# Erzeugung kartenähnlicher Graphiken: XML-basierte Verdrängung und Platzierung von Punktsignaturen [1]

(mit 5 Bildern)

*Von Karl Neumann, Julia Petri und Christian Wolf, Braunschweig*

**ZUSAMMENFASSUNG:** Geodaten, die in GML kodiert sind, lassen sich mit der XML-basierten Transformationssprache XSLT so in Dokumente der Vektorgraphiksprache SVG abbilden, dass kartenähnliche Graphiken entstehen. Wie wir gezeigt haben, können dabei auch die durch den Signaturenkatalog SK25 gegebenen Spezifikationen berücksichtigt werden. Im vorliegenden Text skizzieren wir nun die Integration von Ansätzen der kartographischen Verdrängung in diesen rein XML-basierten Prozess. Dazu betrachten wir als Beispiele Straßen und Baumreihen und demonstrieren, wie deren Geometrien durch eine XSLT-Transformation relativ zu einander verdrängt und platziert werden können. Dabei stehen nicht die recht einfachen ausgewählten Verdrängungs- und Platzierungsalgorithmen im Vordergrund, sondern die Tatsache, dass diese ausschließlich mit XSLT realisiert wurden.

**ABSTRACT:** By means of the XML based transformation language XSLT, GML-encoded geo data can be transformed into documents of the vector graphics language SVG, resulting in graphics that bear some resemblance to maps. As we showed, also the specifications given by the signature catalog SK25 can be systematically incorporated into that transformation. In this paper we outline how to further integrate elements of cartographic displacement into this purely XML based process. For that purpose we consider roads and tree rows as examples and demonstrate, how their geometry can be displaced relatively to each other by using an XSLT transformation. Of course, the quite simple displacement algorithms are not in the center of attention, rather the fact that these were realized exclusively with XSLT.

## 1 Einleitung

Inzwischen setzen sich auch im Bereich der Geodaten und der rechnerbasierten Kartennutzung Techniken und spezielle Sprachen durch, die auf der Trägersprache XML aufbauen, wie etwa die XML-Anwendung GML (Geography Markup Language). GML ist ein offener Standard zum Austausch von Geodaten, der bereits in der Version 3 vom Open Geospatial Consortium herausgegeben wird (*OGC* 2005) und z.B. im ATKIS-ALKIS-AFIS-Projekt eingesetzt wird (*AdV* 2005). Mit einer weiteren XML-basierten Sprache, der Extensible Stylesheet Language for Transformation (XSLT) lassen sich alle Arten von XML-Dokumenten – und damit auch GML-Dokumente – in anders strukturierte XML-Dokumente überführen. Es lag daher der Gedanke nahe, in GML kodierte Geoobjekte

[1] Erscheint in *Mitteilungen des Bundesamtes für Kartographie und Geodäsie, Frankfurt M., 2006.*

mittels XSLT auf graphische Objekte so abzubilden, wie man es aus kartographischen Anwendungen gewohnt ist (*Neumann/Mathiak/Kupfer* 2004, *Neumann et al.* 2004). Als Zielsprache bietet sich dazu wiederum eine XML-basierte Sprache an, nämlich Scalable Vector Graphics (SVG). Die so generierten kartenähnlichen Graphiken lassen sich verbessern, indem vorhandene Signaturenkataloge wie etwa der SK25 für die Topographische Karte 1:25000 systematisch bei dem Transformationsprozess berücksichtigt werden (*Neumann/Kupfer/Mathiak* 2005).

Auf diesen Arbeiten aufbauend skizzieren wir im vorliegenden Text nun die Integration von Ansätzen der kartographischen Verdrängung in unseren rein XML-basierten Prozess. Dazu geben wir im nächsten Abschnitt einen kurzgefassten Überblick über das Verfahren, nur mit XML-basierten Sprachen Geodaten – genauer: ATKIS-Daten des Digitalen Landschaftsmodells DLM25 – sowohl zu modellieren als auch kartenähnlich zu visualisieren (eine Langfassung dieser Beschreibung ist z.B. in *Mathiak/Kupfer/Neumann* 2004 und *Mathiak/Kupfer/Neumann* 2005 zu finden). In Abschnitt 3 betrachten wir die GML-basierten Objektklassen Straßen und Baumreihen näher. Diese Objektklassen sind Beispiele dafür, wie es durch die Nichtbeachtung des kartographischen Kontextes zu unbefriedigenden Darstellungen kommen kann. Ein einfacher Ansatz zur Verdrängung der Baumreihen-Geometrien relativ zu den Straßen-Geometrien und dessen Implementierung mit XSLT wird daher in Abschnitt 4 vorgestellt, und in Abschnitt 5 diskutieren wir ein XSLT-basiertes Verfahren zur besseren Platzierung der einzelnen Baumsymbole. Im letzten Abschnitt fassen wir die Ergebnisse unserer Arbeit kurz zusammen und geben einen Ausblick auf weitere mögliche Aktivitäten.

