DISTRIBUTED KERNELIZED LOCALITY-SENSITIVE HASHING FOR FASTER IMAGE BASED NAVIGATION

THESIS

Scott A. Hutchison, CPT, US Army

AFIT-ENG-MS-15-M-070

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-15-M-070

DISTRIBUTED KERNELIZED LOCALITY-SENSITIVE HASHING FOR FASTER IMAGE BASED NAVIGATION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

Scott A. Hutchison, B.S.C.S.

CPT, US Army

March 2015

AFIT-ENG-MS-15-M-070

DISTRIBUTED KERNELIZED LOCALITY-SENSITIVE HASHING FOR FASTER IMAGE BASED NAVIGATION

THESIS

Scott A. Hutchison, B.S.C.S.
CPT, US Army

Committee Membership:

Dr. G. L. Peterson
Chair

Dr. D. D. Hodson
Member

Maj B. G. Woolley, PhD
Member

AFIT-ENG-MS-15-M-070

Abstract

Content based image retrieval (CBIR) remains one of the most heavily researched areas in computer vision. Different image retrieval techniques and algorithms have been implemented and used in localization research, object recognition applications, and commercially by companies such as Facebook, Google, and Yahoo!. Current methods for image retrieval become problematic when implemented on image datasets that can easily reach billions of images.

In order to process extremely large datasets, this thesis evaluates distributing the computation across a cluster of machines using software such as Apache Hadoop. There are many different algorithms for conducting content based image retrieval, but this research focuses on kernelized locality-sensitive hashing (KLSH). For the first time, a distributed implementation of the KLSH algorithm using the MapReduce programming paradigm performs CBIR and localization using an urban environment image dataset. This new distributed algorithm is shown to be 4.8 times faster than a brute force linear search while still maintaining localization accuracy within 8.5 meters.

Acknowledgements

First and foremost, I would like to thank my wife for her support and encouragement during the last year and a half and over a decade before that also. I would not be where I am today with it. I'd also like to thank the fellow Army officers who never shied away or backed down from having a problem or discussion bounced off them. Many thanks are also owed to my Mother and Aunt for taking the time to perform proof reading for this document, regardless of the many other things which were occurring in their lives at the time. Thanks to you all!

Table of Contents

Page

List of Figures

Figure Page

List of Tables

Table Page

Table Page

List of Symbols

List of Abbreviations

Abbreviation		Page
GPS	Global Positioning System	1
DoD	Department of Defense	1
EXIF	Exchangeable Image File Format	2
KLSH	Kernelized Locality-Sensitive Hashing	2
HDCE	Hadoop Distributed Computing Environment	3
HDFS	Hadoop Distributed File System	4
SIFT	Scale-Invariant Feature Transform	7
ISR	Intelligence, Surveillance, and Reconnaissance	8
HDFS	Hadoop Distributed File System	9
CBIR	Content Based Image Retrieval	11
FLANN	Fast Library for Approximate Nearest Neighbors	13
LSH	Locality-Sensitive Hashing	13
FR	Full Representation	14
BoW	Bag-of-Words	14
DIRS	Distributed Image Retrieval System	17
RBF	Radial Basis Function	26
DARPA	Defense Advanced Research Projects Agency	39
ASPN	All Source Positioning and Navigation	39
RANSAC	Random Sample Consensus	83

DISTRIBUTED KERNELIZED LOCALITY-SENSITIVE HASHING FOR FASTER IMAGE BASED NAVIGATION

I. Introduction

1.1 Background

Modern navigation has been greatly aided by the advent and implementation of Global Positioning System (GPS) technology. Though the GPS system is a popular and widely used method for navigation, it is not infallible. GPS technology has been successfully spoofed to provide false position information [2], and even the GPS satellites themselves could be subject to an attack [3] which could disable or disrupt attempts at using the GPS system for navigation. It is because of the potential vulnerabilities with the GPS system that the Department of Defense (DoD) has sought to supplement and confirm this GPS location information with additional information that can be gathered from the environment.

Examples of navigation with addition information include navigating previously mapped locations by using visual cues, using laser range finders or radar, or by using speed and heading information [4]. This research finds the location of unknown images by searching a large database of images with known locations. Unfortunately, the volume of image data required to adequately map a large area quickly scales beyond the processing power and storage capabilities of any single computer. One way to overcome this limitation is to use clusters of machines working in concert to turn huge volumes of data into useful information. This research focuses on a computer vision problem that merges three different active areas of research: content

based image retrieval, localization, and distributed algorithms.

Content based image retrieval involves matching images by using the contents of the image, rather than using file meta-data, Exchangeable Image File Format (EXIF) data, or other similar information. Comparing images to one another and determining similar images to a query image is a problem that is fairly easy for a human to do, but this task is a difficult one for computers to do well. There are many different algorithms and techniques computer vision researchers have employed to tackle this problem. This research takes an existing algorithm (Kernelized Locality-Sensitive Hashing (KLSH) [5]) and modifies it to run on a cluster of machines.

Localization refers to the attempt to identify an unknown location by using previously collected information [6]. In the case presented by this research, a large database of images from an urban environment which have known locations is given. In other words, it is assumed that the GPS information was recorded and associated to the images when they were acquired. Then, by performing CBIR on a query image of an unknown location in the same city, any correctly matching images from the database with known locations will provide the location of the unknown query image. In addition to localization of information collected by ISR assets, the problem of localization also has forensic applications. Computer forensic specialists may recover an image of a crime being committed at an unknown location. Searching a database of known location images may provide a fruitful place to start looking for the scene of the crime.

Distributed algorithms run on a several machines instead of just one machine. Using several machines to cooperatively tackle a problem allows a larger dataset to be examined at rates much quicker than any single machine could process that same data. As more and more data is being collected every day, this has become a popular area of research both for the tech industry and the DoD. This research utilizes Apache Hadoop, which provides an open source implementation of both a means of

distributed processing using the MapReduce programming model and a distributed file system which can store a large number of files. The MapReduce distributed processing and distributed file system both make up the Hadoop Distributed Computing Environment (HDCE).

1.2 Problem Statement

This research examines the feasibility and scalability of conducting localization through CBIR by using a cluster of machines and a distributed implementation of the KLSH algorithm and a large database of images from an urban environment.

1.3 Research Hypothesis

The hypothesis of this research is that modifying the KLSH algorithm to run on a cluster of computers using Apache Hadoop can achieve significant retrieval speedup without loss of accuracy by conducting localization through the use CBIR. Two requirements to evaluate this hypothesis are as follows:

1. Develop a distributed implementation of the KLSH algorithm to perform content based image retrieval and localization from a large database of images from an urban environment.

2. Compare and contrast a brute force linear search with the distributed KLSH search by evaluating retrieval time and accuracy of retrieval.

1.4 Approach

In order to test the hypothesis, the KLSH [5] algorithm is modified and distributed using the MapReduce programming framework [1] and the distributed file system provided by Apache Hadoop [7]. Images are then be matched using the CBIR provided

by this distributed algorithm. The time required to service a query and the accuracy of the results returned are measured and compared to a brute force linear search. Due to limited available resources, the computing nodes making up the cluster of machines are virtualized using VMWorkstation on three separate physical blade servers. First, a non-distributed implementation of the KLSH and linear search algorithms is implemented and tested to ensure they conform to expected behavior. These two algorithms are then modified to use the Apache Hadoop MapReduce framework for distributed processing and the Hadoop Distributed File System (HDFS) and the scalability of these two distributed algorithms is examined.

The algorithms are evaluated by measuring the time required to conduct image retrieval and the accuracy of the images which are retrieved. The time of the algorithms begins when the query image is submitted and ends when the results are returned by the algorithms. The accuracy measurement involves how many of the results returned correctly match the query image, followed by a calculation of how large the physical distance is between the each of the correct matches and the query image.

1.5 Assumptions/Limitations

One assumption made by this research is that the Earth is a sphere with radius of 6,371 km. This assumption is required because the great-circle distance is calculated between two positions within the city of Columbus, OH. Since Columbus is relatively flat, the distances represented in this research are calculated with the haversine function for finding the closest distance between two points on a sphere. This research finds the distance between two points given their latitude and longitude in radians with 15 digits of precision. The relatively small elevation changes between the points on the ground was not considered.

One limitation of this research was the number of machines used in the cluster. Several other researchers ([8] [9] [10]) conducting similar computer vision problems with Apache Hadoop used hundreds (or even thousands) of machines to demonstrate and test the scalability of their algorithms. This research only virtualized 30 machines due to limited resources. Although more would have been better, 30 machines was still enough to demonstrate and test the scalability of the two algorithms examined by this research. Also, 30 machines provided enough results to project the performance of the algorithms if they were scaled up beyond what was tested by this research.

1.6 Research Contributions

This research successfully demonstrated that quick and accurate localization through CBIR can be achieved using a distributed implementation of the KLSH algorithm. This had never been done before. The distributed KLSH algorithm performed 4.8 times faster than the brute force linear search, while still maintaining a localization accuracy of approximately 8.5 meters.

1.7 Thesis overview

Chapter II provides some background information and concepts which must be understood in order to properly understand and follow the remainder of the document. Chapter III then goes into the detailed implementation of the algorithms examined during this research. Chapter IV discusses in detail the design of the experiments conducted, and Chapter V describes the results of the experiments and any conclusions discovered. Finally, Chapter VI offers a summary of this document and provides recommendations for future research that could be based on this research.

II. Background and Related Research

This research focuses on scaling localization through content based image retrieval from a large database. This problem can be broken down into the following sections that provide background used in the implementation:

1. Feature extraction - In order to compare images, they must somehow be represented quantitatively. Feature extraction techniques are discussed in Section 2.1.

2. MapReduce and distributed file systems - Because this research utilizes distributed computing to modify an existing algorithm, the MapReduce programming paradigm must be understood. Section 2.2 discusses this in detail.

3. Nearest neighbor algorithms - Several different algorithms and techniques have been applied to the problem of image retrieval from a large database. These techniques are discussed in Section 2.3.

4. Distributed image matching - MapReduce has been used for other image matching applications. These are discussed in Section 2.4.

2.1 Feature Extraction

An important area of research within the field of computer vision and image processing is feature extraction. Feature extraction involves reducing the dimensionality of image data in order to speed computations on the data being represented. A digital image is typically represented using pixels, which contain quantized values for brightness and color. Advanced cameras today are capable of taking images that contain over 50 million pixels. Attempting to quickly conduct computations on all the pixels of an image becomes difficult due to the large amount of information images contain.

Instead of using all of an image's pixel information, image data is summarized by using various different feature extraction techniques. There are many different techniques for conducting feature extraction, but for this research, images are compared to one another by using their Scale-Invariant Feature Transform (SIFT) [11] features.

SIFT features represent the key points of an image and are useful for conducting image comparisons because they are invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes. Each SIFT feature is expressed with the following attributes:

- Magnitude - A float value describing the prominence of a feature. These are sorted from most prominent to least prominent to ensure the most prominent features are compared to each other
- Orientation - a float value between 0 and 360.
- Location - the x and y coordinates of the feature.
- Descriptors - 128 integer numbers between 0 and 255.

Figure 2.1 shows a query image with its SIFT key points extracted and overlaid on top of the original image. Each circle on the right-hand image represents one key point detected and extracted by the SIFT algorithm. The location of each key point is the center of a circle. The magnitude of each key point is depicted by the radius of the circle. Circles with larger radii indicate that the key point has a larger magnitude. The orientation of each key point is depicted using the radius line contained in each circle.

The 1,000 most prominent key points are extracted from each image used for this research. The key points are then sorted by magnitude and stored into a flat-text file for use by the search algorithms. This file is referred to the image's scene file for the remainder of this document.

(a) Query image (b) Query image with SIFT keypoints displayed

Figure 2.1. A query image (left) and the same query image with SIFT key points extracted and overlaid on the image (right).

2.2 The MapReduce Programming Model and Apache Hadoop

In 2012, over 53 terabytes of data were collected every day in Afghanistan alone [12]. In order to turn this deluge of data collected by Intelligence, Surveillance, and Reconnaissance (ISR) assets into useful information, they must be processed, sifted through, and indexed for useful trends or correlations. Tech industry leaders like Facebook, Google, and Yahoo! sort and index even larger volumes of "big data" daily. When attempting to process volumes of data that large, the problem of indexing and querying quickly scales beyond the processing power and storage capabilities of a single computer.

Google introduced the MapReduce programming paradigm [1], which was developed to deal with such large volumes of data. Apache Hadoop's [7] is an open source implementation of Google's MapReduce programming paradigm [13] which has been used for many different things. Using Apache Hadoop, Yahoo! sorted 500 GB of data in 59 seconds with 1,406 nodes, and then sorted 1 PB of data at a rate of over 1.03 TB per minute with 3,658 nodes [14]. Facebook, also using Hadoop, scans over 2 PB of data daily using thousands of machines using Apache Hadoop [15]. These are just

a few of the many tech industry leaders who are using Apache Hadoop every day.

Processing large amounts of data by using a cluster of machines is an inherently difficult operation [16]. When writing distributed applications, issues such as how to efficiently parallelize computations, distribute data to the different nodes in the cluster, and gracefully handle task failures quickly complicate even simple operations. In 2008, Dean and Ghemawat [1] introduced a programming model which had been implemented at Google to process the vast amounts of data which Google collects each day and deal with these challenges. In this programming model, the programmer specifies a *map* function. One map task is a single execution of this map function. Each map task processes a key/value pair and produces a set of intermediate key/value pairs. Identical intermediate keys leaving different map tasks are then combined and processed further during the *reduce* function. This programming model is easily scalable, as each map task occurs independently from the others. Another benefit of the MapReduce programming model is that the number of machines in the cluster of computers can scale up or down and programmers need not change their code. Failures are also gracefully detected, and failed map tasks are reassigned to other active nodes to be completed, with no input necessary from the programmer.

Apache Hadoop [7] is an open source implementation of the MapReduce programming model, as well as an implementation of a distributed file system. As previously mentioned, sharing data with different computing nodes is one difficulty with writing distributed applications. The Hadoop Distributed File System (HDFS) is modeled after the Google File System [17] and it allows many machines to cooperatively store files such that all machines in the cluster have quick access to the data stored on the HDFS. Files on the HDFS are stored in blocks (64 MB per block by default) [13], and these blocks are replicated across several machine to provide redundancy and durability for the data on the HDFS should a node in the HDFS fail. The HDFS is a

master/slave architecture [18], where the master is the NameNode, and the slaves are the DataNodes. The NameNode is a process that runs on only one machine, whereas an instance of the DataNode process runs on each machine in the cluster which is participating in the HDFS. The NameNode manages the file system's namespace and maintains a directory tree structure of which file is stored in what block and on which nodes these blocks are stored. The DataNode processes do the actual work of storing and retrieving blocks when instructed by the NameNode (or users of the HDFS). The DataNode processes also periodically update the NameNode with the blocks they are storing.

Apache Hadoop also implements the MapReduce programming model, which is also a master/slave architecture [1]. The master machine in the cluster runs a process called the JobTracker. Each slave node in the cluster runs a process called the TaskTracker. The JobTracker coordinates, schedules, and assigns map and reduce tasks to the different TaskTrackers in the cluster. Each TaskTracker will run the tasks assigned to them by the JobTracker and will periodically report the status of these tasks back to the JobTracker. In the event of a task failure, the JobTracker will detect this after a certain period of time and reassign that task to a different TaskTracker. The MapReduce process is depicted in Figure 2.2

The steps on Figure 2.2 [1] are:

1. The user submits a program (or job) to the master machine's JobTracker process.

2. The JobTracker assigns map and/or reduce tasks to subordinate TaskTrackers on worker nodes. The user program will specify exactly what is done during each map/reduce task.

3. The worker nodes read an input split from the HDFS and execute their map

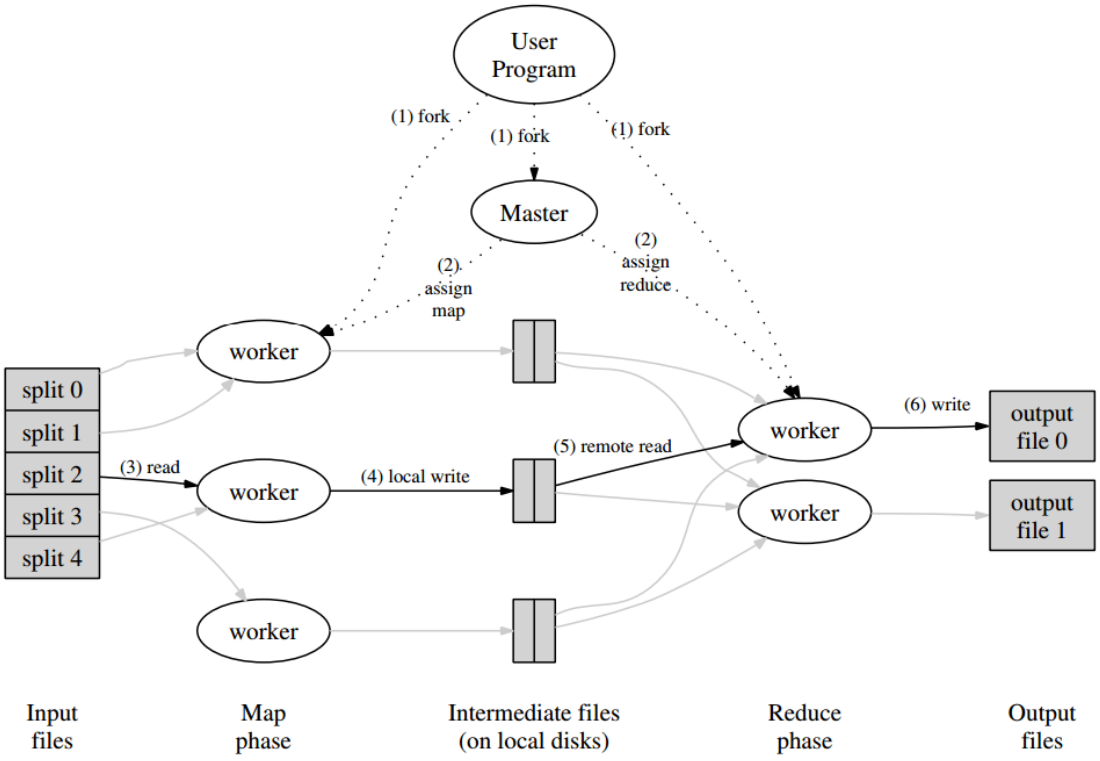


Figure 2.2. The MapReduce Process [1].

tasks according to the instructions provided by the user program.

4. The worker node's map task will output an intermediate key/value pair which will be stored on that machine's local hard drive.

5. These intermediate key value pairs are then read by the worker nodes assigned reduce tasks and identical intermediate keys are combined and processed according to the instructions provided by the user's program.

6. The final key/value pair is written to a file on the HDFS.

2.3 Nearest Neighbor Algorithms

Content Based Image Retrieval (CBIR) refers to a computer vision problem which involves retrieving images from a database similar to a query image by looking at the

content of the image rather than image meta-data [19]. A difficult and important step of this problem is locating the nearest neighbors, or most similar images, to the query image. Image comparison is typically done by extracting and comparing the features of the images to one another. Since this research used SIFT features to represent the images being compared, the SIFT features from all the images were extracted and used to compare images to one another. There are many different algorithms and techniques which have been applied to the problem of locating the closet matches to a query image.

The k-d tree algorithm introduced by Friedman, et al. [20] is a generalization of binary trees to higher dimensions. The k-d tree algorithm has been used for many different things, but recently, Silpa-Anan and Hartley [21] applied k-d trees to the image matching domain by conducting SIFT descriptor matching. Silpa-Anan and Hartley's algorithm creates multiple randomized k-d trees with different structures such that searching any individual k-d tree would be largely independent from searching the other trees. When only one k-d tree was used, the tree must be traversed up and down when it is realized that the current branch is unproductive. By using several independent k-d trees, this expensive back tracking is limited.

A different algorithm for clustering similar data into groups was introduced in 1967 by MacQueen [22]. In this paper, MacQueen introduced the k-means algorithm which categorizes n observations into k clusters, by placing each of the n observations into the cluster with the nearest mean (or cluster center). Once all the observations are grouped into one of the k clusters, the cluster's mean (or center) is recalculated and the process repeats again until the different cluster's centers remain the same. In 2005, Chen, et al. [23] introduced a modification to the k-means algorithm called hierarchical k-means . The hierarchical k-means algorithm improves the center for the clustering of the k-means algorithm by conducting p iterations of the k-means

algorithm and applying a hierarchical clustering algorithm to find better centers for the k different clusters. Recently, this hierarchical k-means algorithm was applied to the computer vision and object recognition problem set by Nister and Stewenius [24]. In Nister and Stewenius' research, images were represented with words. These words were used to build a hierarchical vocabulary tree which was then used for image retrieval and object recognition.

Muja and Lowe [25] introduced a Fast Library for Approximate Nearest Neighbors (FLANN) which automatically optimizes some parameters for the randomized k-d tree and the hierarchical k-means tree algorithms. This open-source project applies some machine learning to the algorithm optimizations to further improve retrieval accuracy and the time required to service queries. Ultimately, FLANN tries to answer the question, "What is the fastest approximate nearest-neighbor algorithm for my data?"

