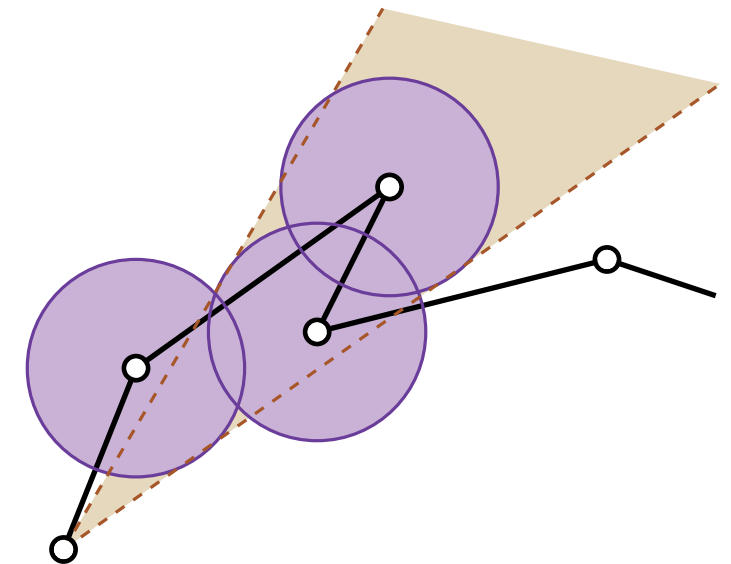
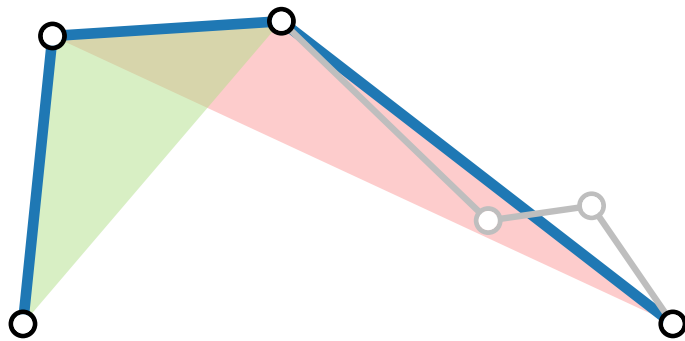


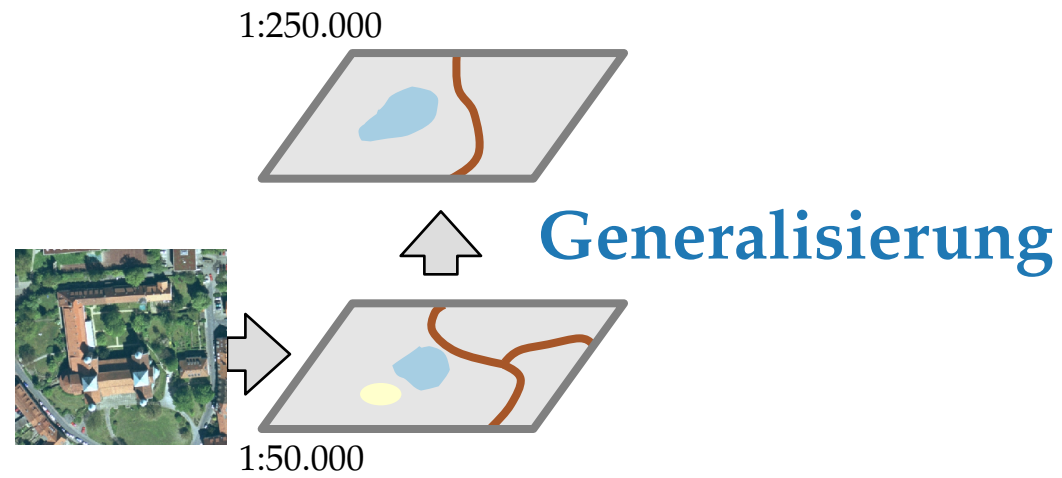
Algorithmen für geographische Informationssysteme

9. Vorlesung Linienvereinfachung

Teil I: Kartografische Generalisierung



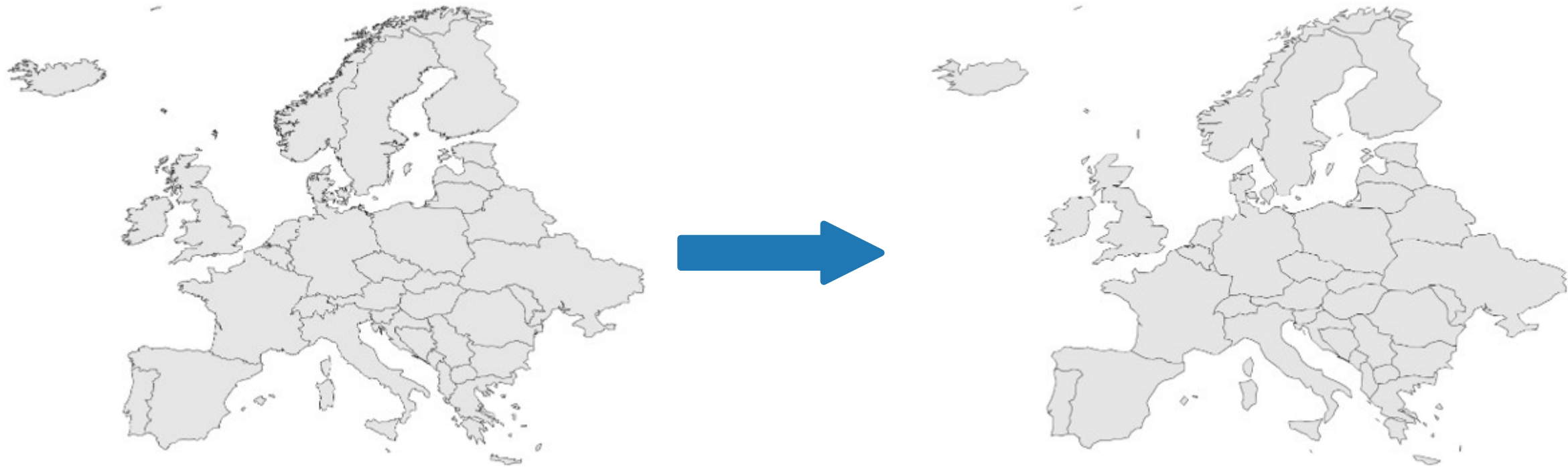
Kartografische Generalisierung



Generalisierung

1. Vereinfachung	2. Vergrößerung	3. Verdrängung	4. Aggregation	5. Auswahl	6. Klassifizierung Typisierung Symbolisierung	7. Bewertung

