die Abbildungskomponente weitergegeben.

Abbildung 3.16 zeigt eine Verkehrsszene und die aus ihr gewonnenen Segmentbäume. Die im Bild enthaltenen Linien, die keinem Segmentbaum zugeordnet wurden, werden ebenfalls bei der nachfolgenden Klassifikation berücksichtigt.

### 3.4.2.2 Abbildung eines Segmentbaums auf das Netzwerk

Die Suche nach einem Match zwischen einem Segmentbaum und dem Netzwerk<sup>47</sup>, ist bereits ein Versuch der Interpretation des Szenenbereichs und damit der erste Schritt zur Klassifikation des beschriebenen Segments. Betrachtet man das Abbildungsproblem aus graphentheoretischer Sicht, so handelt es sich bei der Abbildung des Segmentbaums auf das Netzwerk um die Suche nach einem *Subgraph Isomorphismus*. Es wird eine 1:1 Abbildung  $f$  gesucht, die jedem Knoten des Segmentbaums einen Knoten des Netzwerks zuordnet. Für jede Kante des Segmentbaumes zwischen  $v_1$  und  $v_2$  gibt es eine Kante mit gleicher Bedeutung, die  $f(v_1)$  und  $f(v_2)$  im Netzwerk verbindet. Die Suche nach einem *Subgraph Isomorphismus* ist wesentlich komplexer als die Suche nach einer Isomorphie zwischen zwei Graphen. Das *Subgraph Isomorphismus Problem* fällt in die Klasse der *NP-vollständigen* Probleme. Alle bekannten Algorithmen haben eine Zeitkomplexität, die in Abhängigkeit von der Länge der Eingabe, dies entspricht hier der Anzahl der Knoten und Kanten des Netzwerks und des Segmentbaums, exponentiell steigt.

Die obige Beschreibung läßt die Darstellung von Piktogrammen außer Acht. Da diese im Netzwerk durch einen einzigen Knoten<sup>48</sup>, in der Szene jedoch in den mei-

<sup>47</sup> Bei diesem Vorgang muß nur das Strukturnetz betrachtet werden.

<sup>48</sup> Eine mögliche Integration der Piktogramme in das Netzwerk in Form einer wiederum netzwerk-

sten Fällen durch mehrere Formprimitive dargestellt werden, ist eine 1:1 Abbildung sicherlich nicht möglich. Gleiches gilt für aufgerissene Konturen, die in zwei Formprimitive aufgespalten werden. Einem Piktogrammknoden (PN) des Netzwerks müßten demnach mehrere Formprimitivknoden des Segmentbaums zugeordnet werden, sowie einem Konturknoden (CN) möglicherweise ebenfalls mehrere Formprimitivknoden zugewiesen werden könnten. Dies würde die Komplexität der Abbildung weiter steigern.

Um das Problem in angemessener Zeit lösen zu können, wird die Abbildung des Segmentbaums auf das Netzwerk in mehrere Schritte zerlegt. In einem ersten Schritt werden lediglich Abbildungen auf Konturknoden (CN) des Netzwerks gesucht und die *enthält* Kanten verglichen. Dadurch wird aus dem gesuchten *Subgraph Isomorphismus* ein *zweifacher Subgraph Isomorphismus*, d.h. die Isomorphismen zwischen Subgraphen des Segmentbaums und Subgraphen des Netzwerks werden gesucht. Prinzipiell ist das Problem des *zweifachen Subgraph Isomorphismus* äquivalent zu dem des *Subgraph Isomorphismus*<sup>49</sup>. Hier wird durch den Ausschluß eines bestimmten Knotentyps (PN) eine Einschränkung getroffen. Die Typisierung der Knoten ermöglicht es, einen Knotentyp *auszublenden*. Der Segmentbaum wird somit nur auf das durch die Knoten des Typs CN und die Kanten des Typs RI aufgespannte Subnetz abgebildet. (siehe dazu auch Abbildung 3.17.)

Um eine Abbildung zwischen einem Segmentbaum und dem Netzwerk durchführen zu können, wird ein Ähnlichkeitsmaß benötigt, welches die Güte der Abbildung bestimmt. Dieses Ähnlichkeitsmaß stellt zugleich ein Kriterium für die Klassifikation des Segments dar.

**Der erste Abbildungsschritt** Als erster Schritt wird eine Abbildung gesucht, die eine möglichst große Ähnlichkeit zu einer Substruktur des Netzwerkes hat. Dabei werden die SN- und PN-Knoten des Netzwerkes sowie die CL-Kanten ausgeblendet. Eine Abbildung besteht aus einer Menge von eindeutigen Zuordnungen zwischen Knoten des Segmentbaums und Knoten des Netzwerkes. Jeder Zuordnung wird ein Ähnlichkeitswert zugewiesen.

Der Startknoten *s* und die E-Kanten steuern den Abbildungsvorgang. Dabei wird das Wissen über den Aufbau des Netzwerkes ausgenutzt. Die Eintrittsknoten des Netzwerkes<sup>50</sup> werden mit der Wurzel des Segmentbaumes verglichen. Die Konturbeschreibung, die der Eintrittsknoten speichert, wird mit der durch das Formprimitiv, welches die Wurzel des Baumes darstellt, repräsentierten Kontur verglichen. Dazu

artigen Struktur würde das Problem (teilweise) lösen.

<sup>49</sup>Das Problem des zweifachen Subgraph Isomorphismus kann auf das Problem des Subgraph Isomorphismus reduziert werden. Zu Matching Verfahren siehe auch [BB82] und [San90].

<sup>50</sup>Die durch E-Kanten mit *s* verbundenen Kanten heißen Eintrittsknoten

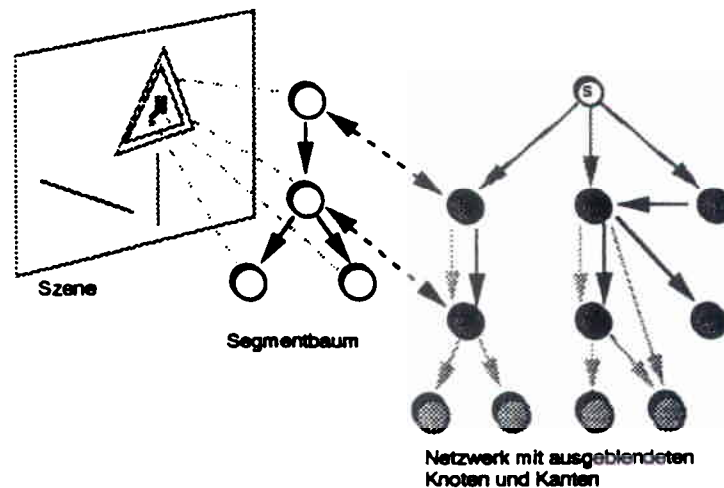


Abbildung 3.17: Abbildung eines Segmentbaumes auf das Netzwerk

wird ein Ähnlichkeitsmaß benutzt, welches in 3.4.2.6 näher beschrieben wird. Weist das Formprimitiv eine ähnliche Kontur wie das Muster auf, besteht aber aus einer anderen Anzahl von Kanten, oder ist offen, so wird versucht, es mit Hilfe des Musters zu rekonstruieren<sup>51</sup>. Danach werden die Eintrittsknoten gemäß ihres zugeordneten Ähnlichkeitswertes in einer Liste *L* zusammengefaßt. Mit Hilfe eines Schwellwertes, werden nun diejenigen aussortiert, die nicht weiter betrachtet werden sollen.

Wird keine ausreichend große Ähnlichkeit zwischen dem Formprimitiv und den Eintrittsknoten festgestellt, so bedeutet dies, daß das Formprimitiv keine äußere Kontur eines Verkehrszeichens darstellt. Es handelt sich bei diesem Formprimitiv demnach um ein Element des Bildhintergrundes oder die äußere Kontur eines Verkehrszeichens wurde aufgerissen und somit nicht in den Baum eingefügt. Im ersten Fall ist die Wurzel des Segmentbaums zu löschen und mit dem (oder den) entstandenen Unterbäumen erneut eine Abbildung auf das Netzwerk zu versuchen. Im letzteren Fall stellt die Wurzel wahrscheinlich die *innere*<sup>52</sup> Kontur eines Verkehrszeichens dar. Es sind also diejenigen Knoten des Netzwerks zu untersuchen, die Information über die inneren Konturen von Verkehrszeichen speichern. Das sind genau die Knoten, die von den Eintrittsknoten aus über eine *enthält*-Kante<sup>53</sup> zu

<sup>51</sup>Ein Formprimitiv des Typs *open.polygonal* kann beispielsweise in ein Formprimitiv des Typs *polygonal* überführt werden.

<sup>52</sup>Besteht ein Zeichen beispielsweise aus zwei ineinanderliegenden Kreisen, so wird der innere Kreis als die innere Kontur des Verkehrszeichens bezeichnet.

<sup>53</sup>Dabei ist zu beachten, daß die Kanten gerichtet sind.

erreichen sind. Wird mit einem dieser Knoten ein Match gefunden, so werden die Formprimitive aus der näheren Umgebung des Wurzelformprimitivs mit den Vätern des gefundenen Netzwerkknotens verglichen. Nach Ergänzung der äußeren Kontur wird fortgefahren, wie im Fall der nichtleeren Liste L.

Ist die Liste L nicht leer, so werden die Söhne des Wurzelknotens des Segmentbaums mit den Knoten des Netzwerks verglichen, die von jeweils einem Knoten A aus L über eine *enthält*-Kante erreicht werden können. Dabei übernimmt A für den nächsten Match die Rolle des Startknotens s und die von A ausgehenden *enthält*-Kanten die Rolle der Eintrittskanten E.

Ist der Baum durchlaufen oder gehen von dem zuletzt gematchten Netzwerkknoten keine *enthält*-Kanten mehr aus, so ist der Abbildungsvorgang beendet. Man erhält eine Menge von Bindungsmengen zwischen Formprimitiven und Knoten des Netzwerks, denen Ähnlichkeitswerte zugeordnet wurden. Diese Mengen stellen das Ergebnis des ersten Abbildungsschrittes dar. Das Ergebnis des ersten Abbildungsschrittes ist eine Menge von möglichen Abbildungen des Segmentbaums auf das Netzwerk. Dabei werden den Formprimitiven (Baumknoten)<sup>54</sup> explizit Knoten im Netzwerk, sowie Ähnlichkeitswerte zugeordnet. Eine solche Abbildung wird durch  $k: \Gamma \rightarrow \text{CN}$  bezeichnet.  $D(k)$  bezeichne die Menge der Formprimitive des Segmentbaums, die durch k auf Netzwerkknoten abgebildet wurden.  $W(k)$  bezeichne die Netzwerkknoten, auf die irgendein Formprimitiv durch k abgebildet wurde. Die Abbildung  $k^{-1}: \text{CN} \rightarrow \Gamma$  ordnet jedem Element N aus  $W(k)$  ein Formprimitiv zu, welches aus einem Formprimitiv A, mit  $k(A)=N$ , des Segmentbaums rekonstruiert wurde. Durch die Vorgehensweise der Abbildung wird zusätzlich garantiert, daß die *enthält*-Lagerrelation zwischen den gematchten Knoten erfüllt ist. Unberücksichtigt sind bisher die anderen modellierten Lagerrelationen. Diese sollen in einem zweiten Abbildungsschritt untersucht werden.

