University Name

Doctoral Thesis

# Thesis Title

*Author:*
John Smith

*Supervisor:*
Dr. James Smith

*A thesis submitted in fulfilment of the requirements*
*for the degree of Doctor of Philosophy*

*in the*

Research Group Name
Department or School Name

February 2015

# Declaration of Authorship

I, John Smith, declare that this thesis titled, 'Thesis Title' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

*"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."*

Dave Barry

UNIVERSITY NAME (IN BLOCK CAPITALS)

# *Abstract*

Faculty Name
Department or School Name

Doctor of Philosophy

**Thesis Title**

by John Smith

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

# *Acknowledgements*

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

# Contents

# List of Figures

# List of Tables

# Abbreviations

**LAH** **L**ist **A**bbreviations **H**ere

# Physical Constants

Speed of Light $c = 2.997\ 924\ 58 \times 10^8\ \mathrm{ms}^{-\mathrm{S}}$ (exact)

# Symbols

| | | |
|---|---|---|
| $a$ | distance | m |
| $P$ | power | W ($Js^{-1}$) |
| $\omega$ | angular frequency | $rads^{-1}$ |

*For/Dedicated to/To my...*

# Chapter 1

# Background

## 1.1 Nearest Neighbors Search

## 1.2 Approximate Nearest Neighbors

## 1.3 k-d Trees

### 1.3.1 Overview

The k-d tree was originally developed as "a data structure storage of information to be retrieved by associative searches" [1]. k-d trees are efficient both in the speed of associative searches and in storage requirments.

A k-d tree is a binary tree which stores points in a k dimension space. Each node contains a single k-dimensional point, a split dimension, and up to two children nodes. Each node represents a hyperplane which lies perpendicular to the split dimension, and passes through the stored point. The left subtree of a node contains all points which lie to the left of the hyperplane, while the right subtree represents all points which lie to the right of the hyperplane. Thus, each node partitions all below it into two half-spaces. Because only a single split dimension is used, each splitting hyperplane is axis-aligned.

### 1.3.2 Construction

The construction of a k-d tree is performed recursively with input parameters of a list of points. Pseudo code is shown below in 0.

**function** KDTREE(pointList)

```
    splitDim = selectAxis()

    medianPoint = selectMedian(pointList, splitDim)
    leftList = select points ≤ medianPoint along splitDim
    rightList = select points > medianPoint along splitDim

    treenode node = new treenode()
    node.splitDim = splitDim
    node.splitPoint = medianPoint
    node.leftChild = kdtree(leftList)
    node.rightChild = kdtree(rightList)

    return node
end function
```

Axis selection can be performed in multiple ways. The classical approach is to deterministically alternate between each dimension. Another approach known as spatial median splitting selects the the longest dimension present in the current pointList to split on [2]. The downside of this method is that a linear traversal is required to select the split dimension. Another popular approach is to randomly select the split dimension with an equal probability of selection each dimension. This approach is often applied when using multiple k-d trees as because of the additional randomness trees are more likely to be different.

While a linear time algorithm for determining the median of an unordered set is possible [3] a heuristic approach is typically used to approximate the median. A common heuristic is to take the median of five randomly chosen elements, however many other methods can be used such as the triplet adjust method [4].

At the termination of of 0, the root of the k-d tree is returned, and each node contains exactly one point. The runtime of this algorithm is O(N*log(N)) where N is the number of points in pointList. While the median can be approximated in constant time, partitioning pointList along that median is an O(N) operation. Since the k-d tree is a binary tree in which each node holds one point, assuming it is relatively balanced, its height is O(log(N)).

### 1.3.3 Nearest Neighbor Query

A simple algorithm exists to apply the k-d tree to a nearest neighbor query. Pseudocode for this algorithm is shown in 0.

```
function SEARCHKDTREE(kdTreeNode, searchPoint, currBest)
    dim = kdTreeNode.splitDim
    searchDir = searchPoint[dim] < kdTreeNode.splitPoint[dim]
    searchFirst = searchDir ? kdTreeNode.left : kdTreeNode.right
    searchSecond = searchDir ? kdTreeNode.right : kdTreeNode.left
    searchkdtree(searchFirst, searchPoint, currBest)
    if distance(kdTreeNode.splitPoint, searchPoint) ¡ distance(currBest, splitPoint)
then
|     currBest = kdTreeNode.SplitPoint
    end
end function
```

Because all hyperplanes are axis aligned, the computation to determine whether a closer point can possibly exist in the second subtree is very simple. The closest possible point in the halfspace represented by the second subtree will lie a distance of $\epsilon$ from the hyperplane, where $\epsilon$ is very small.

### 1.3.4 Modification

# Appendix A

# Appendix Title Here

Write your Appendix content here.

# Bibliography

[1] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[2] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. In *ACM Transactions on Graphics (TOG)*, volume 27, page 126. ACM, 2008.

[3] Nimrod Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the ACM (JACM)*, 31(1):114–127, 1984.

[4] Sebastiano Battiato, Domenico Cantone, Dario Catalano, Gianluca Cincotti, and Micha Hofri. An efficient algorithm for the approximate median selection problem. In *Algorithms and Complexity*, pages 226–238. Springer, 2000.