Wednesday June 7, 2023.

GitHub Repo: Scalable-Recognition-using-Vocabulary-Tree

Enhanced Vocabulary Trees for Real-Time Object Recognition in Image and Video Streams

Sugam Jaiswal[1], Arsheya Raj[1], and Josiah Zacharias[1]
[1]University of Washington Computing and Software Systems, Bothell WA USA

Corresponding author: Clark Olson (e-mail: cfolson@uw.edu).

Special thanks to the University of Washington Bothell for sponsoring the class, and to David Nistér, Henrik Stewénius, and the U of Kentucky Computer Science team for their effort.

ABSTRACT This research paper explores the application of Hierarchical k-Means clustering trees in the field of computer vision, focusing on the influential work by Nister and Stewenius (CVPR 2006) and its extensions. The primary objective is to investigate the effectiveness and efficiency of this approach for visual recognition tasks. The study delves into the hierarchical structure of k-means clustering trees and evaluates their impact on recognition accuracy and computational efficiency. The research methodology involves an in-depth analysis of the original work and subsequent literature, along with implementing a program design based on the proposed methodology. The design includes steps such as dataset installation, feature detection using the ORB detector, k-means clustering, max voting, cluster reassignment, and noise calculations.

To evaluate the performance of the approach, experiments are conducted using publicly available datasets, with plans to explore larger datasets such as Microsoft COCO. The research also considers the exploration of alternative feature detectors like SIFT, BRISK, and AKAZE, aiming to reduce noise and achieve higher accuracy. The findings of this research contribute to greater understanding of k-means clustering trees in computer vision and provide insights into their practical implementation. By comparing the results with the original research paper and exploring different methodologies, the study seeks to highlight the strengths and weaknesses of this approach and inspire further advancements in the field.

INDEX TERMS Computer vision, image processing, feature extraction, clustering algorithms and pattern clustering, hierarchical systems

I. INTRODUCTION

In recent years, computer vision has emerged as a vital field of research, with applications spanning various domains, including image recognition, object detection, and scene understanding. The ability to extract meaningful information from visual data plays a crucial role in numerous real-world applications, such as autonomous driving, surveillance systems, and augmented reality. One of the fundamental challenges in computer vision is developing efficient and accurate algorithms that can handle the complexity and variability of visual data.

This research paper focuses on the application of k-means clustering trees in computer vision, inspired by the seminal work presented by Nistér and Stewénius in their paper "*Scalable Recognition with a Vocabulary Tree*"[1]. The primary objective is to explore the effectiveness and efficiency of this approach for visual recognition tasks. By investigating the hierarchical structure of k-means clustering trees, the study aims to enhance recognition accuracy while minimizing computational complexity.

The motivation behind utilizing k-means clustering trees lies in their potential to address the limitations of traditional methods in visual recognition. Conventional approaches often struggle to efficiently handle large-scale datasets and achieve robust recognition in the presence of noise and variations. The hierarchical structure of k-means clustering trees offers a systematic framework for organizing visual data, enabling efficient search and retrieval.

The research methodology involves a comprehensive analysis of the original work by Nistér and Stewénius, along with subsequent literature and implementations that have built upon their foundations. Furthermore, a program design is developed, incorporating steps such as dataset installation, feature detection using the ORB detector, k-

means clustering, max voting, cluster reassignment, and noise calculation. The performance of the proposed approach is evaluated using publicly available datasets, with plans to explore larger datasets like COCO in the future.

By examining the strengths and weaknesses of the k-means clustering tree approach, this research aims to contribute to the existing body of knowledge in computer vision. Through empirical evaluation and comparison with the results presented in the original research paper, this study seeks to shed light on the practical implications and potential improvements in visual recognition tasks. The findings and insights derived from this research can pave the way for future advancements in the field, leading to more accurate and efficient computer vision systems.

II. APPROACH & METHODOLOGY

A. Feature Extraction

Like the authors, we've adopted the bag-of-visual-words representation, where local image features (e.g., SIFT descriptors) are extracted and quantized into visual words using clustering algorithms like k-means. This representation captures the frequency distribution of visual words in an image.

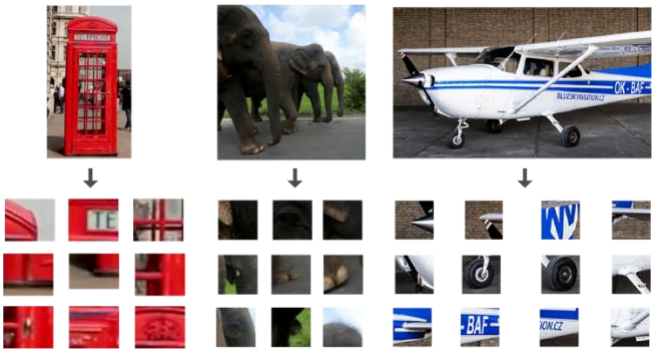


FIGURE 1. Feature extraction: which we used SIFT, ORB, BRISK, and AKAZE.