Unsere Ausführungen in den Abschnitten 4 und 5 stellen im Wesentlichen eine stark komprimierte Sicht auf die Arbeiten von *Petri* 2005 und *Wolf* 2005 dar, auf die wir deshalb an dieser Stelle ausdrücklich verweisen wollen.

# 2 XML-basierte Erzeugung kartenähnlicher Graphiken

Ausgangspunkt für unsere XML-basierte Erzeugung kartenähnlicher Graphiken sind in GML modellierte und dann auch kodierte Objekte des DLM25. Zur Modellierung des Digitalen Landschaftsmodells mit GML haben wir einen komplexen Typ eingeführt, genannt “AtkisMemberType”, der als Exemplare “AtkisMember” enthalten kann. Wie vom GML-Rahmen vorgegeben, wurde “AtkisMemberType” als Featuretyp definiert, also vom GML-Typ “AbstractFeatureType” abgeleitet. Als Geometrie-Elemente passen hier gut die von GML vorgegebenen: “location” für Punkte, “centerLineOf” für Linien und “polygonMember” für Polygone. Wir haben zahlreiche weitere Klassen deklariert: So sind die einzelnen Objektarten des DLM25 wie “Straße” oder “Weg” in unserer Modellierung Subtypen des “AtkisMemberType”. Dieser Struktur folgend können Objektexemplare angegeben werden, z.B. konkrete Straßen, Grenzen oder Verwaltungsbezirke mit ihren jeweiligen Koordinaten.

Um solche Sammlungen von GML-kodierten Objekten des DLM25 auf Graphikobjekte abzubilden, benutzen wir die XML-Sprache XSLT. Sie ähnelt einer funktionalen Pro-

grammiersprache und überführt XML-Dokumente – und damit auch GML-Dokumente – in XML-Dokumente anderer Struktur (*Kay* 2001). Bei der Transformation wird nach dem Prinzip des "Pattern Matching" vorgegangen, d.h. das Quelldokument wird auf das Vorkommen von Mustern hin untersucht. Diese Muster werden in Form von XSLT-Templates vorgegeben. Trifft ein Muster zu, steht ebenfalls im jeweiligen Template, wie ein Stück des Zieldokumentes konstruiert werden soll.

Wir haben Templates für die verschiedenen Objektarten des DLM25 vorgesehen, also solche für Flüsse, Seen, Wälder, Wohnbauflächen usw. Die Templates für Objekte mit punktförmiger Geometrie rufen dann ein Template auf, das alle Arten von Punktsignaturen zeichnet, während für die linienförmigen Objekte ein anderes Template zuständig ist. Diesen Templates wird jeweils eine Style-Klasse als Parameter mitgegeben. Die verschiedenen Style-Klassen werden so in die erzeugten SVG-Zeichenbefehle eingebaut, dass z.B. für Bundesstraßen nacheinander zwei verschieden dicke und verschieden farbige Linien gezeichnet werden, die zusammen die für Straßen mit dieser Widmung gewünschte Liniensignatur ergeben. Die zahlreichen verschiedenen Style-Definitionen haben wir in Form von Cascading Style Sheets (*Meyer* 2002) vorher geeignet implementiert, wobei wie erwähnt die Vorgaben des Signaturenkatalogs SK25 (*AdV* 2002) berücksichtigt wurden. In Bild 1 ist ein Ausschnitt aus einer so generierten kartenähnlichen Graphik dargestellt. Es handelt sich dabei um die Visualisierung von Testdaten des Landesvermessungsamtes Niedersachsen.