**Der zweite Abbildungsschritt** Nachdem eine Menge von Abbildungen des Segmentbaumes auf das Netzwerk gefunden worden ist, müssen noch die modellierten Lagerrelationen geprüft werden. Für jede mögliche Abbildung werden jetzt die Ähnlichkeiten zwischen den in der Szene vorliegenden Lagerrelationen und den im Netzwerk gespeicherten Musterrelationen ermittelt. Eine Lagerrelation ist im Netzwerk durch eine Funktion  $R: \Gamma \times \Gamma \rightarrow [0,1]$  repräsentiert.

Im zweiten Abbildungsschritt werden jetzt zu jeder vorliegenden Knotenabbil-

<sup>54</sup>Die Zuordnung zwischen Knoten im Netzwerk und Formprimitiven werden in Form einer Liste von Paaren gespeichert. Dabei wird jedoch anstelle des ursprünglichen Formprimitivs eine korrigierte und gegebenenfalls in einen anderen Formprimitivtyp konvertierte Kopie gespeichert, um die Berechnung der Lagerrelationen zu ermöglichen, bzw. zu vereinfachen. Weitere Ausführung dazu in 3.4.2.6.

*Matching-Schritt-1*  
 ( Segmentbaum T,  
 Knotenmenge E,  
 Umgebungsdaten U)

```

BEGIN
W:= Wurzel von T
A:=()
L:=()
FOR ALL x aus E DO
  IF SP(W,x) > Match-Schwellwert
  THEN Füge ( x , W, SP(W,x)) in L ein (geordnet nach SP(W,x))
END FOR
IF L = ()
THEN
  NewE:= {x|enthält(y,x) ∈ RL ∧ y ∈ E ∧ y unmarkiert}
  A:= Matching-Schritt-1(T, NewE,U)
  FOR ALL x aus A DO
    Suche einen Match a zwischen dem Vater des obersten
    gematchten Knotens aus x und einem Element aus U.
    Füge a in x ein.
  END FOR
ELSE
  FOR ALL x aus L DO
    Markiere den gematchten Knoten aus x
    FOR ALL y , y ist Unterbaum von T DO
      NewE:= {x|enthält(y,x) ∈ RL ∧ y ∈ E ∧ y unmarkiert}
      Q:= Füge x in Matching-Schritt-1(y,NewE,U) ein
      A:=A+Q
    END FOR
    Lösche die Markierung von x
  END FOR
RETURN A
END

```

Ein Unterbaum von T ist hierbei der Teilbaum von T, der einen Sohn von W als Wurzel hat und W selbst nicht enthält. Die Zahl der möglichen Unterbäume von T ist gleich der Anzahl der Söhne von W.

Abbildung 3.18: Algorithmus für den ersten Abbildungsschritt



Abbildung  $k$ , die durch den ersten Schritt gefunden wurde, alle Lagerrelationen berechnet, die zwischen den Netzwerkknoten  $W(k)$  durch RL-Kanten modelliert wurden. Sei  $R_x$  die Relationenfunktion, die der Kante  $x \in RL$  zugeordnet ist, dann bezeichnet  $B_k$  die Menge der Kanten/Ähnlichkeitswert-Paare, die das Ergebnis des zweiten Abbildungsschrittes ist.

$$B_k := \{(x, w) \mid x \in RL, x = (a, b), a, b \in W(k), w = R_x(k^-(a), k^-(b))\}$$

Dabei ist zu beachten, daß die *enthält*-Relation nicht berechnet werden muß, da ihre Gültigkeit (d.h. Ähnlichkeitswert = 1) bereits durch den ersten Abbildungsschritt gewährleistet ist.

Das Ergebnis der Abbildung besteht demnach aus einer Menge von Knotenabbildungen  $k$  sowie der jeder Abbildung  $k$  zugeordneten Menge  $B_k$ .

### 3.4.2.3 Umrißklassifikation

Die Menge der im Netzwerk modellierten Verkehrszeichen ist nach ihren Konturen in Klassen eingeteilt. Eine Verkehrszeichenklasse bilden dabei diejenigen Zeichen, deren aus den CN-Knoten und RL-Kanten aufgespannte Subnetze des Verkehrszeichensubnetzes identisch sind. Ziel der Umrißklassifikation ist es, diese Verkehrszeichenklasse zu bestimmen.

Nachdem die Abbildungen eines Segmentbaumes auf das Netzwerk gefunden worden sind, werden die Ähnlichkeitswerte der einzelnen durch eine Abbildung gematchten Netzwerkkomponenten über CL-Kanten an die Verkehrszeichenknoten weitergegeben. Da jeder CN-Knoten und jede RL-Kante durch gewichtete CL-Kanten mit mehreren Verkehrszeichenknoten (SN) verbunden ist, wird jeder Ähnlichkeitswert an mehrere Verkehrszeichen weitergegeben. Jeder Verkehrszeichenknoten, dem über eine Kante ein Ähnlichkeitswert übermittelt wurde, wird in eine Menge von zu untersuchenden Knoten aufgenommen. An diesen Verkehrszeichenknoten findet nun eine Verknüpfung der Ähnlichkeitswerte und der Gewichte der Kanten, die die Ähnlichkeitswerte *transportiert* haben, statt. Dabei werden alle CL-Kanten, die von dem untersuchten Verkehrszeichen ausgehen in Betracht gezogen, d.h. liegt von einem CN-Knoten kein gefundener Ähnlichkeitswert vor, so wird dieser auf 0 gesetzt und geht mit in die Berechnung eines Ähnlichkeitswerts für das Verkehrszeichen ein. Eine Ausnahme bilden hier die bei der Abbildung ausgeblendeten PN-Knoten sowie die mit diesen verbundenen Kanten. Da sie für die Bestimmung der Verkehrszeichenklasse nicht von Bedeutung sind, werden sie in diesem Schritt nicht berücksichtigt.

**Bestimmung des Ähnlichkeitswertes** Der an dieser Stelle für die Verkehrszeichen berechnete Ähnlichkeitswert stellt die Ähnlichkeit der in einem Szenenseg-



ment gefundenen Konturen zu den modellierten Verkehrszeichenkonturen dar. Ein Verkehrszeichen soll jetzt in der bei der Definition eines Verkehrszeichensubnetzes angeführten Listenform dargestellt werden. Dabei werden die Komponenten des Verkehrszeichens, die Piktogramme oder die mit Piktogrammen verbundenen RL-Kanten darstellen, sowie die zugeordneten Gewichte aus den Listen entfernt. Den Elementen der Komponentenliste werden durch eine Abbildung Ähnlichkeitswerte zugeordnet. Betrachtet man anstelle der Komponentenliste die Liste der den Komponenten zugeordneten Ähnlichkeitswerte, so erhält man zu jedem Verkehrszeichen und jeder Abbildung zwei Listen  $n$  und  $w$ , wobei  $n = n_1, \dots, n_m$  die Liste von Ähnlichkeitswerten darstellt, und  $w = w_1, \dots, w_m$  die Liste der Gewichte der CL-Kanten ist, die die Ähnlichkeitswerte an die Verkehrszeichenknoten weitergeleitet haben. Um einen Ähnlichkeitswert der Verkehrszeichenstruktur zu einem Segmentbaum zu erhalten, müssen die beiden Listen verknüpft werden. In 3.4.2.7 folgen nähere Betrachtungen der möglichen Verknüpfung von Ähnlichkeitswerten und Gewichten. Hier wird eine Verknüpfung in Form einer normierten gewichteten Summe vorgenommen, die auch zur Verknüpfung mehrerer Lagerrelationen in 3.4.1.3 benutzt wurde<sup>55</sup>.

$$C_{\text{Umriß}}(n, w) = \frac{n_1 w_1 + \dots + n_m w_m}{\sum_{i=1}^m w_i}$$

Diese Ähnlichkeit zwischen Verkehrszeichen und Szenensegment wird für alle betroffenen Verkehrszeichen und alle Abbildungen berechnet. Dann werden diejenigen Verkehrszeichen ausgewählt, deren Ähnlichkeitswert einen Schwellwert überschreitet.

Die so erhaltenen möglichen Verkehrszeichen werden in Klassen zusammengefaßt. Der Ähnlichkeitswert einer Klasse wird durch den höchsten Ähnlichkeitswert eines zu der Klasse gehörenden Zeichens bestimmt. Diese Verkehrszeichenklassen bilden das Ergebnis der Umrißklassifikation.

#### 3.4.2.4 Piktogrammklassifikation

Die oben beschriebene Umrißklassifikation liefert eine Menge von möglichen Verkehrszeichenklassen. Jeder Klasse ist eine strukturelle Beschreibung eines Verkehrszeichens zugeordnet, wobei keine Piktogramme in dieser Beschreibung enthalten sind. Anhand der Piktogramme können die Verkehrszeichen einer Klasse unterschieden werden. Ihre übrigen Strukturen sind nach Definition einer Verkehrszeichenklasse identisch.

<sup>55</sup>Die normierte gewichtete Summe ist auch die Standardeinstellung zur Verknüpfung von Ähnlichkeitswerten, die vom Klassifikationsprogramm benutzt wird.

```
Umrißklassifikation(K)
Eingabe : Eine Menge von Abbildungen K
Ausgabe : Eine Liste von
Verkehrszeichenklasse/Ähnlichkeitswert/Abbildung-Tripeln
BEGIN
R:=()
FOR ALL k aus K DO
  Übermittle die durch k und  $B_k$  bestimmten Ähnlichkeitswerte
  über CN-Kanten an die SN-Knoten.
  FOR ALL n aus SN DO
    Bestimme den Vektor der Ähnlichkeitswerte n
    Bestimme den Gewichtsvektor w
    Berechne  $C_{Umriß}(n, w)$ 
    IF  $C_{Umriß}(n, w) > \text{Umriß-Schwellwert}$ 
      THEN  $R := R + (n, C_{Umriß}(n, w), k)$ 
  END FOR
Erg:= Fasse die Elemente aus R in Klassen zusammen
RETURN Erg
END
```

Abbildung 3.19: Algorithmus zur Umrißklassifikation

Betrachtet man das Subnetz des Verkehrszeichennetzwerks, welches aus den Knoten des Bildbereich  $W(k)$  einer Abbildung  $k$ , sowie den Kanten aus  $B_k$  besteht, so kann durch die Menge der zu überprüfenden Piktogramme eingeschränkt werden. Die Verkehrszeichen einer von der Umrißklassifikation gefundenen Verkehrszeichenklasse enthalten Piktogramme, deren PN-Knoten mit dem oben erwähnten Subnetz verbunden sind. Für jede gefundene Abbildung können somit die Piktogramme bestimmt werden, die es zu überprüfen gilt. Außerdem kann jedem Piktogramm seine (ungefähre) Position im Bild angegeben werden, wodurch die Aufgabe eines *Piktogrammklassifikators* vereinfacht wird. Dadurch, daß jeder Piktogrammknoten mit einem Netzwerkknoten des Typs CN verbunden ist, dem durch die Abbildung ein Formprimitiv zugeordnet wurde, kann aus den Koordinaten des Formprimitivs sowie den modellierten Lagerrelationen zwischen CN-Knoten und Piktogrammknoten, die ungefähre<sup>56</sup> Lage des Piktogramms im Bild berechnet werden.

