Echtzeit–Gelände–Triangulierung mittels eines adaptiv–hierarchischen Interpolationsverfahrens

A. Paul, K. Dobler, G. Sachs, Ch. Zenger

Technische Universität München

20.11.1997

Motivation

- Navigations- und Flugführungsprobleme unter schlechten Außensichtbedingungen
- Flugführungsanzeigen mit hohem Interpretationsbedarf

$\Rightarrow$ computergenerierte synthetische Sicht mit integrierter Flugführungssymbolik

Vorgehen:

- Bestimmung der aktuellen Position und Orientierung mittels integriertem Präzisions–Navigationssystem (DGPS/INS)
- Generieren der synthetischen Sicht aus Datenbasis (Höhendaten, Strukturdaten)
- generiertes Bild mit Flugführungssymbolik kombinieren

Ziele und Anforderungen

- **Kompression der Höhendaten**
- **geeignete Approximation des Geländes**
  insbesondere: stufenloser Level–of–Detail und keine Fehler (Löcher) bei der Geländedarstellung
- **modularer, wartungsfreundlicher und portabler Code**
  $\Rightarrow$ C++, Motif, OpenGL
- **25 km Sichtweite bei 30Hz Bildfrequenz**
  Zielplattform: Silicon Graphics Indigo^2 Maximum IMPACT R10000 CPU (200 MHz), 128 MB Speicher

Datenbasis

- **Höhendaten** (DTED)
  Format: reguläres Gitter
  Auflösung: 1×1 Sekunde (∼20×30 m), 3×3 Sekunden (∼60×90 m)
  Genauigkeit: 7m–30m (abhängig vom Gelände)
  Speicherplatzbedarf eines 1×1 Grad–Blocks: ∼ 2.8 MB

- **Strukturdaten** (DFAD)
  - point features (z.B. Gebäude)
  - line features (z.B. Straßen, Flüsse)
  - areal features (z.B. Wälder, Seen, Ortschaften)

  Speicherplatzbedarf eines 1×1 Grad–Blocks: ∼ 2.6 MB

Ansatz zur Geländetriangulierung

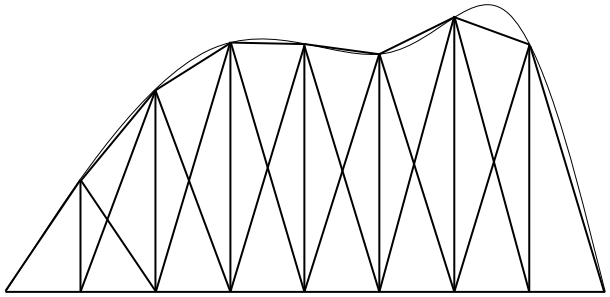
Hierarchische Interpolation

Vorteile:

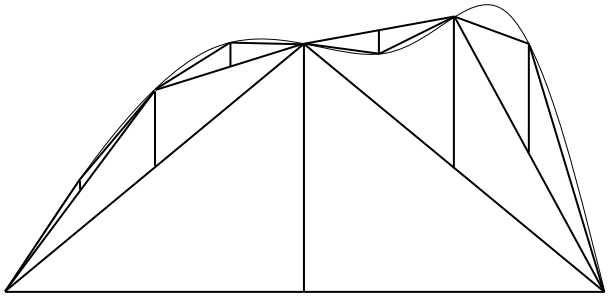
- leichte Kontrolle des Interpolationsfehlers
- inhärente Adaptivität
- Datenkompression

Nachteile:

