Triangulering i 3D vha. kd-træer

Micky Christensen

&

Nadeem Gulzar

Problemformulering

- Givet en tilfældig rækkefølge af *n* knuder i rummet,

$$v_i \in \mathbb{R}^3, 1 \leq i \leq n$$

  som er uniformt distribueret, giv da en korrekt triangulering, så samtlige knuder anvendes og former en lukket overflade

- Med korrekt menes, at hver kant præcis skal anvendes 2 gange og deles af 2 tilstødende trekanter

Trianguleringen

- Aktiv kant-liste initialiseres med en start-trekants 3 kanter
- **Hver aktive kant finder sin kandidatknude og forbinder sig til den**
- Der dannes en ny trekant og de 2 nye kanter tilføjes den aktive kant-liste
- Trianguleringen stopper når der ikke er flere aktive kanter

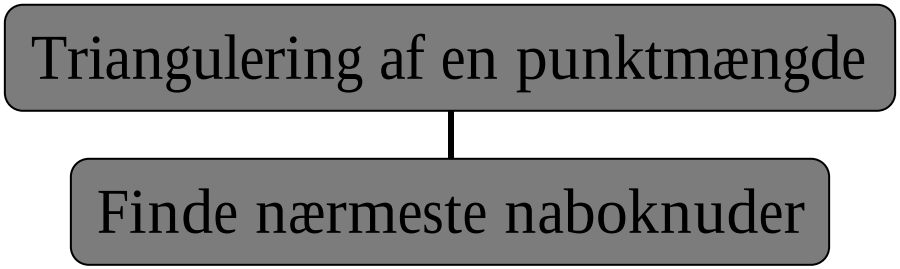
Trianguleringen

- Eulers formel: $V - E + F = 2$
- For enhver triangulering gælder: $E = 3n + 6$
- Antallet af trekanter (facetter):

  $F = 2 - V + E = 2 - V + (3V - 6) = 2V - 4$
- Dette er generelt kun gældende for ét trianguleret område
- For en trianguleringen opdelt i M lukkede områder:

  $F = 2V - 4M$

Nabo-søgning



Trianguleringsprocessen består hovedsageligt af at finde et nabolag omkring en kant, hvorefter den bedste kandidat findes mellem disse naboknuder.

Det er i forbindelse med at finde en nabolag, at kd-træer kan benyttes.

Den lineære løsning

- For hver kant ledes der lineært igennem knudemængden for at finde en nabokandidat
- Køretiden for én kant er da