First, we extracted descriptors for each patch relative to the canonical directions, which are vectors that describe the local appearance of the patch (Fig 1). Once we extracted the features from all the images in the test, we quantized the descriptors with a vocabulary tree—a hierarchical structure that clusters similar descriptors into nodes.

B. Hierarchical k-Means Clustering Vocabulary Tree

We initially followed the author's approach to construct a hierarchical data structure called a Vocabulary Tree, which organizes the visual words in a tree-like structure where each node represents a cluster of visual words (Fig 2). The construction process involves recursively partitioning the visual word space using clustering algorithms like k-means. We used a branching factor (k) and tree depth (L) to determine the total number of leaf nodes in the tree.

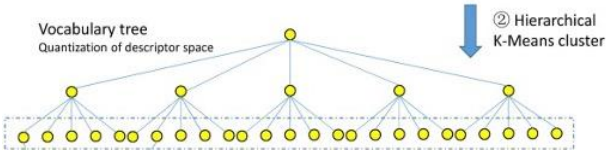


FIGURE 2. Creation of k-means clustering tree: with k=5 and L=2

Each leaf node in the vocabulary tree corresponds to a "visual word" and represents a cluster of similar SIFT descriptors. The leaf nodes are where the actual image descriptors from the training set are stored. To use the vocabulary tree for image retrieval or object recognition, SIFT descriptors are extracted from a query image and then propagated down the tree. Each descriptor "votes" for the most similar leaf node (or visual word) it encounters.

C. Image Indexing and Scoring

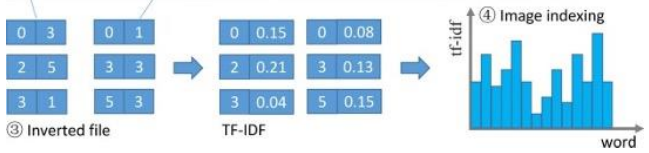
Votes are accumulated in an inverted file to form a histogram that represents the frequency of occurrence of each visual word in the image (Equation 1). The paper refers to this histogram as the "bag of words" representation of an image and we are representing them as same (BoW).

$$\sum_{i=1}^{L} k^i = \frac{k^{L+1} - k}{k - 1} \approx k^L$$

EQUATION 1. Total Descriptor Vectors in Tree: Computational cost is logarithmic based on the number of leaf nodes, and the size of the tree is approximately Dk^L bytes. As an example, a tree with 128-dimension descriptors, 6 levels and k of 10 results in 1M leaf nodes, using 143MB memory.

To enable efficient retrieval, we used an inverted file indexing technique called TF-IDF (Term Frequency-Inverse Document Frequency). Each leaf node of the Vocabulary Tree corresponds to an inverted file entry that stores the indices of images containing the visual word associated with that node (Fig 3).

Figure 3. Inverted image indexing: mapping visual words to the images where they occur. Allows efficient scoring with large databases, and stores image IDs and term frequency for each node.



Hierarchical scoring reduces the risk of vocabulary size becoming too large, with improvement in retrieval performance based on the number of leaf nodes (more leaf nodes is better performance). We also learned the

importance of a large vocabulary size and avoiding overly strong weights for inner nodes (Equation 2).

$$w_i = \ln \frac{N}{Ni}$$

EQUATION 2. Normalized Difference: Calculated between query and database vectors.

This caused us to consider frequency of occurrence and dependencies within the path, to handle weights for different levels of the vocabulary tree. Our testing revealed a trade-off between distinctiveness and repeatability in quantization cells and vocabulary tree depth.

$$s\ (q,\ d) = \left\| \frac{q}{||q||} - \frac{d}{||d||} \right\|$$

EQUATION 3. Normalized Difference: Calculated between query and database vectors.

We then applied entropy weighting to improve the retrieval performance of the TF-IDF scheme. When we query an image, we populate and sort a query tree (Fig. 4).