In 1999, a different algorithm for locating the nearest neighbors to a query image was introduced by Indyk and Motwani [26]. Rather than using tree based algorithms to find nearest neighbors, Indyk and Motwani used a hashing technique which they called Locality-Sensitive Hashing (LSH). LSH is different from traditional hashing algorithms in that, rather than seeking to minimize collisions, LSH tries to get similar objects to collide with one another. LSH was extended and applied to the computer vision and image matching domain by Kulis and Grauman [5].

Kulis and Grauman [5] mention that LSH has been implemented for quite a few functions (Hamming distance, inner products, ℓ_p norms, normalized partial matching, normalized set intersection, etc.). The intent of many of these functions is to allow the mapping of high-dimensional representations of images into very low-dimensional space. As long as this mapping maintains distances similar to the original data, these lower dimensional mappings can then be easily searched for nearest neighbors with known techniques. Essentially, similar images will map to "keys" which can be used

to encode similar inputs. The two authors' contribution produced an algorithm which would employ *arbitrary* normalized kernel functions for image mapping, while also making no assumptions about the data distribution of inputs, even if the underlying feature space of the data is not known. The authors summarize the KLSH algorithm as follows:

- Select p data points and form a kernel matrix K over this data.
- Form the hash table over database items: for each hash function $h(\phi(x))$, form e_s by selecting t indices at random from $[1, \ldots, p]$, then form $w = K^{-1/2}e_s$, and assign bits according to $h(\phi(x)) = \text{sign}(\Sigma_i w(i)\kappa(x, x_i))$.
- For each query, form its hash key using these hash functions and employ existing LSH methods to find the approximate nearest neighbors.

2.3.1 Existing Performance Comparisons of Algorithms.

Since there are so many different nearest neighbor algorithms, some researchers have attempted to compare and contrast them, to determine their strengths and weaknesses. Aly, Munich, and Perona [27] point out that searching a large dataset for a particular image poses three challenges: storage, computational cost, and recognition performance. The authors compare these variables using seven different methods based on two leading approaches for image retrieval: local feature matching (Full Representation (FR)), and visual words histogram matching (Bag-of-Words (BoW)) representations used SIFT feature extraction with Hessian affine feature detectors.

When comparing images using FR, the authors used 5 different algorithms (or data structures) to determine the nearest neighbors: Exhaustive search, k-d trees, LSH using the L2 hash function, LSH using the Spherical-Simplex & Orthoplex hash function, and Hierarchical k-Means. When comparing images using the BoW repre-

sentation, the authors used two algorithms to determine nearest neighbors: Inverted File, and Min-Hash.

In order to determine the theoretical scaling of these different methods, the authors compare and contrast the runtime and the storage required for all of the different techniques for determining the nearest neighbor of a probe image as the number of distractor images in the dataset grows. As the number of images increases, the authors also discuss the minimum number of computers with 50 GB of memory which would be required to store all the images. For calculating theoretical run time, the image dataset was evenly split amongst that number of computers, and each would return the nearest neighbor to the probe image from the images that machine had available to it. The only exception for the assumption, was for the KD-tree algorithm. They discuss how the *k-d* trees algorithm can improve the run time of the nearest neighbor query by taking advantage of advanced parallelization schemes and efficiently storing images on the leaves of the tree on "leaf" machines.

To validate theoretical scaling empirically, the authors experimented with four different data sets. Over the course of the evaluation, the number of images in the distractor datasets were increased from 100, 1k, 10k, 50k, 100k, and 400k. Recognition performance accuracy was measured by seeing if the one correct image was returned in response to a query. After some parameter tuning to optimize retrieval, the run time, storage, and recognition performance of the different methods was measured, and the results were analyzed.

FR methods provided better recognition performance than BoW, although required much more memory and longer run times. The BoW techniques did offer acceptable accuracy for certain datasets. The FR methods differed from one another by offering tradeoff between speed and accuracy. This speed/accuracy tradeoff is a reoccurring trend with the majority of these search algorithms.

Another paper which compares the performance of different hashing algorithms together was written by Pauleve, Jegou, and Amsaleg [28]. In their paper, they analyze the performance of several different hash functions on a real data set. To begin, the authors apply different lattice structures to quantize the data into nearest neighbor groups, which significantly improved the results of nearest neighbor queries. The authors determined that using k-means hash function to quantize the data, and then LSH to execute queries performed best. This was a similar technique used by Muja and Lowe in FLANN [25]. Then, they adapted and evaluated two recent modifications of the original KLSH algorithm described in Section 2.3, multi-probe KLSH and query-adaptive KLSH. Like many of the different image recall techniques, these offered time/accuracy trade offs. Query-adaptive KLSH adapts its behavior and picks from a large pool of existing random hash-functions in an attempt to locate ones that are the most likely to return the nearest neighbors, on a per-query basis. Query-adaptive KLSH had the highest recall and selectivity at the cost of increased memory and query preparation time. Multi-probe KLSH differs from the original KLSH algorithm in that, at query time, several hash-buckets per hash function are retrieved, instead of one. Between the original KLSH algorithm, query-adaptive KLSH, and multi-probe KLSH, multi-probe KLSH is the least memory-demanding method. The original KLSH algorithm, the method explored in this research, falls in the middle of the two.

2.4 Distributed Image Matching Algorithms

As the amount of image data to be searched grows, it becomes difficult or impossible to conduct image retrieval on a single machine. Much research has been done to take the image matching and retrieval algorithms discussed in Section 2.3 and utilize a cluster of machines to solve the problem. Several of the algorithms discussed

in Section 2.3 have already been implemented to utilize the MapReduce framework which was discussed in Section 2.2. The novel addition this research makes is to do this for the KLSH algorithm.

As was mentioned in Section 2.3, Silpa-Anan and Hartley's [21] optimization of k-d trees produced many different k-d trees, that can be searched independently from one another. This situation is an ideal candidate to be parallelized and solved with the MapReduce programming model. Aly, Munich, and Perona [8] described distributing these k-d trees to do image retrieval from a large database using MapReduce.

A different example of image matching using MapReduce was described by Moise et al. in their paper, "Indexing and searching 100 million images with map-reduce" [9]. They modified the extended cluster pruning algorithm to use MapReduce, and were able to conduct image retrieval on over 100 million images.

Another example of image retrieval using MapReduce was described by Zhang, et al. in their paper "DIRS: Distributed Image Retrieval System Based on MapReduce [29]. They implemented a CBIR system they called Distributed Image Retrieval System (DIRS) which used a Java image retrieval library provided by Lucene Image REtrieval (LIRE). With their system, they were able to store and recall over 20,000 images.

From the Air Force Institute of Technology, Murphy [10] successfully distributed the hierarchical k-means trees algorithm using images which were represented using the bag-of-words representation and fully represented by the SIFT features. Using Hadoop's MapReduce, Murphy was able to index and search over 2 million images.

2.5 Summary

The KLSH algorithm has been explored by many different researchers to conduct CBIR, and it is at the cutting edge of image matching research. Modifying algorithms

to utilize the MapReduce programming paradigm is also cutting edge research, as the amount of data scales beyond the capabilities of individual machines. It is for these reasons this research focuses on combining the two and implementing KLSH to run using the MapReduce programming paradigm. After a thorough literature review, it became apparent that this had not been attempted before.

III. Methodology

The ability to quickly and accurately identify the most similar images to a query image from a database of images is an important research topic for computer vision researchers. This content based image retrieval can aid in localization, autonomous vehicle navigation, and can also assist computer forensic specialists with crime scene location. This research focuses on scaling up the Kernelized Locality-Sensitive Hashing (KLSH) algorithm [5] to run in the Apache Hadoop [7] distributed computing environment (HDCE). Specifically, this research compares non-distributed and distributed implementations of a brute force linear search algorithm and the KLSH search algorithm and examines the scalability of these two algorithms.

This chapter discusses in detail the implementation of the algorithms evaluated by this research. First, the process of feature extraction and image representation is discussed. Then, the implementation of the non-distributed linear search and the non-distributed KLSH search is described in detail. Finally, the modifications required to make these two searches run using the HDCE is described.

3.1 Algorithm Overview

The general approach for determining the closest matches to a query image from a database of images is depicted in Figure 3.3. This process is broken into two portions, the off-line portion and the on-line portion. The off-line portion conducts non-query related image pre-processing tasks, while query related tasks occur during the on-line portion. At the end of the off-line portion of the process, several flat-text files are written and stored on the hard disk in preparation for future queries. At the beginning of the on-line portion of the process, the files generated during the off-line portion are loaded into memory and then a query algorithm is executed. The output from

the query algorithm, and the goal of the process, is to find the most similar images from the database to the query image. For consistency, only the on-line portion of all querying processes is timed.

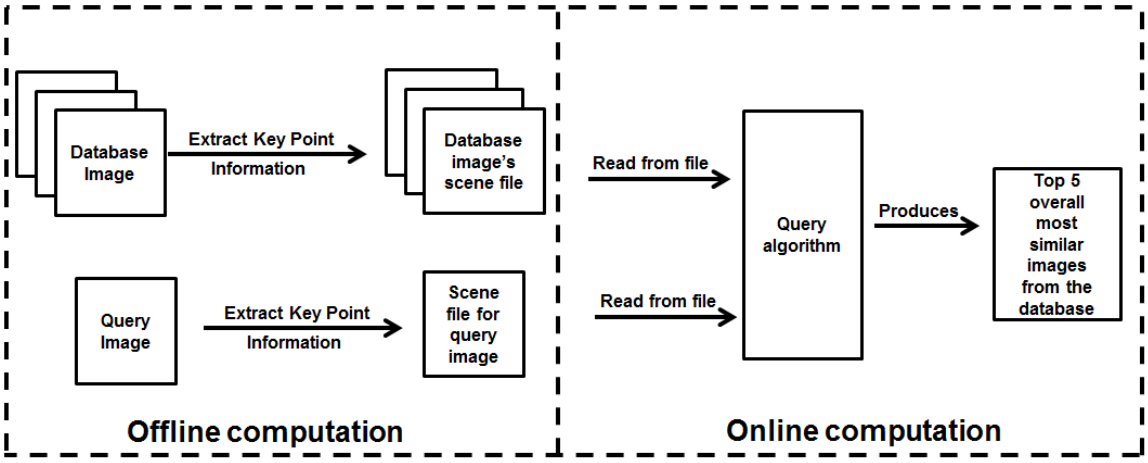


Figure 3.3. The general query process.

3.2 Off-line Computations

Various pre-processing calculations must occur before the querying process begins. First, in order to conduct CBIR, all images must be represented quantitatively. This is accomplished through Scale-Invariant Feature Transform (SIFT) [11] feature extraction. After this occurs, the KLSH search requires that hash and weight tables be generated. Since it is assumed that the database of images is accessible prior to when queries occur, this computation can be conducted during the off-line portion of the search process. These pre-query calculations are described in the following subsections.

3.2.1 Feature Extraction.

The OpenCV [30] implementation of the SIFT feature extraction algorithm detects and extracts all the features from all images in the data base and the query image.

For each image, the feature information is stored in a flat text file, which is referred to as the image's scene file. An example of a scene file is found in Appendix A.

When the SIFT algorithm extracts features from an image, the magnitude or prominence of each feature is calculated. During the off-line phase of the searches, each image's features are extracted and sorted by their magnitudes, and then they are stored in a flat text file for use during the on-line portion of the search. In order to compare one image to another, the images must first be quantified and represented by their features and descriptors. For a given maximum feature depth, D, an image, I, can be represented using its features and descriptors in the manner depicted in Figure 3.4.

$$I = \begin{array}{c} \\ F_0 \\ F_1 \\ F_3 \\ \vdots \\ F_D \end{array} \begin{pmatrix} d_{0,0} & d_{0,1} & d_{0,2} & \dots & d_{0,127} \\ d_{1,0} & d_{1,1} & d_{1,2} & \dots & d_{1,127} \\ d_{3,0} & d_{3,1} & d_{3,2} & \dots & d_{3,127} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{D,0} & d_{D,1} & d_{D,2} & \dots & d_{D,127} \end{pmatrix}$$

Figure 3.4. An image represented by its features and descriptors.

Where D is the maximum feature depth, F_n is the n^{th} feature of an image, and $d_{n,m}$ is the m^{th} descriptor of the n^{th} feature. An illustration of the max feature depth, D can be seen in Figure 3.5.

All images used for this research were represented by this quantitative method. Even though images can now be represented quantitatively, it is still necessary to develop a method to compare them to one another. In order to determine how different or similar features are from one another, the following method is given:

In order to look at a consistent number of features per image, the user must specify a D, only the first D features are considered for comparison. For example,

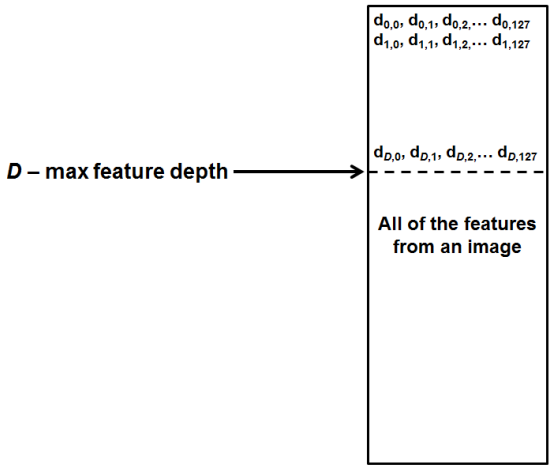


Figure 3.5. A graphical depiction of D.

if D was set to 300, only the 300 most prominent features as determined the SIFT algorithm from each image would be considered and compared. If an image contains fewer than D features, its remaining features and descriptors are populated with all zeros. In order to conduct image comparison, their features and descriptors are compared to one another and the similarity between the two features is expressed using a similarity measurement. The "similarity score" between two features (F_1 and F_2) can be expressed by Equation 3.1. Two identical features would have a similarity score of zero, and the closer the similarity score is to zero, the more similar the two features are.

$$Similarity\ Score = \sum_{j=0}^{127} ABS\Big((d_{0,j} \text{ for } F_1) - (d_{0,j} \text{ for } F_2)\Big) \tag{3.1}$$

where ABS is the absolute value.

3.2.2 KLSH Table Building.

Though the details of a KLSH search are discussed in Section 3.4.2, the general process can be seen in Figure 3.6. Several things occur during the offline portion of the KLSH search. First, all the image's key features are extracted and stored in their scene files, as was described in Section 3.2.1. Then, the radial basis function is applied a subset of the features from scene files of the images from the database. The resulting K matrix has its eigendecomposition taken and the resulting eigen values and eigen vectors are used to generate hash and weight tables. These hash and weight tables, along with some other information, are stored in a flat text file, which will be called the KLSH table files for the remainder of this document. These table files, along with the query image's scene file, are used during the on-line portion of the KLSH search in order to find the top five matches to the query image. An example of a KLSH table file appears in Appendix B.

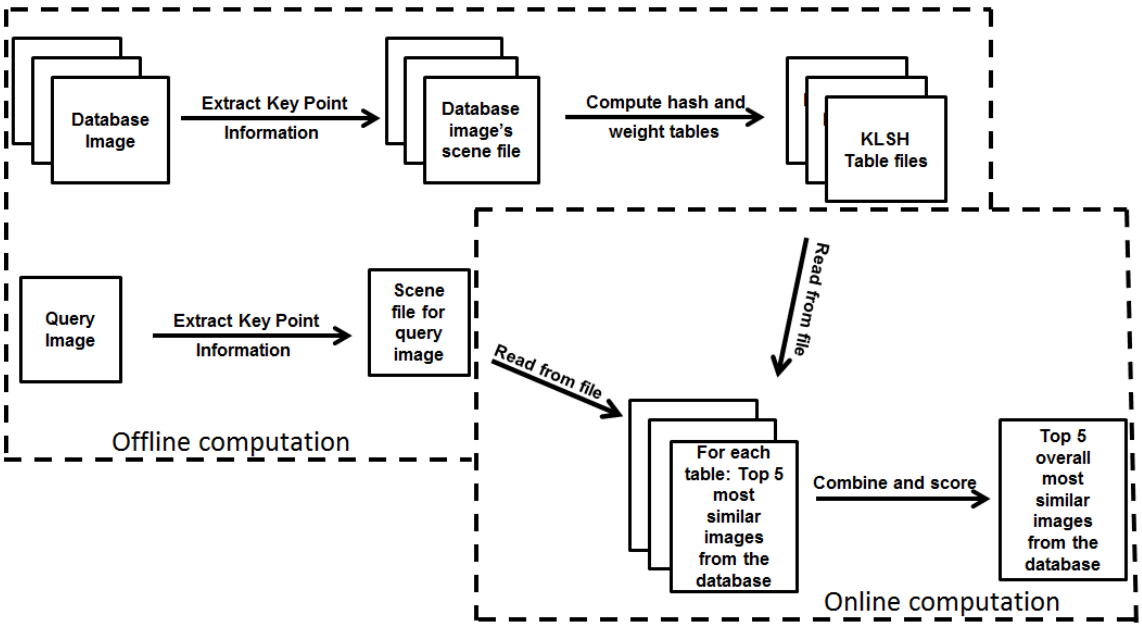


Figure 3.6. The process diagram for the non-distributed KLSH Search.

The table files are built from a subset of the features and descriptors from each

database image must be taken. This is done by using the following input parameters which are illustrated in Figure 3.7:

1. Maximum feature depth (D) - Only the most D prominent feature from each image will be considered up to the maximum feature depth. This is identical to the D which was defined for the non-distributed linear search described in Section 3.4.1

2. Number of features to be used per image (n) - n features are randomly sampled from those in the most prominent set as thresholded in Step 1.

3. Number of descriptors to be used per feature (d) - d descriptors are randomly chosen out of the 128 descriptors of each feature.

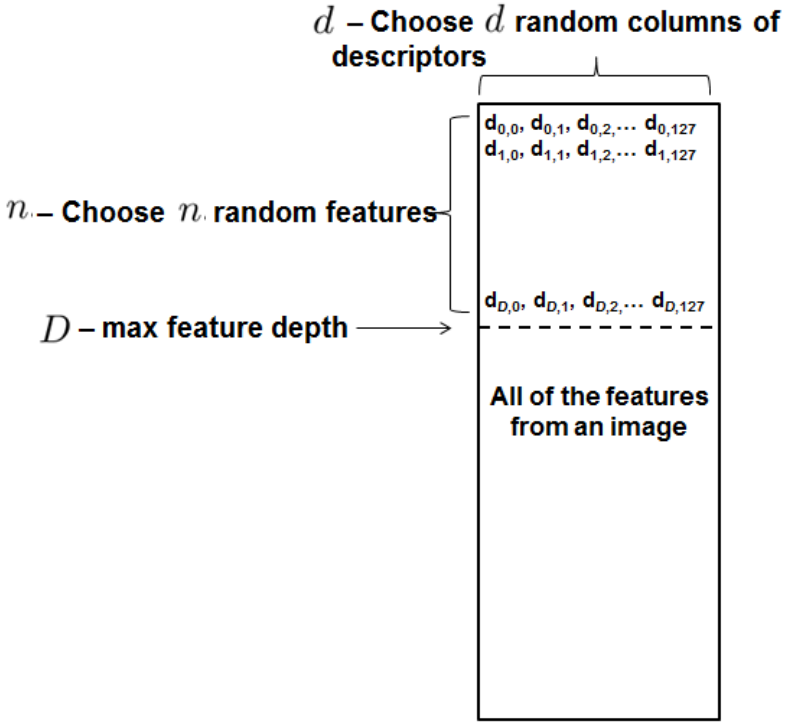


Figure 3.7. Depictions of D, n, and d.

Each table file includes feature representations for N images from the database. To build one table file, the same randomly selected features and descriptors are ex-

tracted from the scene files of each image in the database. These subsets of each image's scene file are combined and placed into a $(N * n) \times c$ matrix as seen in Figure 3.8. To build each table file, n new random features and d new random columns of descriptors are chosen. This is so that a different subset of the overall image database is represented by each table. Through the use of multiple table files, a consensus is built on which images from the database are the closest matches to the query image.