Linienvereinfachung



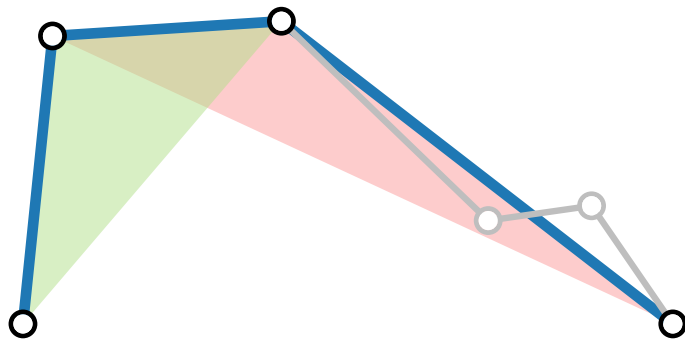
Demo.

<https://bost.ocks.org/mike/simplify>

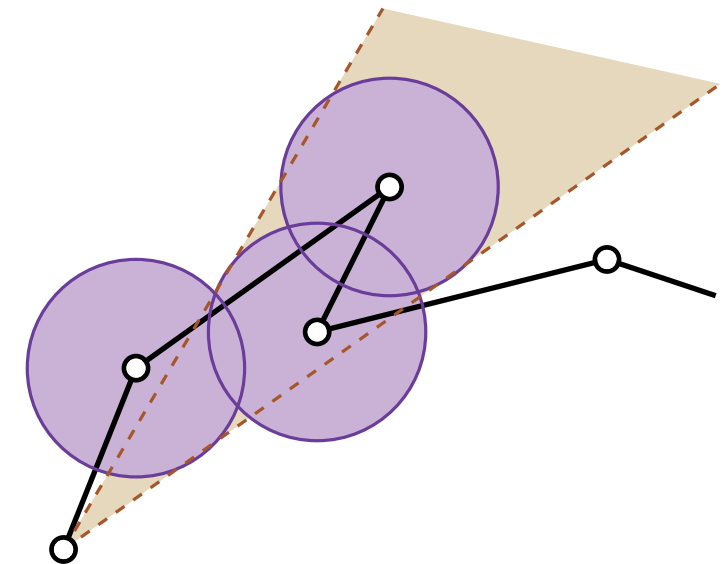
Algorithmen für geographische Informationssysteme

9. Vorlesung Linienvereinfachung

Teil II: DOUGLAS-PEUCKER



Philipp Kindermann



Wintersemester 2024/25



Douglas & Peucker (1973)

Geg. ■ Ein (kartographischer) Linienzug $L = (v_1, \dots, v_n)$

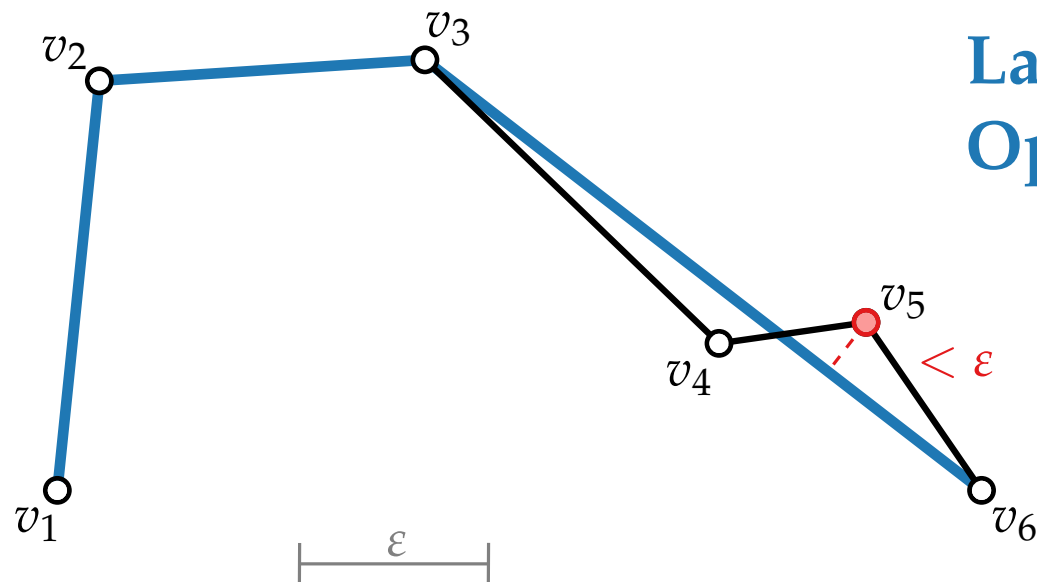
■ Eine geometrische Toleranz ε

Ges. ■ Eine Teilfolge L' von L (von v_1 nach v_n)

■ Für jedes Segment $S = (v_i, v_j)$ in L' gilt
 $\vec{d}_{\text{Hausdorff}}(L', L[i \dots j]) \leq \varepsilon$.

Demo.

<https://karthaus.nl/rdp>



Laufzeit? $\mathcal{O}(n^2)$
Optimal?