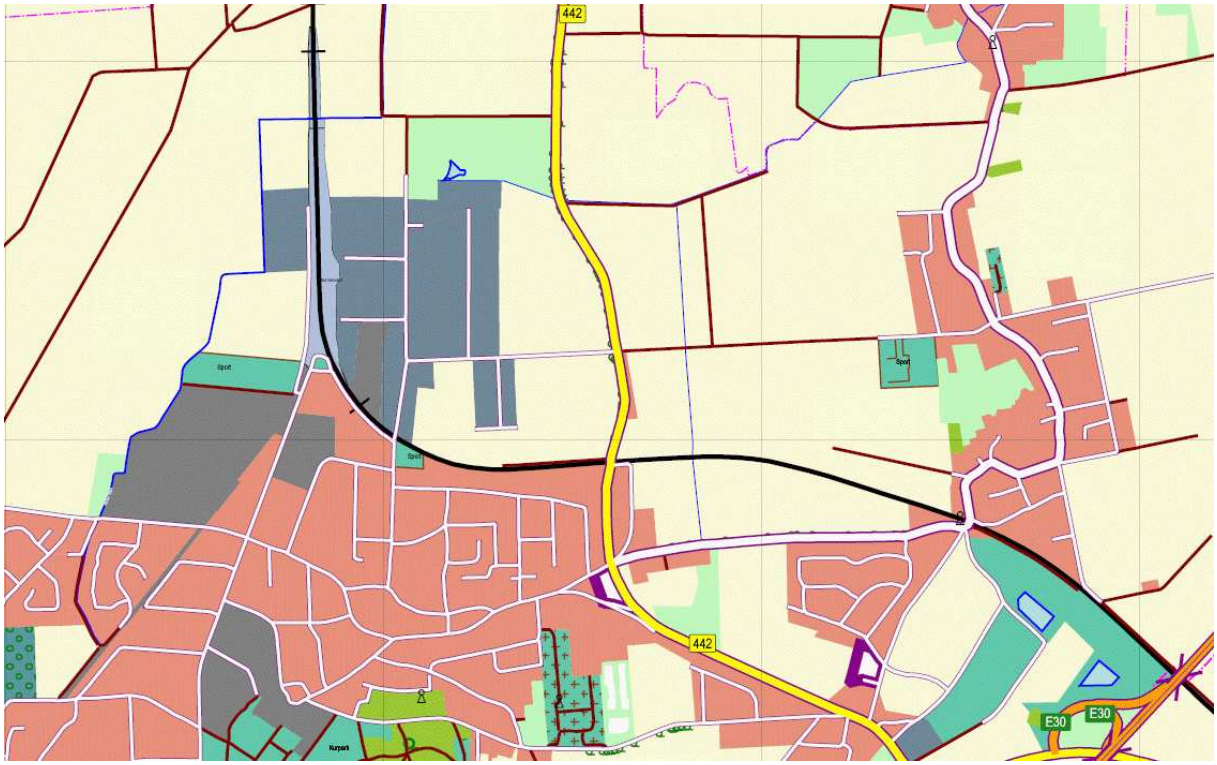


Bild 1 - Ausschnitt aus einer generierten Graphik

Ein Schwachpunkt unserer XML-basierten Erzeugung kartenähnlicher Graphiken ist, dass sämtliche Koordinaten der Graphikobjekte nur durch direkte Transformation der ur-

sprünglichen Koordinaten der DLM-Objekte entstehen. Die resultierenden Graphikobjekte werden zwar genau nach den im SK25 angegebenen Darstellungsprioritäten und auch mit der jeweils umgesetzten Signatur sowie Farbe generiert, es wird jedoch kein weitergehender kartographischer Kontext nebeneinander liegender Objekte beachtet. So kommt es in unseren Darstellungen vor, dass z.B. die Signaturen von Baumreihen durch eine linienhafte Straßensignatur teilweise verdeckt werden (vgl. Bild 1 Mitte: fast völlig durch die Darstellung der Bundesstraße 442 verdeckte Baumreihe). Hier müsste die weniger wichtige Baumreihensignatur nicht einfach durch die wichtigere Straßensignatur überzeichnet werden, sondern sie müsste vor dem Zeichnen verdrängt werden (vgl. z.B. *Hake/Grünreich/Meng* 2002).

# 3 Straßen und Baumreihen

Sowohl Straßen als auch Baumreihen haben als Objekte des DLM25 eine linienhafte Geometrie. In unserer GML-Repräsentation sind beide Subtypen von "AtkisMember" (vgl. Abschnitt 2), und ihre Liniengeometrie wird hier durch den Typ "centerLineOf" ausgedrückt. Straßen weisen neben der Geometrieangabe noch einen Namen auf und besitzen weitere inhaltliche Elemente, z.B. Zustand oder Widmung. Dabei ist das Widmungsattribut für die zu betrachtende Verdrängung von besonderer Bedeutung (vgl. Abschnitt 4). Das folgende Beispiel zeigt unsere GML-Kodierung eines Exemplars einer Gemeindestraße, wobei zahlreiche Stützpunkte der Liniengeometrie aus Platzgründen weggelassen wurden.

```
<AtkisMember>
  <Strasse>
    <gml:name> Rodenberger Strasse </gml:name>
    <gml:centerLineOf>
      <gml:coord> <gml:X> 3523771.360 </gml:X>
                  <gml:Y> 5797513.420 </gml:Y> </gml:coord>
      <gml:coord> <gml:X> 3523769.450 </gml:X>
                  <gml:Y> 5797519.730 </gml:Y> </gml:coord>
       ...
      <gml:coord> <gml:X> 3523733.730 </gml:X>
                  <gml:Y> 5797679.600 </gml:Y> </gml:coord>
    </gml:centerLineOf>
    <Attribute>
      <Zustand> in Betrieb </Zustand>
      <AnzahlDerFahrstreifen Bedeutung="tatsaechliche Anzahl"> 2
      </AnzahlDerFahrstreifen>
      <Funktion> Strassenverkehr </Funktion>
      <VerkehrsbedeutungInneroertlich> Anliegerverkehr
      </VerkehrsbedeutungInneroertlich>
      <Widmung> Gemeindestrasse </Widmung>
    </Attribute>
  </Strasse>
</AtkisMember>
```

