1. Introduction

a. Problem and Context

There is a need for searching for semantically similar images to a given query image through a large database of images. For instance given an image of horses on a field, one maybe interested in finding all or a certain number of horse's images in the database. This problem can be solved using a supervised learning algorithm such as deep learning. An alternative solution is to index the images based on their content. In this approach, each image of the database is represented by a set of features derived from a mathematical process such as wavelet transformation.

In this project, such as content based image retrieval (CBIR) system is designed and implemented using the so called wavelet based image indexing (WBIIS) and search proposed by Wang et al [1]. The average accuracy was found to be around 40%. The algorithm is robust if the background image is clean, other it performs poorly. The compression ratio was at least 16:1.

2. System Design and Implementation

a. System overview

The CBIR system designed has two major components. The first component consists in building the database of feature vectors that semantically represent the database of images. Let name this phase the building phase.
The second phase is the process of finding similar images that best match a given query image which, we have called the query phase. Each phase will be further broken down into its constituent sub-components. The assumptions made will be presented before that.

a. Assumptions

No prioritisation of colour variation over intensity or vis-versa

In this project, the colour variation and intensity were deemed equally important for the different queries. As such, when computing the distance between the query image and the candidate image, the corresponding weights w_{ci} were set to 1. Please refer to Wang et al for more information [1].

No prioritisation of horizontal, vertical or diagonal components

The horizontal, vertical and diagonal components of the wavelet multilevel wavelet transform were also considered equally important. Their corresponding coefficients $W_{1,1}$, $W_{1,2}$, $W_{2,1}$, $W_{2,2}$ were equated to 1. For more information on the matter, please refer to [1].

Optimal Wavelet type and Multiresolution Level

Following the solution presented in Wang et al [1], Daubechies 8 wavelets and 4-layer wavelet transform were assumed optimal.

Optimal Image size

Based on Wang et al [1], we assumed that 128x128 were the optimal image rescaling dimensions. However, to explore the performance of a 4-layer wavelet transform, 256x256 rescaling dimensions was considered.

b. Building Phase

During the building phase, each image is loaded, rescaled to a particular size using bilinear interpolation, and the rescaled image in converted from RGB to the three components colour space C1, C2, and C3 as presented in [1]. Together these steps form the pre-processing phase.

For each image, the output of the pre-processing phase is a 3x128x128 matrix. Using Debauchies 8, the multiresolution wavelet coefficients of this matrix are computed. For an image of size 128x128, the maximum possible level of the multiresolution wavelet transform is 3. Beyond this level, some coefficients will be corrupt since Debauchies 8 filter length is 16 will be larger than the output matrix at level > 3. So, the rescaling size needed to be increased to at least, 256x256 in order to be able to use a 4-layer wavelet transform. This will increased the computation time without any significant increase in performance.
The current implementation therefore caters for 3-layer and 4-layer transforms according to the rescaling size.

For each image, the output of the wavelet transform is a 4x3xNxN (*submatrix times component times submatrix height times submatrix width*), where N = 16 for a 3-layer transform and N = 8 for 4-layer transform. Note that the coefficients of the last layer is stored and the lower layers weights are discarded. This results in a compression rate of: (128x128x3) / (4x3x16x16) = 16:1, for a 128x128 image with 3-layer transform. This is a more realistic compression ratio as opposed to the 64:1 ratio given in [1], which, did not consider the details coefficients used for the fine search.
Furthermore, the standard deviation of the upper left, NxN submatrix of wavelet coefficients is computed and stored together with the 4x3xNxN matrix as the feature vector.

The standard deviations of all the images were saved in a text file on a permanent memory. Similarly, the 4x3xNxN wavelet matrix of each image was stored in a separate text file. The feature vector of each image was indexed using its name as the unique identifier. The submatrices were indexed as shown in the table below.

cA	cH
cV	cD

cA is the upper left wavelet approximation, cH, cV and cD are the details coefficients [1], [2]. The components C1, C2, C3 were indexed accordingly.

Wang et al claimed to have stored the weights of another 5-layer wavelet transform of each image as part of the feature vector to be used in an additional search stage. Given that a 5-layer wavelet transform cannot be computed without corruption, this feature was discarded. In this implementation, the feature vector therefore consists of the 3- or 4-layer wavelet transform depending on the rescaling size and the standard deviation of the lowest frequency weights.

c. Query Phase

The query phase was simplified to 2-stage comparison since the claimed 5-layer wavelet computation is not possible with a 128x128 nor with a 256x256 input matrix.

The first stage is an attempt to speed up the search through the database of features by discarding statistically different images to the query image. The formula provided in [1] was used as the “filtering” condition. Practically, the feature vector of the query image was computed

using the procedure described in the building process. The default value of beta was set to 50 as suggested by Wang et al [1].