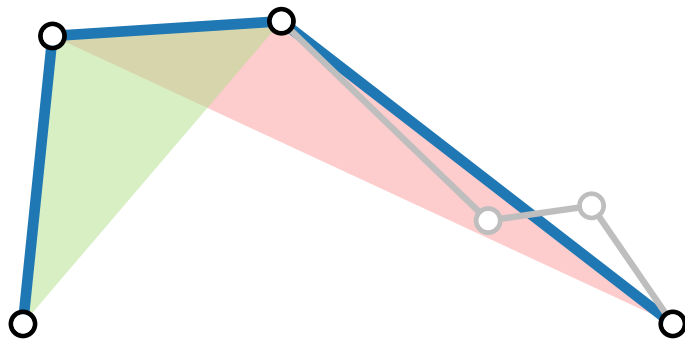
```

DOUGLASPEUCKER( $L[i \dots j], \varepsilon$ )
   $d_{\max} = -\infty$ ;  $p = 0$ 
  for  $k = i + 1$  to  $j - 1$  do
     $d = \text{Abstand } v_k \text{ zu } (v_i, v_j)$ 
    if  $d > d_{\max}$  then
       $d_{\max} = d$ ;  $p = k$ 
  if  $d_{\max} > \varepsilon$  then // nehme  $p$  auf
     $L_1 = \text{DOUGLASP}(L[i \dots p], \varepsilon)$ 
     $L_2 = \text{DOUGLASP}(L[p \dots j], \varepsilon)$ 
    return  $L_1 \circ L_2$ 
  else
    return  $(v_i, v_j)$ 
  
```

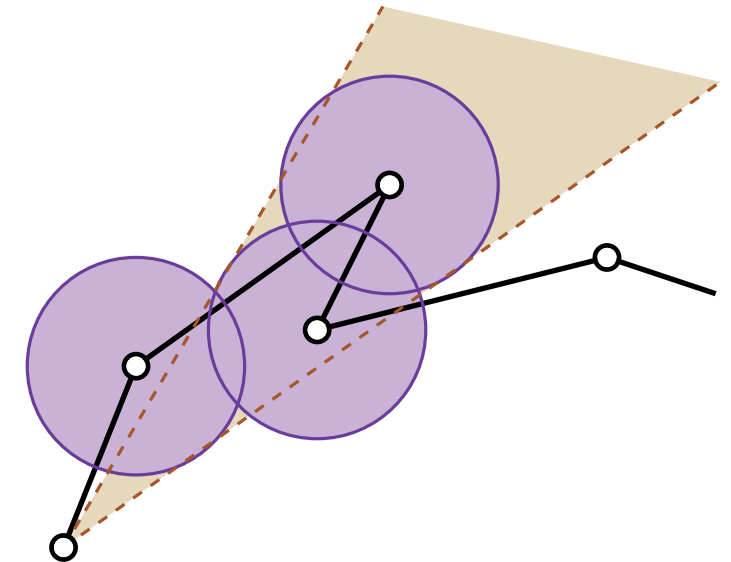
Algorithmen für geographische Informationssysteme

9. Vorlesung Linienvereinfachung

Teil III:
IMAI-IRI



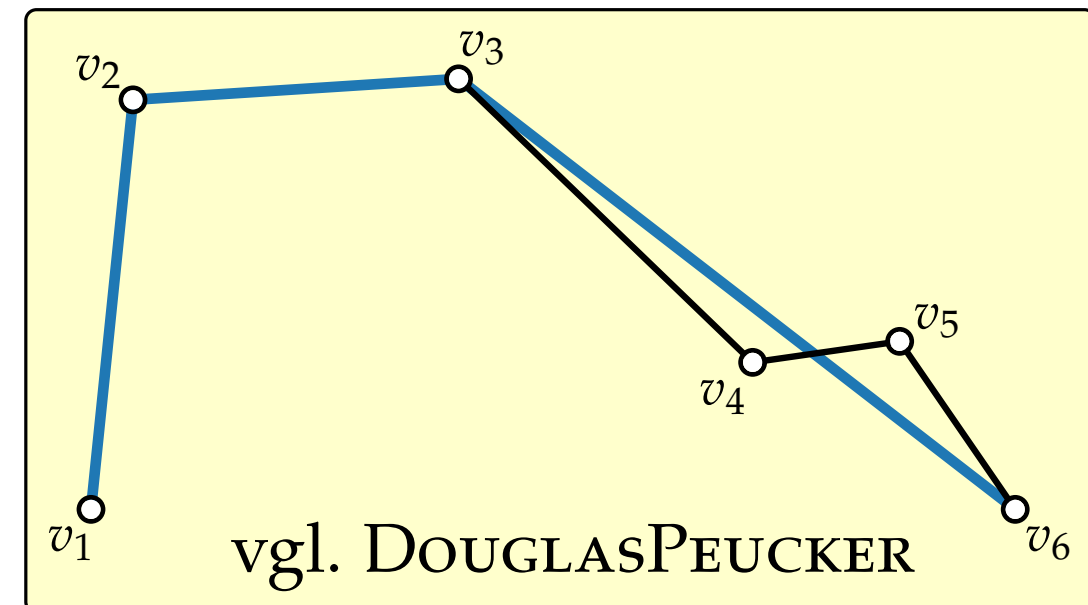
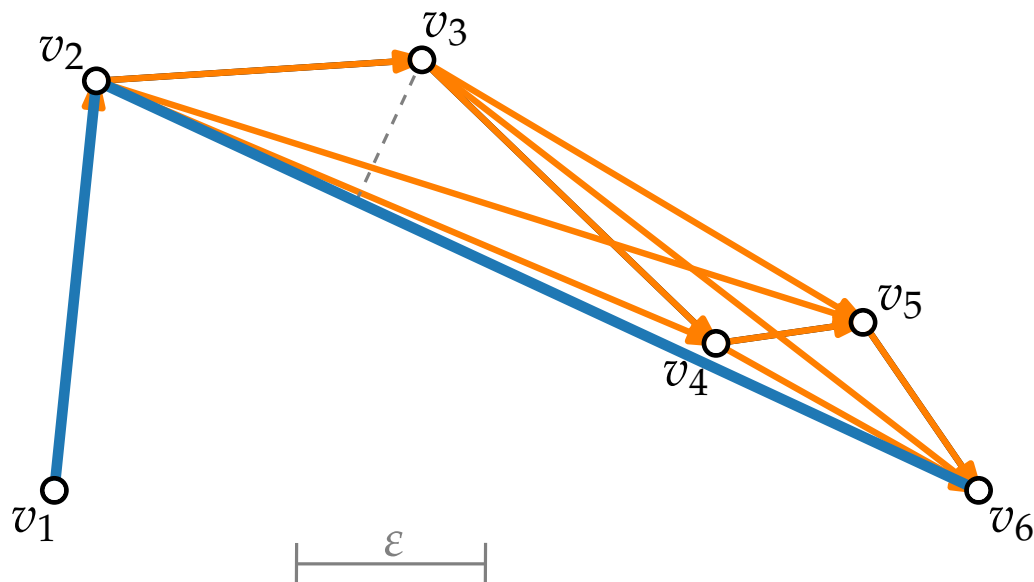
Philipp Kindermann