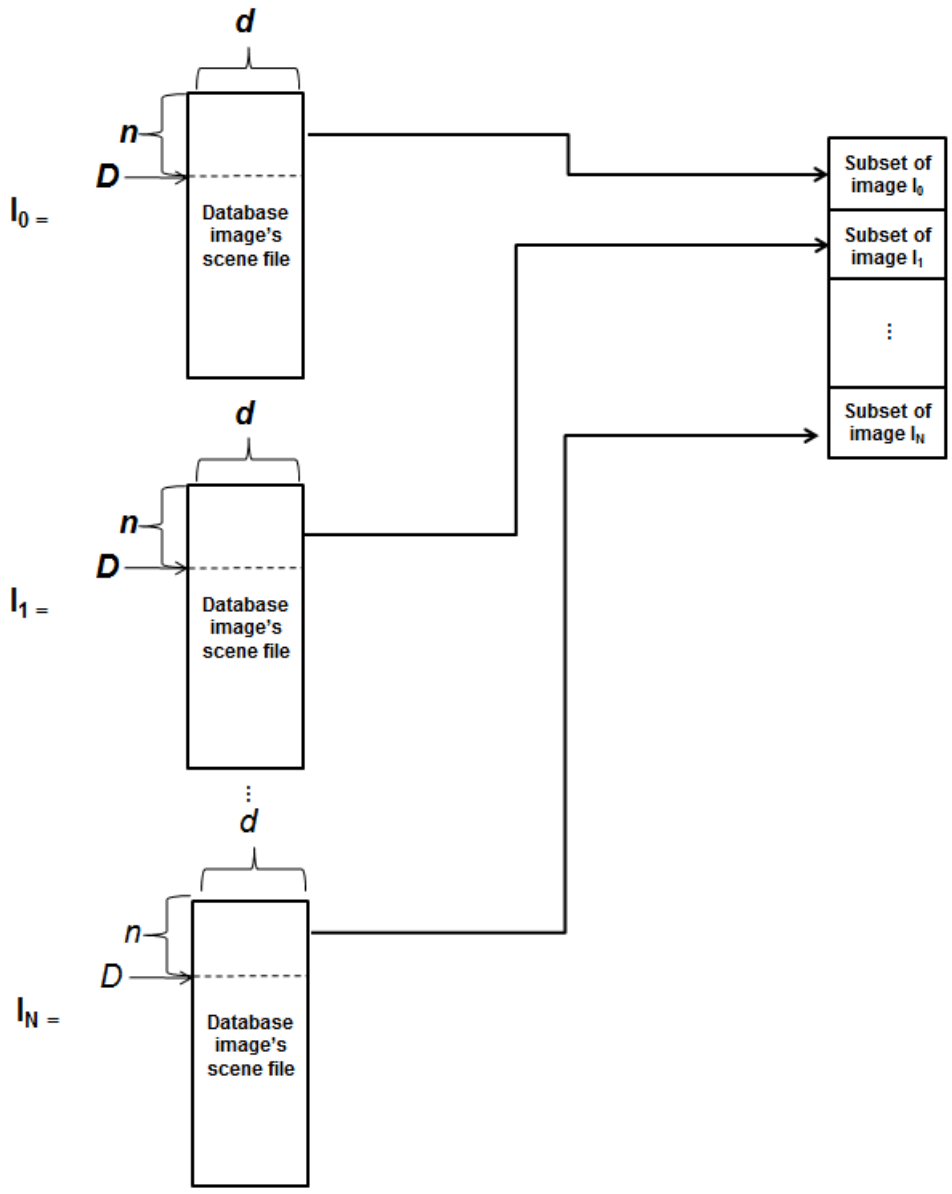


Figure 3.8. Combining the subsets of the database images into a matrix.

For each table file, the Radial Basis Function (RBF) is then applied to the combined subsets matrix described in the previous paragraph. The resulting matrix has its eigendecomposition taken. The resulting eigenvalues and eigenvectors are used to generate a hash table and a table of weights that represent how similar each image's features are to all the another sampled features. This process is shown in Figure 3.9. The whole process is repeated T times, to generate the T tables used during the on-line portion of the KLSH search.

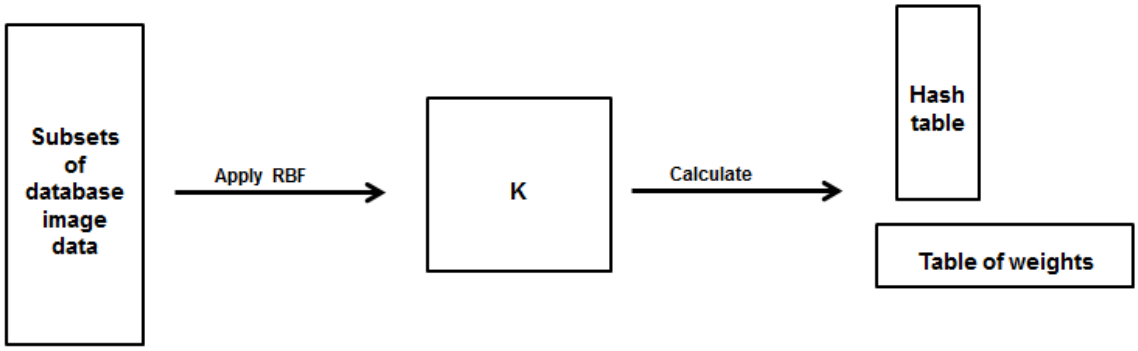


Figure 3.9. Calculating the hash table and the table of weights.

3.3 Combining and Scoring Features

The querying process ultimately finds *features* which are similar to one another. This information must be taken one step farther because this research is conducting image matching. The feature matches returned by the search algorithms must be combined and scored to determine which images are the most similar to the query image. When the search algorithms locate the five most similar features from the database to features from the query image, these features from the database may belong to several different images from the database. In order to combine and score the feature matches and determine which image from the database is the most similar to the query image, the images from the database are awarded points for each of their features which was returned as a match by the image matching algorithms. This

method is illustrated with the following example:

I_{query} has features F_0, F_1, and F_2.

The database is comprised of three images I_0, I_1, and I_2. The features belonging to these three images are as follows:

I_0: F_{00}, F_{01}, F_{02}, F_{03}, and F_{04}

I_1: F_{10}, F_{11}, F_{12}, F_{13}, and F_{14}

I_2: F_{20}, F_{21}, F_{22}, F_{23}, and F_{24}

Each image in the database, I_n, begins with zero "points." Every feature evaluated from the query image has five most similar features from the database as shown in Table 3.2. Points are awarded according the Table 3.1. The image from the database with the most points at the conclusion of the scoring is deemed to be the most similar to the query image. The image with the second most points is the second most similar to the query image, and so on.

Table 3.1. Scoring table used for all searches.

Position	Number of points
Most similar	5
Second most similar	4
Third most similar	3
Fourth most similar	2
Fifth most similar	1

After conducting the combining and scoring depicted in Table 3.2, I_0 has 26 points, I_1 has 7 points, and I_2 has 12 points. Therefore, I_0 would be considered the most similar to I_{query}, I_2 would be considered the second most similar, and I_1 the third. This combining and scoring process can be applied to both the most similar features, as was depicted in Table 3.2, or it could also be applied to images. As will be described soon, during the KLSH search, after processing one KLSH table, the top five images which match the query image are discovered. By applying the point values from Table 3.1 again to the images, the most similar images can also be

Table 3.2. Scoring Table Example.

	Features from the database				
Query Feature/- Points Awarded	Most similar	Second most similar	Third most similar	Fourth most similar	Fifth most similar
F_0	F_{00}	F_{20}	F_{02}	F_{01}	F_{14}
Points awarded	I_0 awarded 5 points	I_2 awarded 4 points	I_0 awarded 3 points	I_0 awarded 2 points	I_1 awarded 1 point
F_1	F_{23}	F_{10}	F_{04}	F_{02}	F_{03}
Points awarded	I_2 awarded 5 points	I_1 awarded 4 points	I_0 awarded 3 points	I_0 awarded 2 points	I_0 awarded 1 point
F_2	F_{04}	F_{03}	F_{23}	F_{11}	F_{00}
Points awarded	I_0 awarded 5 points	I_0 awarded 4 points	I_2 awarded 3 points	I_1 awarded 2 points	I_0 awarded 1 point

discovered using the technique described in this section. This technique of combining and scoring is used and referenced in the difference searches described in the Sections to come.

3.4 On-line Computations

Two different search algorithms are implemented for this research, a linear search and a search using KLSH. The non-distributed implementation of these algorithms is examined first. Then, the two algorithms are modified to use the MapReduce programming paradigm and Hadoop distributed file system (HDFS) and their scalability is examined.

3.4.1 Non-distributed Linear Search.

The linear search is a brute force search that returns the most similar images for each query image. The general process of the on-line portion of the linear search is depicted in Figures 3.10 and 3.11. The extraction of the SIFT key points is assumed to have already occurred during the off-line portion of the process, as was depicted

in Figure 3.3.

The first step of the on-line portion of the linear search compares each feature in the query image up to D, to each feature of every image in the database up to the same D by using the "similarity score" described in Section 3.2.1. Figure 3.10 shows this step occurring for only the first feature of the query image, F_0. The lowest five difference scores for F_0 are the most similar five features to F_0 from all the features of all the images in the database. This process is repeated for every feature in the query image up to D. Once the most similar features to each of the query image's features have been found for each feature in the query image, these results are combined and scored as described in Section 3.3 in order to determine the overall five most similar images from the database to the query image. This whole search process is non-distributed because all computations occur on one computer and neither the HDCE nor the HDFS are used.

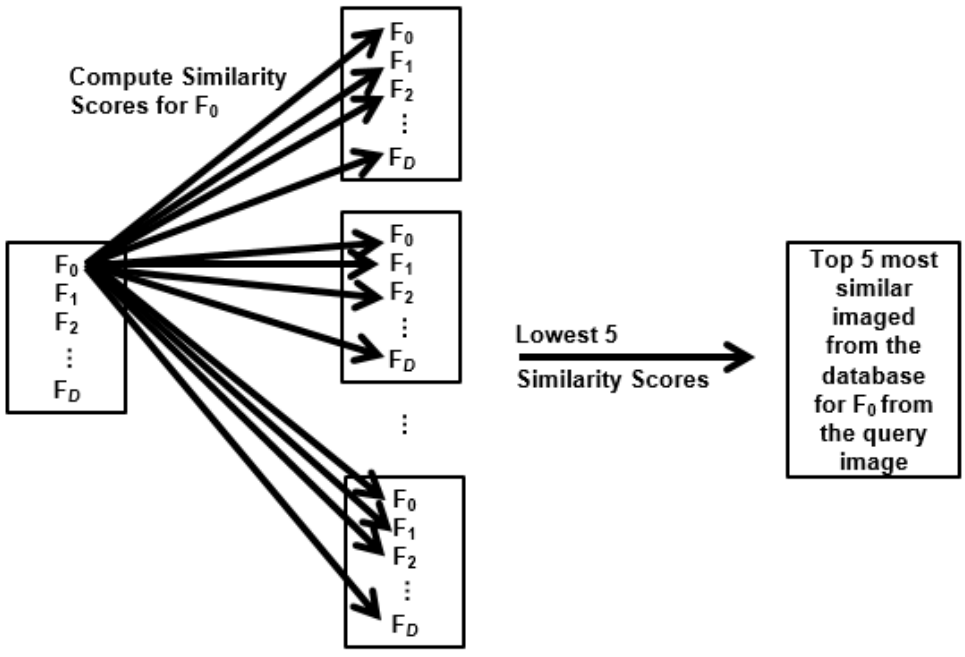


Figure 3.10. A process diagram of the first step of the on-line portion of the linear search.

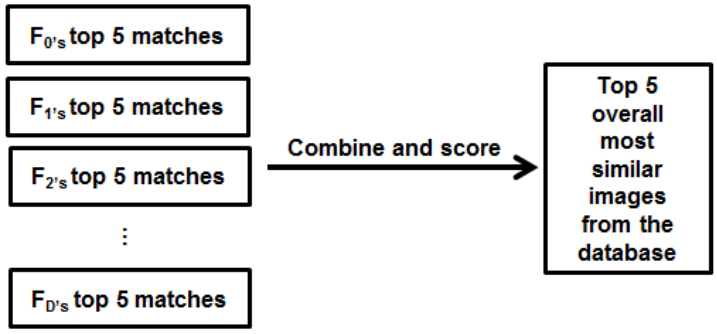


Figure 3.11. A process diagram showing the final step of the linear search.

3.4.2 Non-distributed KLSH Search.

Prior to the start of the on-line portion of the KLSH search, it is assumed that the query image's features have already been extracted into the query image's scene file and the KLSH hash tables and table of weights have already been computed. At the beginning of the on-line portion of the KLSH search, the query image's scene file and each table file are read from the flat text files into memory. Each table presumably has a different randomly chosen set of features and columns of descriptors. For each table, the same subset of features and descriptors is taken and appended to the subsets of the database images, as shown in Figure 3.12.

The same RBF is applied to this new matrix and a portion of this matrix is extracted. This portion relates the features from the query image to the features in the subset of the database images that were chosen. This portion is then matrix multiplied with the table of weights which was previously calculated. This matrix multiplication results in correlation values between negative one and one. Values greater than zero mean that two features are more strongly correlated to one another. Then a logical matrix of bits is created, where all values greater than or equal to zero

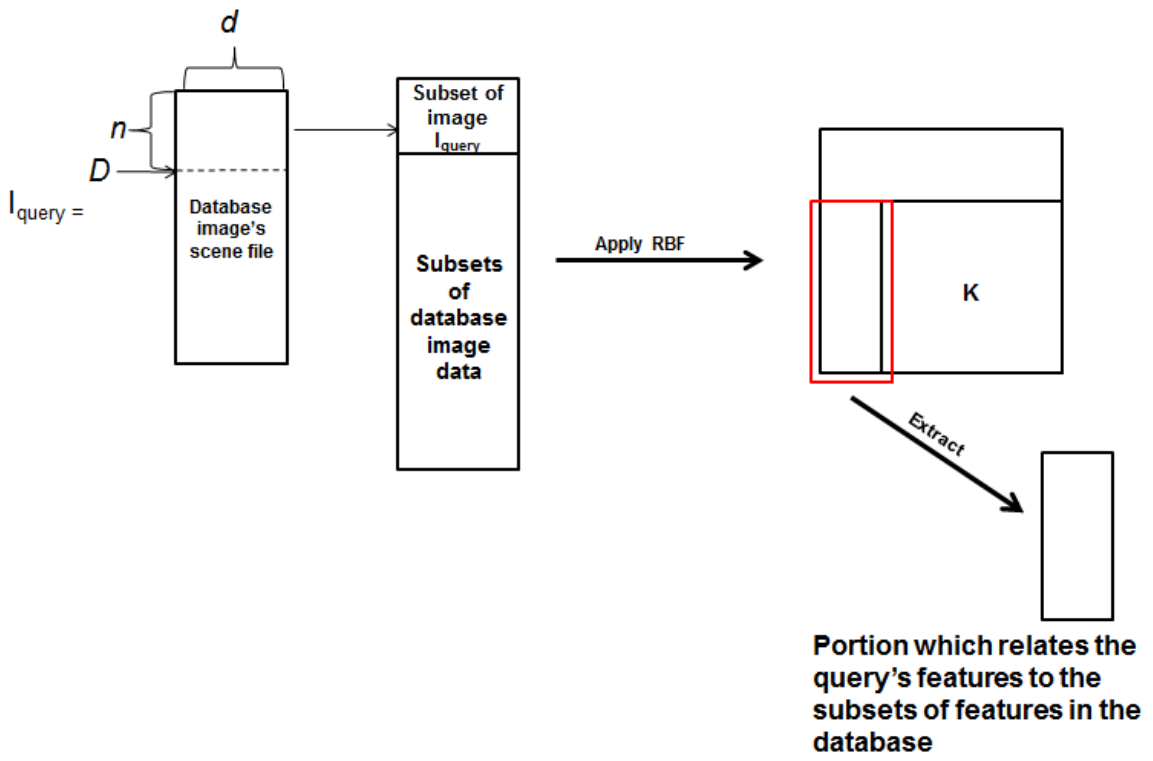


Figure 3.12. Extracting a portion of the K matrix which relates the query features to the features from the database.

turn into 1's and values less than zero turn into 0's. This process is shown in Figure 3.13. Finally, the Hamming distance is taken between this logical matrix and the hash table previously calculated. The results of the Hamming distance calculation yields the overall distance matrix. This distance matrix tells how closely related the features from the query image are to each of the features sampled from the database images. All that remains is to sort the results and pull the top five matches for each feature for a query image. These matches are scored according to Section 3.3 as was done during the linear search. The five images with the most points after this scoring are the five most similar images *for this table*. This process must occur for T different tables. Once the top five matches for every table have been discovered, they are scored again as was described in Section 3.3. The top five images with the highest overall points after this final scoring are the results returned by the KLSH search as

the top five most similar images to the query image. This final scoring process is depicted in Figure 3.14.

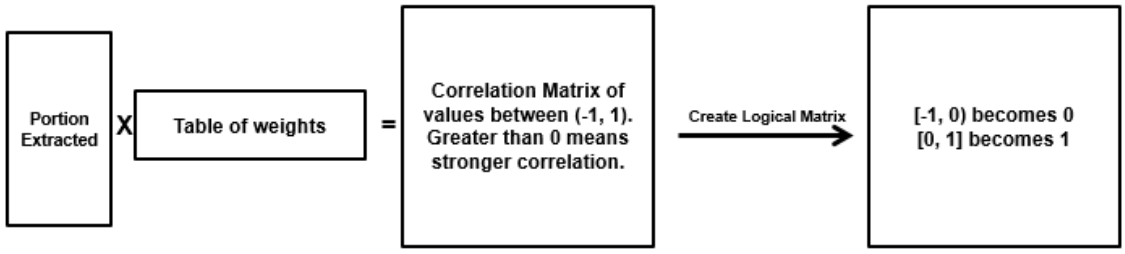


Figure 3.13. Creating a logical matrix after matrix multiplication.

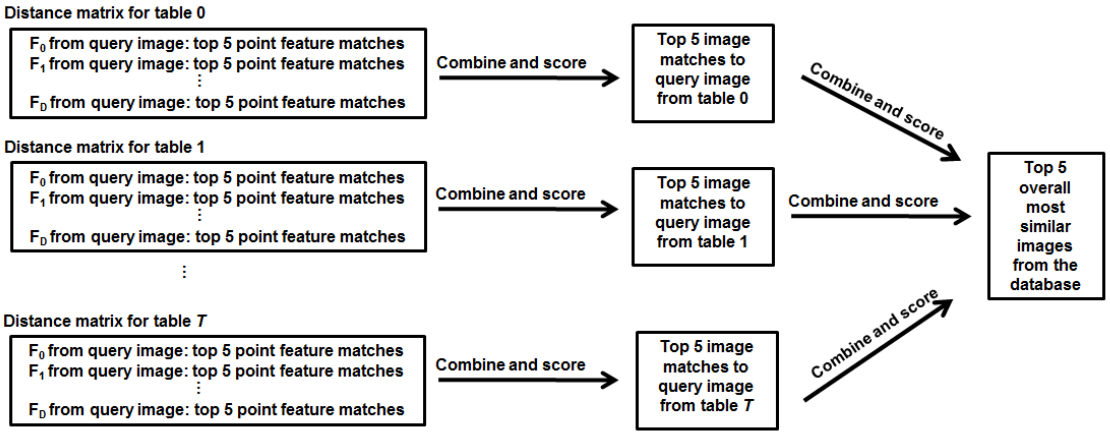


Figure 3.14. Final scoring of the KLSH search algorithm.

3.4.3 Distributed Linear Search.

The distributed linear search is modified to use the MapReduce framework provided by the HDCE and the distributed file system provided by the HDFS. During the off-line portion of the search, the scene files belonging to the query and database images are uploaded to the HDFS, so all computing nodes can access them. As shown in Figure 3.15, each map step of the distributed linear search processes one image from the database and compares it to the query image. During the map step, each feature

of the query image up to depth D is compared to every feature from the one image from the database up to depth D. For each feature in the query image, the features from the database with the lowest five difference scores are sent to the reduce step. The reduce step sorts the difference scores from all the images from the database in order to determine which images contain the overall lowest difference score for each feature of the query image. Finally, this is scored again according to Section 3.3 in order to determine the overall five most similar images from the database to the query image.

3.4.4 Distributed KLSH Search.

The distributed KLSH search is modified to use the MapReduce framework provided by the HDCE and the distributed file system provided by the HDFS. During the off-line portion of the search, the KLSH table files representing the database images are uploaded to the HDFS, so all computing nodes can access them. As shown in Figure 3.17, each map step of the distributed KLSH search processes one KLSH table file and compares it to the query image.

The generation of the table files for the distributed version of the KLSH search differs from the non-distributed version. During the non-distributed version, all files in the database were represented in each table file. Building the table files for the KLSH search is a fairly computationally expensive step which takes quite a long time. Some pilot testing showed that a table file that represents more than a few hundred images is impractical to generate given current computer technology. Because of this, the images from the database are grouped into more manageable chunks. It is also important that each image file from the database appears in roughly the same number of table files.

The chunking processes uses p to represent the number of database image files

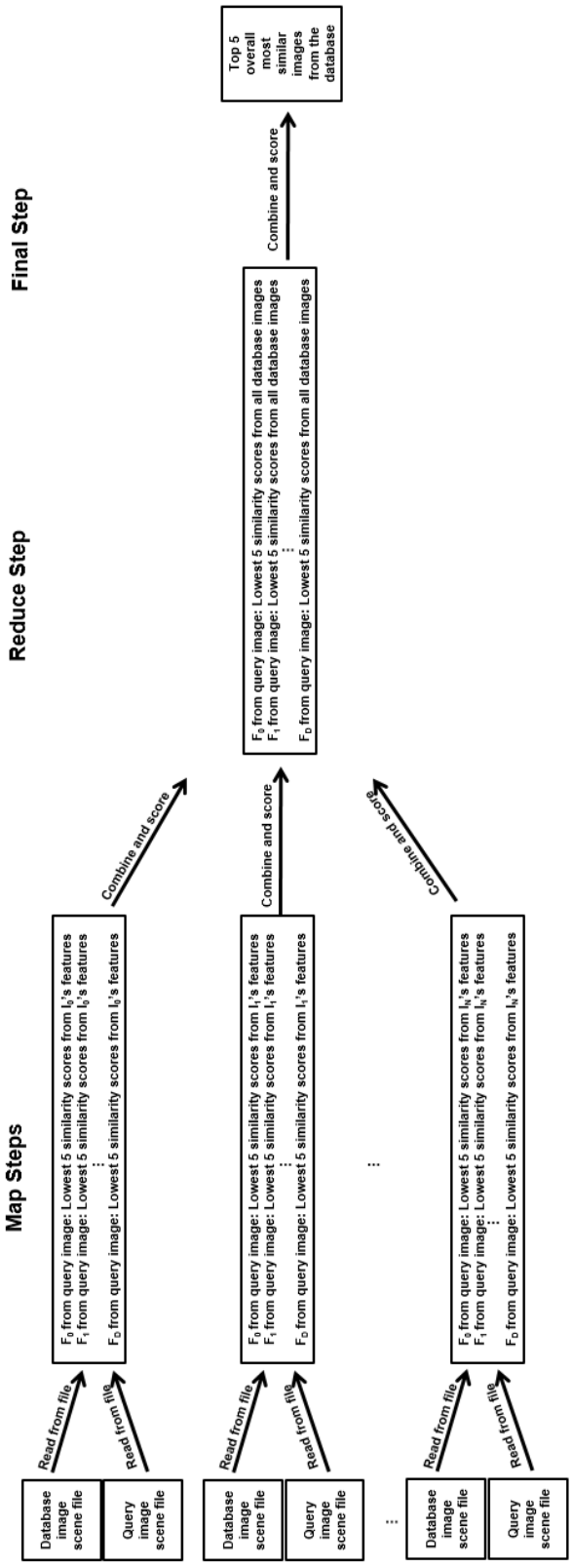


Figure 3.15. Map-reduce steps for the distributed linear search.

represented *per table* and i represent the total number of images in the database. To build the table files for the distributed KLSH search, the user specifies that T tables must be built. First, the order of all the files is shuffled and p images are grouped together to generate $\lfloor \frac{i}{p} \rfloor$ tables. Then, all the files are randomly shuffled again and the process repeats until the required T tables have been built. If p evenly divides i, then each file in the database is represented at most one fewer times than any other file in the database. Figure 3.16 depicts what would happen when $i = 10$ and $p = 5$.