- komplizierte Datenstrukturen (Bäume)
- aufwendige Funktionsauswertung

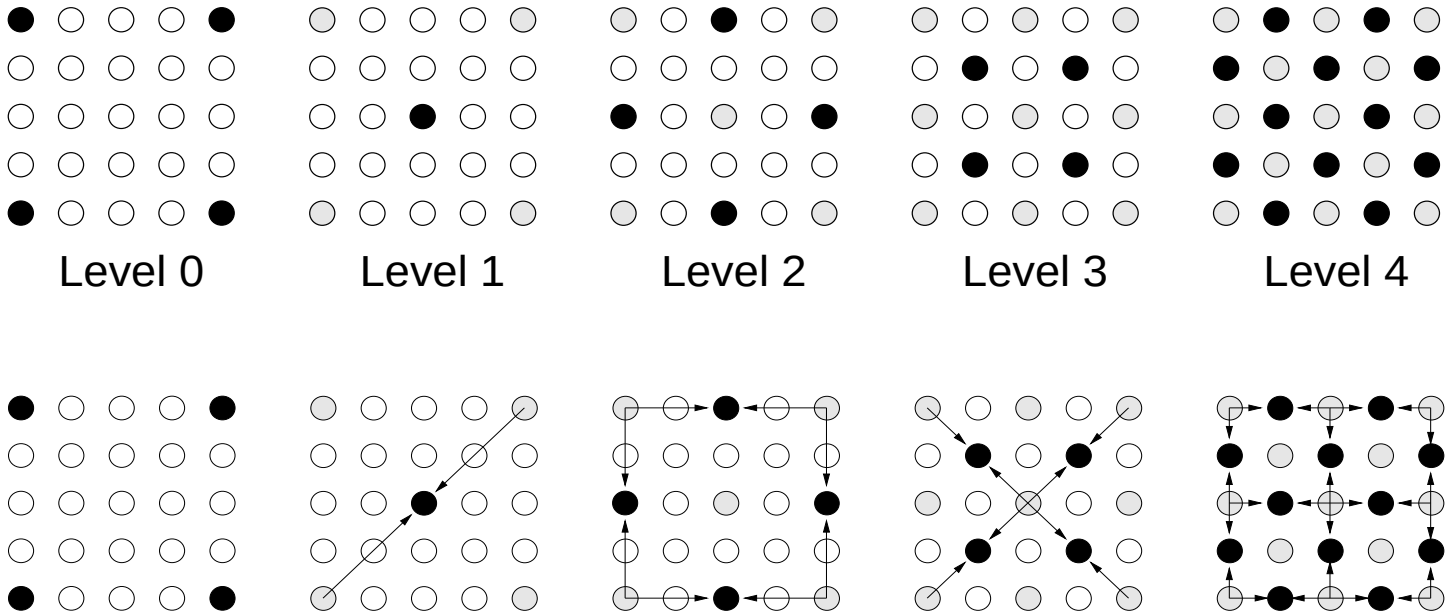


Stützpunktbasis



hierarchische Basis

Hierarchie (5×5 Gitter)



zur Interpolation verwendete hierarchische Vorgänger

Interpolation

Bottom–Up–Durchlauf durch die Höhendaten

- lineare Interpolation jedes Knotens
- Speichern des Interpolationsfehlers für jeden Knoten (in extra Array)

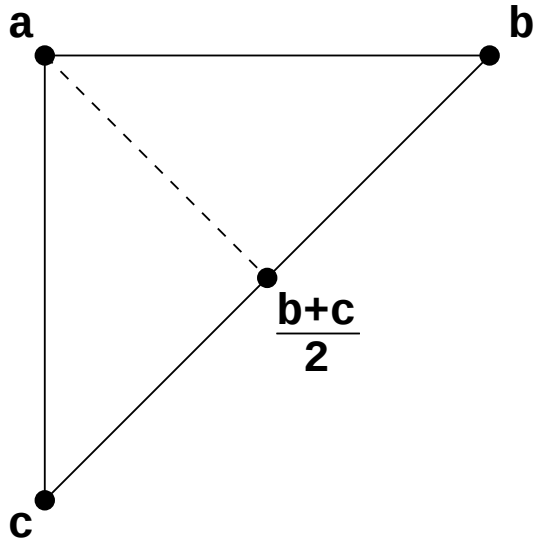
Bottom–Up–Durchlauf durch das Array von Interpolationsfehlern