Die Menge der zu untersuchenden Piktogramme sowie der ihnen zugeordnete Bildbereich können nun einer *Piktogrammklassifikationskomponente* übergeben werden, die den angegebenen Bildbereich näher untersucht. Als Ergebnis dieser Piktogrammklassifikation wird eine Menge  $P_k$  erwartet, in der jedem Piktogramm  $p$  eine Ähnlichkeit  $w$  zu dem mitgelieferten Bildbereich zugeordnet ist.

$$P_k = ((p_1, w_1), \dots, (p_n, w_n))$$

Durch eine Schwellwertoperation werden diejenigen Piktogramme aus  $P_k$  entfernt, die nicht erkannt werden konnten. Die zur Lokalisierung des Suchbereichs benutzten Kanten werden in  $B_k$  mit einem Ähnlichkeitswert von 1 eingefügt, sofern die durch sie mit dem Netzwerk verbundenen PN-Kanten in  $P_k$  sind.

Die für die Piktogrammerkennung verwendete Methode wird hier nicht festgelegt. Eine auf Pixelebene operierende Mustererkennungskomponente ist hier genauso denkbar wie eine symbolische Komponente, die nach einer Vergrößerung des interessanten Bereichs auf ähnliche Weise funktioniert wie die oben beschriebene Umrißklassifikation.

#### 3.4.2.5 Verkehrszeichenklassifikation

Der letzte Klassifikationsschritt faßt die beiden vorhergehenden Schritte zusammen. Aufgrund der vorliegenden Ähnlichkeitswerte für Knoten und Kanten wird den Verkehrszeichen, wie bereits bei der Umrißklassifikation, ein Ähnlichkeitswert zugeordnet. Dieser errechnet sich jetzt aus den Ähnlichkeitswerten aller durch die CL-Kanten mit einem Verkehrszeichenknoten verbundenen Netzwerkkomponenten.

<sup>56</sup> Es kann ein Bereich bestimmt werden, in dem sich das Piktogramm befindet.

Piktogrammklassifikation (k,vk)  
 Eingabe : Eine Abbildung k und eine Menge von  
 Verkehrszeichenklassen vk  
 Ausgabe : Eine Menge von Piktogramm/Ähnlichkeitswert-Paaren  
 BEGIN  
 P:= Liste der Piktogramme, die in Verkehrszeichen der Klassen vk  
 enthalten sind.  
 B:= Liste der Bereiche, in denen die Piktogramme aus P zu suchen  
 sind. (Ermittelt aus k)  
 R:= *Piktogrammklassifikator*(P,B)(Aufruf einer externen Prozedur)  
 RETURN R  
 END

Die Listen n und w sind jetzt nicht mehr auf die auf CN-Knoten und RL-Kanten beschränkte Verkehrszeichenbeschreibung beschränkt.

$$C_{\text{Zeichen}}(n, w) = \frac{n_1 w_1 + \dots + n_m w_m}{\sum_{i=1}^m w_i}$$

Für jede Abbildung k mit den zugeordneten Mengen  $B_k$  und  $P_k$  wird jetzt den bereits durch die Umrißklassifikation in ihrer Zahl eingeschränkten Verkehrszeichen ein Ähnlichkeitswert zugeordnet. Übersteigt dieser Ähnlichkeitswert einen für die Klassifikation vorgegebenen Schwellwert, so wird das Verkehrszeichen dem Szenenbereich zugeordnet.

Tritt der Fall ein, daß mehrere Verkehrszeichen den Schwellwert überschreiten, so stellt dies einen Konflikt dar, der in einem nächsten Schritt zu lösen ist. Damit ein solcher Fall überhaupt eintreten kann, muß es sich bei den beiden (oder mehreren) Verkehrszeichen um sehr ähnliche Verkehrszeichen<sup>57</sup> handeln. Abbildung 3.20 zeigt als Beispiel die Zeichen 237 und 254. Die strukturelle Beschreibung des Zeichens 237 ist Teil der Beschreibung des Zeichens 254. Betrachtet man die Strukturnetze der Verkehrszeichensubnetze der beiden Zeichen, so fällt auf, daß diese sich nicht enthalten. Anhand eines Beispiels sollen jetzt die durch die Ähnlichkeit der beiden Verkehrszeichen entstandenen Probleme verdeutlicht werden. In einer Verkehrsszene erscheine das Verkehrsschild 254. Durch eine Störung wird die äußere Kontur des Verkehrszeichens nicht erkannt, so daß der innere Kreis die Wurzel eines Segmentbaums bildet, welcher klassifiziert werden soll. Es wird eine Abbildung

<sup>57</sup> Wird der Schwellwert zu niedrig gewählt, so kann dieser Fall auch bei völlig verschiedenen Verkehrszeichen auftreten. Auf eine geeignete Wahl des Schwellwertes ist also zu achten.



Abbildung 3.20: Konflikt durch ähnliche Verkehrszeichen

auf das Netzwerk gefunden, die dem Kreis einen Eintrittsknoten zuordnet. Eine weitere Kontur wird nicht gefunden, so daß die Umrißklassifikation die Verkehrszeichenklasse liefert, deren äußere Kontur ein Kreis ist, und die keine weitere Kontur enthält<sup>58</sup>. Die Piktogrammklassifikation untersucht den Bereich innerhalb des Kreises und findet eine Ähnlichkeit zu dem Piktogramm *Fahrrad*. Gehen wir davon aus, daß für beide Komponenten, sowohl für den Kreis als auch für das Fahrrad, ein hoher Ähnlichkeitswert gemessen wurde, so wird der Ähnlichkeitswert für das Zeichen 237 sicherlich über dem Klassifikationsschwellwert liegen. Es bestünde also kein Grund, nach einer eventuell fehlenden äußeren Kontur zu suchen. In diesem Fall wäre das Klassifikationsergebnis sogar katastrophal, da das *erkannte* Zeichen eine völlig andere *Bedeutung* als das in der realen Szene vorhandene Zeichen hat!

Um das Problem zu lösen, kann dem Klassifikationsalgorithmus eine *Untersuchungstiefe* angegeben werden. Ist der *Matching-Schritt-1* mit der Knotenmenge  $E$  durchlaufen, so wird zusätzlich angenommen, daß die äußere Kontur nicht im Segmentbaum vorhanden ist und der Abbildungsalgorithmus erneut durchlaufen. Diesmal werden jedoch die Knoten  $\{x | \text{enthält}(y, x) \in RL \wedge y \in E\}$  anstelle der Knotenmenge  $E$  übergeben. Die *Untersuchungstiefe* gibt an, wie oft dieser Schritt wiederholt werden soll.

Durch diese Maßnahme wird zwar gewährleistet, daß die äußere Kontur des Zeichens gefunden wird, jetzt wird es jedoch zu einem Konflikt kommen. Dazu muß nur in Betracht gezogen werden, daß die Menge der Abbildungen sich nur vergrößert hat, also daß keine Abbildung weggefallen ist. Damit wird weiterhin das Verkehrs-

<sup>58</sup>Ist der Schwellwert sehr niedrig gewählt, so kann auch noch die Klasse der Verkehrszeichen mit zwei ineinanderliegenden Kreisen als Konturen gefunden werden. Dies wird hier zunächst ausgeschlossen.

zeichen 237 klassifiziert, hinzu kommt die Klassifikation des Zeichens 254. Eine mögliche Lösung des Konfliktes wäre der direkte Vergleich der Ähnlichkeitswerte. Bei näherer Betrachtung erweist sich diese Konfliktlösungsmethode jedoch nicht als *fair*. Durch die größere Komponentenzahl der Zeichens 254 wird das Zeichen anfälliger für Störungen. Da die Komponenten des Zeichens 237 auch Komponenten des Zeichens 254 sind, wirkt sich nur die Störung der zusätzlichen Komponenten des Zeichens 254 beim Vergleich der beiden Verkehrszeichen negativ aus.

Eine zweite Lösungsmöglichkeit, die im vorliegenden System realisiert wurde, ist die Gewichtung der Verkehrszeichen. Diese Gewichtung wird nur im Konfliktfall in die Klassifikation einbezogen und hat ansonsten keine Bedeutung. Verkehrszeichen deren Beschreibung in der Beschreibung eines anderen Verkehrszeichens enthalten ist, bekommen ein niedrigeres Gewicht zugeordnet. Dadurch wird eine Bevorzugung der Verkehrszeichen erreicht, die eine umfangreichere Beschreibung haben, eine Fehlklassifikation wird dadurch jedoch nicht ausgeschlossen. Die Gewichtung kann automatisch erfolgen. Im ersten Schritt wird jedem Verkehrszeichen das Gewicht 1 zugeordnet. Die Beschreibung eines Verkehrszeichens kann in der Beschreibung eines zweiten Zeichens enthalten sein, dessen Beschreibung wiederum in der eines dritten enthalten ist. Die Länge einer solchen Kette bestimmt das Gewicht der in ihr enthaltenen Verkehrszeichenknoten. Eine *Initialgewichtung* wie sie durch den Algorithmus 3.21 beschrieben wird, wird durchgeführt bevor die Klassifikation gestartet wird.