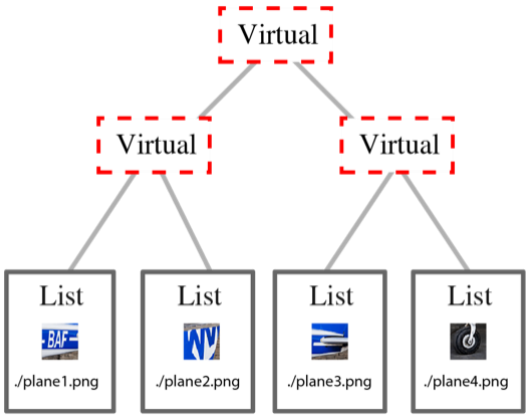


Figure 4. Database structure: example with 2 levels and branching factor 2. The leaf nodes contain inverted files, with the parent (inner) nodes have virtual inverted files which are produced by concatenating the inverted files of the leaf nodes.

The L1 norm, also known as the Manhattan norm or the taxicab norm, is a way to measure the magnitude of a vector in mathematics. It is defined as the sum of the absolute values of the individual components of the vector.

For a vector $x = [x_1, x_2, ..., x_n]$, the L1 norm ($||x||_1$) is calculated as:

$$||x||_1 = |x_1| + |x_2| + ... + |x_n|$$

In other words, it measures the distance between the origin (0, 0, ..., 0) and the point (x_1, x_2, ..., x_n) by summing the absolute differences along each axis.

The L1 norm has various applications, including in optimization problems, machine learning algorithms, signal processing, image similarity scoring. In terms of our score calculation and testing, we found that L1-norm performs better than L2-norm for retrieval (Equation 4).

$$\begin{aligned} \| q - d \|_p^p &= \sum_i |qi - di|^p \\ &= \sum_{i|di=0} |qi|^p + \sum_{i|qi=0} |di|^p + \sum_{i|qi\neq 0,\ di\neq 0} |qi - di|^p \\ &= \| q \|_p^p + \| d \|_p^p + \sum_{i|qi\neq 0,\ di\neq 0} (|qi - di|^p - |qi|^p - |di|^p) \\ &= 2 + \sum_{i|qi\neq 0,\ di\neq 0} (|qi - di|^p - |qi|^p - |di|^p) \end{aligned}$$

EQUATION 4. L1-norm: Calculated between query and database vectors.

The L2 norm, also known as the Euclidean norm or the 2-norm, is a way to measure the magnitude of a vector in mathematics. It is defined as the square root of the sum of the squares of the individual components of the vector.

For a vector $x = [x_1, x_2, ..., x_n]$, the L2 norm ($||x||_2$) is calculated as:

$$||x||_2 = \sqrt{(x_1^2 + x_2^2 + ... + x_n^2)}$$

In other words, it measures the distance between the origin (0, 0, ..., 0) and the point (x_1, x_2, ..., x_n) by calculating the square root of the sum of the squared differences along each axis.

The L2 norm is commonly used in various fields, including optimization, statistics, machine learning, and image processing.

For our implementation, the L2-norm is a simplified version of L1, allowing for faster retrieval (Equation 5).

$$\| q - d \|_2^2 = 2 - 2 \sum_{i|qi\neq 0,\ di\neq 0} q_i d_i$$

EQUATION 5. L2-norm: Calculated between query and database vectors.

Our image matching pipeline consists of two main steps: spatial verification and homography estimation using the RANSAC algorithm:

In the spatial verification step, we take as input a query image and a list of image paths. For each image in the list, we find correspondences with the query image using a specified feature detection method. These correspondences are pairs of points, one from each image, that we believe correspond to the same point in the scene being imaged.

We then apply the RANSAC algorithm to these correspondences to estimate an optimal homography matrix, which describes the transformation from the query image to the current image in the list. RANSAC operates by repeatedly selecting a random subset of the correspondences, computing the homography for this subset, and counting the number of inliers - correspondences that are well explained by the computed homography. The homography that yields the most inliers is considered optimal.

The output of our pipeline is a list of tuples, each containing the path of an image from the list, the number of inliers for

this image, and the correspondences between the query image and this image. As for the corresponding equations, the primary equation used in the RANSAC function is the homography projection:

Let pt1 and pt2 be corresponding points in the two images and H be the homography matrix. The projected point pt1' is calculated as:

$$pt1' = H * pt1\_homogeneous$$

where pt1_homogeneous is pt1 in homogeneous coordinates.

The loss function used to decide if a correspondence is an inlier is the Euclidean distance between the projected point pt1' and pt2:

$$loss = norm(pt2 - pt1'.t())$$

A correspondence is considered an inlier if loss <= 20. We chose the loss <=20 because it was giving us good results and we also ran the RANSAC algorithm for different number of rounds. The choice of a good value for the loss threshold in RANSAC depends on the specific problem, the characteristics of the data, and the desired trade-off between accuracy and robustness. There is no one-size-fits-all loss threshold, and the appropriate value for the loss threshold can vary significantly.