Wintersemester 2024/25

Imai & Iri (1988)

- Geg.**
- Ein (kartographischer) Linienzug $L = (v_1, \dots, v_n)$
 - Eine geometrische Toleranz ε
- Ges.**
- **Kleinste** Teilfolge L' von L (von v_1 nach v_n)
 - Für jedes Segment $S = (v_i, v_j)$ in L' und jeden Punkt v_k mit $i < k < j$ gilt: $d(S, v_k) \leq \varepsilon$.
- Idee.**
- Definiere Graph $G = (V, A)$ mit zulässigen **Abkürzungen** A (*shortcuts*)
 - Finde kürzesten Weg in G von v_1 nach v_n





Imai & Iri (1988) – Abkürzungsgraph

- Definiere Graph $G = (V, A)$ mit zulässigen **Abkürzungen** A (*shortcuts*)

ABKÜRZUNGSGRAPH(L)

$V = \{v_1, \dots, v_n\}$

$A = \emptyset$

foreach $1 \leq i < j \leq n$ **do**

$d_{\max} = 0$

for $k = i + 1$ **to** $j - 1$ **do**

$d = \text{Abstand } v_k \text{ zu } (v_i, v_j)$

if $d > d_{\max}$ **then** $d_{\max} = d$

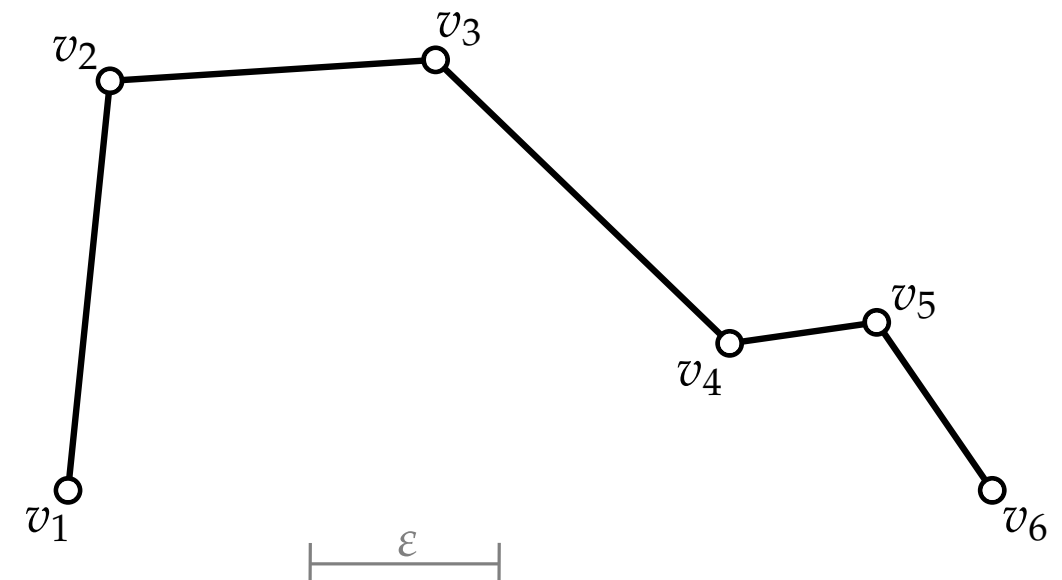
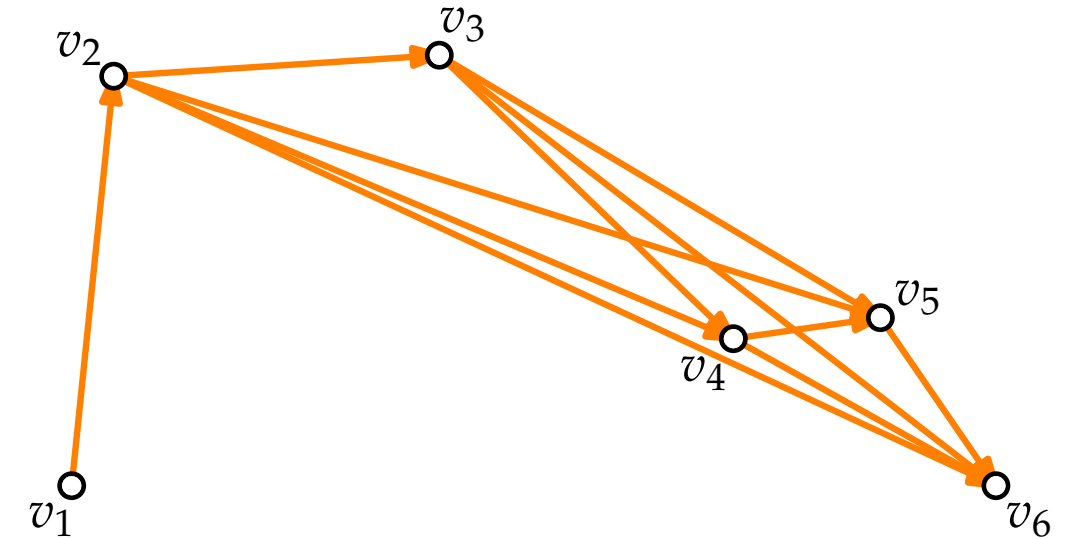
if $d_{\max} \leq \varepsilon$ **then**

$A = A \cup \{(v_i, v_j)\}$

return $G = (V, A)$

Laufzeit?