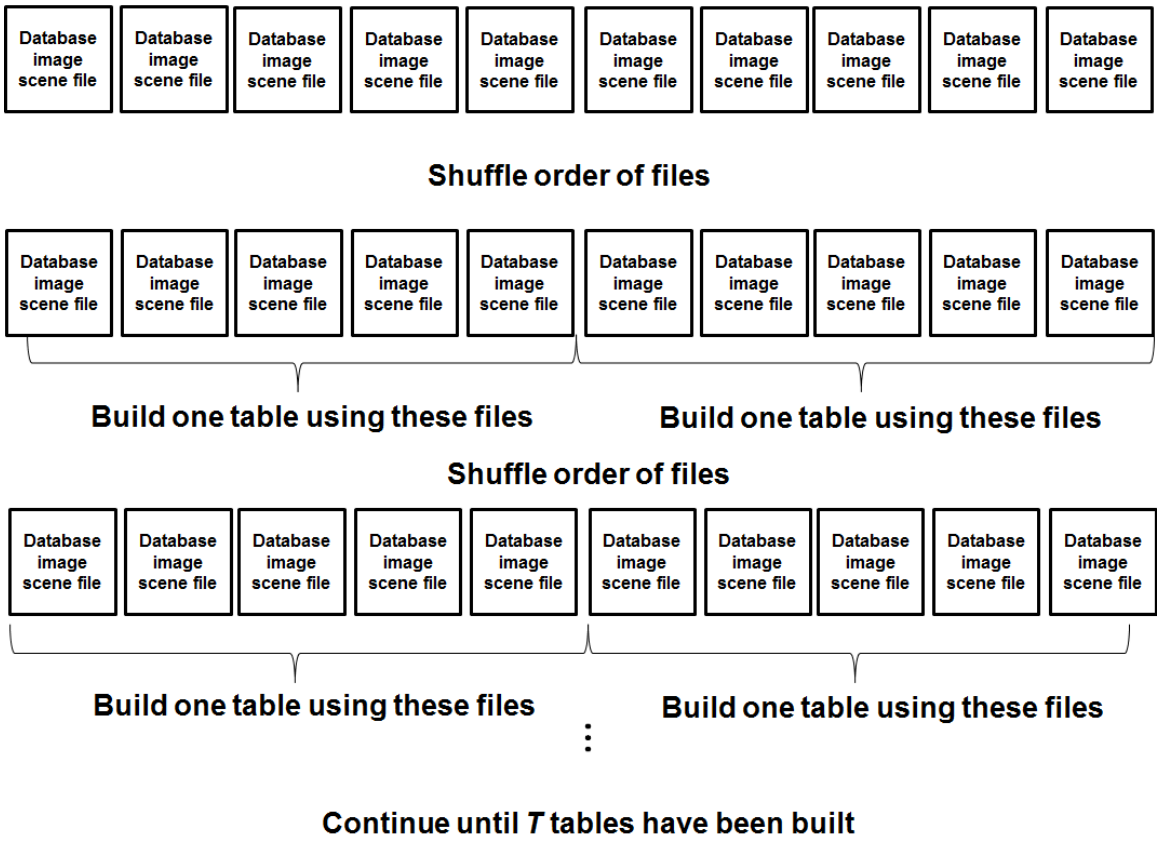


Figure 3.16. Building table files by selecting groups of files from the whole database of image files.

Figure 3.17 shows how the map and reduce steps of the distributed KLSH algorithm work. Each map step processes the query image's scene file and one table file. The output from each map step is the five most similar images to the query image from the images represented by that table. The reduce step of the distributed KLSH

search algorithm takes this information and scores it according to Table 3.1. After scoring, the five images with the most points are returned as the most similar images to the query image.

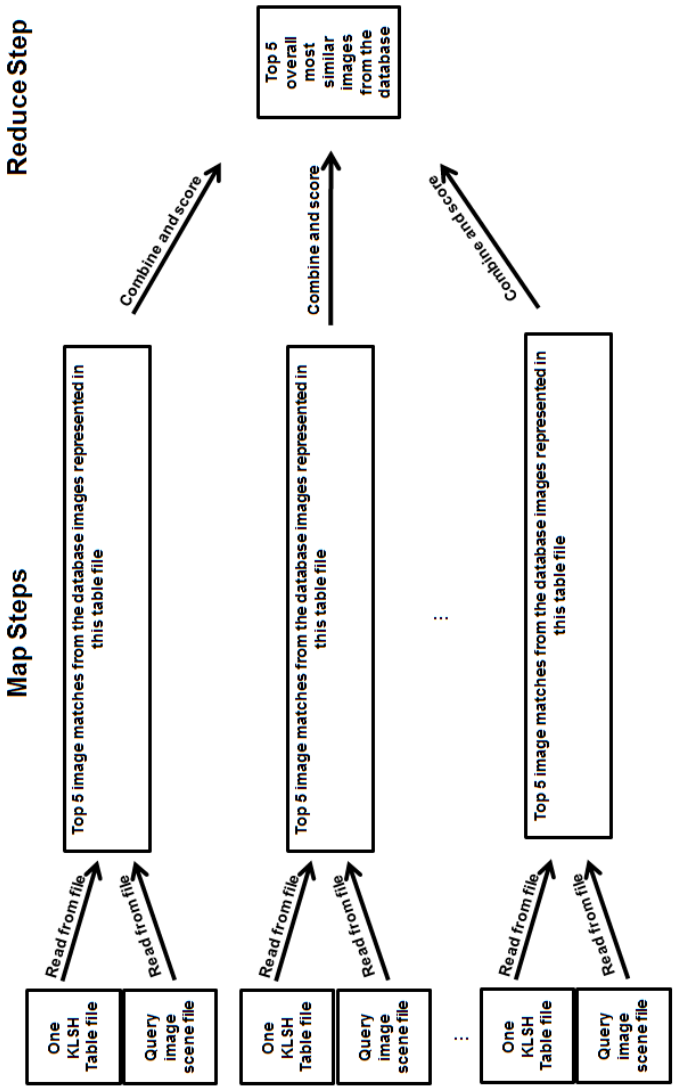


Figure 3.17. Final scoring of the KLSH search algorithm.

3.5 Distance Calculation

Once images matching the query image are successfully retrieved from the database, the great-circle distances between the query image and each match to that query image are calculated by using the haversine formula given in Equation 3.2. These distances are then averaged together to determine the overall distance metric for that particular query.

$$distance = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \quad (3.2)$$

Where *distance* is the distance between two points along a sphere, r is the radius of the sphere, ϕ_1 and λ_1 is the latitude and longitude of one image, and ϕ_2 and λ_2 are the latitude and longitude of the other image. Distances were calculated assuming the Earth was a perfect sphere with r=6,371 km. This distance calculation is conducted after the on-line portion of all queries, so the time to conduct this calculation is not included in the overall time metric.

3.6 Summary

By distributing the calculations of the KLSH search, it is hoped that quick and accurate localization can be obtained through CBIR. The images in the large database are assumed to have known locations associated with them. By conducting CBIR on a query image taken from an unknown location, correctly matching images from the database with known locations will uncover the location of the unknown image. By also distributing the computations of the linear search, comparisons and contrasts can be made between the time and accuracies of the two searches.

IV. Experimental Design Overview

Image matching algorithms are typically evaluated by measuring how long they take (speed) and whether they return the correct results (accuracy). Additionally, since this research is concerned with conducting localization through the use of Content Based Image Retrieval (CBIR), a third metric of distance is used to evaluate the algorithms. This distance metric expresses how many meters separate the GPS location of the query image from GPS locations of the results returned by the algorithms.

This chapter discusses the design of the experiments conducted. Effective experimental design is critical in order to obtain statistically significant results. First in this chapter, the test environment is described. Then, the dataset used for the experiments is discussed. Next, the performance metrics of time and accuracy are laid out and described in detail. The different types of scalability explored by this research are then defined. Finally, the test runs for the different searches are described in detail.

4.1 Test Environment

One of the advantages of the Hadoop software is that it is designed to be run on commodity hardware [13]. The hardware and software used for the experiments for this research are described below.

4.1.1 Cluster Hardware.

Virtualization was employed for the nodes in the Hadoop cluster. Three identical servers acted as the host machines. Each host machine contained two AMD Opteron processors with six cores each and clock speeds of 2.60 GHz. Each server contained 64 GB of RAM. The virtualized guest machines were assigned one processor each

and two GB of RAM each. The master guest machine was assigned one processor and three GB of RAM. No more than 11 guest machines were hosted on any physical server at any time, so one processor was reserved for the host operating system. Each of the three servers had two network interface cards. One network interface card was used by the host operating system for remote control of the servers. The other network interface card was connected to a Cisco 3560G PoE 24 port switch. The three ports of the switch were assigned their own VLAN in order to prevent non-experiment related network traffic from interfering with the data. Using VMWare Workstation, all guest operating system's virtual network interfaces were bridged to the network interface card which was connected to the private VLAN.

4.1.2 Cluster Software.

The host machines were running Windows Server 2008 R2 Enterprise 64 bit edition. Virtualization was accomplished by using VMWare Workstation version 10.0.1. Guest operating systems were Ubuntu Linux server version 12.04.5 LTS 64 bit edition. Java version 1.7.0_65 was used by all guest machines. Apache Hadoop version 1.2.1 was used as it was to most current stable version at the time of this research.

4.2 Database and query images

The database and query images were drawn from almost 15,000 images collected for the Defense Advanced Research Projects Agency (DARPA) All Source Positioning and Navigation (ASPN) program. This program focused on trying to supplement GPS with other sensor data, such as laser range finders and cameras [4]. To build the DARPA ASPN dataset, a vehicle was driven around Columbus, OH with a front facing camera which captured images four times per second. The image database used for this research is a subset of the DARPA ASPN database of images.

Since the DARPA ASPN dataset contained almost 15,000 images, it was necessary to further sub-divide this dataset to form the query images and database used in this research. This was accomplished by pseudo-randomly selecting groups of six images, where each group of six images would be separated by at least 2.5 seconds on either side from any other group of six. An example group of six images can be seen in Figure 4.18. The sixth image from each group of six was placed in the pool of query images, while the remaining five images were placed into the group of images to form the database. If the sixth image were queried into the database, the other five image are the "known best matches," and are what we would expect our search algorithms to return. This guarantees that each query image has at least five matches in the database.

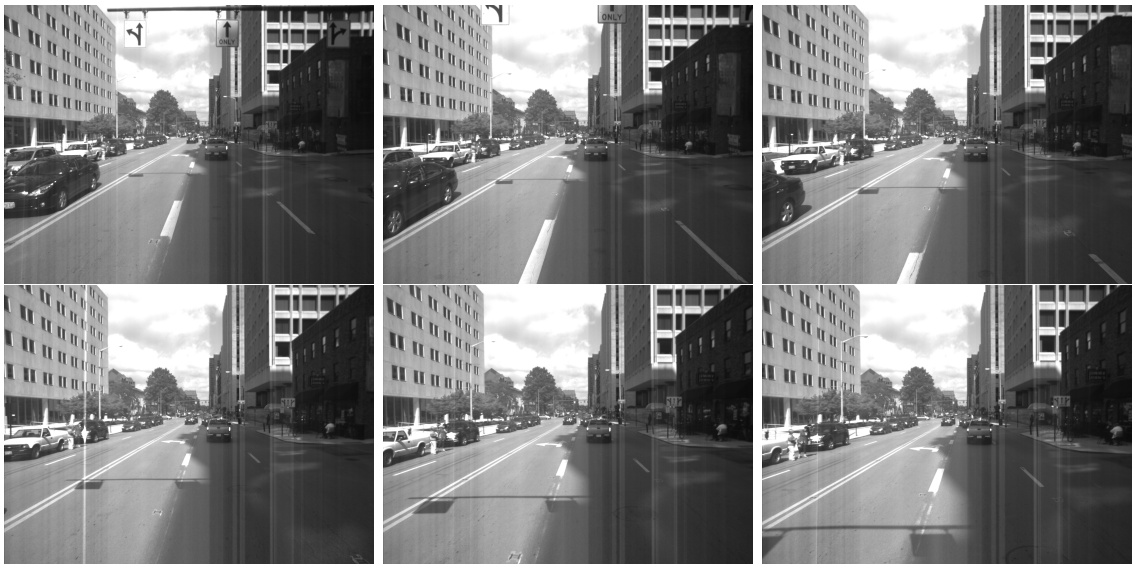


Figure 4.18. An example group of six images pulled from the ASPN dataset.

4.3 Definitions

4.3.1 Performance measures.

When a search is executed on a query image to find the most similar images from a database of images, there are two performance metrics of interest, query response time and the accuracy of the images returned.

1. Query time - The time taken by the algorithm to return the top five matches to the query image from the database of images. This metric is measured in seconds (s) by the system clock on the computer executing the query.

2. Accuracy - The accuracy of the results of a search are the number of correct matches to the query image that this search was able to find. The accuracy percentages can be found in Table 4.3. The position of the match (i.e. best match, second best match, etc.) does not impact the accuracy score.

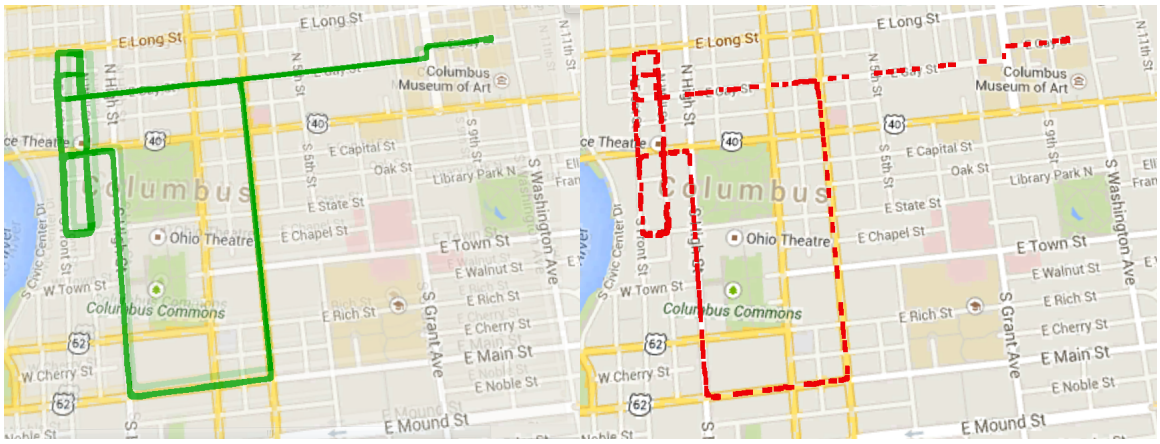
3. Distance - Once the correct images matches to the query image are located by the search algorithms, the great-circle distance is calculated between them and the query image using the haversine formula [31] with the GPS locations of the two images.

Table 4.3. Accuracy rating of matches returned by the algorithms.

Number of correct matches out of 5 possible	Accuracy
0	0 %
1	20 %
2	40 %
3	60 %
4	80 %
5	100 %

The DARPA ASPN dataset was obtained by driving around Columbus, OH, and the route driven is depicted in Figure 4.19. The groups of six images which made up

the dataset for this research are also depicted in Figure 4.19. Despite there being at least five known matches in the database for every query image, it was still necessary to manually score the results returned by the algorithms. As stated in Section 4.2, each group of six images is separated by at least 2.5 seconds from every other group of six images. However, the route used when collecting the DARPA ASPN dataset included several loops, and the crew collecting the dataset revisited the same locations repeatedly but at different times. Also, duplicate locations could appear at different times if the vehicle was stopped for longer than 2.5 seconds at a stop sign or red light. Figure 4.20 show several groups of six images which were taken at the same location but at different times. These database images still show the same location, but they belong to a different group of six images. This other group of images are also considered matches to the query image.



(a) Entire DARPA ASPN dataset (b) Sampled dataset used for this research

Figure 4.19. The path traveled during the collection of the DARPA ASPN (green dots on left map) and the sampled subset of this data used for this research (red dots on right map).

The manual scoring was aided by a program written for this purpose. The program initially knows no correct matches beyond the initial groups of six images. If one of the search algorithms returns an image which is not a known match, it is scored by the user as being a match or a non-match. If it is a match, all images in

the two groups are then stored as valid matches to each other.

Figure 4.20. An example of images from different group of six which would still considered matches to one another.

4.3.2 Scalability.

The scalability of a distributed algorithm deals with how well or poorly the algorithm performs as the number of computing nodes active in the cluster, C, is increased. Therefore, C will be the factor explored for both the distributed linear search, and the distributed KLSH search. The two different forms of scalability are:

1. Strong scalability - The performance of a distributed algorithm when the size of the problem is fixed and the number of computing nodes in the cluster is varied. [32]

2. Weak scalability - The performance of a distributed algorithm when the size of the problem *per node* is fixed and the number of computing nodes in the cluster is varied. [32]

For both the KLSH and linear search distributed algorithms, the query time and accuracy metrics are measured while varying several factors associated with strong

and weak scalability. These factors are described in detail in Sections 4.4.3 and 4.4.4.

4.4 Experimental Runs

Several different test runs with different input parameters are required in order to explore the feasibility and scalability of the different search algorithms. The different test runs are described in the following paragraph.

4.4.1 Non-distributed Linear Search.

The only factor which affects the linear search is the maximum feature depth, D. D is varied at four levels, 100, 300, 500, and 700 in order to determine the impact D has on both accuracy and query time. For the non-distributed linear search, the number of images in the database, i, is fixed at 100 images.

For each of the four levels of D, 20 different images are queried into the database five times each. This yields 100 observations per query depth. This process is randomized and repeated three times. The metrics to be measured are the time taken to respond to a query and the accuracy of the results returned.

4.4.2 Non-distributed KLSH Search.

The version of the KLSH search used for this experiment is a translation of the implementation of Kulis and Grauman's work [5] from MATLAB into Java. Though all efforts were made to translate the implementation line by line, it will still be important to reach statistical certainty that the translated algorithm conforms to expected behavior. The different factors for the non-distributed KLSH search are the maximum feature depth ,D, the number of KLSH tables, T, the number of features used, n, and the number of descriptors used per feature, d. These factors were depicted graphically in Figure 3.7.

For the non-distributed KLSH search, the number of images in the database, i, is fixed at 100 and each KLSH table represents all 100 images in the database. Table 4.4 shows the factors and how they are varied in order to test their impact on both the time taken to service a query request and the accuracy of image retrieval. The base case for the non-distributed KLSH search is when $D = 500$, $T = 20$, $n = 15$, and $d = 60$. This base case was chosen through pilot testing that varied the factors in an attempt to find a point where the non-distributed versions of KLSH and the linear search took approximately the same amount of time.

Table 4.4. Table of Runs for Non-Distributed KLSH Search.

Test Number	Number of Features (n)	Maximum Feature Depth (D)	Number of Descriptors used per Feature (d)	Number of Tables (T)	Purpose
Base case	15	500	60	20	All factors are varied around the base case
1	10	500	60	20	Factor varied: n Factors held constant: D, d, and T
2	20	500	60	20	
3	25	500	60	20	
4	15	500	40	20	Factor varied: d Factors held constant: n, d, and T
5	15	500	80	20	
6	15	500	100	20	
7	15	500	60	15	Factor varied: T Factors held constant: n, D, and d
8	15	500	60	25	
9	15	500	60	30	
10	15	300	60	20	Factor varied: D Factors held constant: n, d, and T
11	15	700	60	20	

For each of the tests in Table 4.4, 20 different images are queried into the

database five times each. This yields 100 observations per set of factors. This process is randomized and repeated three different times. The metrics measured are the time taken to respond to a query and the accuracy of the results returned.