- An jedem Knoten: $v_{aktuell} = max\{|v_{aktuell}|, V_{Nachfolger}\}$

$\Rightarrow$ größter Interpolationsfehler im hierarchisch höchsten Knoten

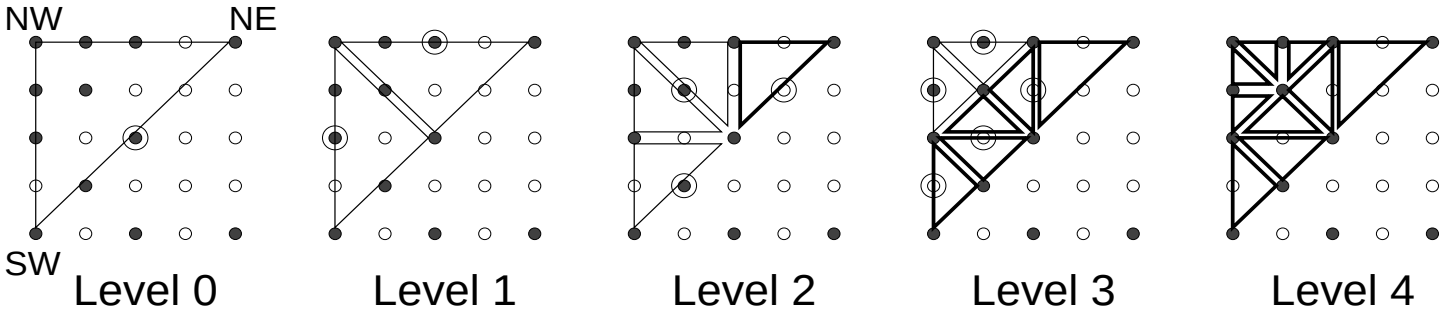
Triangulierung

```
untersuche_dreieck(a,b,c)
{
  if(Interpolationsfehler am Knoten (b+c)/2 < ε)
    zeichne_dreieck(a,b,c);
  else {
    untersuche_dreieck( (b+c)/2, a, b);
    untersuche_dreieck( (b+c)/2, c, a);
  }
}
```