Eine Baumreihe ist nach der Definition des DLM25 "eine reihenförmige Anordnung von Bäumen außerhalb von Wald oder Forst", und ihre GML-Struktur ist deutlich einfacher als die der Straßen, da Baumreihen neben der Geometrieangabe lediglich ein Vegetationsmerkmal als Attribut haben. Das folgende GML-Beispiel zeigt eine "Laubholz-Baumreihe"; es wurden wieder viele Stützpunkte weggelassen.

```
<AtkisMember>
  <Baumreihe>
    <gml:centerLineOf>
      <gml:coord> <gml:X> 3524258.170 </gml:X>
                  <gml:Y> 5800238.690 </gml:Y> </gml:coord>
      <gml:coord> <gml:X> 3524256.190 </gml:X>
                  <gml:Y> 5800220.270 </gml:Y> </gml:coord>
       ...
      <gml:coord> <gml:X> 3524581.650 </gml:X>
                  <gml:Y> 5799674.000 </gml:Y> </gml:coord>
    </gml:centerLineOf>
    <Attribute>
      <Vegetationsmerkmal> Laubholz </Vegetationsmerkmal>
    </Attribute>
  </Baumreihe>
</AtkisMember>
```

Zur Verbesserung unserer kartenähnlichen Graphiken ist es nun nötig, die Geometrien von Baumreihen, die einen zu geringen Abstand zu Geometrien von Straßen aufweisen, durch eine XSLT-Transformation zu ändern. Nach dieser Änderung können alle Objekte wie in Abschnitt 2 skizziert visualisiert werden, d.h. die Transformation von GML nach SVG wird von der Änderung der Baumreihen-Geometrien nicht berührt.

# 4 Verdrängung mit XSLT

Unser Verdrängungs-Algorithmus, den wir mit XSLT implementiert haben, ist recht einfach und naheliegend: Es werden alle in einem Datenbestand vorhandenen Baumreihenobjekte in der Reihenfolge ihres Auftretens dahingehend inspiziert, ob für das gerade betrachtete Baumreihenobjekt b eine Straße s existiert, deren Abstand zum Baumreihenobjekt b einen minimalen Abstand unterschreitet. Dabei hängt der Wert des minimalen Abstands von der Widmung der Straße s ab, da z.B. Bundesstraßen breiter dargestellt werden als Gemeindestraßen, die Baumreihen also weiter von Bundesstraßen als von Gemeindestraßen platziert werden müssen. Wenn solch ein Paar Baumreihenobjekt b und Straßenobjekt s gefunden wurde, wird die Baumreihe b verdrängt.

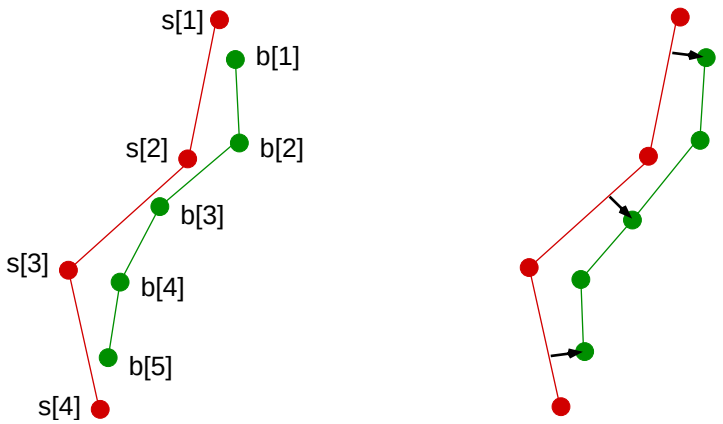


Bild 2 - Verdrängung von Baumreihen-Stützpunkten

Dazu werden alle Stützpunkte der Baumreihe b betrachtet, und es wird für jeden Stützpunkt ein Referenzsegment der Straße bestimmt. Rechtwinklig zu diesem Referenzsegment wird dann der jeweilige Stützpunkt der Baumreihe so weit verschoben bis der minimale Abstand erreicht ist. Bild 2 illustriert das Verfahren: Die roten Punkte und Strecken stellen die Geometrie einer Straße s dar, die grünen die einer Baumreihe b. Die Stützpunkte b[1], b[3] und b[5] unterschreiten den vorgegebenen minimalen Abstand zur Straßengeometrie, und die zugehörigen Referenzsegmente der Straße sind: für b[1] s[1]–s[2], für b[3] s[2]–s[3] und für b[5] s[3]–s[4]. Rechtwinklig zu diesen Segmenten der Straße werden die drei Stützpunkte b[1], b[3] und b[5] verschoben (Bild 2, rechts).