The second stage is the fine search based on the image content represented by the wavelet weights. As such, using the square Euclidean distances between the query image and each candidate image were calculated and sorted in an ascending order. The Euclidean distances were squared to optimised runtime since only the relative distances are of interest.
The sorting algorithm used is the python standard Timsort algorithm, which is a hybrid sorting derived from merge sort and insertion sort [3].

In sum, for a given query image, the feature vector of the query image is computed, the standard deviation of the image database are loaded for the first stage comparison. Then, the wavelet coefficients of the candidate images that passed the first stage are loaded to compute their respective distances to the query image. These distances are ordered. Finally, the first M (specified by the user) images are displayed and stored as the best matches.

d. Implementation Softwares

The implementation was done in python 2.7 [3] together with various external packages. These packages will now be discussed.

OpenCV for image operations

OpenCV is an open source computer vision library [4]. In this work, the python version 3.4.1 was used to perform basic image operations. It was used for loading, resizing, interpolating, displaying and saving images. OpenCV 3.4.1 requires the scientific computing package numpy [5] to work.

PyWavelet for wavelet transform

To compute the wavelet transforms, the python package called PyWavelet [2] was used.

e. Unimplemented Feature: Partial Query

In this work, the partial query feature was not implemented.

3. Analysis and Discussions

a. Test Corpus Description

The test corpus is an image database of 1000 images divided into 10 classes. Each class therefore has 100 elements [6]. Among others, the classes are as follows: horses on a field, flowers, beach scenes, buses, dinausors, and elephants.
The images are jpg images of size either 256x384 or 384x256.

b. Hardware Description

An entry level laptop computer: Asus X554L, was used for both the implementation and the test. The basic specifications of this hardware are listed below.

- Processor: Intel® Core™ i3-5010U CPU @ 2.10GHz
- RAM: 4,00 GB

- Permanent Memory: Hard disk drive

c. Test Procedure

Accuracy

The test was performed as follow. The first element of each was taken as the query image and the percentage accuracy of matches were calculated based on the elements of the class returned. The results are presented in Table 1. Any outside image was not considered because finding the best matches of a particular class is independent of other classes. The algorithm does not learn anything about inter-class or inner-class dependency. So, when testing a given class, an 'outside' image is effectively used for the other classes.

Runtime

The time taken to compute all the wavelet weights is on average: 5 seconds

The time taken to compute and save the feature vector database is on average: 45 seconds

The time taken to query is on average: 7 seconds.

d. Results

Table 1: Accuracies of each class

Class	Culture	Beach	Buildings	Buses	Dinaus-ors	Elepha-nts	Flowers	Hors-es	Mountains	Food
Accuracy (%)	18	26	21	9	99	45	66	66	47	5

The average accuracy is 40.2%. The best accuracy is 99% which is an outlier. This is because the background of this images are clean: mostly white background without any noise. This corresponds to the images of a dinausors as shown in the pictures below.

The worst accuracy: 5% is for the food class. This is because this class as opposed to the dinausor class does not have a particular local structure. It therefore most rely on colour variation which can be confused other classes.

4. Conclusions

This work was about implementing an efficient CBIR system using daubechies' wavelet transform. Thus, given a database of images and a query image, the semantically similar images needed to be found. The implemented solution was WBIIS was presented in Wang et al [1].

In this solution, the images were described using wavelet weights and a multistage comparison algorithm was employed to search for the closest match. This method guarantees some form of output which does not necessarily satisfy the user's request.
It was found that for images without any noise, the algorithm performs perfectly. However, when the object is buried in multiple colours, the matches are not satisfactory in general.

The wavelet representation allows a compression of at least 16:1 ratio. The first stage comparison does not reduce significantly the size of the fine search as claimed in [1].

5. Recommendations

The following recommendations can be implemented to improve this project:

- Test the assumptions made recoding the optimal image size, the type of wavelet, the number of wavelet layer, …
- Implement the partial query feature.
- Perform PCA on the feature vector.
- Investigate the optimal value for beta in the first stage comparison.

6. References

[1] O. F. James Ze Wang Gio Wiederhold and S. X. Wei, "Content-based Image Indexing and Searching Using Daubachies' Wavelets," *Int. J. Digit. Libr.*, vol. 1, pp. 311–328, 1997.

[2] and C. Lee G, Wasilewski F, Gommers R, Wohlfahrt K, O'Leary A, Nahrstaedt H, "PyWavelets - Wavelet Transforms in Python." [Online]. Available: https://github.com/PyWavelets/pywt. [Accessed: 09-Apr-2018].

[3] Python Software Foundation, "Python Language Reference." .

[4] Itseez, "Open Source Computer Vision Library." 2015.

[5] T. Oliphant, "NumPy," 2006. [Online]. Available: https://github.com/numpy/numpy. [Accessed: 09-Apr-2018].

[6] J. Z. Wang, "CBIR Database." [Online]. Available: http://wang.ist.psu.edu/docs/related/. [Accessed: 10-Apr-2018].