Die Komponente Verkehrszeichenklassifikation ordnet jedem Segment, welches in einer Szene gefunden werden konnte, eine Verkehrszeichenklasse, ein Verkehrszeichen oder *not classified* zu. Die Funktion *Klassifikation* bearbeitet einen einzelnen Segmentbaum, während die Funktion *Verkehrszeichenklassifikation* das gesamte, in Form einer Liste von Segmentbäumen vorliegende, Bild betrachtet. Durch die Angabe von Untersuchungstiefen *Ut1*, *Ut2* und *Ut3* kann der Ablauf gesteuert werden. *Ut1* beschreibt, wie oft die äußere Kontur, d.h. der Wurzelknoten, eines Segmentbaumes entfernt werden soll, falls es für diesen keinen Match mit einem Eintrittsknoten des Netzwerks gibt. *Ut2* beschreibt, wie oft dies durchgeführt werden soll, auch wenn es einen Match gibt, um einen eventuell besseren Match nicht zu übergehen. Eine dritte Untersuchungstiefe *Ut3* gibt an, wie oft die Annahme getroffen werden soll, daß die äußere Kontur eines Verkehrszeichens aufgrund von Störungen nicht in den Segmentbaum aufgenommen wurde. Das oben angeführte Beispiel beschreibt den Nutzen der Angabe einer Untersuchungstiefe.

```
Gewichtung(K)
Eingabe: Eine Menge von Knoten des Typs SN
Effekt : Ordnet den Verkehrszeichenknoten ein Gewicht zu
BEGIN
  FOR ALL x aus K DO
    IF Beschreibung von x enthalten in der
      Beschreibung eines y aus K,  $y \neq x$ 
    THEN
       $g(x) := 0.95 * g(x)$ 
       $NewK := NewK + x$ 
    END FOR
  IF NewK ist nicht leer
  THEN Gewichtung(NewK)
  END
```

```
Initialgewichtung
BEGIN
  FOR ALL x aus SN DO
     $g(x) := 1$ 
  END
  Gewichtung(SN)
END
```

Abbildung 3.21: Bestimmung der Gewichte der Verkehrszeichenknoten



```

Klassifikation(T,U,N,Ut3)
Eingabe : Ein Segmentbaum T, Umgebungsdaten U,
          Netzwerk N, Untersuchungstiefe Ut3
Ausgabe : Ein Tupel bestehend aus Segmentbaum T,
          Verkehrszeichen(klasse) und Ähnlichkeitswert
BEGIN
Nodes:=Eintrittsknoten von N
A:=()
RV:=()
RC:=()
FOR x:=1 TO Ut3 DO
  A:=A  $\cup$  Matching-Nodes-1(T, Nodes, U)
  Nodes:={x | enthält(y, x)  $\wedge$  y  $\in$  Nodes}
END FOR
Umrissklassen:= Umrissklassifikation(A)
IF Umrissklassen = ()
THEN RETURN ()
ELSE
  FOR ALL x aus Umrissklassen DO
    k := Abbildung von x
    vk := Verkehrszeichenklasse von x
    Pk := Piktogrammklassifikation(k,vk)
    Übermittle die durch k, Bk und Pk bestimmten
    Ähnlichkeitswerte über CN-Kanten an die SN-Knoten von N.
    FOR ALL v aus SN DO
      Bestimme den Vektor der Ähnlichkeitswerte n.
      Bestimme den Gewichtsvektor w.
      Berechne CZeichen(n, w)
      IF CZeichen(n, w) > Zeichen-Schwellwert
      THEN RV := RV + (V(v), CZeichen(n, w), k)
      ELSE
        IF Ähnlichkeitswert von x > Klassen-Schwellwert
        THEN RC := RC + x
      END FOR
    END FOR
  END FOR
  IF RV enthält ein Element
  THEN RETURN Erstes Element von RV
  ELSE
    IF RV  $\neq$  ()
    THEN
      Multipliziere die Ähnlichkeitswerte der Elemente
      aus RV mit den Verkehrszeichengewichten.
      Wähle das Element e mit dem größten Ähnlichkeitswert.
      RETURN e
    ELSE
      IF RC = ()
      THEN RETURN ()
      ELSE
        Wähle das e  $\in$  RC mit dem größten Ähnlichkeitswert.
        RETURN e
      END
    END
  END
END

```

Abbildung 3.22: Algorithmus zur Klassifikation eines Segments

```

Verkehrszeichenklassifikation(T,U,N,Ut1,Ut2,Ut3)
Eingabe : Eine Liste von Segmentbäumen T, Umgebungsdaten U,
          Netzwerk N, Untersuchungstiefen Ut1,Ut2 und Ut3
Ausgabe : Eine Liste von Zuordnungen zwischen Segmentbäumen und
          Verkehrszeichen oder Verkehrszeichenklassen
BEGIN
Erg:=()
IF T=() THEN RETURN Erg
WHILE T nicht leer DO
  t := erstes Element von T
  Lösche erstes Element von T
  k := Klassifikation(t,U,N,Ut3)
  IF (k leer  $\wedge$  Ut1 > 0)  $\vee$  Ut2 > 0
  THEN
    s := Liste der Unterbäume von t
    z := Verkehrszeichenklassifikation(s,U,N,Ut1-1,Ut2-1,Ut3)
    IF k leer
    THEN
      Erg := Erg  $\cup$  z
    ELSE
      IF  $\forall x \in z$ : Ähnlichkeitswert von k > Ähnlichkeitswert von z
      THEN Erg := Erg + k
      ELSE Erg := Erg  $\cup$  z
    IF k nicht leer
    THEN Erg := Erg + k
    ELSE Erg := Erg + (t,not classified,0)
  END WHILE
RETURN Erg
END

```

Abbildung 3.23: Algorithmus der Verkehrszeichenklassifikation

### 3.4.2.6 Erkennung und Rekonstruktion von Konturen

Um eine Abbildung zwischen Segmentbaum und Netzwerk zu finden, müssen die im Segmentbaum vorhandenen Formprimitive mit den Konturen verglichen werden, die den CN-Knoten des Netzwerks zugeordnet sind. Handelt es sich bei den Formprimitiven um aufgerissene, oder veränderte Strukturen, so müssen diese korrigiert bzw. rekonstruiert werden, so daß die Lagerrelationen berechnet werden können. Wenden wir uns zuerst der Erkennung von Formprimitiven zu.

**Erkennung von Formprimitiven** Will man ein Formprimitiv mit einer Kontur vergleichen, so muß zunächst eine gemeinsame Darstellungsform gefunden werden. Aus der Eckpunktbeschreibung des Formprimitivs kann eine Konturbeschreibung in Winkel/Längen-Form abgeleitet werden. Da den Formprimitiven *circular* und *circular.line* keine Konturbeschreibung dieser Form zugeordnet werden kann, werden diese zunächst aus der Betrachtung ausgeklammert.

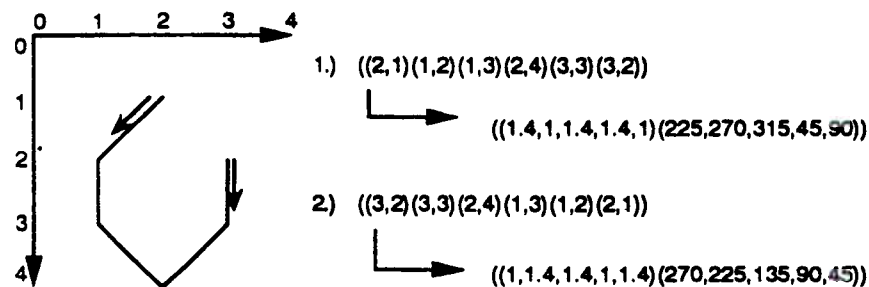
Bezeichne  $angle(line1, line2)$  den Winkel zwischen zwei Linien und wird eine Linie durch ihren Anfangs- und Endpunkt bestimmt, so kann eine Funktion  $f_\Gamma: \Gamma \rightarrow \Theta$  bestimmt definiert werden, die einem Formprimitiv  $A$  eine Konturbeschreibung in Winkel/Längen-Form zuordnet.  $\Gamma_G$  bezeichne die Menge der geschlossenen,  $\Gamma_O$  die Menge der offenen Formprimitive.

$$f_\Gamma(A) = \begin{cases} (f_{Length}(list.of.corners(A)), f_{Angle}(list.of.corners(A))) & A \in \Gamma_O \\ (f_{Length}(list.of.corners(A) + first(list.of.corners(A))), & A \in \Gamma_G \\ f_{Angle}(list.of.corners(A) + first(list.of.corners(A)))) & \end{cases}$$

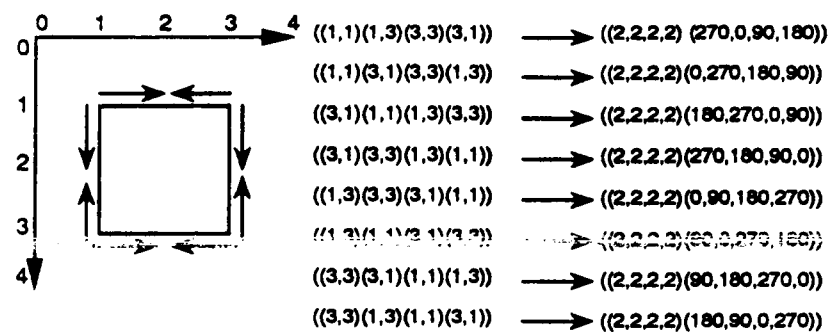
$$\begin{aligned} f_{Length}((x_1, y_1), \dots, (x_n, y_n)) &= \\ &(\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}, \dots, \sqrt{(x_{n-1} - x_n)^2 + (y_{n-1} - y_n)^2}) \\ f_{Angle}((x_1, y_1), \dots, (x_n, y_n)) &= \\ &(angle[(x_1, y_1), (x_2, y_2)][(0, 0)(1, 0)], \dots, angle[(x_{n-1}, y_{n-1}), (x_n, y_n)][(0, 0), (1, 0)]) \end{aligned}$$

Die Winkel und Kantenlängen der Musterkontur sind als Vektoren gespeichert, womit eine Reihenfolge gegeben ist. Da die Reihenfolge der Winkel und Kantenlängen der aus dem Formprimitiv gewonnenen Konturbeschreibung eine andere sein kann, muß das Muster *gedreht* und *gespiegelt* werden können. Abbildung 3.24 zeigt eine Kontur und verschiedene mögliche Beschreibungen. Die *Spiegelung* einer Konturbeschreibung entspricht der Änderung der Richtung in der die Kanten eines Objekts

**a) Mögliche Konturbeschreibungen für offene Primitive**



**b) Mögliche Konturbeschreibungen für geschlossene Primitive**



**Abbildung 3.24: Mögliche Winkel/Längen-Beschreibungen einer Kontur**

durchlaufen werden. Bei einer *Drehung*, die nur bei geschlossenen Objekten durchgeführt wird<sup>59</sup>, wird eine andere Kante ausgewählt, die als erste in die Konturbeschreibung eingehen soll.

