

# WHASSUPP: A Novel Approach to Query-by-Sketch Using Wavelet Coefficients and Color Histograms

Ankit Patel (abpatel@fas.harvard.edu)  
Mark Tonkelowitz (mtonkel@fas.harvard.edu)  
Harvard University

**Abstract:** In this paper we present the Wavelet-Histogram Aggregate Scoring System for User Painted Pictures (WHASSUPP), an improved method for content-based image retrieval from an image database. Rather than rely on a single signature for each image, our algorithm computes two indices, one based on a wavelet decomposition of the image; the other based on a fixed palette color histogram. This method appears to be superior to either comparison metric alone because it allows for objects in user sketches to be scaled, rotated, or translated with respect to their locations in the target image.

## 1. Introduction

In 1995, Jacobs, Finkelstein, and Salesin showed that algorithms using wavelet coefficients as a metric for comparison of images were surprisingly accurate and efficient in “query by sketch” applications [1]. However, while wavelets are robust with respect to color shifts, they are not invariant to scaling, rotation, or translation of the input signal (sketch image). As a result, some other method is needed to account for these transformations. Jacobs et al compare their work to image retrieval algorithms involving color histograms [1]. The latter method is invariant to scaling, rotation, and translation, but performs poorly when there is a color shift between the sketch and the target images. This raises an interesting question: Can we combine these two approaches?

We believe that a combination of these two methods would result in an algorithm that yields greater accuracy and that is better tuned to the needs of query by sketch applications. Although Jacobs et al showed impressive retrieval accuracy for databases of 20,000 images [1], today we must consider the

existence of even larger databases, with millions of images.<sup>1</sup>

The WHASSUPP project attempts to answer the question of what combination, if any, of both wavelet- and histogram-based indexing methods leads to the best image retrieval accuracy. However, our evaluation also takes into account performance issues. While our scheme may produce more accurate results than either comparison metric alone, is the longer running time of the algorithm justified? We certainly believe so.

At the heart of the WHASSUPP project is a simple indexing and retrieval scheme that can be easily implemented with any general commercial database system. The rest of this paper explores the merits and shortcomings of this indexing scheme.

### 1.1 Related Work

Since 1995 there have been several improvements upon the scheme presented by Jacobs et al, but none of them have involved a combination of wavelet coefficients and color

---

<sup>1</sup> Although we were only able to obtain a sample database of 10,000 images, we speculate about the performance of WHASSUPP on larger databases in Section 5.

histograms. In 1998, Wang et al showed that using Daubechies wavelets and a more complex scoring scheme results in greater search accuracy [2]. However, their work again failed to address the case where the user sketch is some transformation of the target image (i.e., scaled, translated, rotated). Later that year, in a paper on filtering out pornographic images, Wang et al added color histograms to their matching metric, but only to determine if an image was above a certain threshold of yellow [3]. Natsev et al used wavelets and dynamic programming in the 1998 WALRUS project to perform efficient region-based image analysis [4]. While their method provides solutions to the traditional wavelet shortcomings regarding scale and translation variance, they do not address rotational transformations.

## 1.2 Paper Overview

In this paper, we will explore the advantages and disadvantages of an image indexing scheme involving both wavelet and color histogram methods. In Section 2, we provide a brief introduction and summary of the mathematical theory behind wavelets, along with the theory of color histograms and our own contribution to their unification, the WHASSUPP system. Section 3 discusses the implementation and issues of the Java Client/Server architecture that was developed and used for testing purposes. Sections 4 and 5 discuss the performance of WHASSUPP in relation to either the wavelet or color histogram method alone and possible future work.

## 2. Development of an Image Querying Metric

The motivation for our algorithm comes from two existing paradigms for image retrieval: Haar wavelet decomposition and color histogram analysis. In Section 2.1, we give an overview of wavelets and their mathematically

useful properties. The algorithm and indexing scheme we use for wavelet-based image retrieval is explained. In Section 2.2, we examine methods of color histogram computation. We explain our decision to use a fixed palette and follow with our histogram-based algorithm and indexing scheme. Finally, in Section 2.3, several hybrid scoring methods, including WHASSUPP, that make use of both the wavelet and color histogram indices are motivated.