4.4.3 Distributed Linear Search.

As the strong and weak scalability of the linear search algorithm is examined, the number of computing nodes, C, will be the factor varied. Since each map step processes one image's scene file from the database, the number of database items must be fixed or varied as C is increased. Table 4.5 shows the different test runs which examine the strong and weak scalability of the distributed linear search.

Table 4.5. Table of runs for distributed linear search.

Test Number	Number of Computing Nodes (C)	Number of Images in the Database	Number of map tasks per node	Search type explores
1	10	1000	100	Linear weak scalability
2	10	2000	200	Linear strong scalability
3	20	2000	100	Linear strong and weak scalability
4	30	2000	67 (rounded up from 66.6)	Linear strong scalability
5	30	3000	100	Linear weak scalability

Test numbers 1, 3, and 5 explore the strong scalability of the distributed linear search, and the number of overall map tasks is fixed at 2000 while the number of nodes is varied. Test numbers 2, 3, and 4 explore the weak scalability of the distributed

linear search, and the number of map tasks *per node* is fixed at 100. Additionally, a single map task can not be split between nodes, so any fractional map task is rounded up to the next higher whole number. For the distributed linear search algorithm, D is fixed at 300. In other words, only the first 300 most prominent features from each image are considered.

4.4.4 Distributed KLSH search.

As the strong and weak scalability of the KLSH search algorithm is examined, C is the factor varied. Since each map step processes one KLSH table file, the number of table files must be fixed or varied with the number of nodes in the cluster. Table 4.6 shows the different test runs which examine the strong and weak scalability of the distributed KLSH search.

Table 4.6. Table of runs for distributed KLSH search.

Test Number	C	T	Number of Images in the Database	Number of map tasks per node	Search type explores
1	10	120	2000	12	KLSH weak scalability
2	10	240	2000	24	KLSH strong scalability
3	20	240	2000	12	KLSH strong and weak scalability
4	30	240	2000	8	KLSH strong scalability
5	30	360	2000	12	KLSH strong scalability

Test numbers 2, 3, and 4 explore the strong scalability of the distributed KLSH search The number of overall map tasks is fixed at 12 while the number of nodes is varied. Test numbers 1, 3, and 5 explore the weak scalability of the distributed KLSH

search and the number of map tasks *per node* is fixed at 12. For the distributed KLSH search algorithm, each table is built using $n = 20$, $D = 300$, and $d = 60$. Additionally, i is fixed at 2,000 images for all tests.

4.5 Summary

The non-distributed algorithms are examined first, in order to validate that they behave according to expected behavior as their factors are varied. Then, the strong and weak scalability of the distributed algorithms are examined using the metrics of time, accuracy, and distance.

V. Results and Analysis

The hypothesis of this research is that modifying the KLSH algorithm to run on a cluster of computers using Apache Hadoop can achieve significant retrieval speedup without loss of accuracy by conducting localization through the use CBIR. In order to evaluate this hypothesis, the test runs must first show that the non-distributed search algorithms conform to expected behavior. Once this is shown to be true, the distributed test runs must demonstrate the scalability and accuracy of the distributed search algorithms. What is ultimately shown is that the distributed KLSH algorithm scales well, is 4.8 times faster than the brute force linear search, and is still able to perform localization within an average of 8.5 meters.

This chapter begins by explaining the statistical methods which were used in order to evaluate the results of the experimental runs. Next, the procedure for obtaining random results is explained. Next, the time and accuracy results for each of the non-distributed and distributed tests are explained. Then, the results of the distance metric for the distributed searches are explained. Finally, some interpretive analysis about the scalability of the distributed KLSH algorithm is discussed.

5.1 Statistical Procedure

The statistical tests used to evaluate all the different search techniques followed the same general procedure:

1. Determine the normality of the accuracy data for all test runs combined by graphing the frequency of the accuracy and/or using the Shapiro-Wilk's normality test.
2. Based on the normality of the data, conduct either a t-test (for normal data) or a Mann-Whitney U test (for non-normal data) to determine if the difference

in the mean of the accuracy of the search is statistically significant from the accuracy of randomly guessing.

3. For each factor explored in the tests, determine the normality of the accuracy and time data by graphing the frequency of the data and/or using the Shapiro-Wilk's normality test.
4. Based on the normality of the data, conduct either a one way ANOVA test (for normal data) or a Kruskal-Wallis test (for non-normal data) to determine if the differences in the averages of the accuracy and time metrics as each of the factors were varied could have occurred due to random chance.
5. For the time and accuracy metrics, conduct the Dunn test with Bonferroni correction for multiple comparisons to determine if there are statistically significant differences between the time and accuracy metrics for each of the different levels of the factors which were varied.

5.2 Randomly Guessing

The accuracy of all searches must be shown to be statistically better than randomly guessing. In order to determine the accuracy of a completely random guess, 150 sets of six pseudo-random numbers are generated using Python's random function. The first number generated was the "query image," and the remaining five images were the "responses to the search." These pairs were scored using the same scoring program that was used to score all other results, as was described in Section 4.3.1. Randomly guessing yielded an accuracy of 5.3%, which is depicted on several of the accuracy charts for comparison purposes.

5.3 Non-Distributed Tests

Prior to exploring the feasibility and scalability of the distributed search algorithms, it must be shown that the non-distributed algorithms conform to expected behavior. The time and accuracy of the non-distributed linear search and non-distributed KLSH algorithm are evaluted using the statistical procedure outlined in Section 5.1 while the various factors are altered.

5.3.1 Non-Distributed Linear Search.

Table 5.7 depicts the results of the non-distributed linear search, and Table 5.8 shows a summary of the statistically significant findings for the non-distributed linear search. These Tables can be referenced for clarity as the detailed results are explained.

Table 5.7. Accuracy and Time Results for Non-Distributed Linear Search Runs.

Test number	Maximum Feature Depth (D)	Average Accuracy (%)	Average Time (s)
1	100	99%	3.81 s
2	300	97%	18.94 s
3	500	99%	48.45 s
4	700	98%	86.72 s

Table 5.8. Statistically Significant Factors of the Non-distributed Linear Search.

		Are factors statistically significant?	
		Time	Accuracy
Response Variable	D	Yes	Yes, for some values of D
Statistically better than a random guess		Yes	

The overall mean accuracy of the linear search was 98.25% when all linear search test runs data are combined. As mentioned in Section 5.2, the overall accuracy of

randomly guessing was 5.3%. Figure 5.21 shows that the datasets are not normally distributed, and the Mann-Whitney U test yields a p-value of $2.2 * 10^{-16}$. This means that it is highly unlikely that difference in the means of the accuracy occurred by random chance. Therefore, the higher accuracy of the linear search *is* statistically different from that of randomly guessing.

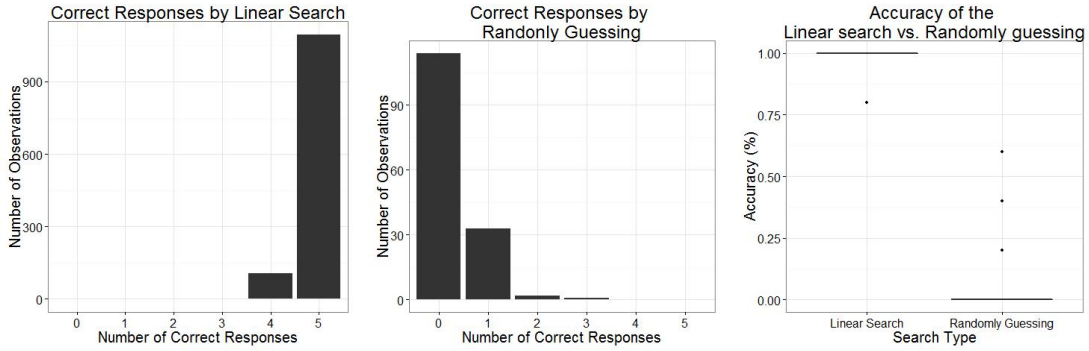


Figure 5.21. Linear search vs. Randomly guessing.

Impact of Maximum Feature Depth on Time/Accuracy.

Figure 5.22 shows the interactions between time and accuracy as the maximum feature depth, D, is varied. The only factor which effected the non-distributed linear search was D. Based on the test runs that varied D, it was discovered that varying D *does* have a statistically significant effect on the time and *does* have a statistically significant effect on the accuracy of image retrieval for some values of D.

For the time metric, the assumption of normality was violated for the four different values of D (100, 300, 500, and 700) as assessed by the Shapiro-Wilk's normality test (p-values $= 2.16 * 10^{-16}$). The Kurskal-Wallis test determined that the differences in the times for the four different values of D *was* statistically significant (p-value $= 2.16 * 10^{-16}$). Table 5.9 shows the results of the Dunn test, which found that all six pair-wise comparisons had statistically significant differences between the mean of the times. These differences in the means of the time as D is increased are

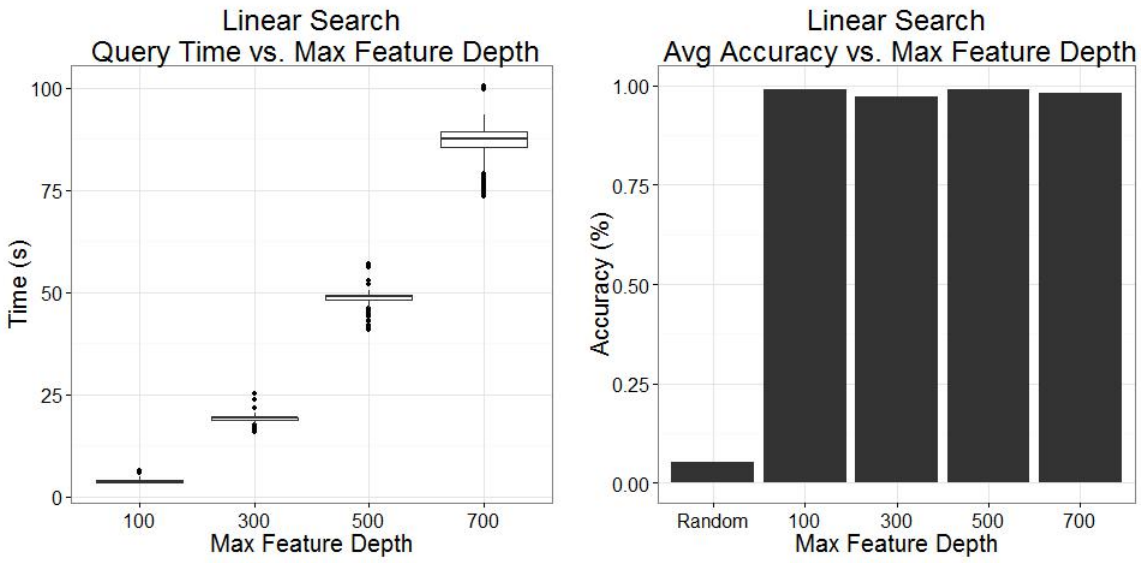


Figure 5.22. Linear search - How varying D effects query time and accuracy.

clearly illustrated in the left graph of Figure 5.22.

Table 5.9. Dunn Test Results Depicting the Statistically Significant Differences of Times for the Different Levels of D.

Are There Statistically Significant Difference Between Levels?			
Maximum Feature Depth (D)	100	300	500
300	Yes (p-value = $2.2 * 10^{-16}$)		
500	Yes (p-value = $2.2 * 10^{-16}$	Yes (p-value = $2.2 * 10^{-16}$	
700	Yes (p-value = $2.2 * 10^{-16}$	Yes (p-value = $2.2 * 10^{-16}$	Yes (p-value = $2.2 * 10^{-16}$

For the accuracy metric, the assumption of normality was violated for the four different values of D (100, 300, 500, and 700) as assessed by the Shapiro-Wilk's test (p-values = $2.2 * 10^{-16}$). The Kurskal-Wallis test determined that the differences in

the means of the accuracy for the two different values of D *was* statistically significant (p-value = $1.045 * 10^{-5}$). Table 5.10 shows the results of the Dunn test, which found that two out of the six pair-wise comparisons had statistically significant differences between the mean of the accuracies. Even though varying D *was* found to be statistically significant for two values of D, in practice, there is little difference between the accuracies for the different values of D. It is clear from Table 5.7 that all accuracies for the linear search were very close to 100% and were all within 2% from one another. It is, therefore, not surprising that some of the values were not statistically different from one another.

Table 5.10. Dunn Test Results Depicting the Statistically Significant Differences of Accuracy for the Different Levels of D.

Are There Statistically Significant Difference Between Levels?			
Maximum Feature Depth (D)	100	300	500
300	No (p-value = 0.091)		
500	No (p-value = 0.091)	Yes (p-value = $2.2 * 10^{-16}$)	
700	No (p-value = 0.091)	Yes (p-value = 1)	Yes (p-value = $2.2 * 10^{-16}$)

After analyzing the results from the non-distributed linear search tests, it is reasonable to conclude that varying D *does* have an effect on the time required to conduct a query and *does* have an effect on the accuracy for some of the D's which were measured. It can also be concluded that the accuracy of the linear search *is* statistically different from that of randomly guessing.

5.3.2 Non-Distributed KLSH Search.

Table 5.11 depicts the results of the non-distributed KLSH search, and Table 5.12 shows a summary of the statistically significant findings for the non-distributed KLSH search. These Tables can be referenced for clarity as the detailed results are explained.

Table 5.11. Accuracy and Time Results for Non-Distributed KLSH Search Runs

Test number	n	D	d	T	Average Accuracy (%)	Average Time (s)
Base case	15	500	60	20	80%	39.98 s
1	10	500	60	20	65%	26.43 s
2	20	500	60	20	78%	50.14 s
3	25	500	60	20	87%	60.41 s
4	15	500	40	20	67%	36.76 s
5	15	500	80	20	77%	40.76 s
6	15	500	100	20	81%	42.27 s
7	15	500	60	15	70%	29.41 s
8	15	500	60	25	78%	48.78 s
9	15	500	60	30	82%	58.71 s
10	15	300	60	20	75%	38.63 s
11	15	700	60	20	75%	40.45 s

The overall mean accuracy of the KLSH search was 76.25% when all KLSH search test runs data are combined. As mentioned in Section 5.2, the overall accuracy of randomly guessing was 5.3%. Figure 5.23 shows that the datasets are not normally distributed, and the Mann-Whitney U test yields a p-value of $2.2 * 10^{-16}$. This means that it is highly unlikely that difference in the means of the accuracy occurred by random chance. Therefore, the higher accuracy of the KLSH search *is* statistically

Table 5.12. Non-distributed KLSH - Statistically significant factors.

		Are factors statistically significant?	
		Time	Accuracy
Response Variable	D	Yes	No
	n	Yes	Yes (5 out of 6)
	d	Yes	Yes (3 out of 6)
	T	Yes	Yes (3 out of 6)
Statistically better than a random guess?		Yes	

different from that of randomly guessing.

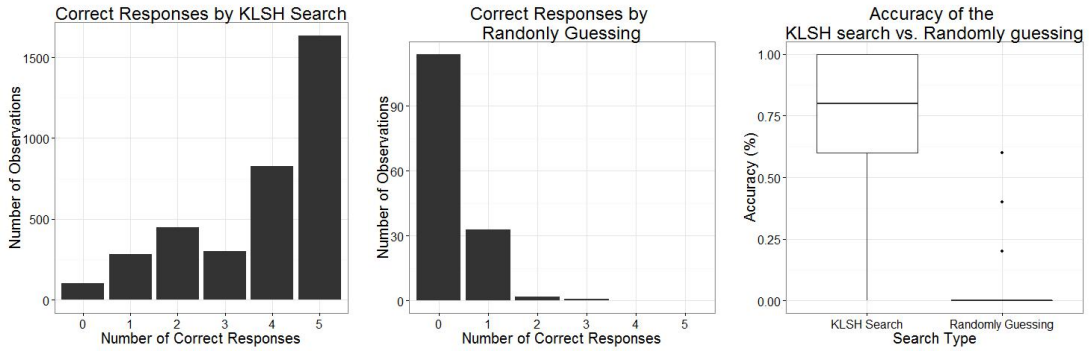


Figure 5.23. KLSH search vs. Randomly guessing.

The factors which effected the non-distributed KLSH search were the maximum feature depth, D, the number of features, n, the number of descriptors chosen per feature, d, and the number of tables used, T. These factors are varied in order to determine the impact on the time and accuracy of the non-distributed KLSH search.

Impact of Maximum Feature Depth on Time/Accuracy.

Figure 5.24 shows the interactions between time and accuracy as the number of features chosen, D, is varied. Based on the test runs that varied D, it was discovered that varying D *does* have a statistically significant effect on the time and *does not* have a statistically significant effect on the accuracy of image retrieval.

For the time metric, the assumption of normality was violated for the three

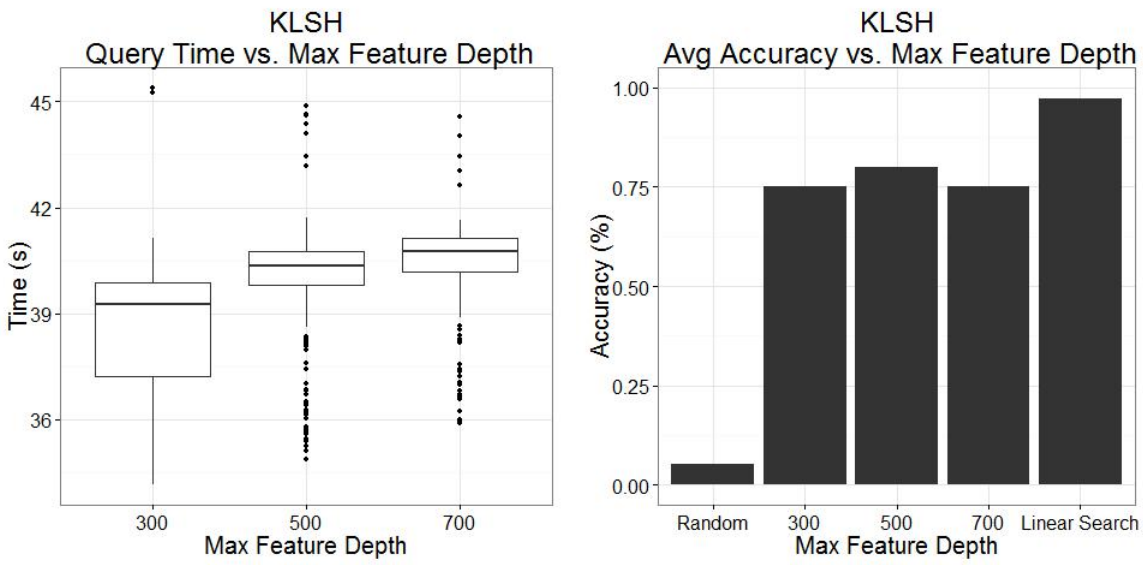


Figure 5.24. KLSH search - How varying D effects query time and accuracy.

different values of D (300, 500, and 700) as assessed by the Shapiro-Wilk's normality test (p-values $= 2.16 * 10^{-16}$). The Kurskal-Wallis test determined that the differences in the times for the three different values of D *was* statistically significant (p-value $= 2.16 * 10^{-16}$). Table 5.13 shows the results of the Dunn test, which found that all three different pair-wise comparisons had statistically significant differences between the mean of the times. This is somewhat unexpected, as varying D should not have a great effect on the time taken by the algorithm. Table 5.14 shows the summary statistics for the time when D is varied. It is thought that because the 95% confidence intervals are so small and not overlapping with one another, the differences in the means of the time was detected as statistically significant. Though Figure 5.24 shows there is a slightly upward trend in the time as D is increased, it is thought that the upward trend would no longer be observed if more levels of D were chosen.

For the accuracy metric, the assumption of normality was violated for the three different values of D (300, 500, and 700) as assessed by the Shapiro-Wilk's test (p-values $= 2.2 * 10^{-16}$). The Kurskal-Wallis test determined that the differences in the means of the accuracy for the three different values of D *was not* statistically signif-

Table 5.13. Dunn Test Results Depicting the Statistically Significant Differences of Time for the Different Levels of D.

Are There Statistically Significant Difference Between Levels?		
Maximum Feature Depth (D)	300	500
500	Yes (p-value = $2.2 * 10^{-16}$)	
700	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)

Table 5.14. Summary Statistics for Time when D is varied.

Depth Levels	Number of Observations	Average Time (s)	Standard Deviation	95% Confidence Interval
300	300	38.63 s	1.73	38.44 to 38.83
500	300	39.98 s	1.56	39.80 to 40.16
700	300	40.45 s	1.25	40.31 to 40.59

icant (p-value = 0.69). Table 5.15 shows the results of the Dunn test, which found that none of the three different pair-wise comparisons had statistically significant differences between the mean of the accuracies. This is unsurprising, since varying D should have little to do with the accuracy of the KLSH search. If anything, increasing D should have a negative overall effect on the accuracy, as it would be possible to choose less prominent features when building the tables used by the KLSH algorithm. It is thought that the downward trend in accuracy as D is increased would become apparent if more levels of D were chosen. The random and linear search accuracy levels have been added to Figure 5.24 for comparison purposes.

Table 5.15. Dunn Test Results Depicting the Statistically Significant Differences of Accuracy for the Different Levels of D.

Are There Statistically Significant Difference Between Levels?		
Maximum Feature Depth (D)	300	500
500	No (p-value = 0.67)	
700	No (p-value = 0.71)	No (p-value = 1)

Impact of the Number of Features Chosen on Time/Accuracy.