Es soll jetzt eine Funktion  $ft_t: \Theta \rightarrow \Theta$  bestimmt werden, die eine Spiegelung und Drehung der Musterkonturbeschreibung durchführen kann. Durch den Index  $t$ ,  $t \in [-\|\alpha(x)\|, +\|\alpha(x)\|]$ , wird die Anzahl der Drehungen sowie die Richtung bestimmt.  $ft_t$  ist wie folgt rekursiv definiert:

$$ft_t([(l_1, \dots, l_n), (\alpha_1, \dots, \alpha_n)]) = \begin{cases} ft_{t-1}([(l_2, \dots, l_n, l_1), (\alpha_2, \dots, \alpha_n, \alpha_1)]) & t > 0 \\ [(l_1, \dots, l_n), (\alpha_1, \dots, \alpha_n)] & t = 0 \\ ft_{-t+1}([(l_n, \dots, l_1), \\ ((180 + \alpha_n) \bmod 360), \dots, (180 + \alpha_1) \bmod 360)]) & t < 0 \end{cases}$$

<sup>59</sup>Die für diese Anwendung benötigten Konturmuster gehören ausschließlich zu geschlossenen Objekten. Aus diesem Grund wird der Fall eines offenen Konturmusters nicht betrachtet. Anstelle einer Drehung würde hier das der erste Winkel und die erste Kantenlänge der Beschreibung weggelassen.

Es soll nun ein Ähnlichkeitsmaß definiert werden, welches die Ähnlichkeit zwischen einer Musterkontur und der Kontur eines Formprimitivs bestimmt. Dazu werden die Winkel- und Kantenlängenvektoren komponentenweise verglichen bis die Abweichung einen Schwellwert übersteigt.  $S_\alpha$  bezeichnet dabei eine maximal zulässige Winkelabweichung,  $S_l$  die maximal zulässige Abweichung von Kantenlängen. Die Funktion *diff* berechnet die Differenz zwischen zwei Winkeln<sup>60</sup>. Die normierten Längen der Musterkonturbeschreibung müssen an die Längen des Vergleichskandidaten angepaßt werden. Dabei dient der Quotient der Länge der zweiten Kanten als Streckungsfaktor<sup>61</sup>. Die Funktionen  $T_\alpha$  und  $T_l$  bilden die gemessene Distanz zwischen den Winkeln bzw. Kantenlängen auf einen Ähnlichkeitswert ab. Dabei sind  $T_\alpha$  und  $T_l$  Funktionen der Form  $T(x) = 1 - \left(\frac{x}{1+x}\right)^s$ . Außerdem erfolgt eine Gewichtung der beiden Vergleiche (Kantenlänge und Winkel) mit Hilfe der Faktoren  $w_\alpha$  und  $w_l$ . Durch diese Gewichtung, sowie durch die Wahl von  $s$  kann das Ähnlichkeitsmaß eingestellt werden<sup>62</sup>.

Ein Teil der aus dem Formprimitiv  $F$  gewonnenen Kontur kann zu einem Konturmuster  $M$  ähnlich sein (siehe dazu auch Abbildung 3.25). Die durch den Vektor  $\alpha(M)$  dargestellte Winkelfolge, kann eine Teilwinkelfolge von  $\alpha(F)$  sein. Die Musterkonturbeschreibung muß demnach nicht nur mit der kompletten Konturbeschreibung des Formprimitivs verglichen werden, sondern auch mit allen Teilkonturbeschreibungen.

$$\text{sim}_{\alpha,l}([\alpha_M, l_M], [\alpha_F, l_F]) =$$

$$\begin{aligned} \text{Max} \{ & \frac{\sum_{i=1}^{k-1} w_\alpha \cdot T_\alpha(\text{diff}(\alpha_{M,i}, \alpha_{F,i+o})) + w_l \cdot T_l(\text{abs}((l_{M,i} \cdot \frac{l_{F,2}}{l_{M,2+o}}) - l_{F,i+o}))}{w_\alpha \cdot \|\alpha_M\| + w_l \cdot \|l_M\|} \\ & | o \in [0, \|\alpha(F)\| - 1] \\ & k = \text{Min}\{j \mid \text{diff}(\alpha_{M,j}, \alpha_{F,j}) > S_\alpha \vee \text{abs}(l_{M,j} - l_{F,j}) > S_l \vee i + o > \|\alpha(F)\|\} \} \end{aligned}$$

Das oben definierte Ähnlichkeitsmaß  $\text{sim}_{\alpha,l}$  bestimmt die Ähnlichkeit zwischen zwei Vektorpaaren, dabei ist die Drehung und Spiegelung der Musterkonturbeschreibung noch nicht berücksichtigt worden. Berechnet man für jede mögliche Drehung und Spiegelung der Musterkonturbeschreibung die Ähnlichkeit bezüglich  $\text{sim}_{\alpha,l}$ , so ist die Ähnlichkeit zwischen Musterkontur und Formprimitiv durch den maximalen errechneten Ähnlichkeitswert gegeben. Das so erhaltene Ähnlichkeitsmaß wird in Form

<sup>60</sup>Die Winkeldifferenz zwischen 355 Grad und 5 Grad ist beispielsweise nicht 350 sondern 10 Grad. Maximale Winkeldifferenz ist 180 Grad.

<sup>61</sup>Da die erste Kante der Konturbeschreibung des Formprimitivs, falls es sich um ein aufgerissenes handelt, meist durch eine Störung verkürzt oder verlängert wurde, wird eine Kante ausgewählt, die als sicherer betrachtet wird.

<sup>62</sup>Weitere Ausführungen zur Steuerung der Klassifikation durch Schwellwerte, Gewichte und Transformationsfunktionen folgen in 3.4.2.7.

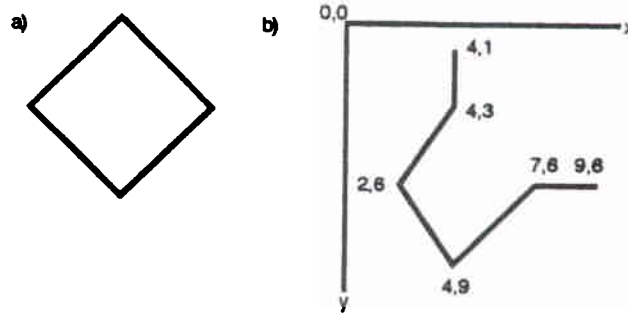


Abbildung 3.25: Vergleich von Formprimitiv und Konturmuster

einer Erkennungsfunktion gespeichert, die einem Paar, bestehend aus Musterkonturbeschreibung und Formprimitiv einen Ähnlichkeitswert zuordnet.

Bisher wurde die Kontur eines Kreises oder Kreisbogens aus der Betrachtung ausgeschlossen. Als Musterkontur tritt bei Verkehrszeichen<sup>63</sup> lediglich eine Kreiskontur auf, so daß hier auf den Vergleich zwischen zwei Kreisbögen verzichtet werden kann. Wird eine Kreiskontur mit einer Kontur verglichen, die aus einer Kantenfolge besteht, so wird die Ähnlichkeit zwischen den Vergleichsobjekten auf 0 festgelegt. Der Vergleich zwischen Kreisbogen und Kreis liefert die Länge des Kreisbogens relativ zur Länge der Kontur des vervollständigten Kreises.

**Definition : Konturerkennungsfunktion**

Sei  $\Gamma$  die Menge der Formprimitive und  $\Theta$  die Menge der Konturbeschreibungen, dann wird eine Funktion  $SP, SP: \Gamma \times \Theta \rightarrow [0,1]$ , als Konturerkennungsfunktion bezeichnet.

$$SP(A, M) = \begin{cases} \text{Max}\{sim_{\alpha,l}(\alpha(ft_t(M)), l(ft_t(M)), \alpha(fr(A)), l(fr(A))) \\ \quad | t \in [-||\alpha(M)||, +||\alpha(M)||]\} \\ \quad \text{für } M \neq \text{circular.shape} \wedge A \notin \Gamma_{\text{circular}} \cup \Gamma_{\text{circular.line}} \\ 1 \quad \text{für } A \in \Gamma_{\text{circular}} \wedge M = \text{circular.shape} \\ 0 \quad \text{für } A \in \Gamma_{\text{circular}} \cup \Gamma_{\text{circular.line}} \wedge M \neq \text{circular.shape} \\ \frac{\text{contour.length}(A)}{2 \cdot \pi \cdot \text{contour.length}(A)} \quad \text{für } A \in \Gamma_{\text{circular.line}} \wedge M = \text{circular.shape} \end{cases}$$

<sup>63</sup>Werden die Piktogramme mit in die Betrachtung einbezogen, so muß eine andere Form der Konturbeschreibung gewählt werden.

**Beispiel :**

Das Formprimitiv A (siehe Abbildung 3.25b) des Typs *open.polygonal* soll mit dem Muster M (siehe 3.25a) verglichen werden. Dazu wird die Eckpunktdarstellung des Formprimitivs zuerst in eine Winkel/Längen-Beschreibung umgeformt:

$$\text{corners}(A) = ((4, 1), (4, 3), (2, 6), (4, 9), (7, 6), (9, 6))$$

$$f_{\Gamma}(A) = ((2, 3.6, 3.6, 4.24, 2), (270, 236.6, 303.6, 45, 0))$$

Als Winkel-Längenbeschreibung des Musters M liegt folgende Form vor:

$$\text{temp}_M = ((1, 1, 1, 1), (45, 315, 225, 135))$$

Für dieses Beispiel seien die Transformationsfunktionen  $T_{\alpha}, T_l$  wie folgt gewählt:

$$\begin{aligned} T_{\alpha}(x) &= 1 - \left(\frac{x}{1+x}\right)^5 \\ T_l(x) &= 1 - \left(\frac{x}{1+x}\right)^3 \end{aligned}$$

Jetzt kann die Ähnlichkeit der, durch das Formprimitiv A dargestellten, offenen Kontur zu dem Muster M berechnet werden. Um die Rechnung abzukürzen wird nur die Anpassung  $ft_{-4}$  und die Berechnung von  $\text{sim}_{\alpha,l}$  mit  $\alpha = 1$  betrachtet<sup>64</sup>.

$$ft_{-4}(M) = ((1, 1, 1, 1), (225, 315, 45, 135))$$

Als Schwellwerte seien die Werte  $S_{\alpha} = 15$  und  $S_l = 5$  gewählt und es sei eine Gewichtung durch  $w_{\alpha} = 2$  und  $w_l = 1$  gegeben. Daraus ergibt sich für  $\text{sim}_{\alpha,l}$ :

$$\text{sim}_{\alpha,l}((1, 1, 1, 1), (225, 315, 45, 135), (2, 3.6, 3.6, 4.24, 2), (270, 236.6, 303.6, 45, 0))$$

$$\begin{aligned} &= \frac{2}{12} \cdot (T_{\alpha}(11.6) + T_{\alpha}(11.4) + T_{\alpha}(0)) \\ &\quad + \frac{1}{12} \cdot (T_l(0) + T_l(0) + T_l(0.64)) \\ &= \frac{0.34+0.34+1}{6} + \frac{1+1+0.94}{12} \\ &= 0.28 + 0.245 \\ &= 0.525 \end{aligned}$$

Es kann also eine Ähnlichkeit von 0.525 zwischen dem Szenenelement und der Musterkontur festgestellt werden.

