Prof. G. Zachmann
R. Weller

University of Bremen
School of Computer Science
CGVR Group
June 1, 2015

Sommersemester 2015

# Assignment on Geometric Data Structures for Computer Graphics - Sheet 4

Due Date 10. 06. 2015

## Exercise 1 ($k$-Nearest-Neighbour-Suche mit $kd$-Trees, *4 Credits*)

Erweitern Sie die Nearest-Neighbour-Suche für $kd$-Trees so, dass sie nicht nur einen einzigen nächsten Nachbarn, sondern die $k$ nächsten Nachbarn für einen Anfragepunkt $q$ liefert. Beschreiben Sie Ihren Algorithmus in Pseudocode.

## Exercise 2 (Bereichsanfragen mit $kd$-Trees, *3+5 Credits*)

Mittels $kd$-Trees über Punkten kann, wie schon von den Quadtrees bekannt, *Range Queries* sehr einfach und relativ effizient beantworten. Dabei ist ein Anfragerechteck gegeben, und gesucht sind alle diejenigen Punkte, die in diesem Rechteck liegen.

a) Geben Sie einen Algorithmus an, der dieses leistet.

b) Zeigen Sie, daß rechteckige Bereichsanfragen in einem $kd$-Tree mindestens $\Omega(\sqrt(n))$ dauern.

## Exercise 3 ($kd$-Trees und Dreiecke, *3+5 Credits*)

Ein $kd$-Tree kann auch für dreieckige Range Queries verwendet werden.

a) Zeigen Sie, daß die Laufzeit für dreieckige Range Queries mit einem 2-dimensionalen $kd$-Tree im schlimmsten Fall linear ist, selbst wenn das Dreieck keinen Punkt des Baumes enthält.
*Hinweis*: Betrachten Sie die Punkte auf der Gerade $y = x$.

b) Im folgenden beschränken wir uns auf Dreiecke in der Ebene, deren Kanten horizontal oder vertikal ausgerichtet sind, oder eine Steigung von $-1$ bzw $+1$ aufweisen.
Geben Sie eine Datenstruktur an, die für diese speziellen Dreiecke effizienter arbeitet (Hier ist kein formaler Beweis zur Laufzeit gefordert, sondern lediglich eine Begründung warum Ihre Datenstruktur effizienter ist).
*Hinweis*: Erweitern Sie Ihren 2-dimensionalen $kd$-Tree.