In general, a good value for the loss threshold should be selected based on the following considerations:

1. Noise level: If the data is relatively clean with low noise, a smaller loss threshold may be appropriate. This allows for a more accurate model estimation by including only data points that closely fit the model.
2. Outlier presence: If the data contains a significant number of outliers or grossly inaccurate measurements, a larger loss threshold may be necessary. This helps to make the RANSAC algorithm more robust by allowing some degree of tolerance for outliers.
3. Model complexity: The complexity of the model being estimated can also influence the choice of the loss threshold. More complex models may require a smaller loss threshold to ensure a good fit, while simpler models may be more tolerant to larger loss values.
4. Computational efficiency: The choice of the loss threshold can also impact the computational efficiency of the RANSAC algorithm. A smaller threshold may require more iterations (number of rounds) to achieve a desired level of accuracy, resulting in longer execution times.

It is often a matter of trial and error to determine a suitable loss threshold. Experimentation and evaluation on representative datasets or through cross-validation can help in identifying an appropriate value that balances accuracy and robustness for a specific problem.

The RANSAC algorithm is repeated several times specified by num_rounds, and the homography matrix H that produces the maximum number of inliers is selected as the optimal homography matrix.

D. Input Datasets

For testing and building the tree, we used Microsoft COCO (Common Objects in Context)—a large-scale dataset designed for various computer vision tasks, such as object detection, segmentation, and captioning. COCO contains over 200,000 labeled images with more than 1.5 million object instances across 80 object categories.

It provides rich annotations, object bounding boxes, segmentation masks, and captions—ideal for training and evaluation in our set. We created different subsets of the training 2017 dataset and validation 2017 dataset from COCO to build the databases of varying sizes. To download the images, python's pycocotools library was used to extract images from their respective instances stored in the JSON format.

The rich annotations in the Microsoft COCO dataset enable the development and evaluation of algorithms for tasks such as object detection, instance segmentation, image captioning, and visual relationship understanding. It has become a standard benchmark for measuring the performance of computer vision models on these tasks, facilitating advancements in the field.

The availability of the COCO dataset, along with evaluation metrics and associated challenges, has fostered significant research and progress in computer vision, driving the development of state-of-the-art algorithms for understanding and interpreting visual data.

III. RESULTS & DISCUSSION

Image Retrieval: Given a query image, the Vocabulary Tree is traversed by assigning the query's visual words to the appropriate nodes in the tree. The traversal path forms a histogram, and image retrieval is performed by searching the inverted file entries corresponding to the histogram bins. The retrieved images from the database are ranked and scored based on their similarity to the query.

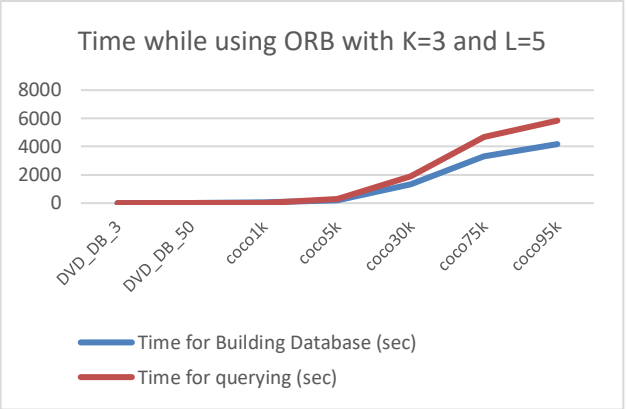


Figure 5. ORB Detector 1: Calculated time using ORB for K=3 and L=5

In our study, we conducted experiments using a Vocabulary Tree with a branching factor (K) of 3 and levels (L) of 5. We utilized various image sets from the Microsoft COCO dataset, each consisting of a different number of images. During our experiments, we measured the time required to build our database and the time taken to query an image within the database.

By analyzing the recorded timings, we gained insights into the efficiency and effectiveness of our approach. The results provided valuable information about the performance of our system, highlighting the trade-offs between database construction time and query processing time. This data will help us optimize and refine our methodology for future applications.

Our experiments demonstrate the significance of the COCO dataset as a valuable resource for evaluating and benchmarking computer vision algorithms. By leveraging this dataset, we were able to conduct rigorous experiments and obtain meaningful results, contributing to the advancement of research and development in the field of computer vision.

Our observations revealed an interesting relationship between dataset size, vocabulary tree parameters (K and L), and querying time. When using smaller values of K and L, which resulted in fewer leaf nodes in the vocabulary tree, we noticed that querying larger datasets took more time.