$$O(n)$$

hvor n er antallet af knuder

Den optimerede løsning

- Ved brug af kd-træer optimeres søgerutinen til

$$O\left(\sqrt[3]{n^2} + k\right)$$

hvor k er antallet af rapporteret knuder

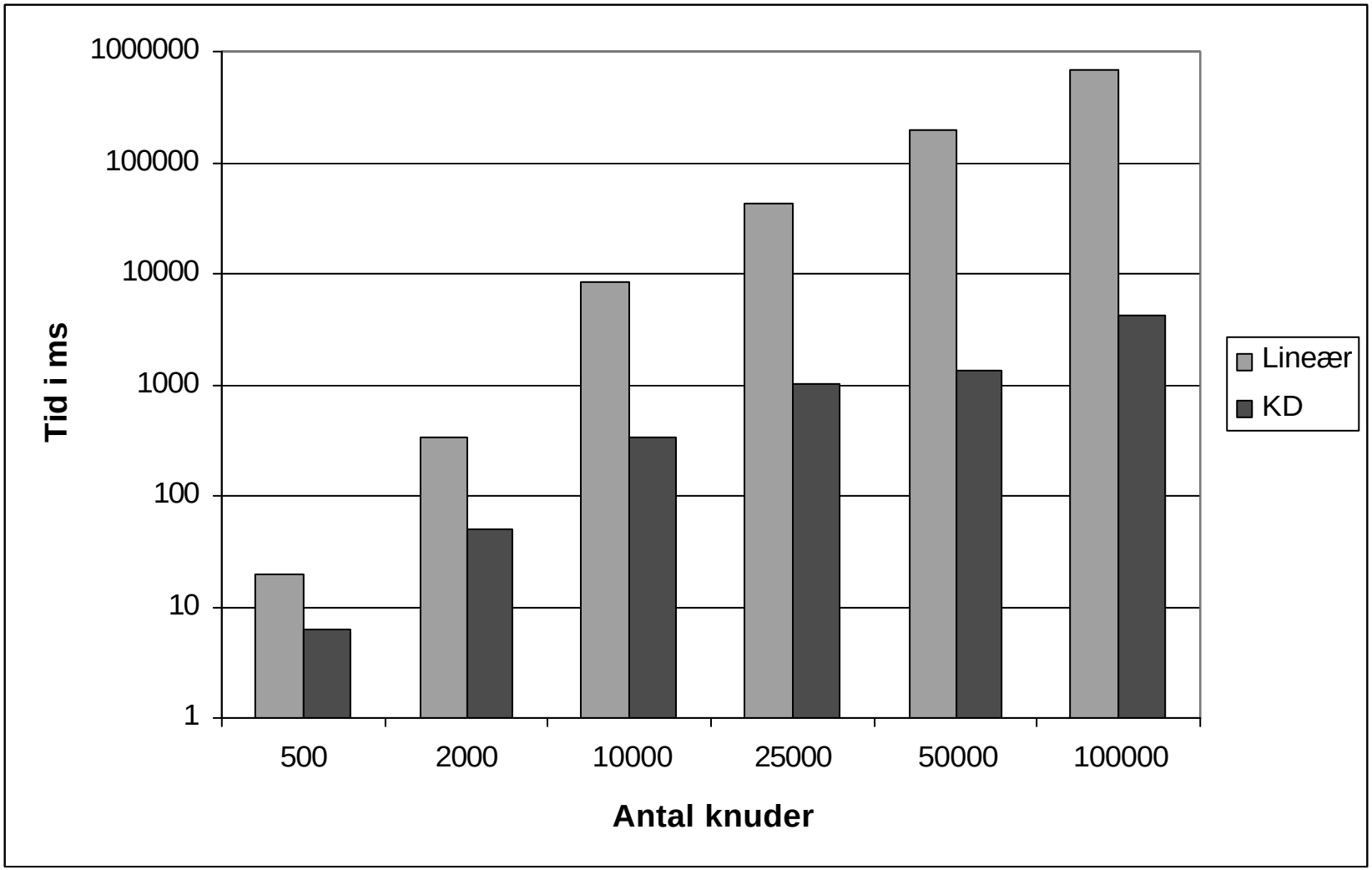
Design af søgeradius

- Der søges med en kugle (halvkugle)
- k-faktoren skal holdes så lille som muligt ved brug af ”passende” søgeradius
- 2-step søgning (først med angivet radius dernæst med dobbelt radius, hvis der ikke findes nogen knuder i den første søgning)

Resultater (tilfældige knuder)

Antal knuder	Lineær algoritme (tid i ms)	Kd-træer (tid i ms)
500	20	6
2000	331	50
10.000	8.287	336
25.000	43.208	1.017
50.000	192.589	1.377
100.000	677.767	4.076

Resultater (tilfældige knuder)