$\mathcal{O}(n^3)$



Imai & Iri (1988)

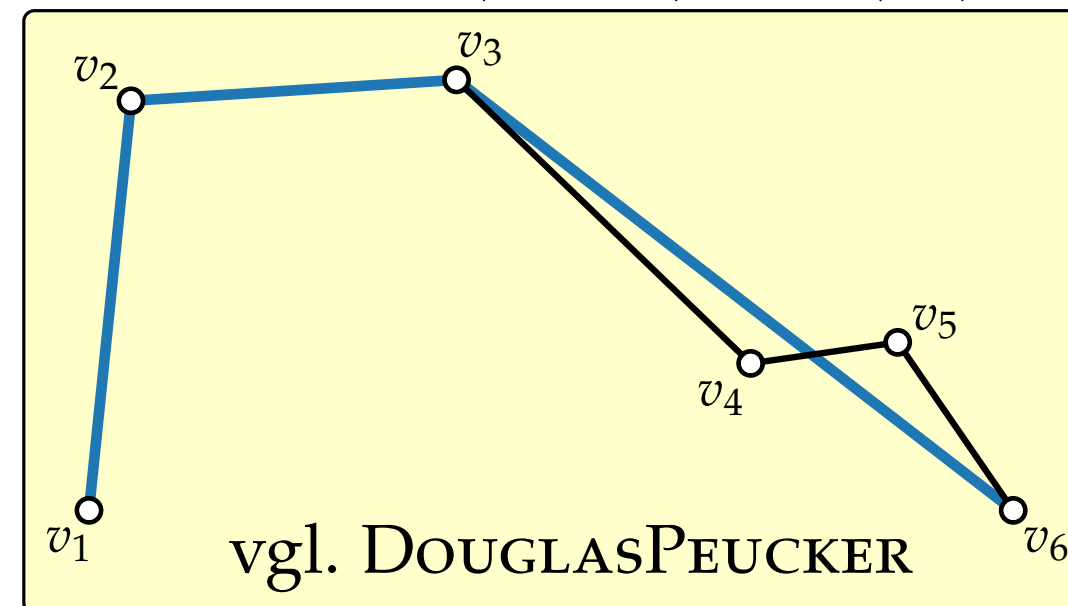
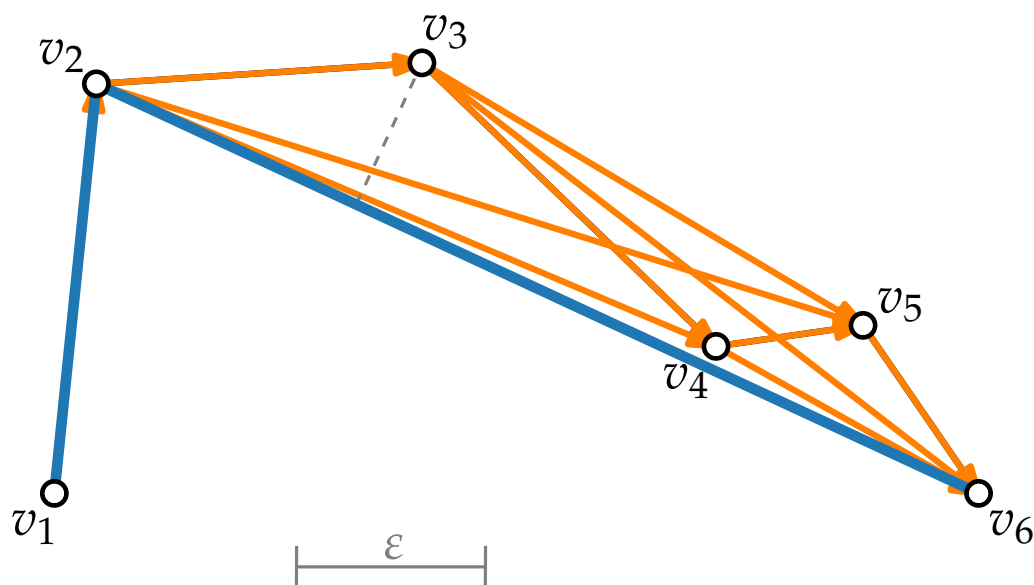
Geg. ■ Ein (kartographischer) Linienzug $L = (v_1, \dots, v_n)$
 ■ Eine geometrische Toleranz ε

Ges. ■ **Kleinste** Teilfolge L' von L (von v_1 nach v_n)
 ■ Für jedes Segment $S = (v_i, v_j)$ in L' und jeden Punkt v_k mit $i < k < j$ gilt: $d(S, v_k) \leq \varepsilon$.

Idee. ■ Definiere Graph $G = (V, A)$ mit zulässigen **Abkürzungen** A (*shortcuts*)
 ■ Finde kürzesten Weg in G von v_1 nach v_n

$$\mathcal{O}(n^3)$$

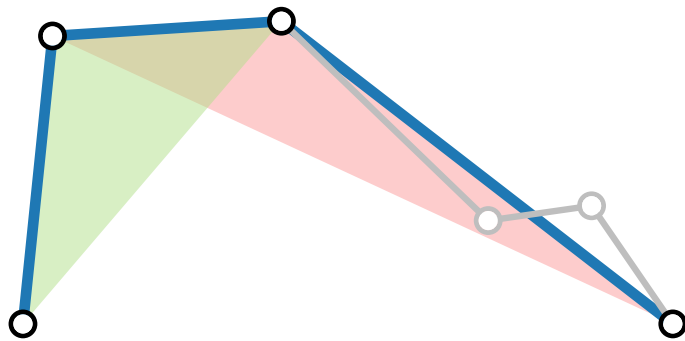
$$\mathcal{O}(n + m) = \mathcal{O}(n^2) \text{ (BFS)}$$



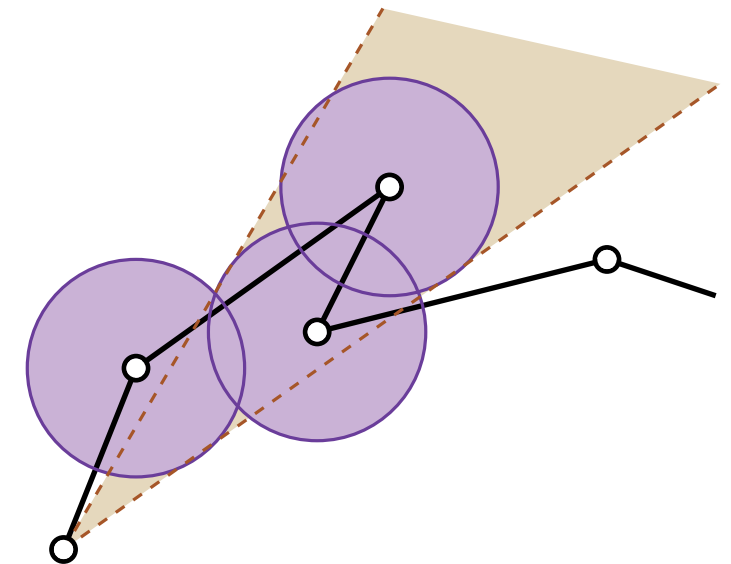
Algorithmen für geographische Informationssysteme