<sup>64</sup>Diese liefert auch das Ergebnis mit der größten Ähnlichkeit.



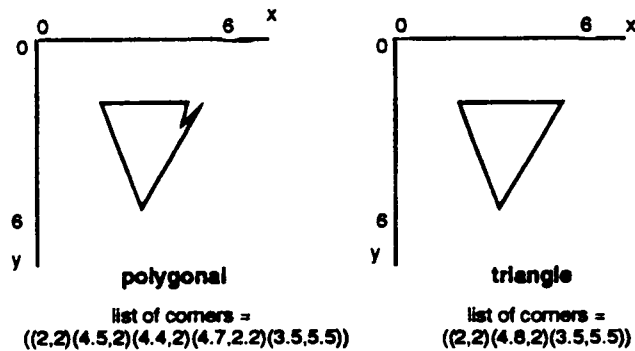


Abbildung 3.26: Konvertierung von Formprimitiven

**Rekonstruktion von Konturen** Die Wissensbasis enthält Knoten mit Konturinformation und Kanten, die Lagerrelationen speichern. Diese Lagerrelationen beziehen sich auf bestimmte Attribute von Formprimitiven (Mittelpunkt, Radius, Kantenlängen,...). Formprimitive verschiedenen Typs enthalten auch verschiedene Attribute, so daß eine Lagerrelation auf einem bestimmten Formprimitivtyp definiert ist. Es muß also dafür Sorge getragen werden, daß die Parameter der Lagerrelation vom richtigen Typ sind. Wird beispielsweise einer als Formprimitiv des Typs *polygonal* abgelegten Kontur eine hohe Ähnlichkeit zu einer Musterkonturbeschreibung *triangle.up* zugeordnet, so muß das Formprimitiv in ein neues Formprimitiv des Typs *triangle* konvertiert werden um später die Lagerrelation *double.up.triangle*<sup>65</sup> berechnen zu können (siehe auch Abbildung 3.26).

Betrachtet man aufgerissene Strukturen wie in Abbildung 3.25, so wird deutlich, daß bei der Berechnung einer Lagerrelation, die eine Kontur der Form des in Abbildung 3.25 dargestellten Musters erwartet, Berechnungen sehr kompliziert werden können<sup>66</sup>.

Bei der Abbildung eines Segmentbaums auf das Verkehrszeichennetzwerk, wird deshalb nicht das Formprimitiv in die Bindungsmenge aufgenommen, welches sich im Segmentbaum befindet, sondern eine korrigierte, gegebenenfalls konvertierte Kopie. Als Eingabe einer Lagerrelation liegen also stets Formprimitive des vorgeschriebenen Typs vor.

Die gestörte Kontur, die in einer Szene auftritt, liegt als Formprimitiv vor. Handelt es sich um eine offene Kontur, so ist das Formprimitiv vom Typ *open.polygonal* (oder *line*). Wie Abbildung 3.25 verdeutlicht, müssen in einem ersten Schritt die

<sup>65</sup>Die Lagerrelation legt das Größenverhältnis sowie den Abstand der (parallelen) Linien voneinander fest.

<sup>66</sup>Insbesondere müßte die Suche nach einer zu der erwarteten Kontur passenden Teilkontur der Eingabe, welche bereits bei der Berechnung der Ähnlichkeit durchgeführt worden ist, gegebenenfalls wiederholt werden.

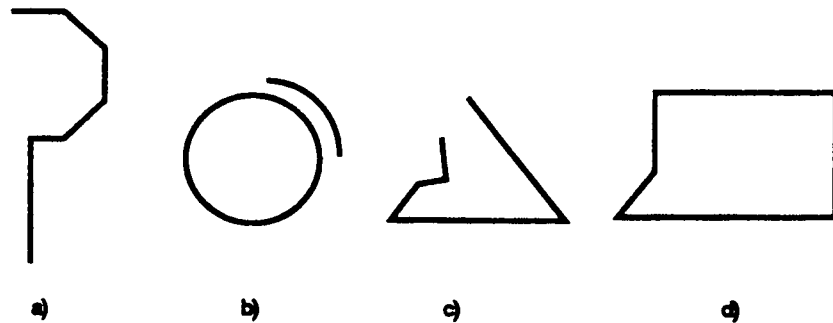


Abbildung 3.27: Veränderte Konturen

Teilkantenfolge bestimmt werden, die die größte Ähnlichkeit zu einer Musterkontur hat. Dies geschieht analog zur Berechnung der Ähnlichkeit zwischen Formprimitiv A und Muster M durch eine Funktion  $F_{fit}(A, M)$ , die diese Kantenfolge bestimmt<sup>67</sup>.

Die Kontur eines Objekts kann aus verschiedenen Gründen verändert, bzw. gestört werden. Abbildung 3.27 zeigt verschiedene mögliche Arten einer Veränderung der Kontur.

- 3.27a zeigt die aufgerissene Kontur eines Stoppschildes. Eine häufige Ursache für das Aufreißen von Konturen ist durch die Anbringung des Verkehrszeichens gegeben. An der Stelle treffen die Kontur des Schildes und die der Anbringung aufeinander. Die Bildvorverarbeitung muß jetzt entscheiden, mit welcher Linie die Kontur weitergeführt wird<sup>68</sup>.
- 3.27b zeigt eine Teilkontur eines Kreises, die zu einem Verbotsschild gehört. Die Kontur eines Kreises kann aufgebrochen werden indem Teilkonturen als Linie identifiziert werden oder, wie in diesem Fall anzunehmen, durch zu geringen Kontrast zum Hintergrund. Die Rekonstruktion eines Kreises aus dem Kreisbogen stellt jedoch kein größeres Problem dar.
- 3.27c zeigt ein aufgerissenes und in seiner Kontur verändertes Dreieck. Diese

<sup>67</sup> Bei der Berechnung der Ähnlichkeit zwischen Formprimitiv und Muster wurde die Kantenfolge gesucht, die den größten Ähnlichkeitswert liefert.  $F_{fit}$  liefert lediglich die gefundene Kantenfolge (als Liste von Eckpunkten), anstelle des dieser zugeordneten Ähnlichkeitswerts.

<sup>68</sup> Da kein Formprimitiv eine solche Verknüpfung von 3 oder mehr Linien an einem Punkt erlaubt, muß an dieser Stelle zwangsläufig aufgebrochen werden. Dabei wird noch kein Wissen über die Konturen von Verkehrszeichen verwendet, so daß das Aufbrechen mehr oder weniger willkürlich geschieht.



**Eine fehlende Kante :** Die übrigen Kanten stimmen in Ausrichtung und Länge mit den Kanten der Musterkontur (ungefähr) überein. Die Endpunkte der übereinstimmenden Kantenfolge können also einfach verbunden werden. Geht man von einer durch eine Störung hervorgerufenen Verkürzung oder Verlängerung der Anfangs- und Endkante des Kantenzuges aus, so kann man die fehlende Kante in dem durch die Musterkontur vorgegebenen Winkel an beiden Seiten ergänzen und die geeignetere Kante auswählen. Dabei sollen die vorhandenen Kanten möglichst nicht verkürzt werden. Die zweite Endkante wird dann verlängert<sup>69</sup>, um den Kantenzug zu schließen.

**Zwei fehlende Kanten :** An beiden Enden des Kantenzuges wird je eine neue Kante in dem durch die Musterkontur vorgegebenen Winkel eingefügt. Der Schnittpunkt dieser beiden Kanten ergibt den fehlenden Eckpunkt des Kantenzuges.

**Keine fehlende Kante :** Die beiden Endpunkte des Linienzuges werden gelöscht und die zwei jetzt fehlenden Kanten mit Hilfe der bekannten Winkel der gelöschten Kanten ergänzt. Wie im vorhergehenden Fall wird ein Schnittpunkt der Kanten gesucht und dieser als fehlender Eckpunkt in der Eckpunktbeschreibung der Kontur ergänzt.

Es sollen auch Konturen rekonstruiert werden können, von denen nur ein kleiner Teil der Kanten erkannt worden ist. Dabei wird der erkannte Kantenteil genutzt, um die Musterkontur zu fixieren. Um eine solche Rekonstruktion durchführen zu können, muß man von sehr sicheren erkannten Kanten ausgehen. Deshalb werden die erste und letzte Kante der Konturbeschreibung des zu rekonstruierenden Kandidaten gestrichen. Die übrigen Kanten sind aber bereits bestimmten Kanten der Musterkontur zugeordnet, so daß durch die Streichung der zwei Kanten die Eindeutigkeit der Rekonstruktion nicht in Frage gestellt wird. Es bleibt somit mindestens eine Kante, die in Richtung, Länge und konkreten Koordinaten in der aktuellen Szene festgelegt ist. Diese Information reicht aus, um das Muster zu fixieren. Die Kantenlängen werden gemäß der vorgegebenen Kantenlänge des Kandidaten gestreckt. Die Abweichung zwischen den Winkeln der Kante des Kandidaten und der zugeordneten Musterkante wird berücksichtigt<sup>70</sup>. Abbildung 3.29 zeigt Beispiele zu den angeführten Rekonstruktionsverfahren.

<sup>69</sup>Tritt der Fall ein, daß beide einfügbaren Kanten die jeweils andere Endkante des Kantenzuges schneiden, so wird diejenige Endkante gekürzt, bei der die nötige Kürzung geringer ausfällt.

<sup>70</sup>Dadurch werden die Winkel der erzeugten Kontur nicht mit denen des Musters übereinstimmen. Die Musterkontur wird also, je nach Abweichung der Winkel des Kandidaten, (leicht) gedreht.