Figure 5.25 shows the interactions between time and accuracy as the number of features chosen, n, is varied. Based on the test runs that varied n, it was discovered that varying n *does* have a statistically significant effect on the time and *does* have a statistically significant effect on the accuracy of image retrieval for most values of n.

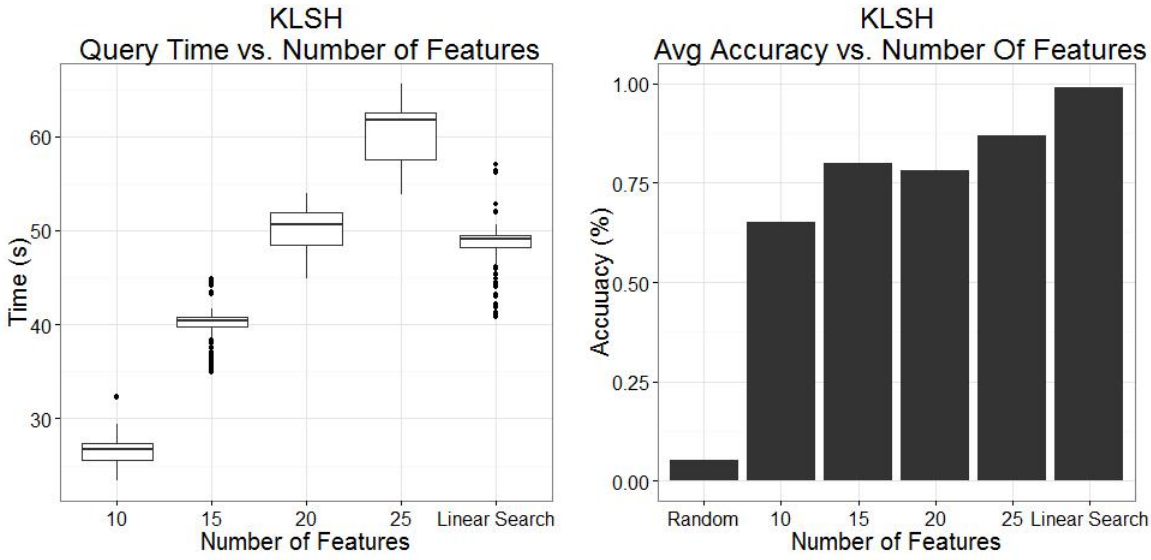


Figure 5.25. KLSH search - How varying n effects query time and accuracy.

For the time metric, the assumption of normality was violated for the four different values of n (10, 15, 20, and 25) as assessed by the Shapiro-Wilk's normality

test (p-values = $1.69 * 10^{-13}$, $2.2 * 10^{-16}$, $1.48 * 10^{-10}$, $2.2 * 10^{-16}$ respectively). The Kurskal-Wallis test determined that the differences in the times for the three different values of n *was* statistically significant (p-value = $2.16 * 10^{-16}$). Table 5.16 shows the results of the Dunn test, which found that all six different pair-wise comparisons had statistically significant differences between the mean of the times. These differences in the means of the time for different levels of n are clearly illustrated in the left graph of Figure 5.25.

Table 5.16. Dunn Test Results Depicting the Statistically Significant Differences of Time for the Different Levels of n.

Are There Statistically Significant Difference Between Levels?			
Number of Features (n)	10	15	20
15	Yes (p-value = $2.2 * 10^{-16}$)		
20	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)	
25	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)

For the accuracy metric, the assumption of normality was violated for the four different values of n (10, 15, 20, and 25) as assessed by the Shapiro-Wilk's test (p-values = $1.69 * 10^{-13}$, $2.2 * 10^{-16}$, $1.48 * 10^{-10}$, $2.2 * 10^{-16}$ respectively). The Kurskal-Wallis test determined that the differences in the means of the accuracy for the four different values of n *was* statistically significant (p-value = $2.2 * 10^{-16}$). Table 5.17 shows the results of the Dunn test, which found that five out of six different pair-wise comparisons had statistically significant differences between the mean of the accuracies. This is unsurprising, since increasing n should have positive effect on the

accuracy of the results which are returned. This upward trend in accuracy as n is increased is apparent when viewing the right plot of Figure 5.25. The random and linear search accuracy has been added to this chart for comparison purposes.

Table 5.17. Dunn Test Results Depicting the Statistically Significant Differences of Accuracy for the Different Levels of n.

Are There Statistically Significant Difference Between Levels?			
Number of Features (n)	10	15	20
15	Yes (p-value = $2.2 * 10^{-16}$)		
20	Yes (p-value = $2.2 * 10^{-16}$)	No (p-value = 1)	
25	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = 0.0005)	Yes (p-value = 0.0001)

Impact of Number of Descriptors Chosen per Feature on Time/Accuracy.

Figure 5.26 shows the interactions between time and accuracy as the number of descriptors used per feature, d, is varied. Based on the test runs that varied d, it was discovered that varying d *does* have a statistically significant effect on the time and *does* have a statistically significant effect on the accuracy of image retrieval for some values of d.

For the time metric, the assumption of normality was violated for the four different values of d (40, 60, 80, and 100) as assessed by the Shapiro-Wilk's normality test (p-values = $2.2 * 10^{-16}$). The Kurskal-Wallis test determined that the differences in the times for the four different values of d *was* statistically significant (p-value =

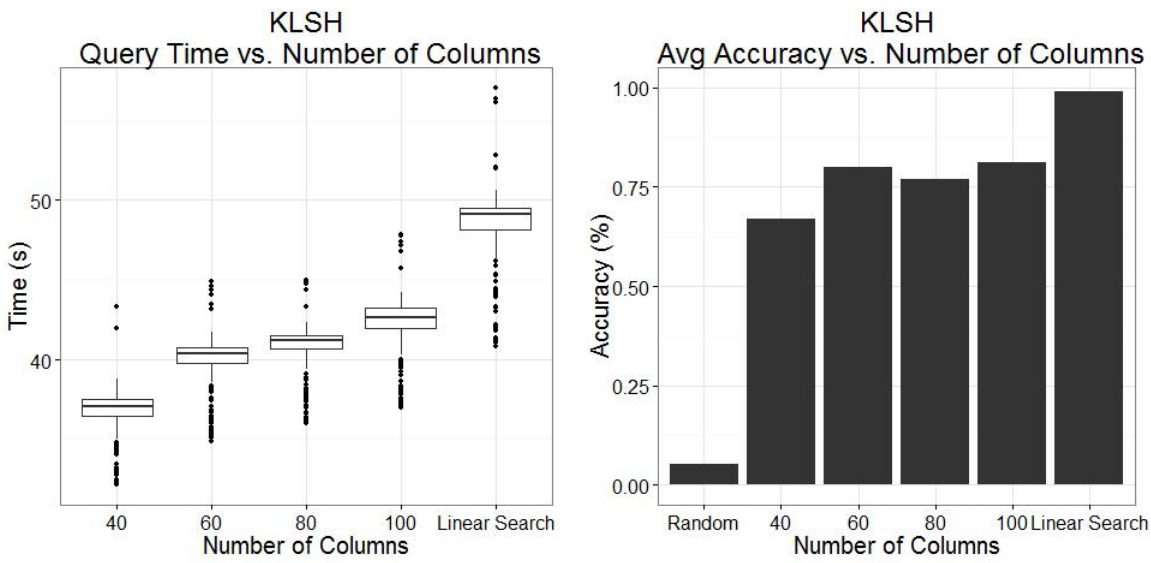


Figure 5.26. KLSH search - How varying d effects query time and accuracy.

$2.16 * 10^{-16}$). Table 5.18 shows the results of the Dunn test, which found that all six different pair-wise comparisons had statistically significant differences between the mean of the times. These differences in the means of the time for different levels of d are clearly illustrated in the left graph of Figure 5.26. The linear search time was added to this graph for comparison purposes.

For the accuracy metric, the assumption of normality was violated for the four different values of d (40, 60, 80, and 100) as assessed by the Shapiro-Wilk's test (p-values $= 2.2 * 10^{-16}$). The Kurskal-Wallis test determined that the differences in the means of the accuracy for the four different values of d *was* statistically significant (p-value $= 1.702 * 10^{-7}$). Table 5.19 shows the results of the Dunn test, which found that three out of six different pair-wise comparisons had statistically significant differences between the mean of the accuracies.

This is somewhat unexpected, because if Figure 5.26 is viewed, the accuracy level appears to generally increases as d increases. Figure 5.27 shows the number of correct responses as d is varied. When d=60, 80, and 100, the accuracy data is skewed to the right, whereas when d=40, the data is skewed towards its edges. When d=40,

Table 5.18. Dunn Test Results Depicting the Statistically Significant Differences of Time for the Different Levels of d.

Are There Statistically Significant Difference Between Levels?			
Number of Descriptors	40	60	80
60	Yes (p-value = $2.2 * 10^{-16}$)		
80	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)	
100	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)

Table 5.19. Dunn Test Results Depicting the Statistically Significant Differences of Accuracy for the Different Levels of d.

Are There Statistically Significant Difference Between Levels?			
Number of Descriptors	40	60	80
60	No (p-value = 1)		
80	No (p-value = 0.86)	No (p-value = 0.53)	
100	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = 0.0001)	Yes (p-value = $2.2 * 10^{-16}$)

this skewness is caused by the high number of observations with only one correct response. This causes the mean to not accurately represent the data when d=40, which may help explain the accuracy data's unusual statistical significance when d is varied. However, the higher number of occurrences with only one correct when d=40 does make sense, because as d decreases, less of the database is being represented

in each KLSH table. The linear search and random columns have been added to Figure 5.26 for comparison purposes only. It is apparent that the KLSH search is less accurate than the linear search because accuracy of the linear search's accuracy is almost 100%. It is also apparent that for the values of d which were chosen, the KLSH search is also faster than the linear search.

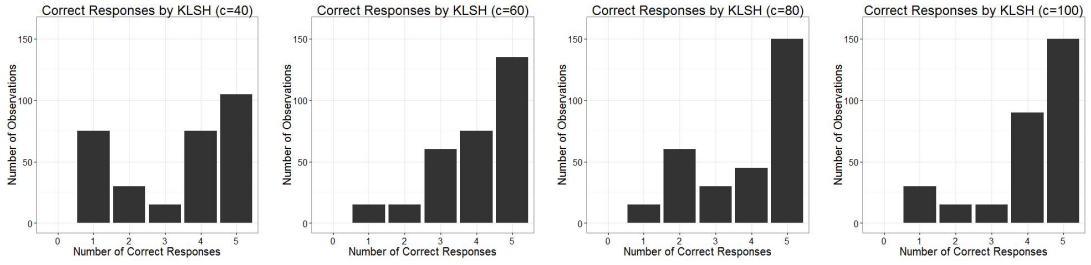


Figure 5.27. Number of correct responses as d is varied.

Impact of Number of KLSH Tables on Time/Accuracy.

Figure 5.28 shows the interactions between time and accuracy as the number of KLSH tables used, T, is varied. Based on the test runs that varied T, it was discovered that varying T *does* have a statistically significant effect on the time and *does* have a statistically significant effect on the accuracy of image retrieval for some values of T.

For the time metric, the assumption of normality was violated for the four different values of T (15, 20, 25, and 30) as assessed by the Shapiro-Wilk's normality test (p-values = $2.88 * 10^{-11}$, $2.2 * 10^{-16}$, $9.6 * 10^{-11}$, $8.14 * 10^{-15}$ respectively). The Kurskal-Wallis test determined that the differences in the times for the four different values of T *was* statistically significant (p-value = $2.16 * 10^{-16}$). Table 5.20 shows the results of the Dunn test, which found that all six different pair-wise comparisons had statistically significant differences between the mean of the times. These differences in the means of the time for different levels of T are clearly illustrated in the left

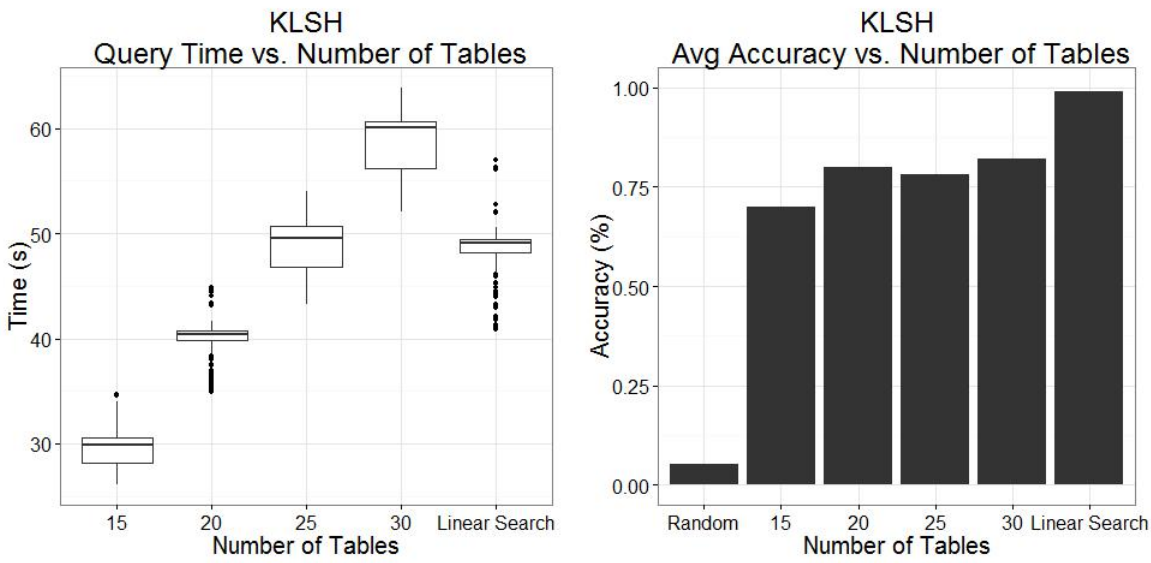


Figure 5.28. KLSH search - How varying T effects query time and accuracy.

graph of Figure 5.28. The linear search time was added to this graph for comparison purposes.

Table 5.20. Dunn Test Results Depicting the Statistically Significant Differences of Time for the Different Levels of T.

Are There Statistically Significant Difference Between Levels?			
Number of KLSH Tables (T)	15	20	25
20	Yes (p-value = $2.2 * 10^{-16}$)		
25	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)	
30	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)

For the accuracy metric, the assumption of normality was violated for the four different values of T (40, 60, 80, and 100) as assessed by the Shapiro-Wilk's test (p-

values = $1.22 * 10^{-15}$, $2.2 * 10^{-16}$, $2.2 * 10^{-16}$, $2.2 * 10^{-16}$ respectively). The Kurskal-Wallis test determined that the differences in the means of the accuracy for the four different values of T *was* statistically significant (p-value = $3.74 * 10^{-9}$)). Table 5.21 shows the results of the Dunn test, which found that three out of six different pair-wise comparisons had statistically significant differences between the means of the accuracies.

Table 5.21. Dunn Test Results Depicting the Statistically Significant Differences of Accuracy for the Different Levels of T.

Are There Statistically Significant Difference Between Levels?			
Number of KLSH Tables (T)	15	20	25
20	Yes (p-value = 0.0001)		
25	Yes (p-value = $2.2 * 10^{-16}$)	No (p-value = 1)	
30	Yes (p-value = $2.2 * 10^{-16}$)	No (p-value = 0.11)	No (p-value = 0.14)

As this implementation of the non-distributed KLSH algorithm was a translation of the work of Kullis and Grauman [5] from MATLAB into Java, these tests were conducted in order to validate that the working algorithm was not compromised during translation from one language to another. Generally speaking, the KLSH algorithm performed as expected during the non-distributed portion of the testing and enough confidence in the performance of the algorithm was reached to proceed with the modifications required for the KLSH algorithm to work as a distributed algorithm.

5.4 Distributed Tests

For the distributed tests, the strong and weak scalability is being examined. Strong scalability is when the problem size is fixed and the number of nodes is varied and the factors of time and accuracy are measured. Weak scalability is when the problem size *per node* is fixed and the factors of time and accuracy are measured.

5.4.1 Distributed Linear Search.

Evaluating the strong and weak scalability of the distributed linear search varies the number of computing nodes in the cluster, C, and the maximum feature depth, D. Table 5.22 depicts the results of the distributed linear search, and Table 5.23 shows a summary of the statistically significant findings for the distributed linear search. These Tables can be references for clarity as the detailed results are explained.

Table 5.22. Accuracy and Time Results for Distributed Linear Search Runs.

Test Number	Number of Nodes (C)	Number of Images in the Database	Number of map tasks per node	Average Accuracy (%)	Average Time (s)	Average Distance (m)
1	10	1000	100	100%	1341.79 s	.46 m
2	10	2000	200	99%	2410.67 s	.79 m
3	20	2000	100	98%	1403.72 s	1.03 m
4	30	2000	67 (rounded up from 66.6)	99.8%	971.63 s	.97 m
5	30	3000	100	100%	1366.43 s	1.17 m

The overall mean accuracy for the distributed linear search was 99.3% when all the distributed linear search rest runs are combined. As mentioned in Section 5.2, the overall accuracy of randomly guessing was 5.3%. Figure 5.29 shows that the

Table 5.23. Distributed Linear Search- Statistically significant factors

		Are factors statistically significant?	
		Time	Accuracy
Response Variable	Strong Scalability C	Yes	No (1 out of 3)
	Weak Scalability C	Yes	Yes (2 out of 3)
Statistically better than random guess?		Yes	

datasets are not normally distributed, and the Mann-Whitney U test yields a p-value of $2.2 * 10^{-16}$). This means that it is highly unlikely that the difference in the means of the accuracy occurred by random chance. Therefore, the higher accuracy of the distributed linear search *is* statistically different from that of randomly guessing.

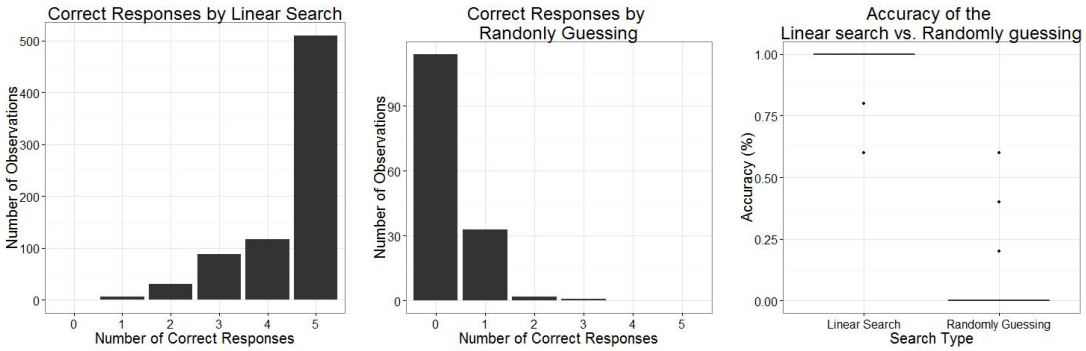


Figure 5.29. Distributed Linear Search vs. Randomly Guessing.

Strong Scalability.

Figure 5.30 shows the interactions between time and accuracy and the number of computing nodes, C, as C is increased and the problem size is fixed at 2000 images. Based on the tests that varied C, it was discovered that varying C *does* have a statistically significant effect on the time, and *does not* have a statistically significant effect on the accuracy for most levels of C chosen.

For the time metric, two of the three different values for C (10, 20, and 30) violated the assumption of normality as assessed by the Shapiro-Wilk's normality

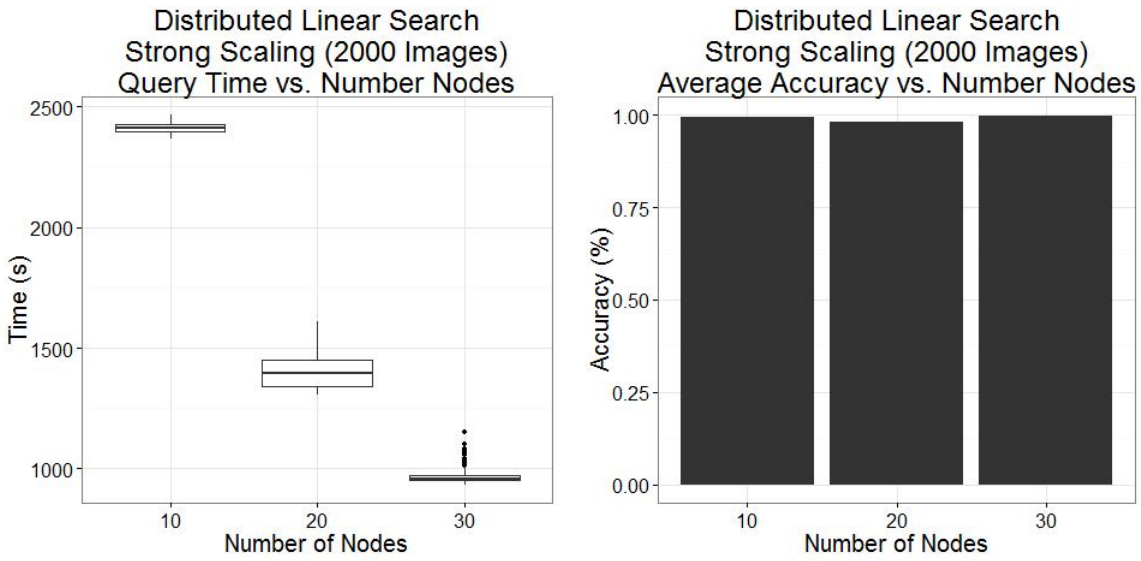


Figure 5.30. Distributed Linear Search - Strong Scalability.

test (p-values = 0.7, $7.3 * 10^{-5}$, and $4.4 * 10^{-15}$, respectively). The Kruskal-Wallis test determined that the differences in the time for the three different level of C *is* statistically significant (p-value = $2.2 * 10^{-16}$). Table 5.24 shows the results of the Dunn test, which found that all three pair-wise comparisons had statistically significant differences in the averages of the times. These differences in the means of the time are clearly illustrated in the left graph of Figure 5.30. This also makes sense logically because more nodes should be able to solve the same sized problem faster.