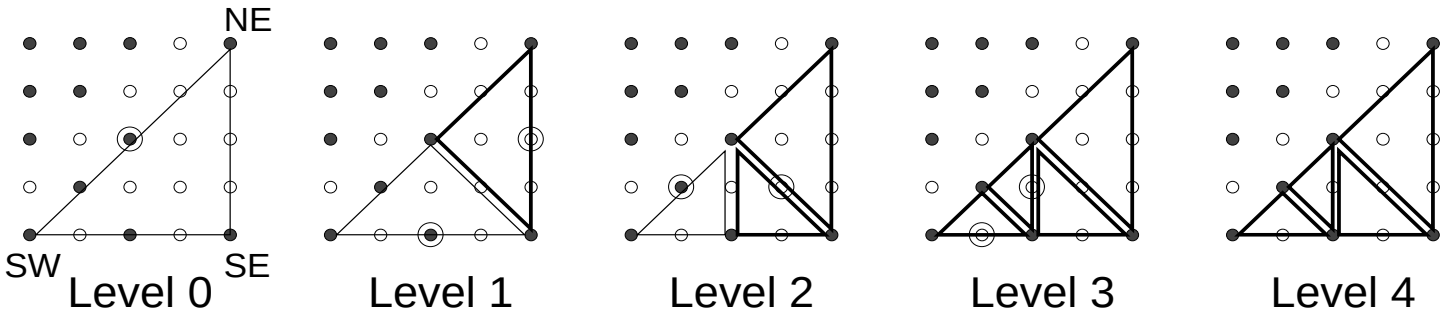


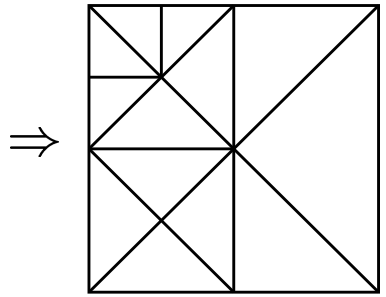
Triangulierung

`untersuche_dreieck(NW,NE,SW);`



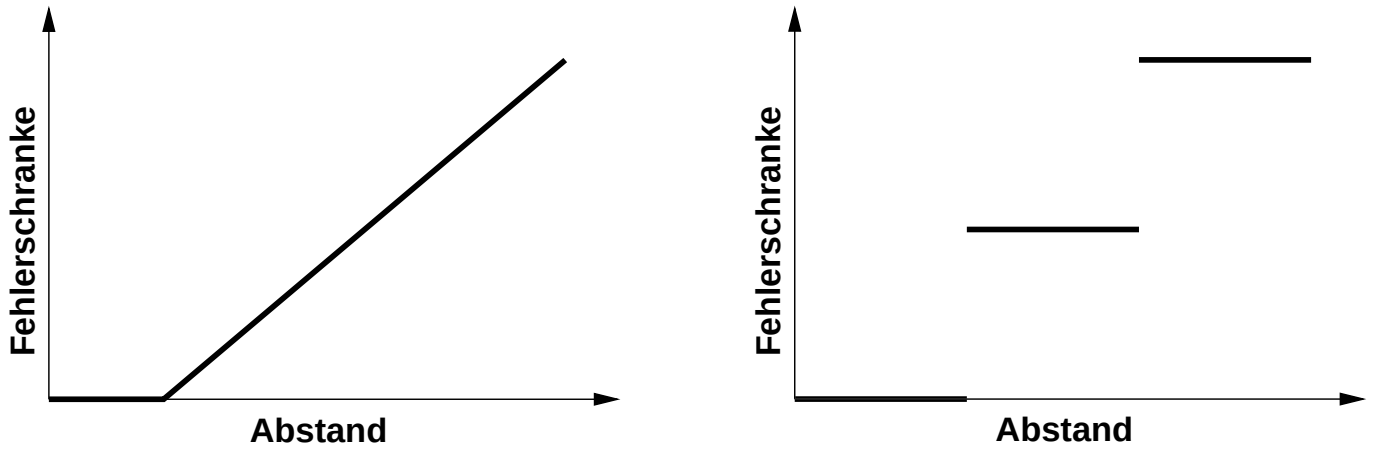
`untersuche_dreieck(SE,SW,NE);`





Level–of–Detail

konstantes $\varepsilon \rightarrow$ Funktion `eps(...)`



Änderung der `if`–Abfrage in `untersuche_dreieck()`:

```
if(Fehler am Punkt (b+c)/2 < eps(d(a,b,c)))
```