Obwohl bei unserem naheliegenden Verdrängungsalgorithmus nur einige wenig komplizierte Formeln und Verfahren umzusetzen sind, wie der Abstand eines Punktes von einer Geraden oder Winkelbestimmungen in Dreiecken, ist die Implementierung mit XSLT nicht zu einfach und zudem recht umfangreich. Dies liegt daran, dass XSLT eine deklarative, funktional-applikative Sprache ist. So gibt es etwa keine Schleifenkonstrukte, und Variablen können nur einmal bei deren Deklaration mit einem Wert versehen werden. Außerdem fehlen viele der aus konventionellen imperativen Programmiersprachen gewohnten eingebauten Funktionen. Einen Eindruck der Implementierung soll der folgende sehr kurze Ausschnitt aus dem XSLT-Code vermitteln. Das Template “PunktGeradeAbstand” berechnet den Abstand der durch die Punkte A und B festgelegten Geraden vom Punkt C. Die Gerade ist dabei eines der Straßensegmente, und der Punkt C ist ein Stützpunkt einer Baumreihe.

```
01 <template name = "PunktGeradeAbstand">
02  <param name = "A"/><!--Strasse-->
03  <param name = "B"/><!--Strasse-->
04  <param name = "C"/><!--Baumreihe-->
05  <variable name = "Ax" select = "$A/gml:X"/>
06  <variable name = "Ay" select = "$A/gml:Y"/>
07  <variable name = "Bx" select = "$B/gml:X"/>
08  <variable name = "By" select = "$B/gml:Y"/>
09  <variable name = "Cx" select = "substring-before($C, ' ')"/>
10  <variable name = "Cy" select = "substring-after($C, ' ')"/>
11  <variable name = "a">
12   <value-of select = "($Ay - $By) div ($Ax - $Bx)"/>
13  </variable>
14  <variable name = "b"> <value-of select = "-1"/> </variable>
15  <variable name = "c"> <value-of select = "$By - $a * $Bx"/> </variable>
16  <variable name = "tmp"> <value-of select = "$a*$a + $b*$b"/> </variable>
17  <variable name = "sqrtab">
18   <call-template name = "sqrt">
19    <with-param name = "number" select = "$tmp"/>
20    ...
21   </call-template>
22  </variable>
23  <variable name = "normierungsFaktor">
24   <if test = "$c &gt; 0"> <value-of select = "-1 * $sqrtab"/> </if>
25   <if test = "$c &lt; 0"> <value-of select = "$sqrtab"/> </if>
26  </variable>
27  <variable name = "cosPhi">
28   <value-of select = "$a div $normierungsFaktor"/>
29  </variable>
30  <variable name = "sinPhi">
31   <value-of select = "$b div $normierungsFaktor"/>
32  </variable>
33  <variable name = "d1">
34   <value-of select = "$cosPhi * $Ax + $sinPhi * $Ay"/>
```

```
35  </variable>
36  <variable name = "d2">
37   <value-of select = "$cosPhi * $Cx + $sinPhi * $Cy"/>
38  </variable>
39   <value-of select = "(1 - 2 * (($d2 - $d1) &lt; 0)) * ($d2 - $d1)"/>
40 </template>
```

Im ersten Teil (Zeilen 1–16) werden Variablen deklariert und Zwischenrechnungen durchgeführt. In den Zeilen 17–22, die hier aus Platzgründen nicht vollständig wiedergegeben sind, wird die Quadratwurzel des bis dahin ermittelten Wertes der Zwischenvariablen “tmp” berechnet, wozu das externe Template “sqrt” benutzt wird, das ein rekursives Verfahren zur Wurzelberechnung realisiert. Der verbleibende Teil (Zeilen 23–40) ermittelt dann den gesuchten Abstand unter Benutzung der Hesseschen Normalform der Geraden durch die Punkte A und B.