Table 5.24. Dunn Test Results Depicting the Statistically Significant Differences of Time for the Different Levels of C.

Are There Statistically Significant Difference Between Levels?		
Number of Nodes (C)	10	20
20	Yes (p-value = $2.2 * 10^{-16}$)	
30	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)

For the accuracy metric, the assumption of normality was violated for the three different values of C (10, 20, and 30) as assessed by the Shapiro-Wilk's normality test (p-values = 0.02, $7.37 * 10^{-5}$, and $3.05 * 10^{-15}$, respectively). The Kruskal-Wallis test determined that the differences in the accuracies for the three different level of C *is not* statistically significant (p-value = 0.07). Table 5.25 shows the results of the Dunn test, which found that only one of the three pair-wise comparisons had statistically significant difference between the means of the accuracies. This is unsurprising, because the overall accuracy of the distributed linear search was very high, and the accuracy differences between each of the different levels of C was less than 2%.

Table 5.25. Dunn Test Results Depicting the Statistically Significant Differences of Accuracy for the Different Levels of C.

Are There Statistically Significant Difference Between Levels?		
Number of Nodes (C)	10	20
20	No (p-value = 0.19)	
30	No (p-value = 0.68)	Yes (p-value = 0.035)

Weak Scalability.

Figure 5.31 shows the interactions between time and accuracy and the number of computing nodes, C, as C is increased and the problem size is fixed at 100 images per node. Based on the tests that varied C, it was discovered that varying C *does* have a statistically significant effect on the time, and *does not* have a statistically significant effect on the accuracy for most levels of C chosen.

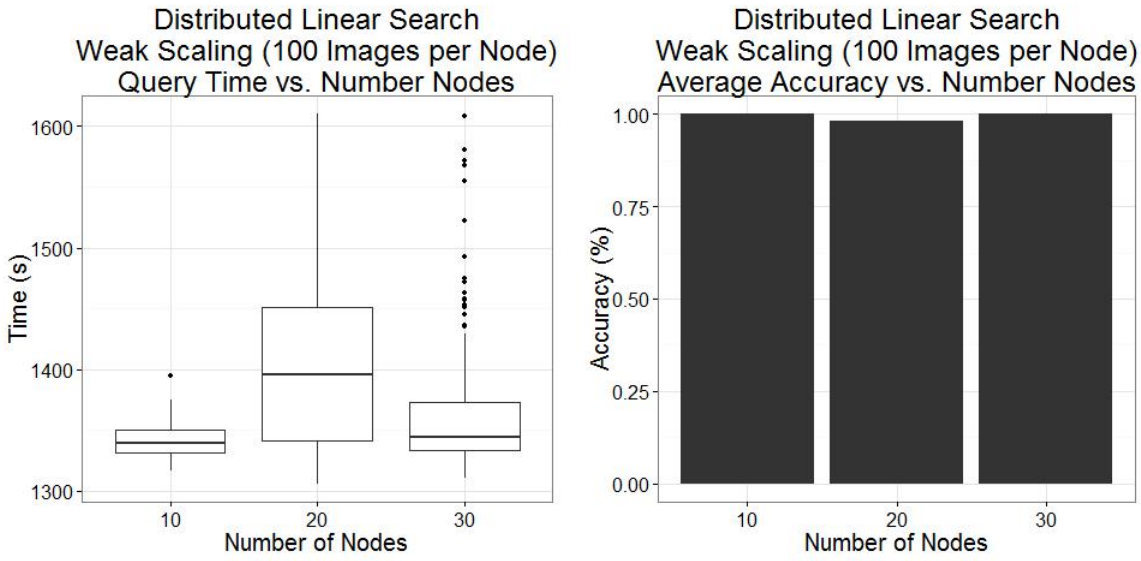


Figure 5.31. Distributed Linear Search - Weak Scalability.

For the time metric, two of the three different values for C (10, 20, and 30) violated the assumption of normality as assessed by the Shapiro-Wilk's normality test (p-values = 0.7, $7.3 * 10^{-5}$, and $4.4 * 10^{-15}$, respectively). The Kruskal-Wallis test determined that the differences in the time for the three different level of C *is* statistically significant (p-value = $2.2 * 10^{-16}$). Table 5.26 shows the results of the Dunn test, which found that all three pair-wise comparisons had statistically significant differences in the averages of the times. This was somewhat unexpected, as the size of the problem *per node* remained the same. It is thought that if more levels of C were chosen, the differences in the means of the time between the levels would

become less pronounced. The left graph of Figure 5.31 even shows how the box plots for the three levels of C which were tested all overlap with one another. It remains an open question why the differences in the means of the times was determined to be statistically significant.

Table 5.26. Dunn Test Results Depicting the Statistically Significant Differences of Time for the Different Levels of C.

Are There Statistically Significant Difference Between Levels?		
Number of Nodes (C)	10	20
20	Yes (p-value = $2.2 * 10^{-16}$)	
30	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)

For the accuracy metric, the assumption of normality was violated for the three different values of C (10, 20, and 30) as assessed by the Shapiro-Wilk's normality test (p-values = $2.2 * 10^{-16}$). The Kruskal-Wallis test determined that the differences in the accuracies for the three different level of C *is* statistically significant (p-value = 0.03). Table 5.27 shows the results of the Dunn test, which found that only one of the three pair-wise comparisons had statistically significant difference between the means of the accuracies. This is unsurprising, because the overall accuracy of the distributed linear search was very high, and the accuracy differences between each of the different levels of C was less than 2%.

Table 5.27. Dunn Test Results Depicting the Statistically Significant Differences of Accuracy for the Different Levels of C.

Are There Statistically Significant Difference Between Levels?		
Number of Nodes (C)	10	20
20	No (p-value = 0.39)	
30	No (p-value = 0.20)	Yes (p-value = 0.013)

5.4.2 Distributed KLSH Search.

Evaluating the strong and weak scalability of the distributed KLSH search varies the number of computing nodes in the cluster, C, and the number of KLSH tables, T. Table 5.28 depicts the results of the distributed KLSH search, and Table 5.29 shows a summary of the statistically significant findings for the distributed KLSH search. These Tables can be references for clarity as the detailed results are explained.

Table 5.28. Accuracy and Time Results for Distributed KLSH Search Runs.

Test Number	C	T	Number of map tasks per node	Average Accuracy (%)	Average Time (s)	Average Distance (m)
1	10	120	12	61%	270.64 s	9.98 m
2	10	240	24	68.9%	474 s	8.29 m
3	20	240	12	68.7%	293.55 s	7.21 m
4	30	240	8	68.4%	223.03 s	10.08 m
5	30	360	12	76%	302.11 s	6.97 m

The overall mean accuracy for the distributed KLSH search was 68.6% when all the distributed KLSH search rest runs are combined. As mentioned in Section 5.2, the overall accuracy of randomly guessing was 5.3%. Figure 5.32 shows that the

Table 5.29. Distributed KLSH Search- Statistically Significant Factors.

		Are factors statistically significant?	
		Time	Accuracy
Response Variable	Strong Scalability C	Yes	No
	Weak Scalability C	Yes	Yes
Statistically better than a random guess?		Yes	

datasets are not normally distributed, and the Mann-Whitney U test yields a p-value of $2.2 * 10^{-16}$). This means that it is highly unlikely that the difference in the means of the accuracy occurred by random chance. Therefore, the higher accuracy of the distributed KLSH search *is* statistically different from that of randomly guessing.

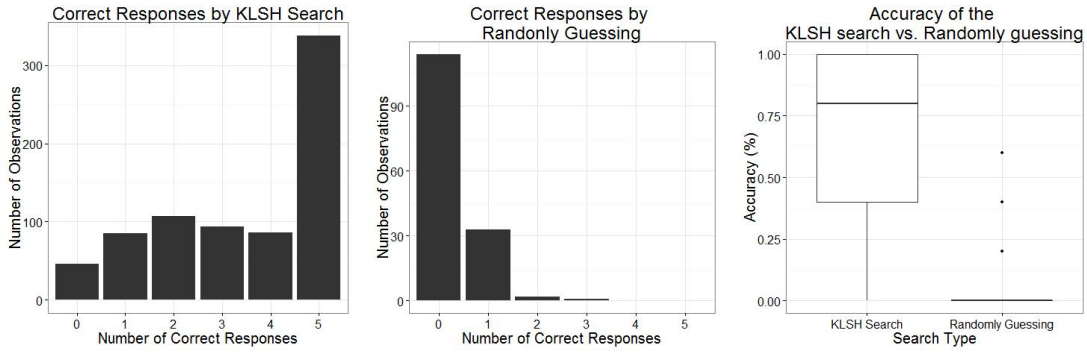


Figure 5.32. Distributed KLSH search vs. Randomly guessing.

Strong Scalability.

Figure 5.33 shows the interactions between time and accuracy and the number of computing nodes, C, as C is increased and the problem size is fixed at 240 KLSH tables. Based on the tests that varied C, it was discovered that varying C *does* have a statistically significant effect on the time, and *does not* have a statistically significant effect on the accuracy for most levels of C chosen.

For the time metric, two of the three different values for C (10, 20, and 30) violated the assumption of normality as assessed by the Shapiro-Wilk's normality

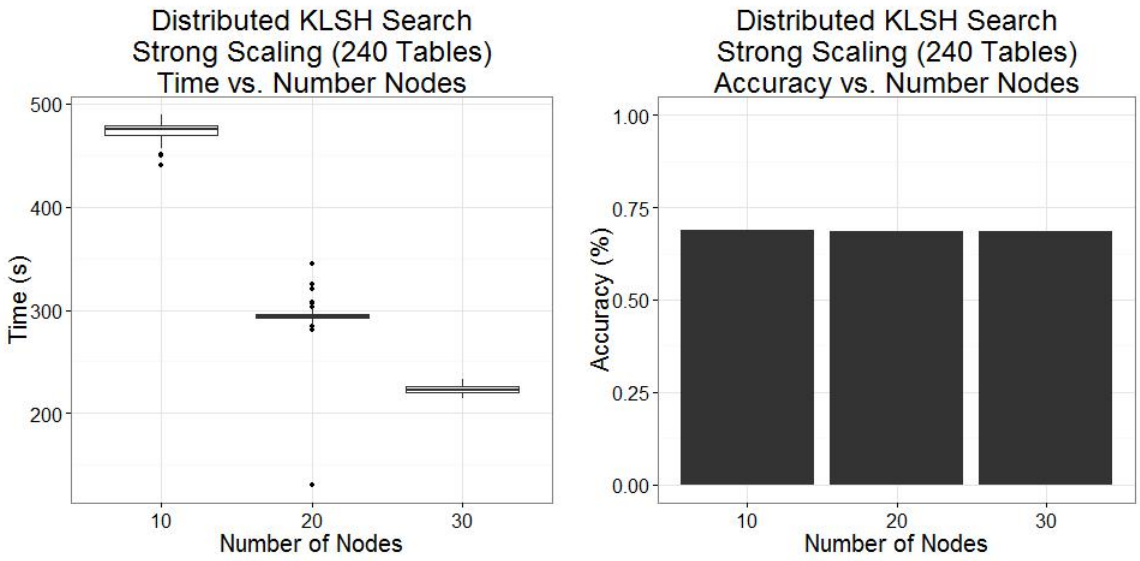


Figure 5.33. Distributed KLSH Search - Strong Scalability.

test (p-values = $3.6 * 10^{-6}$, $2.2 * 10^{-16}$, and .03, respectively). The Kruskal-Wallis test determined that the differences in the time for the three different level of C *is* statistically significant (p-value = $2.2 * 10^{-16}$). Table 5.30 shows the results of the Dunn test, which found that all three pair-wise comparisons had statistically significant differences in the averages of the times. These differences in the means of the time are clearly illustrated in the left graph of Figure 5.33. This also makes sense logically because more nodes should be able to solve the same sized problem faster.

Table 5.30. Dunn Test Results Depicting the Statistically Significant Differences of Time for the Different Levels of C.

Are There Statistically Significant Difference Between Levels?		
Number of Nodes (C)	10	20
20	Yes (p-value = $2.2 * 10^{-16}$)	
30	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)

For the accuracy metric, the assumption of normality was violated for the three different values of C (10, 20, and 30) as assessed by the Shapiro-Wilk's normality test (p-values = $2.2 * 10^{-16}$, $2.48 * 10^{-12}$, and $2.2 * 10^{-16}$ respectively). The Kruskal-Wallis test determined that the differences in the accuracies for the three different level of C *is not* statistically significant (p-value = 0.074). Table 5.31 shows the results of the Dunn test, which found that none of the three pair-wise comparisons had statistically significant difference between the means of the accuracies. This is unsurprising, because the accuracy differences between each of the different levels of C was less than 1%. This also makes sense logically, since the same number of tables were being used to conduct each query, the resolution for the accuracy of the search query should remain about the same.

Table 5.31. Dunn Test Results Depicting the Statistically Significant Differences of Accuracy for the Different Levels of C.

Are There Statistically Significant Difference Between Levels?		
Number of Nodes (C)	10	20
20	No (p-value = 1)	
30	No (p-value = 1)	No (p-value = 1)

Weak Scalability.

Figure 5.34 shows the interactions between the time and accuracy and the number of computing nodes, C, as C is increased and the problem size is fixed at 100 images per node. Based on the tests that varied C, it was discovered that varying C *does* have a statistically significant effect on the time, and *does not* have a statistically significant effect on the accuracy for most levels of C chosen.

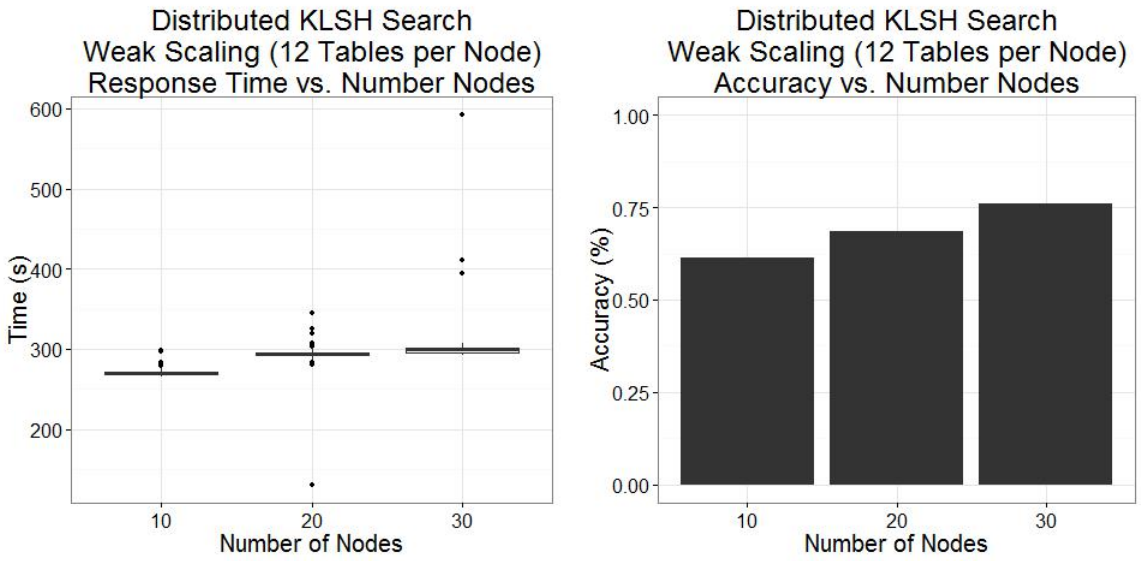


Figure 5.34. Distributed KLSH Search - Weak Scalability.

For the time metric, two of the three different values for C (10, 20, and 30) violated the assumption of normality as assessed by the Shapiro-Wilk's normality test (p-values $= 2.2 * 10^{-16}$). The Kruskal-Wallis test determined that the differences in the time for the three different level of C *is* statistically significant (p-value $= 2.2 * 10^{-16}$). Table 5.32 shows the results of the Dunn test, which found that all three pair-wise comparisons had statistically significant differences in the averages of the times. This was somewhat unexpected, as the size of the problem *per node* remained the same. It is thought that if more levels of C were chosen, the differences in the means of the time between the levels would become less pronounced. Figure 5.34 even shows how the box plots for the three levels of C which were tested are very close to one another. It remains an open question why the differences in the means of the times was determined to be statistically significant.

For the accuracy metric, the assumption of normality was violated for the three different values of C (10, 20, and 30) as assessed by the Shapiro-Wilk's normality test (p-values $= 1.27 * 10^{-12}$, $2.49 * 10^{-12}$, and $2.2 * 10^{-14}$ respectively). The Kruskal-Wallis test determined that the differences in the accuracies for the three different

Table 5.32. Dunn Test Results Depicting the Statistically Significant Differences of Time for the Different Levels of C.

Are There Statistically Significant Difference Between Levels?		
Number of Nodes (C)	10	20
20	Yes (p-value = $2.2 * 10^{-16}$)	
30	Yes (p-value = $2.2 * 10^{-16}$)	Yes (p-value = $2.2 * 10^{-16}$)

level of C *is* statistically significant (p-value = 0.002). Table 5.33 shows the results of the Dunn test, which found that all three pair-wise comparisons had statistically significant difference between the means of the accuracies. The increase in accuracy as the number of nodes is increases is easily observed if the right graph of Figure 5.34 is viewed.

Table 5.33. Dunn Test Results Depicting the Statistically Significant Differences of Accuracy for the Different Levels of C.

Are There Statistically Significant Difference Between Levels?		
Number of Nodes (C)	10	20
20	Yes (p-value = 0.0086)	
30	Yes (p-value = 0.0008)	Yes (p-value = 0.018)

5.4.3 Distance Calculations for Distributed Tests.

The distance metric is depicted as a column in Tables 5.22 and 5.28 and is an average of the distances between all correct matches and the query image. Figure 5.35

shows the interactions between the distance of the images retrieved as the number of nodes is increased. As the accuracy of the distributed linear search was so high, the corresponding distance was very low (0.89 m). This means that the average distance between an image returned by the distributed linear search was less than one meter, which is likely how far the car was able to travel in the quarter of a second which elapsed between images in the database. Not unexpectedly, the distributed KLSH search algorithm's distance improved as the number of tables increased. It is thought that both of these will behave according the logarithmic curves (red lines of Figure 5.35). The average distance of the distributed KLSH search was 8.5 meters, but it the distance was decreasing as the number of tables increased.

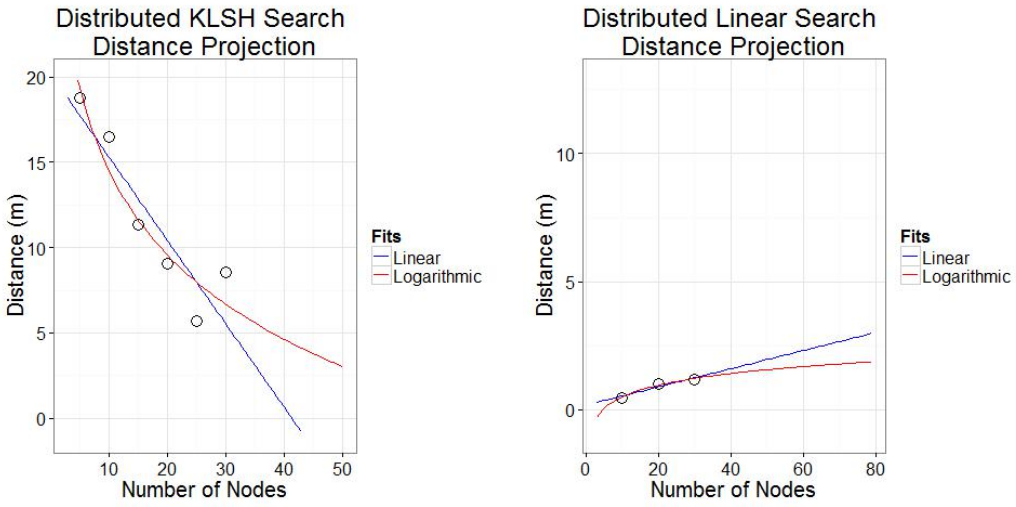


Figure 5.35. Projected Minimum Distance of the Distributed KLSH Search and Maximum of the Distributed Linear Search.

5.5 Additional Interpretive Analysis

The distributed KLSH search required much less time than the distributed linear search. When all data for the two searches is examined at once, the distributed KLSH search took an average of 312.3 seconds (5.2 minutes), while the distributed linear search took an average of 1499.2 second (approx. 25 minutes). Using these numbers,

we can conclude that the distributed KLSH algorithm is 4.8 times faster than a brute force linear search. There was, however, a corresponding trade off in the accuracy: the distributed KLSH search's overall accuracy was 68.6% in contrast to the distributed linear search's 99.3%. This means that the KLSH search could consistently select 3 or 4 matches per query. Depending on the application, this level of accuracy would likely be considered "good enough," especially when considering the time saved.