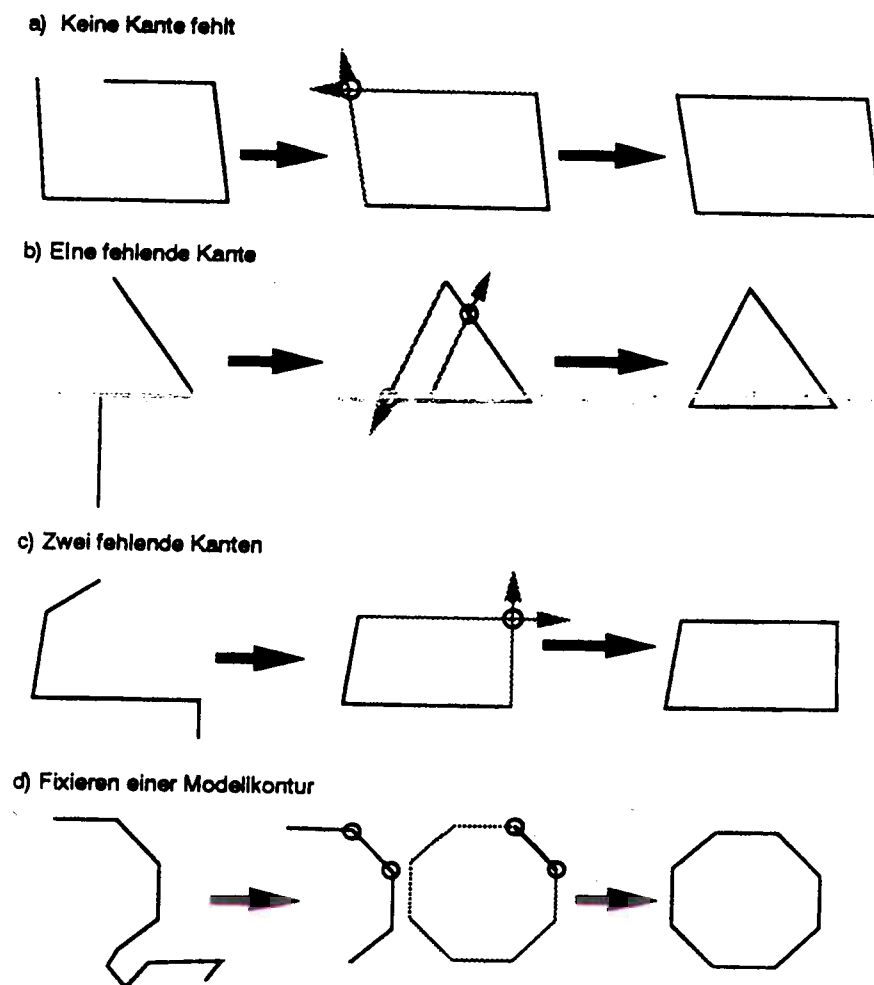


Abbildung 3.29: Methoden zur Rekonstruktion

Eine Rekonstruktionsfunktion, wie sie im System benötigt wird, berücksichtigt die vier oben angeführten Fälle. Grundsätzlich kann die zuletzt beschriebene Rekonstruktionsmethode in jedem Fall angewendet werden, doch diese orientiert sich stark an den durch das Muster vorgeschriebenen Konturen, wobei weniger auf die wirklich in der Szene vorhandenen Teilkonturen geachtet wird. Dies ist für die Rekonstruktion ausgehend von geringer Information nötig. Bei nur leicht gestörten Konturen wird das reale Bild stärker verfälscht als durch die vorher angeführten Methoden. Daher sind diese wenn möglich vorzuziehen.

**Definition : Rekonstruktionsfunktion**

Sei  $\Gamma$  die Menge der Formprimitive und  $\Theta$  die Menge der Konturbeschreibungen. Dann wird eine Funktion  $SR$ ,  $SR: \Gamma \times \Theta \rightarrow \Gamma$ , als Rekonstruktionsfunktion bezeichnet. Der folgende Algorithmus beschreibt den Rekonstruktionsvorgang:

```

Rekonstruktion(A,M)
Eingabe: Ein Formprimitiv A, ein Muster M
Ausgabe: Ein Formprimitiv
BEGIN
Bestimme  $SP(A,M)$  und  $F_{fit}(A,M)$ .71
t:= Der M zugeordnete Formprimitivtyp72
IF  $SP(A,M) < S_{match}$ 
THEN RETURN no match
ELSE
  IF A ist vom Typ circular.line
  THEN
    Erzeuge ein Formprimitiv B vom Typ circular.
    Initialisiere die Attribute von B durch
    übereinstimmende Attribute aus A.
    Setze den Wert von reliability auf  $SP(A,M)$ 
    RETURN B
  ELSE
    pl:= $F_{fit}(A,B)$ 
    kb:= $F_{\Gamma}(A)$ 
    CASE
      es fehlt eine Kante:
        Ergänze eine Halbgerade an beiden Enden
        der Kontur in dem durch M vorgegebenen Winkel.

```

<sup>71</sup> Dies wird in der Implementierung in einem Schritt gemacht.

<sup>72</sup> Jedem Konturmuster ist ein Formprimitivtyp zugeordnet, der die Konturbeschreibung darstellen kann.

```

    Verlängere die jeweils gegenüberliegende
    Endkante der Kontur bis zum Schnittpunkt mit der
    Halbgeraden.
    Wähle den Schnittpunkt, der durch die
    geringere Verlängerung zustande kommt.
    Ersetze den Endpunkt der verlängerten Kante
    in  $pl$  durch den Schnittpunkt.
  es fehlen 2 Kanten:
    Ergänze eine Halbgerade an beiden Enden
    der Kontur in dem durch  $M$  vorgegebenen Winkel.
    Bestimme den Schnittpunkt  $s$  der Halbgeraden.
     $pl := pl + s$ 
  es fehlt keine Kante:
    Lösche den ersten und letzten Punkt aus  $pl$ .
    Ergänze eine Halbgerade an beiden Enden
    der Kontur in dem durch  $kb$  vorgegebenen Winkel.
    Bestimme den Schnittpunkt  $s$  der Halbgeraden.
     $pl := pl + s$ 
  ELSE
     $lf :=$  Der Quotient aus  $l_2(kb)$  und der
    zugeordneten Musterkantenlänge.
     $af :=$  Die Winkeldifferenz zwischen
     $\alpha_2(kb)$  und dem zugeordneten
    Musterwinkel.
    Multipliziere die Kantenlängen des Musters
    mit  $lf$ .
    Addiere  $af$  zu allen Winkeln des Musters.
     $pl :=$  Berechne die Eckpunktdarstellung des
    mit Hilfe des Anfangs- und Endpunktes
    der zweiten Kante aus  $kb$  ( $pl_2$  und  $pl_3$ ).
  END CASE
  Erzeuge ein Formprimitiv  $B$  des Typs  $t$ .
  Initialisiere das Attribut list.of.corners mit  $pl$ .
  Setze den Attributwert von reliability auf  $SP(A,M)$ .
  Berechne die weiteren Attributwerte.
  RETURN  $B$ 
END

```

Ein Problem bei der Erkennung und Rekonstruktion von Konturen stellt die Drehung eines Objektes in der Szene dar. Steht ein Verkehrsschild etwas schräg<sup>73</sup>, so verändern sich die Winkel der Konturbeschreibung. Ist diese Veränderung so stark,

<sup>73</sup>Der gleiche Effekt wird erzielt, wenn die Kamera leicht schräg angebracht ist, und sich somit das Raster (Koordinatenkreuz) verschiebt.



daß die Winkelabweichung den Schwellwert übersteigt, so kann das Verkehrszeichen nicht mehr erkannt werden.

### 3.4.2.7 Kalibrierung

Die Klassifikation kann durch die Wahl von Schwellwerten, durch die verwendeten Ähnlichkeitsmaße und Transformationsfunktionen, sowie durch die Gewichtung der Komponenten eines Verkehrszeichens gesteuert werden. Es stellt sich also die Frage nach einer besonders günstigen Wahl dieser Größen. Im Folgenden sollen die einzelnen Einstellungsmöglichkeiten sowie deren Auswirkung erläutert werden.

**Berechnung der Ähnlichkeit** Bei der Abbildung eines Segmentbaums auf das Verkehrszeichennetzwerk werden verschiedene Ähnlichkeitsmaße benutzt, um die Ähnlichkeit zwischen zwei Konturen, zwischen Lagerrelationen, sowie zwischen dem Segmentbaum selbst und einem modellierten Verkehrszeichen zu berechnen.

Die Ähnlichkeit zwischen Konturen wird berechnet, indem die Abweichungen zwischen Winkeln und Kantenlängen der Konturen gemessen werden. Durch Transformationsfunktionen wird jeder Abweichung ein Ähnlichkeitswert zugeordnet. Aus den einzelnen Ähnlichkeitswerten und einer Gewichtung wird die Ähnlichkeit zwischen den Konturen bestimmt. Auch bei der Berechnung der Ähnlichkeit zwischen Lagerrelationen werden Ähnlichkeiten von einfachen Lagerrelationen gewichtet und zu einem Gesamtwert verrechnet. Der Ähnlichkeitswert eines Verkehrszeichens zu einem Szenensegment wird auf die gleiche Weise, durch Verknüpfung der Ähnlichkeitswerte der Komponenten des Verkehrszeichens, bestimmt.

Jedesmal muß eine Liste (der Länge  $n$ ) von Ähnlichkeiten  $a$  mit einer Liste von Gewichten  $w$  verrechnet werden, um wieder einen Ähnlichkeitswert zu erhalten. Dazu gibt es verschiedene Möglichkeiten. Zwei Mögliche Verknüpfungen sollen hier vorgestellt und verglichen werden.

**Die normierte gewichtete Summe** Die Elemente der beiden Listen  $a$  und  $w$  werden paarweise miteinander multipliziert und aufsummiert. Das Ergebnis wird durch die Summe der Gewichte von  $w$  dividiert.

$$C_1(a, w) = \frac{\sum_{i=1}^n a_i \cdot w_i}{\sum_{i=1}^n w_i}$$

Die Funktion  $C_1$ , die eine normierte gewichtete Summe berechnet liefert Werte im Bereich  $[0,1]$ . Dabei kann der Wert 0 nur dadurch erreicht werden, daß für jedes  $i$  entweder  $a_i$  oder  $w_i$  gleich 0 ist<sup>74</sup>. Genauso ist der Wert 1 nur dadurch zu

<sup>74</sup> Negative Gewichte sind nicht zugelassen, so daß es nicht zu Aufhebungen kommen kann.

erreichen, daß für alle  $i$   $a_i$  den Wert 1 haben. Außerdem wird ausgeschlossen, daß alle  $w_i$  den Wert 0 annehmen.

**Die Verknüpfungsfunktion nach Valdes** Valdes beschreibt eine Methode zur gewichteten Verknüpfung von Evidenzen<sup>75</sup> in Produktionsregeln. Jeder Evidenz wird durch einen *evidence ranking vector*  $r$ ,  $r_i \in \text{Nat}^+$ , ein Gewicht zugeordnet. Zur Verknüpfung der Evidenzen, dargestellt durch einen Vektor  $w$ , und der Gewichtungen wird ein Isomorphismus  $f: (-1,1) \rightarrow \text{REAL}$  bestimmt, mit dessen Hilfe *r-fractions* von  $w$  durch  $f^{-1}(\frac{f(w_i)}{r_i})$  bestimmt werden. Die Evidenz des Elements der Prämisse, welches die niedrigste *r-fraction* besitzt, wird zur Evidenz der gesamten Prämisse<sup>76</sup>.

Benutzt man diese Methode für die Verknüpfung von Ähnlichkeitswerten, so ergibt sich die Berechnung:

$$C_2(a, n) = a_i \text{ mit } f^{-1}\left(\frac{f(a_i)}{w_i}\right) = \text{Min}\left\{f^{-1}\left(\frac{f(a_i)}{w_i}\right) \mid i = 1, \dots, n\right\}$$

Im Gegensatz zu den Evidenzen, können die Ähnlichkeiten nur Werte aus  $[0,1]$  annehmen. Damit wird ein negatives Ergebnis von  $C_2$  ausgeschlossen.  $C_2$  liefert den Wert 0 sobald ein  $a_i$  den Wert 0 annimmt und den Wert 1, falls die zugeordnete *r-fraction* minimal ist<sup>77</sup>.