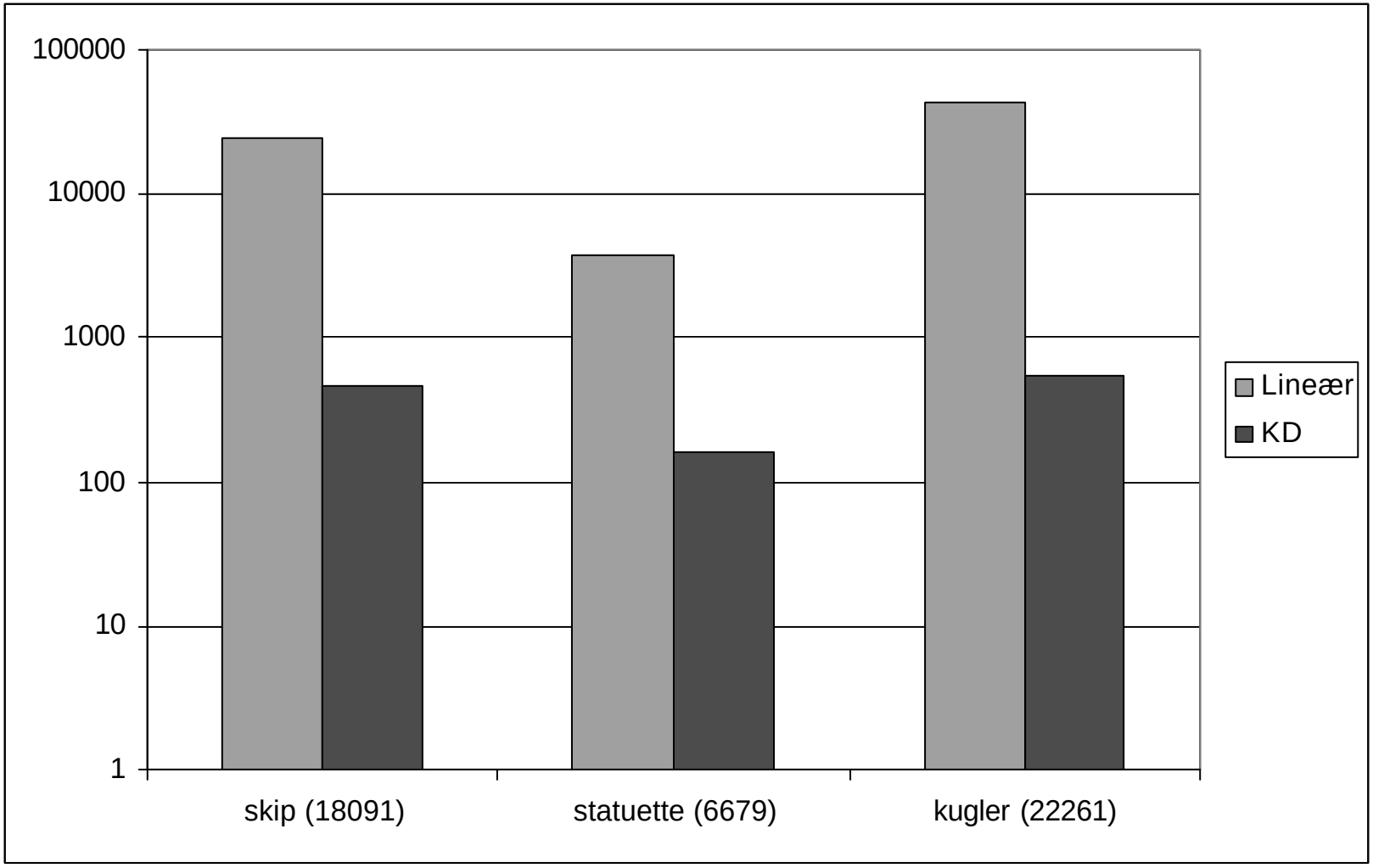


Tests blev udført på en Pentium 4 2.53GHz maskine med 512mb ram

Resultater (objekter)

Objekt (antal knuder)	Lineær algoritme (tid i ms)	Kd-træer (tid i ms)
Skip (18.091)	24.373	466
Statuette (6.679)	3.808	165
Kugler (22.261)	43.124	559

Resultater (objekter)



Tests blev udført på en Pentium 4 2.53GHz maskine med 512mb ram

Konklusion

- Det giver en rigtig fornuftig hastighedsforøgelse i triangulering, ved at bruge kd-træer til nabosøgning

- Hastighedsforskellen er selvfølgelig afhængig af antallet af knuder. Jo større antal knuder vi arbejder med, jo bedre resultater

Spørgsmål

- Spørgsmål ?
- Kommentarer ?

- Vores demoer:
  Neighborhood Search in 3D
  Triangulation in 3D