To further investigate this, we conducted additional experiments with K=6 and L=8, which increased the number of leaf nodes in the vocabulary tree. The results were striking when the number of leaf nodes was higher, the querying time reduced significantly compared to the experiments with K=3 and L=5.

These findings highlight the importance of carefully selecting the vocabulary tree parameters to optimize query performance. By appropriately tuning K and L based on the dataset characteristics, we can achieve more efficient and faster image retrieval.

This observation underscores the significance of parameter selection and fine-tuning in achieving optimal performance in image retrieval systems based on the Vocabulary Tree approach. It also suggests potential avenues for further exploration and optimization to enhance the efficiency and effectiveness of such systems in real-world scenarios.

During our experimentation, we made an interesting observation regarding the impact of the loss threshold on the performance of the ORB Feature Detector. Initially, when working with smaller image sets containing fewer than 1000 distinct images, setting the loss threshold to <= 20 did not yield significantly improved results.

However, we decided to explore the effects of adjusting the loss threshold further. By increasing the threshold value to <= 50, we discovered a notable improvement in the performance of the ORB Feature Detector for these smaller image sets. This adjustment allowed us to achieve more accurate and satisfactory results.

These findings suggest that the choice of the loss threshold plays a crucial role in optimizing the performance of the ORB Feature Detector, especially when working with smaller image sets. By carefully tuning this parameter, we can enhance the detection accuracy and ultimately obtain better outcomes.

It is worth noting that the optimal value of the loss threshold may vary depending on the specific characteristics of the image dataset and the desired level of accuracy. Further investigation and experimentation can help refine our understanding and identify the most suitable threshold for different scenarios.

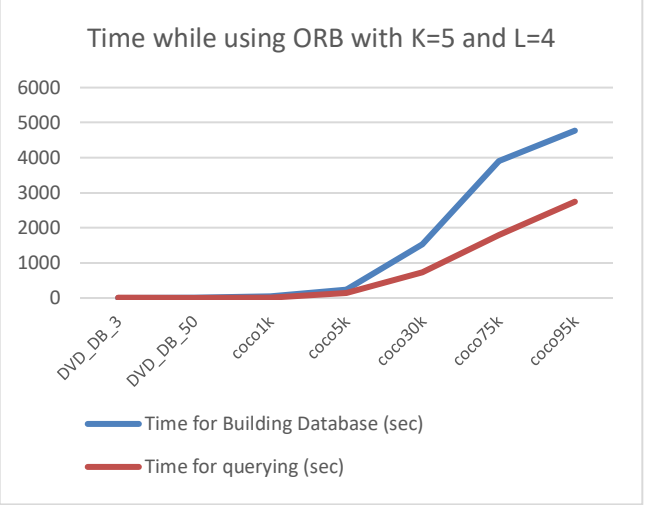


Figure 6. ORB Detector 2: Calculated time using ORB for K=5 and L=4

In our experimentation, we explored different configurations of the Vocabulary Tree, specifically K (branching factor) and L (levels). One configuration we tested was K=5 and L=4. We observed that this configuration resulted in a lower overall run time compared to the previous configuration of K=3 and L=5. The improved performance can be attributed to the increased number of leaf nodes in the Vocabulary Tree. In the K=5, L=4 configuration, the number of leaf nodes (K^L) was 625, whereas it was 243 in the K=3, L=5 configuration. We also tested a higher configuration of K=6 and L=8, which resulted in a significant increase in the number of leaf nodes (K^L = 1679616).

During our experiments, we utilized the ORB feature detector as our primary choice due to its faster processing speed, enabling us to handle larger image sets efficiently. However, we also explored other feature detectors such as BRISK and AKAZE, particularly on the coco1k and coco5k datasets. While BRISK and AKAZE required more computational time compared to ORB, they yielded superior results in terms of feature detection and matching. Thus, we found that BRISK and AKAZE were suitable alternatives

when dealing with smaller image sets that demanded more accurate feature extraction.