9. Vorlesung Linienvereinfachung

Teil IV:
CHAN-CHIN



Philipp Kindermann



Wintersemester 2024/25



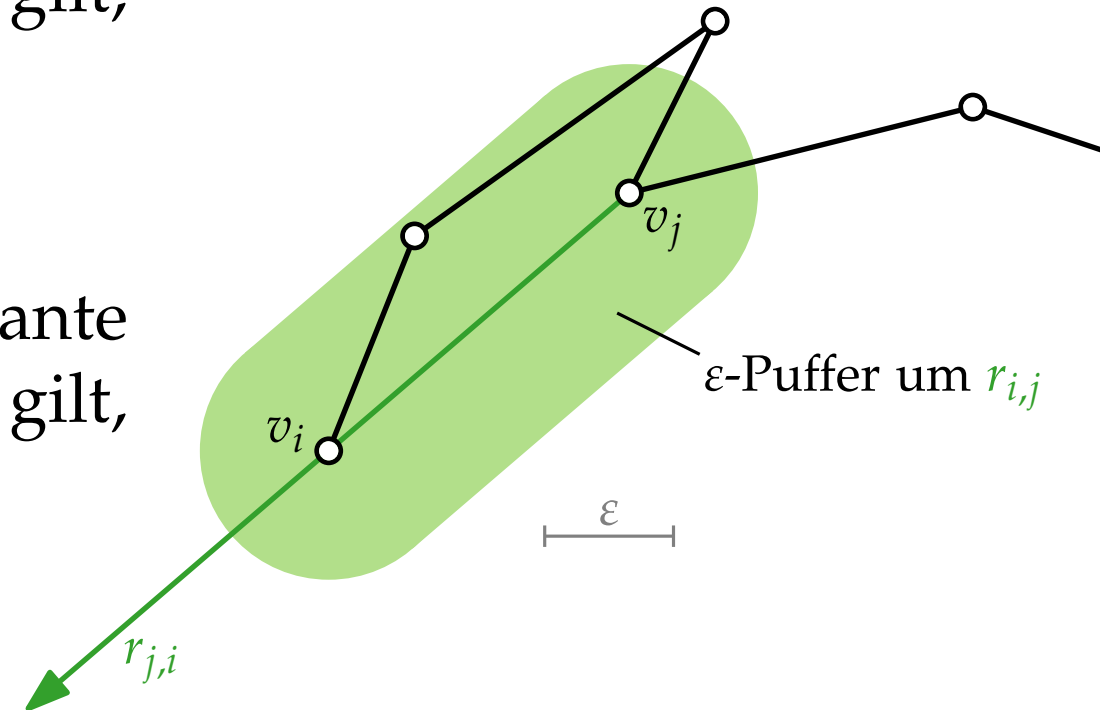
Chan & Chin (1996) – Idee

- Definiere Graph $G = (V, A)$ mit zulässigen **Abkürzungen** A (*shortcuts*)
- Berechne Graph $G' = (V, A')$, der Kante (v_i, v_j) enthält, wenn für jedes $i < k < j$ gilt, dass $d(v_k, r_{i,j}) \leq \varepsilon$.

Achtung: $G \neq G'$!

- Berechne Graph $G'' = (V, A'')$, der Kante (v_i, v_j) enthält, wenn für jedes $i < k < j$ gilt, dass $d(v_k, r_{j,i}) \leq \varepsilon$.
 - G enthält Kante (v_i, v_j) , wenn G' und G'' Kante (v_i, v_j) enthalten.
- $A = A' \cap A''$