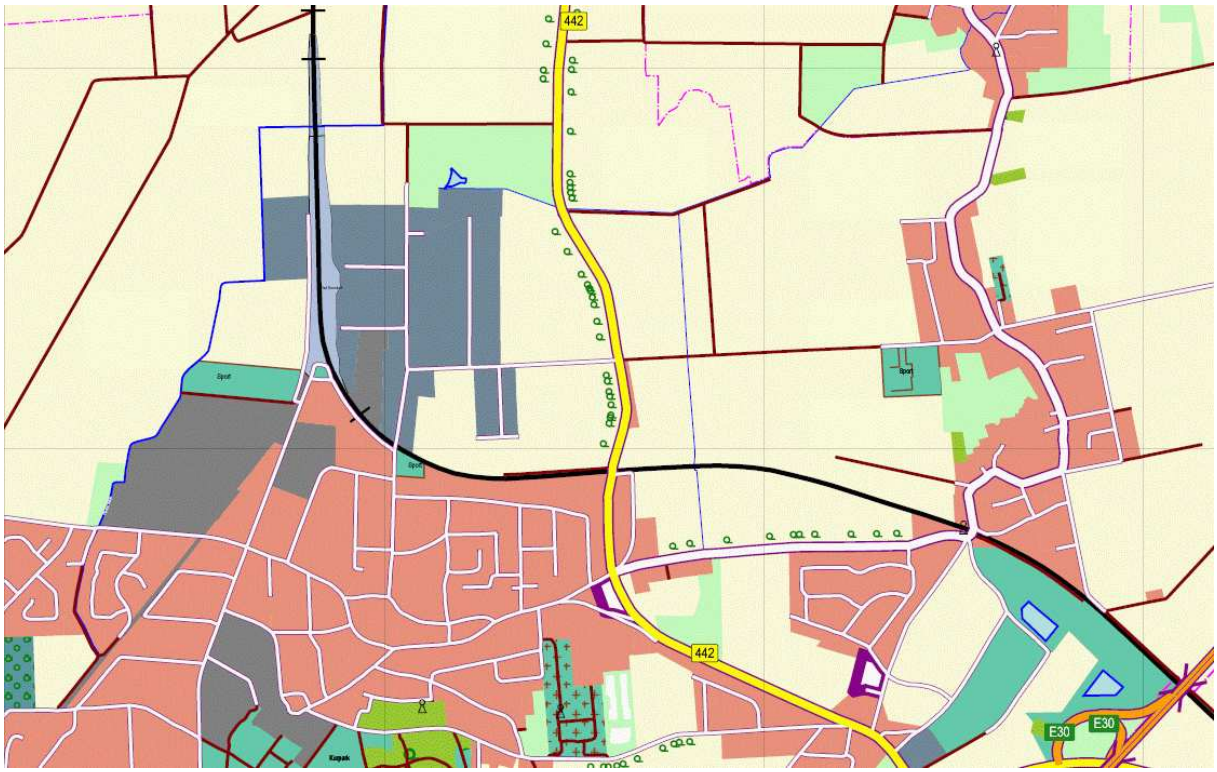


Bild 3 - Ausschnitt aus der generierten Graphik mit Verdrängung

Werden die so geänderten GML-Daten nun wie in Abschnitt 2 skizziert auf SVG-Befehle abgebildet, ergibt sich eine Darstellung wie in Bild 3: Die Baumsignaturen werden jetzt nicht mehr von den Straßensignaturen verdeckt, aber die einzelnen Baumsymbole werden nicht mit konstantem Abstand platziert, wie es an sich wünschenswert wäre und wie es auch vom Signaturenkatalog SK25 vorgegeben wird, sondern auf den Positionen der Stützpunkte der Baumreihen. Wir geben daher eine weitere XSLT-Transformation an, die nun die Baumsymbole mit regelmäßigem Abstand auf dem Linienverlauf von Baumreihen platziert.

# 5 Platzierung mit XSLT

Zur besseren Platzierung der Baumsymbole werden die Geometrien sämtlicher Baumreihenobjekte modifiziert, die in einem Datenbestand auftreten. Dazu wird jede Baumreihe b betrachtet, nun unabhängig davon, ob es in deren Nähe eine Straße gibt oder nicht. Zunächst wird für die aktuelle Baumreihe b deren Länge berechnet und damit und mit einer globalen Abstandskonstanten die Anzahl n der zu platzierenden Baumsignaturen. Danach werden die Koordinaten der n neuen Stützpunkte so ermittelt, dass sie jeweils den geforderten konstanten Abstand von einander haben und auf der Liniengeometrie der ursprünglichen Baumreihe b liegen. In Bild 4 wird das Verfahren illustriert: Oben sieht man eine gegebene Baumreihe b mit ihren 8 Stützpunkten b[1] bis b[8]. Darunter sind die neuen resultierenden Stützpunkte b'[1] bis b'[5] skizziert, die einen einheitlichen Abstand zu einander aufweisen und auf der gestrichelt gezeichneten Liniengeometrie der Ausgangsbaumreihe b liegen.