Figure 5.36 shows a projection of the time and accuracy as the number of nodes is increased beyond what was tested. It is thought that for both factors (time and accuracy), the projection will follow a logarithmic curve (red lines) vs. a linear curve (blue lines). Although only three data points were collected at the lower end of these two projections, the logarithmic curve makes the most sense for the following reasons:

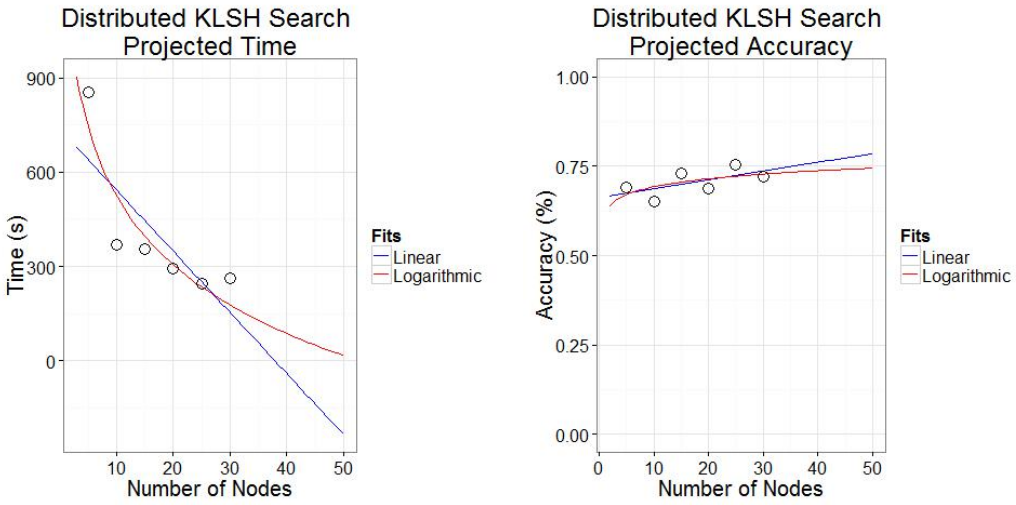


Figure 5.36. Projected Maximum Accuracy and Minimum Time of the Distributed KLSH Search.

1. With a fixed number of tables, increasing the number of nodes will decrease query time only up to the point where all tables are being simultaneously processed. Any additional increase in the number of nodes will result in idle nodes and no decrease in the time to answer the query. This means that there will be a floor to the time required of approximately how long one map takes to be

processed by a node plus the reduce step and the overall overhead time taken by the Hadoop DCE. In practice with the parameters chosen for the tables used for these experiments, the average time for one map to process one table was around 45 seconds. As demonstrated by the non-distributed KLSH tests, the time will vary wildly based on the input parameters used to build the tables.

2. The KLSH algorithm only looks at a small percentage of the features and descriptors. For this distributed implementation of KLSH, when the tables were being built, each table only represented some fraction of the images in the database. Based on the small percentage of the overall database that was being compared to the query image, it is expected that 100% accuracy will not be attainable. Throughout all tests, the accuracy ceiling appeared to be somewhere around 80 to 85%, which also fits the logarithmic curve.

Returning to the hypothesis of this research, a distributed implementation of the KLSH search has been shown to scale well and perform quick and accurate localization through the use of content based image retrieval from a database of images from an urban environment.

VI. Conclusions and Recommendations

The problem of overly relying on the GPS system for navigation is that it may not alway be available. The GPS signals are subject to spoofing, and the satellites themselves could be subject to an attack, which could render GPS navigation unavailable. Because of this, it is important to have alternatives to GPS based navigation. The alternative navigation technique explored by this research was vision based navigation by conducting localization through content based image retrieval (CBIR).

It was hypothesized that modifying the KLSH algorithm to run on a cluster of computers using Apache Hadoop could achieve significant retrieval speedup without loss of accuracy by conducting localization through the use CBIR. This was confirmed by the experiments conducted for this research. The distributed KLSH algorithm performed 4.8 times faster than the brute force linear search, while still maintaining localization accuracy of 8.5 meters. This technology scaled well, and it was demonstrated that this new distributed algorithm is a viable technique for conducting localization and could be used to confirm or supplement GPS based navigation.

6.1 Research Contributions

The main contribution this research made was the distributed implementation and testing of the KLSH search algorithm. This is the first time this particular algorithm has been implemented to use Apache Hadoop and the MapReduce framework. This research demonstrated that the distributed KLSH algorithm is a viable search algorithm which scales well as the number of computing nodes in the cluster is increased.

6.2 Recommendations for Future Work

There are several different extensions to this research which would be of interest or worth exploring:

- Further testing should be done increasing the size of the image database and also the number of computing nodes. In this study, the number of nodes was somewhat limited by the hardware available for this research. Much of the prominent work that is being done with Hadoop uses hundreds or even thousands of machines.

- Another possible extension of this research would be to have a non-static database of images which would incrementally grow over time. Perhaps a robot is exploring a city and adding to its database of images. KLSH tables for these new images could be created on the fly, added to the database, and "learned" as the robot encounters new locations. This would be an interesting extension to this research which was not explored.

- The distributed KLSH algorithm exhibited when choosing different input parameters for building the tables which were used by the distributed tests. This drop in accuracy was never explored by this research. Perhaps, further speed up could be achieved by using fewer tables built with different input parameters, or perhaps representing more or less of the overall database.

- The results from the KLSH search could likely be improved by conducting Random Sample Consensus (RANSAC) on matches returned by the distributed KLSH algorithm to further refine and strengthen the image matching abilities of the algorithm.

Appendix A. Example Scene File

Below is an example of a scene file for the 9_4.png image. These files were used for both the linear search, and to build the tables for the KLSH search. Comments begin with a # symbol. The name of the file is included in the text of the first comment line. The first portion, which is enclosed in # symbols, is the legend for the scene file. Next, the image's height and width information is recorded. For this image, only the first 1000 features were recorded. The remainder fo the scene file following the "## Begin" are paired information about each feature. Information about each feature is stored on first line, and enclosed in square braces are the feature's 128 descriptors, which are values between 0 and 255. The features have been sorted in decreasing magnitude order so the most prominent features come first. Only the first and last few feature are displayed here for the sake of brevity. Finally, the file ends with a #EOF statement, signifying the end of the file.

```
#9_4.scene#####################################################
# Format:
#   Dimension of image (Height, Width)
# Begin - Begin feature data after this line
#   x, y, magnitude, response, orientation
#   128-element descriptor
#   Repeat previous two lines for each feature
#-----------------------------------------------
# 1000 features recorded
################################################
1024 1360
## Begin
925.327270508 392.783569336 168.554733276 0.0359726548195 270.530883789
[  13.   31.   16.   15.   28.    3.    0.    1.    8.   11.    8.   47.
  150.   37.    6.    9.  111.   98.   20.    6.   11.    7.    9.   76.
   19.   33.   44.    0.    0.    2.    7.   14.   15.    7.    2.    5.
   35.    4.    3.   19.   35.   10.    4.   45.  150.   22.    5.   11.
  150.   18.    3.    8.   11.    8.   42.  150.   12.    2.    3.    0.
    0.   10.   80.   34.    6.   16.   30.   37.   39.    2.    2.   12.
   78.   19.    7.  128.  150.    2.    1.    1.  150.   57.    8.   11.
    8.    0.   11.   42.   44.   17.   22.    6.    1.    1.   30.   18.
    0.    9.   20.   17.   21.    0.    0.    0.   41.    8.    2.   44.
   84.    0.    0.    0.  150.   17.    4.    7.    4.    0.    0.    4.
   15.    4.   13.   14.    4.    0.    0.    1.]
989.112426758 629.651977539 167.034713745 0.0227982979268 105.422119141
[   7.    1.    0.    0.   18.   10.    0.    0.   61.    8.    0.    0.
    1.    2.    4.    6.    7.    0.    0.    0.    0.    4.   40.    8.
    0.    0.    0.    0.    0.    0.    9.    2.   33.    2.    0.    0.
```

```
  153.  153.    6.   10.  153.   17.    0.    0.   22.   28.   13.   65.
   48.    3.    1.   11.   13.   16.   73.   33.    0.    0.    0.    6.
    7.    6.   13.    1.   12.    1.   14.   43.  153.  153.    6.    6.
  153.   62.    3.    4.   67.   45.    5.   26.   70.   33.    5.   25.
   27.   11.    4.    5.    0.    0.    0.    9.   17.    8.    6.    0.
    1.    1.   36.   91.  129.   43.   14.    2.  139.   74.   11.   29.
   52.   35.    7.   13.  120.   40.    0.    0.    5.   24.   25.   23.
    0.    0.    1.    4.    9.   23.   31.    1.]

747.640869141 576.204711914 128.163116455 0.0142046399415 90.0196533203
[   5.    0.    2.   32.  137.   32.    2.    4.  137.    2.    0.    2.
   42.    9.    4.   96.   81.    5.    5.   12.    3.    7.   21.   58.
    0.    0.    3.   21.   10.   17.   15.    0.    4.    4.   20.  107.
  137.   33.    8.    9.  137.   28.    8.    4.   22.   17.   13.   77.
   85.   13.   10.   24.   12.    5.    0.   10.    0.    0.    4.   28.
   19.    3.    0.    0.    5.   12.   31.  116.   66.   79.   19.   10.
  137.   69.   19.    7.    2.    5.   12.   76.   76.    8.    0.    0.
   14.   31.   23.   34.    0.    0.    0.    3.   16.   30.   20.    0.
    2.    4.   17.   92.   86.   68.   15.    6.  137.   90.   20.    5.
    2.    4.   10.   48.   98.   25.    0.    2.   37.   41.   21.   26.
    0.    0.   10.    8.   23.   18.   13.    0.]
329.11114502 626.800170898 96.0515136719 0.0368077754974 79.2372436523
[   9.   11.   21.   16.   14.   24.   51.   15.  147.   23.    4.    1.
    0.    3.   27.  131.   27.    1.    0.    2.   12.   51.   37.   38.
    0.    0.    1.    4.    9.   35.   18.    0.   24.   53.   63.   23.
    1.    0.    0.    0.  179.   70.   14.    4.    4.    4.    2.   16.
   84.    3.    1.   12.  108.   34.    6.   11.    0.   11.   13.   17.
   34.   11.    7.    0.   56.   23.   49.   13.    7.   12.   13.   16.
  179.   22.   11.    1.    5.    2.    2.   48.  100.    1.    0.    3.
  167.   20.    0.    8.    0.    2.    8.   17.   69.    4.    0.    0.
   46.   10.   10.    5.    1.   10.   19.   16.  179.   42.    5.    3.
    2.    0.    2.   27.   58.   15.   11.   43.   92.    5.    0.    1.
    0.    0.    2.   24.   70.    4.    0.    0.]

< .....  Many features have been removed for brevity .....>

58.6121482849 181.520843506 6.63190317154 0.0270179584622 171.169754028
[  17.   18.   11.    8.   10.    9.   10.   23.   16.   19.    2.    1.
    1.   25.   52.   29.   42.   28.    1.    1.    1.   25.   83.   34.
   52.   80.   29.   28.   27.   11.   24.   24.   30.   24.    8.    3.
    0.    2.   21.   89.   56.   46.    7.    0.    2.   23.   28.   32.
  135.   20.    1.    0.   13.   36.  102.  114.   13.    2.    3.   24.
  135.   84.   53.   26.   34.   36.    7.    0.    0.    3.   38.   94.
   40.   14.    6.    7.   13.   66.   59.   24.  135.   58.    3.    4.
   32.   22.    9.   27.   41.   16.    5.   13.  135.  109.    2.    3.
   18.   68.   17.    1.    1.    7.   37.   16.   35.   36.   32.   19.
   10.   10.    9.   29.  127.   49.    8.    8.    9.   15.   28.  119.
   11.   18.   13.   30.   94.   72.   20.    8.]
#EOF
```

Appendix B. Example KLSH Table File

Below is an example of a KLSH table file. This file was calculated during the off-line portion of the KLSH search and loaded into memory during the on-line portion of the KLSH search in order to calculate how similar each of the features in the query image are to each of the features in each of the image represented in this table. This particular table was built using N=100, n=15, D=300, and d=40. These flat text files were quite large, and were usually between 15 and 20 MB each. Comments begin with a # sign. The first four lines are comments and are included for the benefit of the viewer. The line beginning with a left square brace lists the file names which were used to build and the order in which they occur in the data to follow. Since N=100 and n=15 for this table, this line would contain 15 different groupings of 100 file names. Because of this, the line has been truncated with a "<...>". The next line following the comment are the indices of the 30 randomly chosen columns. This information is needed so the same columns of descriptors can be chosen from the query image's scene file. The next 1500 lines after the comment would be the precomputed hash tables for each and the next 1500 lines after the comment would be the precomputed weight tables. The next 1500 lines after the comment represent the subset of the database features which were used to build the hash and weight tables. All this information is needed for the on-line portion of the KLSH search. The file ends with a #EOF signifying the end of the file.

```
#=======================
#Information about table 1
#=======================
#Filenames of database files are as follows:
[15_0.scene, 17_1.scene, 1_1.scene, 11_4.scene, 2_0.scene, 10_2.scene <...>
#Random columns chosen are as follows:
92,119,107,117,84,98,37,23,24,62,80,32,65,50,122,86,106,31,30,16,71,5 <...>
#Random features chosen from a feature depth of 300 are as follows:
296,265,292,50,299,97,60,192,136,57,243,175,206,31,172,
#hashTables[1].getHashTable is 1500 by 300
1.0,0.0,1.0,1.0,0.0,0.0,0.0,0.0,1.0,1.0,1.0,0.0,1.0,1.0,0.0,0.0 <...>
0.0,1.0,1.0,1.0,1.0,0.0,0.0,0.0,1.0,1.0,0.0,1.0,1.0,1.0,0.0,0.0 <...>
0.0,0.0,0.0,1.0,0.0,1.0,1.0,1.0,1.0,1.0,0.0,1.0,0.0,1.0,0.0,0.0 <...>
<...>
#hashTables[1].getWeightMatrix is 1500 by 300
4.421632232672155,-0.009401125020833848,-4.066339026168283,-1.4332872 <...>
0.15735213437325377,-5.479548592735836,-0.608062079956929,-1.09183815 <...>
2.0114411706394164,-0.17459113043336139,-1.6443747702425406,-1.921708 <...>
<...>
subsetsOfDatabaseFeatures[8] is 1500 by 40
0.07450980392156863,0.011764705882352941,0.00392156862745098,0.223529 <...>
0.054901960784313725,0.00392156862745098,0.00392156862745098,0.007843 <...>
0.00392156862745098,0.011764705882352941,0.00392156862745098,0.015686 <...>
<...>
#EOF
```

Bibliography

1. J. Dean and S. Ghemawat, "Mapreduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, p. 107, 2008.

2. R. New, "GPS Spoofing Detection via Dual-Receiver Correlation of Military Signals," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 4, 2013.

3. B. Nicholson, "World Fury at Satellite Destruction," January 2007. [Online]. Available: `http://www.theage.com.au/news/national/fury-at-space-destruction/2007/01/19/1169095981210.html`. [Accessed Feb 7, 2015].

4. J. Keller, "Three Join DARPA ASPN Navigation Sensor-Fusion Program to Reduce U.S. Need for GPS," February 2013. [Online]. Available: `http://www.militaryaerospace.com/articles/2013/02/Three-join-ASPN.html`. [Accessed Feb. 7, 2015].

5. B. Kulis and K. Grauman, "Kernelized Locality-Sensitive Hashing for Scalable Image Search," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2130–2137, IEEE, 2009.

6. J. J. Leonard and H. F. Durrant-Whyte, "Mobile Robot Localization by Tracking Geometric Beacons," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 376–382, 1991.

7. Apache Hadoop. [Online]. Available: `http://hadoop.apache.org`. [Accessed Feb. 7, 2015].

8. M. Aly, M. Munich, and P. Perona, "Distributed kd-Trees for Retrieval from Very Large Image Collections," in *British Machine Vision Conference, Dundee, Scotland*, 2011.

9. D. Moise, D. Shestakov, G. Gudmundsson, and L. Amsaleg, "Indexing and Searching 100M Images with Map-Reduce," in *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*, pp. 17–24, ACM, 2013.

10. W. E. Murphy, "Large Scale Hierarchical k-means Based Image Retrieval with MapReduce," Master's thesis, Air Force Institute of Technology, 2014.

11. D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

12. T. Costlow, "Big Data Poses Big Challenge for Military Intelligence," March 2012. [Online]. Available: `http://defensesystems.com/articles/2012/03/29/big-data-military-intelligence-analysis.aspx`. [Accessed Feb. 7, 2015].

13. T. White, *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 1st ed., 2009.

14. O. OMalley and A. C. Murthy, "Winning a 60 Second Dash with a Yellow Elephant," *Proceedings of sort benchmark*, 2009.

15. D. Borthakur, "Looking at the Code Behind our Three uses of Apache Hadoop," December 2010. [Online]. Available: https://www.facebook.com/notes/facebook-engineering/looking-at-the-code-behind-our-three-uses-of-apache-hadoop/468211193919. [Accessed Feb. 7, 2015].

16. D. Thain, T. Tannenbaum, and M. Livny, "Distributed Computing in Practice: The Condor Experience," *Concurrency and computation: practice and experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

17. S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," in *ACM SIGOPS Operating Systems Review*, vol. 37, pp. 29–43, ACM, 2003.

18. K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," in *IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–10, IEEE, 2010.

19. A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-Based Image Retrieval at the End of the Early Years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349–1380, 2000.

20. J. H. Friedman and J. L. Bentley, "An Algorithm for Finding Best Matches in Logarithmic Expected Time," 1975.

21. C. Silpa-Anan and R. Hartley, "Optimised kd-trees for Fast Image Descriptor Matching," in *IEEE Conference on Computer Vision and Pattern Recognition, 2008.*, pp. 1–8, IEEE, 2008.

22. J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, California, USA, 1967.

23. T.-S. Chen, T.-H. Tsai, Y.-T. Chen, C.-C. Lin, R.-C. Chen, S.-Y. Li, and H.-Y. Chen, "A Combined k-means and Hierarchical Clustering Method for Improving the Clustering Efficiency of Microarray," in *Proceedings of 2005 International Symposium on Intelligent Signal Processing and Communication Systems*, pp. 405–408, IEEE, 2005.

24. D. Nister and H. Stewenius, "Scalable Recognition with a Vocabulary Tree," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,*, vol. 2, pp. 2161–2168, IEEE, 2006.

25. M. Muja and D. G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pp. 331–340, INSTICC Press, 2009.

26. P. Indyk and R. Motwaniy, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," in *Proceedings ACM Symposium on Theory of Computing,*, pp. 604–613, 1998.

27. M. Aly, M. Munich, and P. Perona, "Indexing in Large Scale Image Collections: Scaling Properties and Benchmark," in *IEEE Workshop on Applications of Computer Vision (WACV)*, January 2011.

28. L. Paulevé, H. Jégou, and L. Amsaleg, "Locality Sensitive Hashing: A Comparison of Hash Function Types and Querying Mechanisms," *Pattern Recognition Letters*, vol. 31, no. 11, pp. 1348–1358, 2010.

29. J. Zhang, X. Liu, J. Luo, and B. Lang, "Dirs: Distributed Image Retrieval System Based on MapReduce," in *Pervasive Computing and Applications (ICPCA), 2010 5th International Conference on*, pp. 93–98, IEEE, 2010.

30. G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

31. C. Robusto, "The Cosine-Haversine Formula," *American Mathematical Monthly*, pp. 38–40, 1957.

32. I. T. Todorov, W. Smith, K. Trachenko, and M. T. Dove, "Dl_poly_3: New Dimensions in Molecular Dynamics Simulations via Massive Parallelism," *Journal of Materials Chemistry*, vol. 16, no. 20, pp. 1911–1918, 2006.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
26–03–2015	Master's Thesis	Sept 2013 — Mar 2015

4. TITLE AND SUBTITLE

Distributed Kernelized Locality-Sensitive Hashing for Faster Image Based Navigation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Hutchison, Scott, A, CPT, USA

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-15-M-070

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratories – Sensors Directorate
2950 Hobson Way
WPAFB OH 45433-7765
DSN 271-0690, COMM 937-255-3636
Email: gilbert.peterson@afit.edu

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RYRW, AFRL/RIGB

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Content based image retrieval (CBIR) remains one of the most heavily researched areas in computer vision. Different image retrieval techniques and algorithms have been implemented and used in localization research, object recognition applications, and commercially by companies such as Facebook, Google, and Yahoo!. Current methods for image retrieval become problematic when implemented on image datasets that can easily reach billions of images. In order to process extremely large datasets, the computation must be distributed across a cluster of machines using software such as Apache Hadoop. There are many different algorithms for conducting content based image retrieval, but this research focuses on Kernelized Locality-Sensitive Hashing (KLSH). For the first time, a distributed implementation of the KLSH algorithm using the MapReduce programming paradigm performs CBIR and localization using an urban environment image dataset. This new distributed algorithm is shown to be 4.8 times faster than a brute force linear search while still maintaining localization accuracy within 8.5 meters.

15. SUBJECT TERMS

Content Based Image Retrieval, Kernelized Locality-Sensitive Hashing, Hadoop, MapReduce, Localization, HDFS, Feature Extraction

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Gilbert L. Peterson, AFIT/ENG
U	U	U	U	106	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, x4281; gilbert.peterson@afit.edu