Beide Verknüpfungsmethoden sind alternativ im System einsetzbar. Dabei eignet sich die Methode der normierten gewichteten Summe besser für diese Anwendung. Es kann häufig vorkommen, daß eine Komponente der Struktur eines Verkehrszeichens nicht erkannt werden kann, also dem entsprechenden Netzwerkknoten ein Ähnlichkeitswert von 0 zugeordnet wird. Werden die übrigen Komponenten sicher erkannt, so kann trotz der fehlenden Komponente auf das Verkehrszeichen geschlossen werden. Durch eine Verknüpfungsfunktion nach Valdes würde dem Verkehrszeichen jedoch ein Ähnlichkeitswert von 0 zugeordnet werden. Liegen dem System bessere Eingangsdaten vor und werden die Verkehrszeichen gewichtet, so kann auch die aus Valdes Behandlung von Prämissen von Produktionsregeln abgeleitete Verknüpfungsfunktion sinnvoll angewendet werden.

**Transformationsfunktionen** Eine Transformationsfunktion  $T$  ist eine bijektive, ordnungsinvertierende Abbildung mit  $T(0) = 1$ , die ein gegebenes Distanzmaß in ein

<sup>75</sup>Ein Sicherheitswert aus dem Intervall  $[-1,1]$ , der einem Element der Prämisse einer Regel zugeordnet wird.

<sup>76</sup>Weitere Ausführungen sind [Val91] zu entnehmen.

<sup>77</sup>d.h.  $a_i$  ist das höchste Gewicht  $w_i$  zugeordnet.

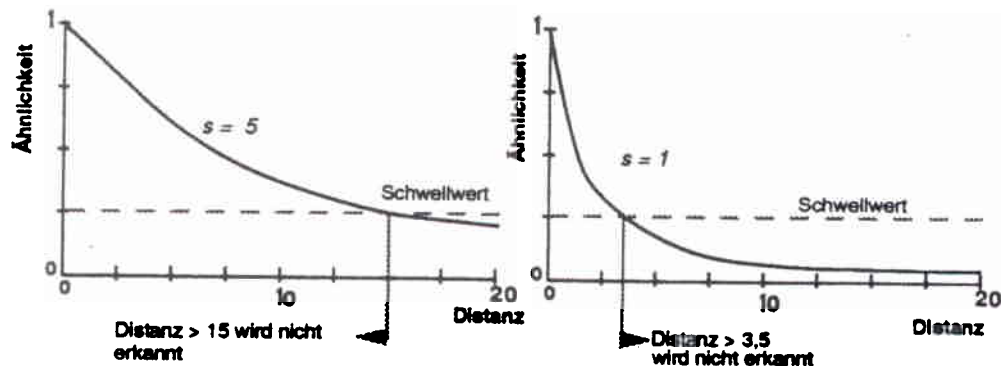


Abbildung 3.30: Toleranzeinstellung durch die Ähnlichkeitskurve

Ähnlichkeitsmaß transformiert. Zwei Arten von Transformationsfunktionen wurden benutzt:

1. Ist der maximale Abstand nicht bekannt oder ist der Abstand nicht begrenzt, so wird eine Transformationsfunktion der folgenden Form benutzt.

$$T(x) = 1 - \left( \frac{x}{1+x} \right)^s \quad s \in \text{Nat}^+$$

2. Ist der Abstand durch max begrenzt, so wird eine Transformationsfunktion des folgenden Typs verwendet.

$$T(x) = 1 - \left( \frac{x}{\max} \right)^s \quad s \in \text{Nat}^+$$

Durch den Parameter  $s$  kann festgelegt werden, wie stark die Ähnlichkeit mit wachsendem Abstand abnimmt. Damit kann die Toleranz eingestellt werden. Eine sehr steil abfallende Ähnlichkeitskurve (siehe auch Abbildung 3.30) bedeutet eine sehr geringe Toleranz. Bereits eine kleine Abweichung eines Kandidatens zum Vergleichsmuster würde die Zuordnung eines niedrigen Ähnlichkeitswerts zur Folge haben.

Durch die Möglichkeit differenzierter Einstellung der Toleranzbereiche kann das Ähnlichkeitsmaß feiner justiert werden. Werden zwei Abstandsmessungen A und B durchgeführt, und ist bekannt, daß bei Messung des Abstands A durch eine Störung häufig kleine Abweichungen verursacht werden, bei Messung des Abstands B jedoch meist korrekte Daten vorliegen, so kann durch eine größere Abweichungstoleranz bei der Transformation von A diesen unterschiedlichen Gegebenheiten Rechnung getragen werden.

**Schwellwerte** Der Klassifikationsvorgang wird durch eine Anzahl von Schwellwerten gesteuert. Ein Schwellwert bestimmt die Grenze bis zu der eine Kontur, Lagerrelation, Verkehrszeichenklasse oder ein Verkehrszeichen als *erkannt* eingestuft werden. Eine zweite Art eines verwendeten Schwellwertes ist die Begrenzung der Rekursion des Matching-Algorithmus. Diese Schwellwerte dienen zur Begrenzung der Rechenzeit. Die Schwellwerte im einzelnen:

1. Der *Match-Schwellwert* legt fest, ab welchem Ähnlichkeitswert eine Kontur *erkannt* wird. Damit kann eingestellt werden, welche Abweichung vom Muster noch akzeptiert werden kann. Dadurch, daß Konturen, die eine zu große Abweichung vom Muster aufweisen, nicht weiter betrachtet werden, wird durch diesen Schwellwert auch die Anzahl der betrachteten Abbildungen und damit die Rechenzeit eingeschränkt.
2. Die Schwellwerte  $S_l$  und  $S_\alpha$  bestimmen, welche Abweichung einer einzelnen Kante einer Kontur akzeptiert werden kann. Während durch eine zu große Wahl der Schwellwerte zu viele Konturen erkannt werden, werden durch zu klein gewählte Schwellwerte korrekte Konturen ausgeschlossen, die lediglich durch die Verzerrung des Originalbildes von dem Muster abweichen. Die Schwellwerte werden lokal für jedes Muster festgelegt, so daß beispielsweise bei einem regelmäßigen Achteck (Stopschild) eine geringere zulässige Winkelabweichung  $S_\alpha$  gewählt werden kann.
3. Der *Umriß-Schwellwert* bestimmt, ab welchem Ähnlichkeitswert eine Verkehrszeichenklasse als *wahrscheinlich* angenommen werden kann, so daß diese Klasse auf vorhandene Piktogramme untersucht werden kann. Durch die Wahl dieses Schwellwertes kann die Menge der zu untersuchenden Verkehrszeichenklassen eingeschränkt werden.
4. Der *Klassen-Schwellwert* bestimmt, ab welchem Ähnlichkeitswert eine Verkehrszeichenklasse erkannt wird, falls kein Verkehrszeichen erkannt werden kann. Dieser Schwellwert ist höher zu wählen als der Umriß-Schwellwert.
5. Der *Zeichen-Schwellwert* bestimmt, ab welcher Ähnlichkeit ein Verkehrszeichen als *erkannt* gilt. Dieser Schwellwert ist so zu wählen, daß Konflikte möglichst ausgeschlossen werden, d.h. so daß nur ein Verkehrszeichen den Schwellwert überschreiten kann.
6. Die Untersuchungstiefen  $Ut_1, Ut_2$  und  $Ut_3$  sind zur Beschränkung der Rechenzeit vorgesehen. Ihre Funktion wurde bereits in 3.4.2.5 beschrieben. Eine zu kleine Wahl der Werte von  $Ut_1, Ut_2$  und  $Ut_3$  führt dazu, daß nicht genügend

Untersuchungen durchgeführt werden, also manche Verkehrszeichen nicht erkannt werden können. Werden die Werte zu groß gewählt, wirkt sich das nur auf die Rechenzeit aus.

**Kantengewichte** Eine wichtige Einflußgröße für die Klassifikation bilden die Gewichte, die den Kanten des Typs CL zugeordnet sind. Von diesen Gewichten kann abhängen, ob eine Verkehrszeichenklasse weiter untersucht wird, ob sie klassifiziert wird, oder ob ein Verkehrszeichen klassifiziert wird. Es stellt sich die Frage, wie diese Gewichte zu wählen sind um eine sichere Klassifikation zu gewährleisten.

Das System stellt eine Heuristik zur Verfügung, mit deren Hilfe die Kantengewichte initialisiert werden können. Das Gewicht einer Kante stellt die Gewichtung des Ähnlichkeitswerts einer Komponente des Strukturnetzes bezüglich eines bestimmten Verkehrszeichens dar. Da Komponenten, die zu einem gewissen Grad *typisch* für ein Verkehrszeichen sind, für die Unterscheidung dieses Verkehrszeichens von anderen ausschlaggebend sind, werden diese mit einem hohen Gewicht versehen. Die Heuristik bestimmt für jede Komponente  $x$  des Strukturnetzes die Anzahl der CL-Kanten  $Z(x)$ , die diese Komponente mit einem Verkehrszeichen verbinden. Für jedes modellierte Verkehrszeichen  $v$  wird nun der  $\max = \text{Max} \{Z(x) | x \in CN_v \cup PN_v \cup RL_v\}$  bestimmt. Dann wird den CL-Kanten, die von einem Verkehrszeichenknoten ausgehen und diesen mit einer Komponente  $x$  des Strukturnetzes verbinden, das Gewicht  $\max-Z(x)+1$  zugewiesen. Aufgrund des höheren Informationsgehaltes von Lagerrelationen gegenüber Konturen und aufgrund der fast eindeutigen Zuordnung zwischen Piktogramm und Verkehrszeichen, werden die Gewichte der CL-Kanten, die auf Lagerrelationen oder Piktogramme verweisen, mit einem Faktor  $f_l$  bzw.  $f_{pikt}$  multipliziert. Eine Ausnahme bildet die Lagerrelation *enthält*. Da diese Relation Voraussetzung für die Berechnung der meisten<sup>78</sup> Lagerrelationen ist, wird ihr nur ein geringes Gewicht zugeordnet. Die Gewichte aller CL-Kanten, die auf *enthält*-Kanten verweisen wird auf 1 gesetzt.

Neben dieser automatischen Gewichtung können die Gewichte für jedes Verkehrszeichen explizit eingestellt werden.

<sup>78</sup> Es ist keine Relation implementiert, die nicht die *enthält* Relation voraussetzt. D.h. es sind nur Relationen zwischen Konturen definiert, die sich enthalten.