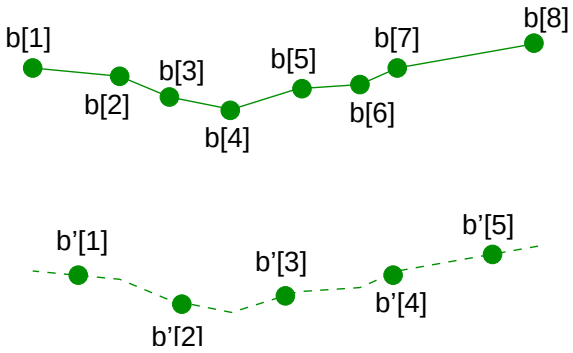


Bild 4 - Platzierung von Baumreihen-Stützpunkten

Die Implementierung des Platzierungsalgorithmus mit XSLT ist einfacher und weniger umfangreich als die des Verdrängungsverfahrens aus dem vorigen Abschnitt. Um einen Eindruck zu vermitteln, geben wir wieder einen kurzen Ausschnitt aus dem XSLT-Code an. Das unten aufgeführte Template "Punktsetzen" wird aufgerufen, nachdem feststeht, dass der nächste Stützpunkt der neuen Baumreihe b' im Abstand von "laenge" ausgehend vom Punkt b[i] auf dem Segment b[i]–b[i+1] der ursprünglichen Baumreihe b zu platzieren ist. Die Koordinaten dieses neuen Punktes werden dann mit Hilfe des Strahlensatzes berechnet.

```
01 <template name="Punktsetzen">
02  <param name="liste"/>
03  <param name="geslaenge"/>
04  <param name="laenge"/>
05  <if test="contains($liste, ' ')">
06   <variable name="p1" select="substring-before($liste, ' ')"/>
07   <variable name="p2" select="substring-after($liste, ' ')"/>
08   <variable name="x1" select="substring(substring-before($p1, ','), 2)"/>
09   <variable name="y1"
10             select="substring-before(substring-after($p1, ','), ')')"/>
11   <variable name="x2" select="substring(substring-before($p2, ','), 2)"/>
12   <variable name="y2"
13             select="substring-before(substring-after($p2, ','), ')')"/>
14   <variable name="dx" select="$x2 - $x1"/>
```

```
15   <variable name="dy" select="$y2 - $y1"/>
16   <variable name="verh" select="$laenge div $geslaenge"/>
17   <variable name="sx" select="$dx * $verh"/>
18   <variable name="sy" select="$dy * $verh"/>
19   <variable name="resX" select="$x1 + $sx"/>
20   <variable name="resY" select="$y1 + $sy"/>
21   <element name="gml:coord">
22    <element name="gml:X"> <value-of select="$resX"/> </element>
23    <element name="gml:Y"> <value-of select="$resY"/> </element>
24   </element>
25  </if>
26 </template>
```

Zunächst werden die Koordinaten der Punkte b[i] und b[i+1] aus der Zeichenkette “liste” extrahiert, die die Baumreihe b enthält, und in die lokalen Variablen “x1”, “y1”, “x2” und “y2” geschrieben (Zeilen 1–13). Danach erfolgen die Berechnungen nach dem Strahlensatz (Zeilen 14–20) und schließlich werden die Koordinaten des neuen Punktes ausgegeben (Zeilen 21–24).

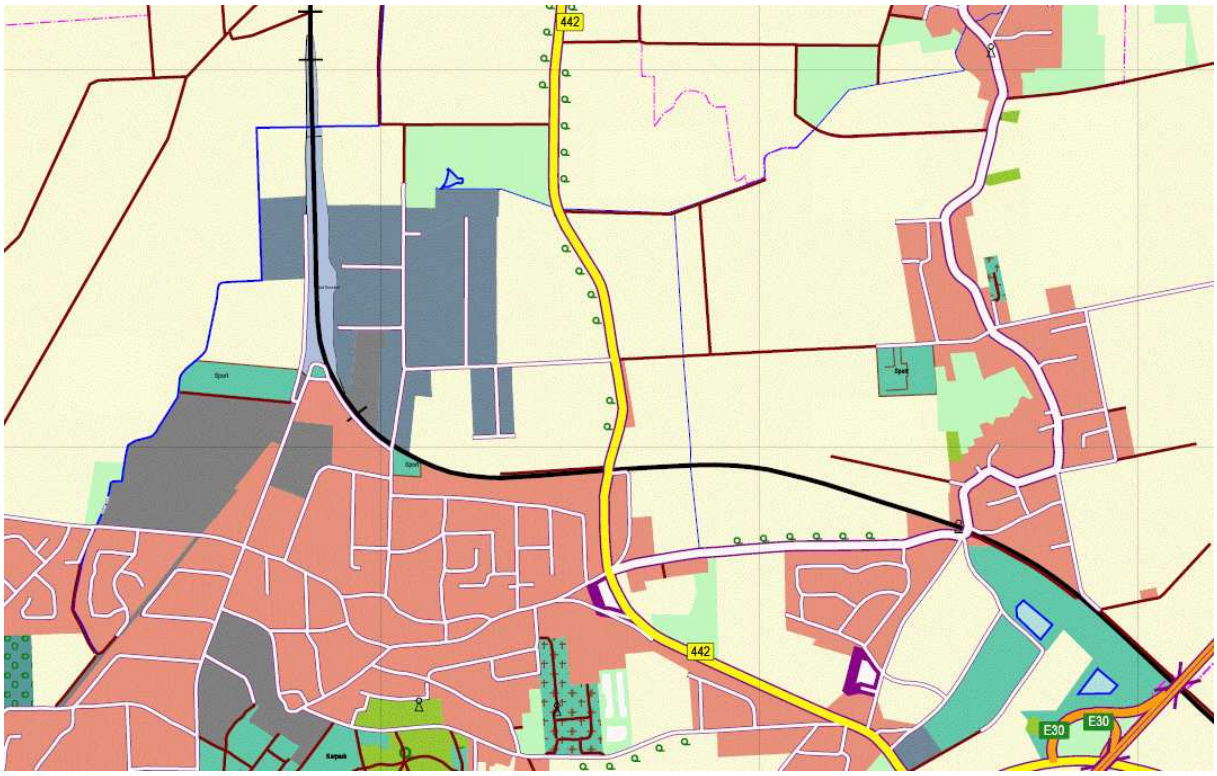


Bild 5 - Ausschnitt aus der generierten Graphik mit Verdrängung und Platzierung

Nach dieser weiteren Transformation der ursprünglichen GML-Daten ergibt die Visualisierung eine Darstellung wie in Bild 5 angegeben: Die Baumsignaturen sind nun äquidistant entlang von Straßensignaturen platziert.