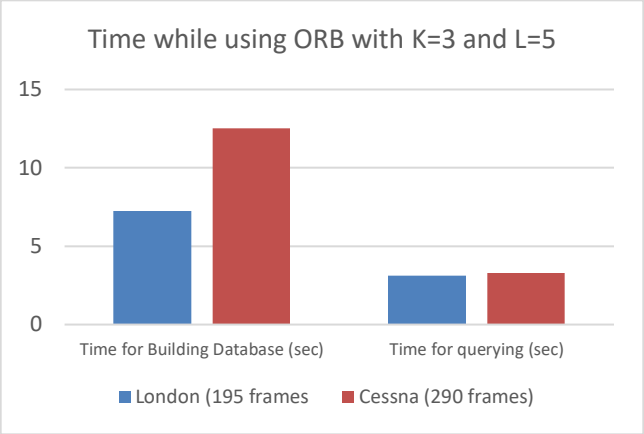


Figure 7. ORB Detector: Calculated times for different videos

During our experiments, we applied the method of constructing a vocabulary tree to video data as well. The vocabulary tree enables us to efficiently index and compare the local features extracted from each video frame and to measure the similarity between a query image and a video frame based on their common visual words. To evaluate the performance of our recognition scheme, we conducted an experiment using two sample videos with different characteristics. The first video was a London landscape video that consisted of 195 frames, and the second video was a Cessna video that contained 290 frames. We measured the time required to build the vocabulary tree and store the database for each video, as well as the time required to perform the query matching for a given image. We found that the query matching time was relatively consistent regardless of the number of frames in the video, whereas the database construction time was longer for the video with more frames than for the video with fewer frames.

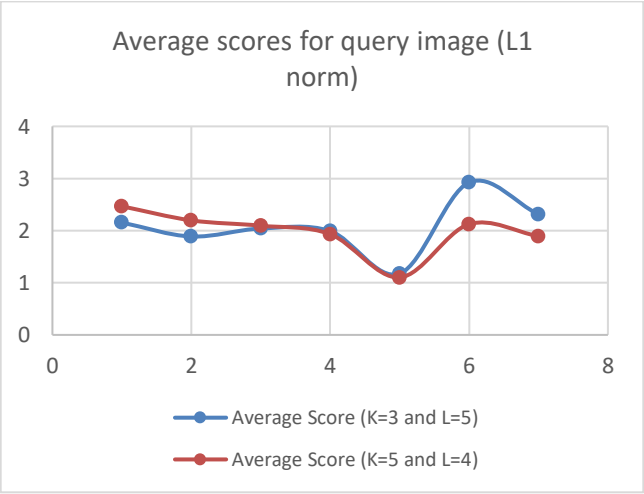


Figure 8. Parameter importance: differing k and L values affect performance.

In our code, we calculated the scores for all the matched images in the image set by computing the L1 norm of the BoW (Bag of Words) mapping between the query image (q) and the database image (d). The L1 norm, also known as the Manhattan distance, was utilized to measure the dissimilarity between the two histograms. The formula used to calculate the L1 norm was as follows:

$$\text{L1 norm} = |q - d| = |q_1 - d_1| + |q_2 - d_2| + \ldots + |q_n - d_n|$$

Where $q_1, q_2, \ldots, q_n$ and $d_1, d_2, \ldots, d_n$ represent the histogram values for the respective visual words in the query and database images. By computing the L1 norm, we obtained a quantitative measure of the dissimilarity between the query and database images, allowing us to rank and retrieve the most relevant matches. The formula used in our code is as follows:

$$\| q - d \| \frac{1}{1} = 2 + \sum_{i|qi\neq 0,\, di\neq 0} (|qi - di| - |qi| - |di|).$$

This is achieved in our code by using the accumulate function as follows:

score_lst[j]. second = 2 + accumulate(begin(q), end(q), 0.0f) - accumulate(begin(t), end(t), 0.0f).

After this, the average score is calculated by taking a mean of all the scores we have for the similar images.

In addition to computing scores based on the L1 norm, we further enhance our similarity measurement by incorporating the RANSAC algorithm. After running RANSAC and identifying the inliers for the similar images, we calculate a final score. This final score is determined by multiplying the number of inliers obtained through RANSAC with the previous scores obtained using the L1 norm.

By combining the inlier count and the L1 norm scores, we obtain a more comprehensive and robust similarity metric. The final scores are then sorted and reversed to ensure that the highest scoring images are ranked at the top. This enables us to effectively retrieve the best similar image from our database for a given query image, improving the accuracy and relevance of the results.

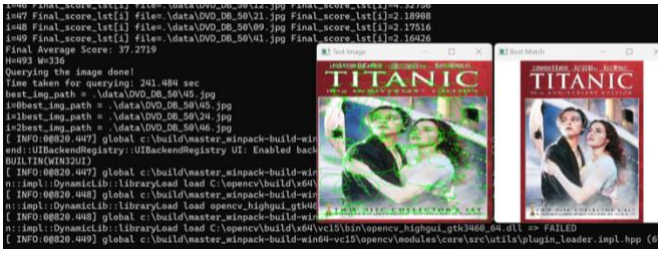


Figure 9. Match detection visualization: top features circled for visibility.