$\Rightarrow$ Problem: Löcher

$\rightarrow$ bis jetzt keine einfache funktionierende Lösung gefunden

Work–Around

Triangulierung in zwei Top–Down–Durchläufen

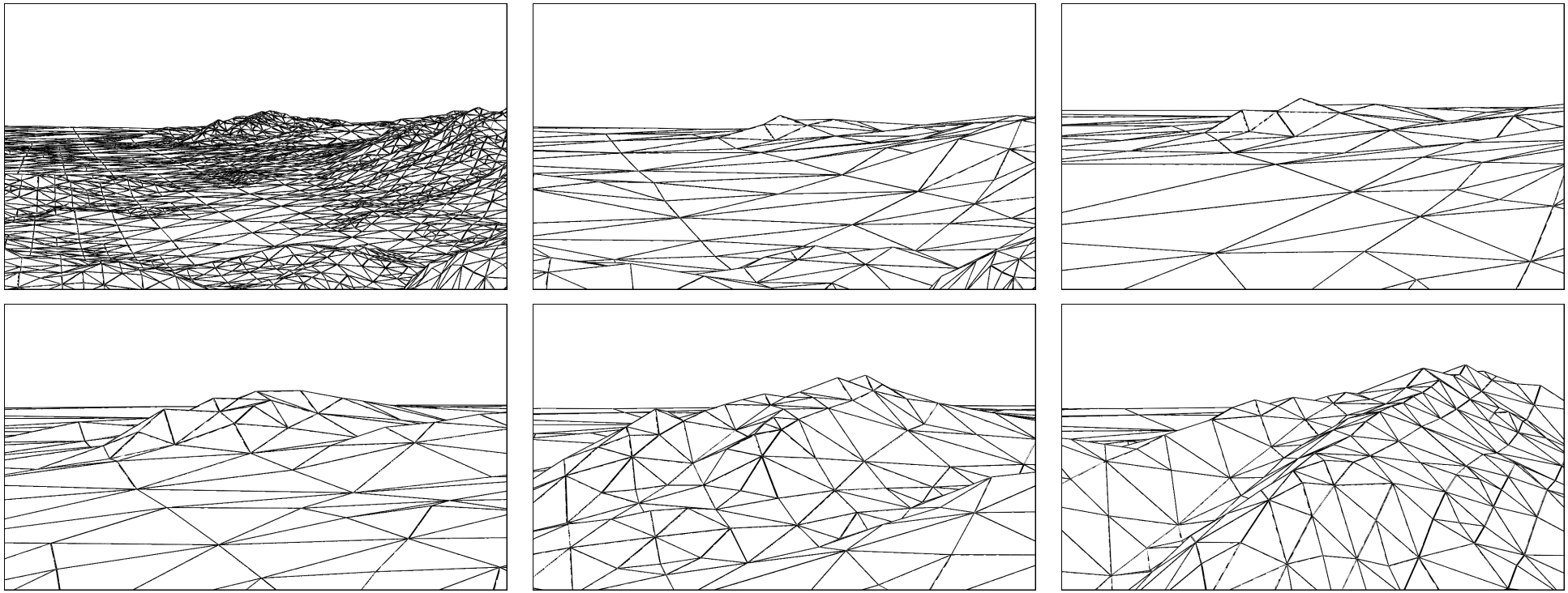
1. Durchlauf: Markieren der benötigten Knoten

2. Durchlauf: Zeichnen der Dreiecke

```
untersuche_und_markiere_dreieck(a,b,c)
{
  if(dreieck_nicht_sichtbar(a,b,c)) return();
  if(Fehler am Knoten (b+c)/2 < eps(d(a,b,c))) markiere_knoten(a,b,c);
  else {  untersuche_und_markiere_dreieck( (b+c)/2, a, b);
          untersuche_und_markiere_dreieck( (b+c)/2, c, a); }
}

untersuche_und_zeichne_dreieck(a,b,c)
{
  if(Knoten (b+c)/2 ist nicht markiert) zeichne_dreieck(a,b,c);
  else {  untersuche_und_zeichne_dreieck( (b+c)/2, a, b);
          untersuche_und_zeichne_dreieck( (b+c)/2, c, a); }
}
```

Ergebnisse



- **stufenloser LOD, keine Löcher in der der Landschaft**
- **25 km Sichtweite bei 30Hz Bildfrequenz**: knapp verfehlt (24Hz)
- **Portabilität**: Programm lief auf einem Linux Laptop
- **Datenkompression**: bisher noch nicht näher untersucht

Ausblick

- Nachladen von Daten während des Fluges
  $\Rightarrow$ zu speicherndes Datenformat / Datenkompression
- Parallelisierung
  $\rightarrow$ warten auf Betriebssystem mit Support für Kernel–Threads