$r_{i,j}$ = Strahl von v_i durch v_j

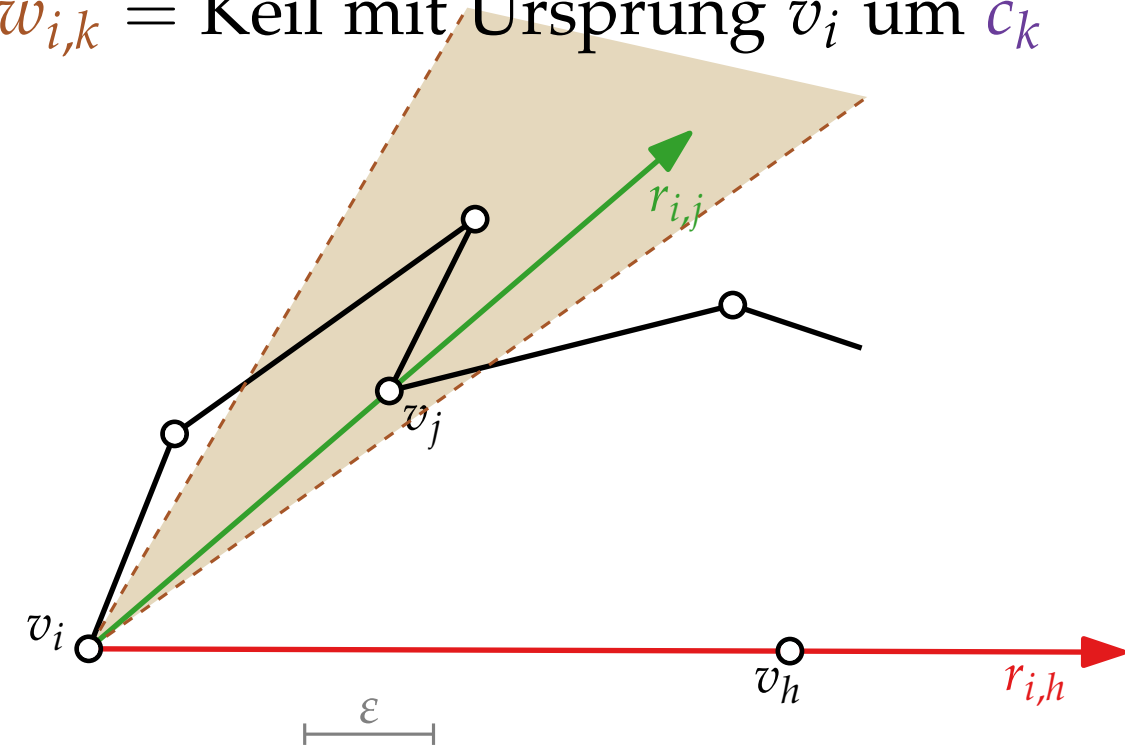




Chan & Chin (1996) – Berechnung

- Definiere Graph $G = (V, A)$ mit zulässigen **Abkürzungen** A (*shortcuts*)
- Berechne Graph $G' = (V, A')$, der Kante (v_i, v_j) enthält, wenn für jedes $i < k < j$ gilt, dass $d(v_k, r_{i,j}) \leq \varepsilon$.
- Für jeden Knoten v_i berechne alle Kanten in G' in $\mathcal{O}(n)$ Zeit.
- (v_i, v_j) ist in G' , wenn $r_{i,j}$ jeden Kreis c_k mit $i < k \leq j$ schneidet.
 - ➔ (v_i, v_j) ist in G' , wenn jeder Keil $w_{i,k}$ mit $i < k \leq j$ den Strahl $r_{i,j}$ enthält.
 - ➔ (v_i, v_j) ist in G' , wenn die Schnittmenge $W_{i,j}$ der Keile $w_{i,k}$ mit $i < k \leq j$ den Strahl $r_{i,j}$ enthält.

$r_{i,j}$ = Strahl von v_i durch v_j
 c_k = Kreis um v_k mit Radius ε
 $w_{i,k}$ = Keil mit Ursprung v_i um c_k



Müssen nur *linken* und *rechten* Rand des Keils berechnen und aktualisieren.



Chan & Chin (1996) – Berechnung

- Definiere Graph $G = (V, A)$ mit zulässigen **Abkürzungen** A (*shortcuts*)
- Berechne Graph $G' = (V, A')$, der Kante (v_i, v_j) enthält, wenn für jedes $i < k < j$ gilt, dass $d(v_k, r_{i,j}) \leq \varepsilon$.

CHANCHIN(L, ε)

for $i = 1$ **to** $n - 1$ **do**

$W_{i,i} = \mathbb{R}^2$

for $j = i + 1$ **to** n **do**

$W_{i,j} = W_{i,j-1} \cap w_{i,j}$

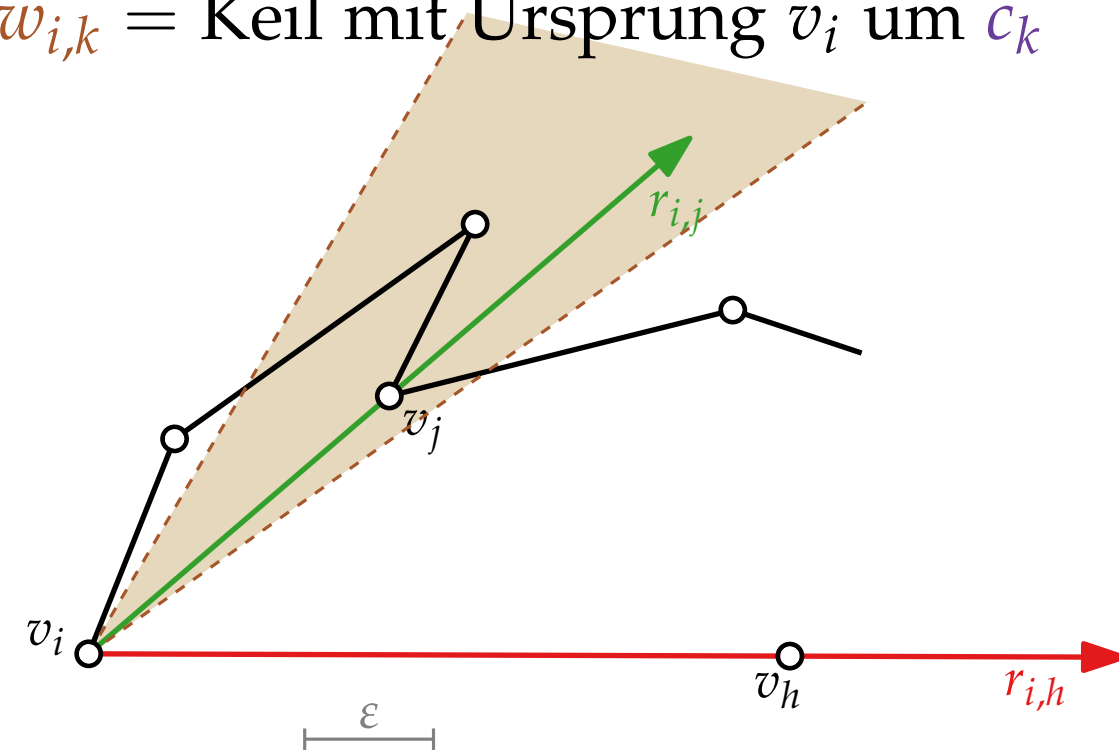
if $W_{i,j} = \emptyset$ **then break**

if $r_{i,j} \subseteq W_{i,j}$ **then**

$A = A \cup \{(v_i, v_j)\}$

return $G = (V, A)$

$r_{i,j}$ = Strahl von v_i durch v_j
 c_k = Kreis um v_k mit Radius ε
 $w_{i,k}$ = Keil mit Ursprung v_i um c_k

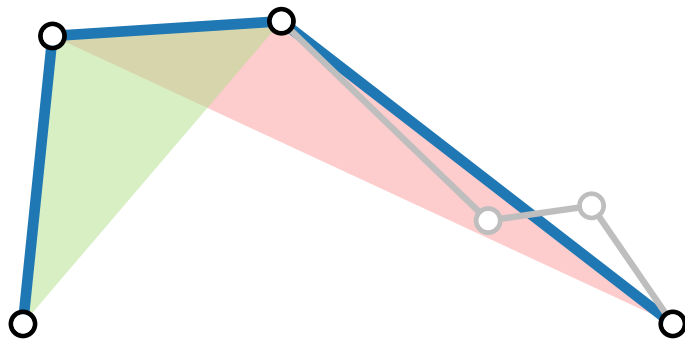


Müssen nur *linken* und *rechten* Rand des Keils berechnen und aktualisieren.