In our experimentation with different feature detectors, we found that SIFT was the most accurate one in finding the best matches while ORB was doing the poorest job at it.

However, the time taken to build the database and query the image using SIFT was the slowest among all the feature detectors. Thus, the choice of feature detectors is mostly relied on the trade-off between speed and accuracy. The above diagram is the output generated while using SIFT to build the vocabulary tree for 50 images. The time taken for the complete execution of the program was much longer than the contemporary configurations, SIFT was among the few that gave satisfactory results. In the example, the query image is a DVD cover of Titanic in a cropped version. The proposed scalable recognition scheme was easily able to locate and display the most similar image from the database I.e., the maximized version of the Titanic DVD cover. We also visualized the features of the query image that matched the best image found as shown by the green circles overdrawn in the diagram.

Scalability and Efficiency: The authors of the reference paper have highlighted the scalability and efficiency of the vocabulary tree method. The Vocabulary Tree allows for a compact representation of visual words, reducing memory requirements. Additionally, the inverted file indexing and hierarchical structure enable efficient image retrieval.

We were able to match most of the processes and techniques followed by the authors. We also saw a massive improvement in the time taken to build the vocabulary tree for larger datasets over the time mentioned in the original paper.

This was mostly due to the difference in computational capacity of our machines and the machines at the time of the original research. As far as scalability is concerned, the recognition scheme was easily able to scale as the number of nodes in the tree increased. We also observed that the higher number of leaf nodes in the tree resulting from higher values for K and L gave us quicker time to query an image but increased the time taken to build the database. On the other hand, the smaller number of nodes in the vocabulary tree led to a decrease in the time to build the database but increased the query time. With the further tuning of different values, it's possible to achieve an even more scalable and efficient image recognition pipeline which can find an optimal trade-off between the database building time and the query time.

IV. CONCLUSION

Building a vocabulary tree using the bag of visual words for images helps to achieve a scalable image recognition system. The tree built using hierarchical k-means clustering resulted in efficiently finding a similar match for the query images. The performance was tested on the Microsoft COCO dataset with up to 100K images and short videos with hundreds of frames. The project was able to accurately find the matching for all the test videos and small-size datasets in a quick span of time. It was seen that the time taken to build the tree and query the image was affected by various factors like the feature detector use, branching factor, levels of the tree, RANSAC threshold, etc. Further experimentation with these factors and their optimization to reduce the final query time could be the possible future direction for this research.