# 6 Ausblick

Mit dem vorliegenden kurzen Text haben wir versucht zu zeigen, dass Ansätze der kartographischen Verdrängung in einen rein XML-basierten Prozess der Generierung kartenähnlicher Graphiken integriert werden können. Dazu haben wir unser bisheriges Verfahren der Visualisierung von mit GML kodierten DLM25-Daten um zwei weitere XSLT-Transformationen ergänzt: Die erste Transformation verdrängt nun Stützpunkte von Baumreihen, die zu dicht an Straßensegmenten liegen. Mit einer zweiten Transformation werden Stützpunkte von Baumreihen äquidistant platziert, und erst die dritte XSLT-Transformation führt die bereits bekannte Abbildung der GML-Daten auf SVG-Zeichenbefehle durch. Die implementierten Verdrängungs- und Platzierungsalgorithmen für Straßen und Baumreihen sind zwar recht einfach, lassen sich jedoch mit wenig Aufwand an zahlreiche andere Objektarten mit linienhafter Geometrie anpassen.

Der Ansatz, kartographische Verdrängung einfach als (weitere) XSLT-Transformation auf vorhandenen GML-kodierten Daten zu realisieren, hat sich als praktikabel erwiesen. Allerdings resultieren aus den an sich einfachen Algorithmen recht umfangreiche XSLT-Programme, was an dem funktionalen Charakter von XSLT liegt. Konventionelle imperative Programmiersprachen, wie etwa Java, wären hier deutlich einfacher einzusetzen und würden auch zu kürzeren Laufzeiten führen, wären aber nicht so interoperabel wie die XSLT-Programme.

# Literatur

*Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV)*: ATKIS-Signaturenkatalog 1:25000 (ATKIS-SK25). Version 4.1, Bonn, 2002.

*Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV)*: Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens (GeoInfoDok). Version 4.0, Bonn, 2005.

*Hake, G.; Grünreich, D.; Meng, L.*: Kartographie – Visualisierung raum-zeitlicher Informationen. de Gruyter, 8. Aufl., 2002.

*Kay, M.*: XSLT Programmer's Reference. 2. Auflage, Wrox Press, 2001.

*Mathiak, B.; Kupfer, A.; Neumann, K.*: Using XML Languages for Modeling and Web-Visualization of Geographical Legacy Data. In Proc. "VI Brazilian Symposium on GeoInformatics: GeoInfo 2004", Iochpe, C.; Camara, G. (Hrsg.), Instituto Nacional de Pesquisas Espaciais, Campos do Jordao 2004, pp. 265–280.

*Mathiak, B.; Kupfer, A.; Neumann, K.*: Modellierung und kartographische Visualisierung von Geodaten mit XML-basierten Sprachen. In Informatik – Forschung und Entwicklung, Band 20, Heft 1–2, pp. 24–32, 2005.

*Meyer, E.A.*: On CSS. Mastering the Language of Web Design with Cascading Style Sheets. New Riders, 2002.

*Neumann, K.; Ahlbrecht, P.; Eckstein, E.; Mathiak, B.; Kupfer, A.*: Visualization of Landscape Data in Digital Maps by Exclusive Use of XML-Based Languages. In Proc. "12th Int. Conf. on Geoinformatics", Brand, S.A. (Hrsg.), Gävle 2004, pp. 370–374.

*Neumann, K.; Kupfer, A.; Mathiak, B.*: Umsetzung des Signaturenkataloges SK25 bei der XML-basierten Erzeugung kartenähnlicher Graphiken. In Mitteilungen des Bundesamtes für Kartographie und Geodäsie, Band 34, Frankfurt M. 2005, pp. 107–118.

*Neumann, K.; Mathiak, B.; Kupfer, A.*: Modellierung und kartographische Visualisierung von Geodaten mit XML-basierten Sprachen. In Proc. "Modellierung 2004", Rumpe, B.; Hesse, W. (Hrsg.), Lecture Notes in Informatics P-45, 2004, pp. 93–107.

*Neumann, K.; Mathiak, B.; Kupfer, A.*: Der Einsatz von GML, XSLT und SVG am Beispiel von ATKIS-DLM-Daten. In Mitteilungen des Bundesamtes für Kartographie und Geodäsie, Band 31, Frankfurt M. 2004, pp. 97–107.

*Open Geospatial Consortium (OGC)*: Geography Markup Language (GML) 3.1.1. 2005, http://schemas.opengis.net/gml/3.1.1/

*Petri, J.*: Realisierung eines Beispiels der kartographischen Verdrängung mit XSLT. Studienarbeit, TU Braunschweig, 2005.

*Wolf, C.*: XSLT-gesteuerte Platzierung von Punktsignaturen auf Liniengeometrien am Beispiel der Objektklasse Baumreihe. Studienarbeit, TU Braunschweig, 2005.