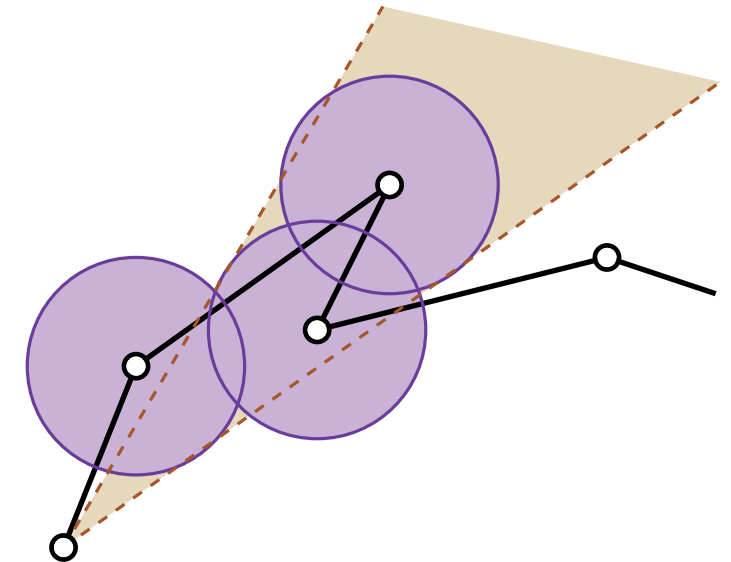
Algorithmen für geographische Informationssysteme

9. Vorlesung Linienvereinfachung

Teil V:
VISVALINGAM-WYATT



Philipp Kindermann



Wintersemester 2024/25



Visvalingam & Wyatt (1993)

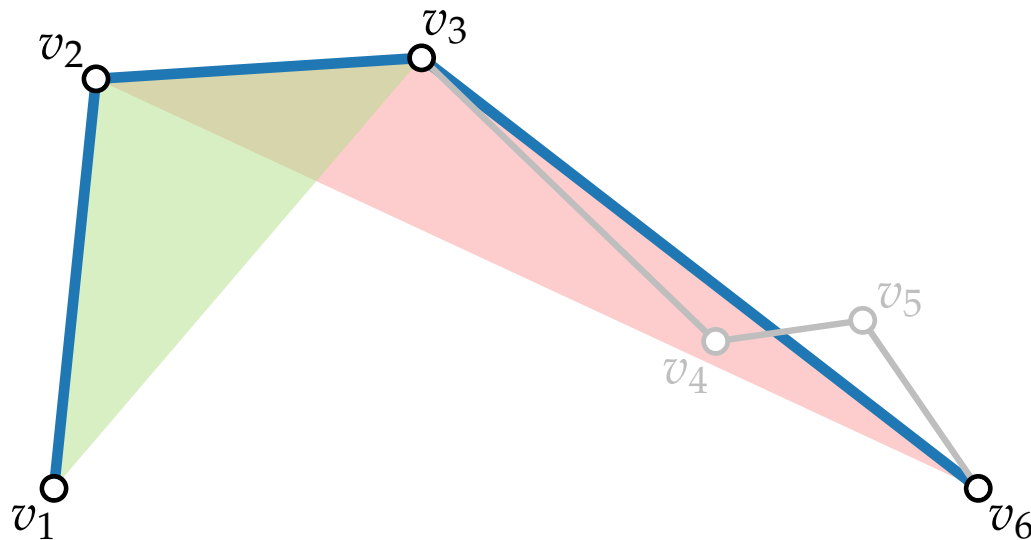
Geg.

- Ein (kartographischer) Linienzug $L = (v_1, \dots, v_n)$
- Eine geometrische Toleranz ε

Ges.

- Eine Teilfolge L' von L (von v_1 nach v_n)
- Für alle drei aufeinanderfolgenden Punkte v_i, v_j, v_k in L' gilt: $\triangle(v_i, v_j, v_k) \geq \varepsilon$.

Laufzeit? $\mathcal{O}(n^2)$
 $\mathcal{O}(n \log n)$ mit MINHEAP



VISVALINGAMWYATT(L, ε)

$L' = L$

$(v_i, v_j, v_k) =$ kleinstes Dreieck in L'

while $\triangle(v_i, v_j, v_k) < \varepsilon$ **do**

 Lösch v_j aus L'

$(v_i, v_j, v_k) =$ kleinstes Dreieck in L'

return L'

Heuristik

[Douglas & Peucker 1973]

Divide & Conquer

$\mathcal{O}(n^2)$

[Hershberger & Snoeyink 1992]

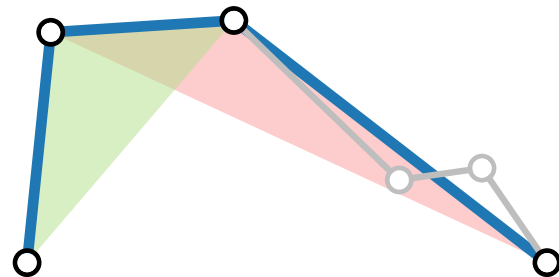
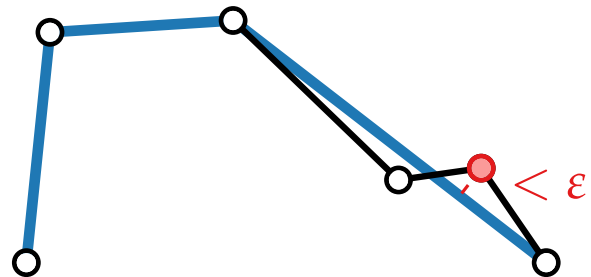
Anpassung von Douglas & Peucker

$\mathcal{O}(n \log n)$

[Visvalingam & Wyatt 1993]

Kleinste Dreiecke entfernen

$\mathcal{O}(n \log n)$



Optimierung

[Imai & Iri 1988]

Abkürzungsgraph + Abkürzungen

$\mathcal{O}(n^3)$

[Chan & Chin 1996]

Schnellerer Abkürzungsgraph
mit Keilen

$\mathcal{O}(n^2)$

[Agarwal et al. 2002]

ϵ -zulässige Vereinfachung mit höchstens
so vielen Punkten wie optimale
($\epsilon/2$)-zulässige Vereinfachung

$\mathcal{O}(n)$