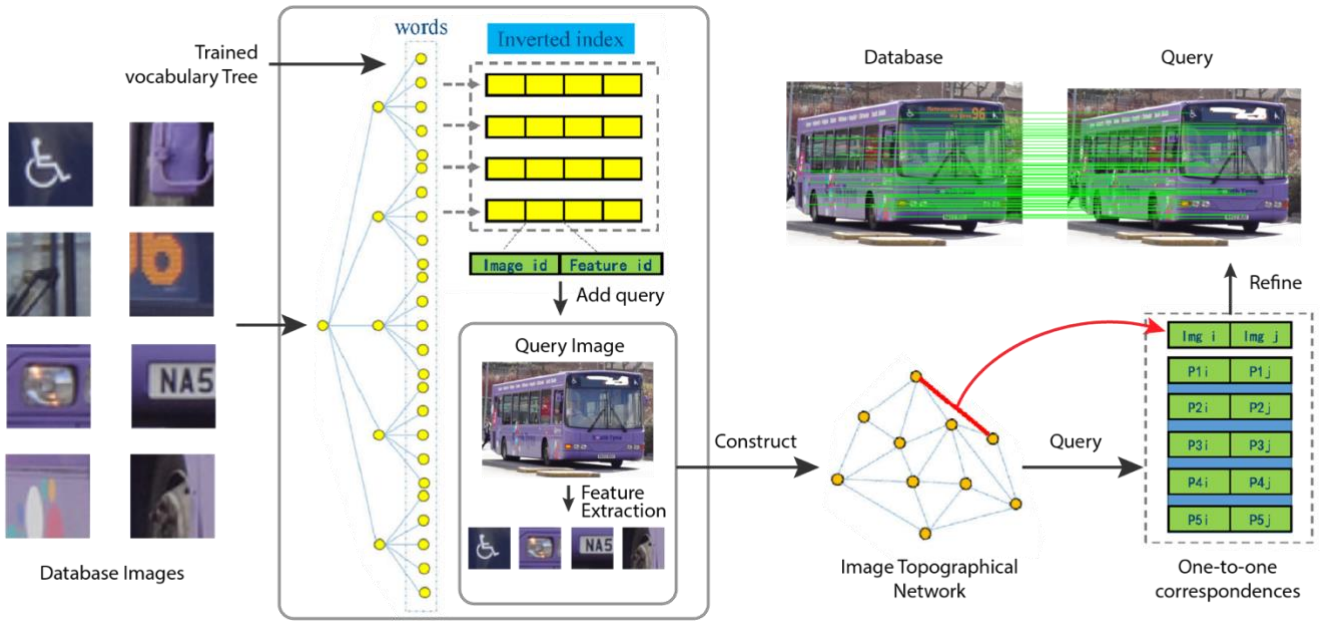


Figure 10. Proposed Algorithm: In the first step, we index all images in the database into a trained vocabulary tree, optimizing the amount of leaf nodes based on the features detected. We then establish the word-image inverted index structure called the Bag of Words. The second step involves processing each query image through the inverted index structure, enabling us to retrieve images that bear a resemblance to the query image. Utilizing all retrieved results, we construct the image topological connection network. In the final step, for every matched pair, we first secure initial matches using the direct index's acceleration feature. Then we refine these matches by integrating both local and global geometric constraints.

ACKNOWLEDGMENT

We express our heartfelt gratitude to all those who helped us to make this research paper. We extend special thanks to our advisor, Professor Clark Olson, for his invaluable guidance, stimulating suggestions, and constant encouragement throughout the quarter (Sp23) and research process.

APPENDIX

FUTURE CONSIDERATIONS

Researchers wishing to replicate this experiment should consider the following points:

1) Correct k and L parameters make a big difference in overall performance and accuracy. Optimizing the performance of our code involves selecting the right configuration parameters, which can sometimes be a challenging task due to the numerous options available. We have several important parameters to consider, such as the number of iterations for k-means, the epsilon value used in the k-means algorithm, the branching factor (K) of the vocabulary tree, the number of levels (L) in the vocabulary tree, the loss threshold for RANSAC, and the number of rounds for RANSAC. Finding the optimal values for these parameters requires careful experimentation and analysis. We need to strike a balance between computational efficiency and accuracy to achieve the best results. By fine-tuning these parameters, we can tailor our code to suit the specific requirements of the dataset and maximize the performance of our image retrieval system.
2) Nature of Feature Detector depending on machine (Mac vs Windows can cause problems with Brute-Force matcher).
3) SIFT, ORB, AKAZE and BRISK are four popular feature extraction algorithms that are used for various computer vision tasks, such as object recognition, image matching, and image retrieval. They have some similarities and differences in terms of their design, performance, and applications. SIFT is the most accurate and robust algorithm among the four, but also the most computationally expensive and slowest one. ORB is the fastest and most efficient algorithm among the four, but also the least robust one. ORB can handle moderate scale and rotation changes, but not affine transformations. BRISK is similar to ORB in terms of speed and efficiency, but slightly more robust to scale and rotation changes. AKAZE is a compromise between SIFT and ORB, offering a good balance between accuracy, robustness, and efficiency.

REFERENCES

[1] Nister, D., & Stewenius, H. (2006). *Scalable Recognition with a Vocabulary Tree*. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06) (Vol. 2, pp. 2161-2168).IEEE. https://inst.eecs.berkeley.edu//~cs294-6/fa06/papers/nister_stewenius_cvpr2006.pdf

[2] Sivic, J., & Zisserman, A. (2003). *Video Google: A text retrieval approach to object matching in videos*. In Proceedings of the Ninth IEEE International Conference on Computer Vision, 1470-1477. https://www.robots.ox.ac.uk/~vgg/publications/2003/Sivic03/sivic03.pdf

[3] Jiang, S.; Jiang, W.; Guo, B. *Leveraging vocabulary tree for simultaneous match pair selection and guided feature matching of UAV images*. ISPRS J. Photogramm. Remote Sens. 2022, 187, 273–293. [ScienceDirect].

Arsheya Raj was born in Patna, India and holds a bachelor's degree in information technology from the Birla Institute of Technology, Mesra (BIT Mesra) and is expected to complete his master's degree in computer science and software engineering from University of Washington in 2024.

Sugam Jaiswal was born in Nepal and holds bachelor's degree in computer science from the Vellore Institute of Technology in India. He is currently pursuing a master's degree in computer science and software Engineering at the University of Washington and expected to graduate in 2024.

Josiah Zacharias was born in Maracay, Venezuela, and holds a bachelor's degree in chemistry and biology from Trinity College, a UX Certificate from the School of Visual Concepts, and is expected to complete his Master of Computer Science and Software Engineering from the University of Washington in 2024.