### 2.1 Overview of Wavelets<sup>2</sup>

In mathematics, we can express a function  $F(t)$  by a (possibly infinite) linear combination of a set of *basis* functions, with a set of scalar coefficients. For example,  $\sin(x) = x - x^3/3! + x^5/5! - \dots$  is the expression of the sine function in terms of the basis functions  $\{x^n : n = 0, 1, 2, \dots\}$ . (This is formally known as the Taylor Series of  $\sin(x)$  centered at  $x=0$ ). Note that this series is an infinite series and that formally an infinite number of terms are needed to evaluate the function exactly; in practice, we only need the dominating terms and thus we may truncate the series.

Of course, in the real world, the world of algorithms and data structures in computers, an infinite dimensional basis is certainly not feasible and instead we use a finite-dimensional basis that is of course less expensive in terms of memory but lacks some expressive power since not all arbitrary functions can be represented exactly and thus are approximated (i.e., truncation of an infinite series).

There are many useful bases other than the Taylor basis; among them, the most important is the Fourier Basis, composed of  $\{\sin(nx), \cos(nx) : n \text{ integer}\}$ . This implies that any

---

<sup>2</sup> Many thanks to Prof. Steven Gortler for his help on Wavelet theory and its application in Computer Graphics. Much of this section appears in his PhD thesis [5].

arbitrary function can be represented in terms of a series of sines and cosines. But despite the popularity and rich applications of Fourier Analysis, the Fourier basis functions have a property known as *global support*, which means that *each* basis function has “influence” over the entire parameter domain. Formally, the basis functions are non-zero over the entire real line. This means that if the curve were to be modified slightly, the effect of this small change would have “influence” over the entire domain (i.e., many or all of the scalar coefficients would change). This property represents what we do not want; intuitively, in an approximation of some function, if some part of the curve changes slightly, we do not want the change to propagate through to all the scalar coefficients, it is simply less efficient computationally. Instead, we desire the property of *compact support*, where the basis functions have only local influence; hence the functions are non-zero over only a finite interval of the domain. (Imagine the hat basis functions over the entire real line).

Another property that is of interest is that of frequency decomposition. In a Fourier basis, sines and cosines of high frequencies correspond to scalar coefficients that describe the “contribution” of that frequency to the entire function or *signal*. Hence, we can extract fine-grained information (high resolution) from a signal as well as coarse information (low resolution) by simply examining the high and low-frequency scalar coefficients of the corresponding Fourier basis functions. This property of frequency decomposition is useful in computer graphics, where one can represent an image as a signal containing fine detail as well as coarse, of high and low frequencies. However, as we mentioned before, the Fourier basis has the unfortunate property of global support, so that a set of coefficients that describes a function  $F(t)$  could be completely different from a set of coefficients that describes another function  $G(t)$ , *even if*  $F(t)$  is approximately equal to  $G(t)$ . We desire a set of basis functions with *compact*

*support* and the ability to extract information from a signal at different frequencies (resolutions) – enter Wavelets.

Wavelets are a family of mathematical basis functions that have the properties that we desire; they provide frequency decomposition along with compact, or finite, support. Unlike previous bases we have mentioned where each function is different, a wavelet basis consists of multiple copies of the *same* function. The copies are simply translations and scalings of this original function. We will now examine the Haar basis, our basis of choice for this paper.

### 2.1.1 The Haar Basis

The simplest and oldest example of a wavelet basis is the Haar basis. It consists of a single box function and many step functions, each of successively higher resolution (i.e., the width of the step). Consider two adjacent step functions  $\Phi(L,0)$  and  $\Phi(L,1)$ . They can be replaced by the basis consisting of a box function twice as wide  $\Phi(L-1)$ , and a step function (called the Haar function)  $\Psi(L-1,0)$ . This process can be repeated indefinitely to produce a basis with as fine a resolution capability as needed. In essence, this is the idea behind a *hierarchical basis*, one that captures information at varying scales. Without delving further into the mathematics of wavelets, we list some of the properties of the Haar basis that are important for our purposes:

- multiresolution property: ability to extract information at any scale or resolution
- hierarchical representation: allows for economic representation of a function (constant areas of little detail will have very small coefficients and can be removed with no significant loss)
- efficient coefficient calculation: transforming from the box basis to

the Haar basis is cheap in terms of computation time.

The reason for our choice of the Haar basis is simple. Our user-painted queries will primarily consist of constant-colored regions of little or no fine detail, and thus the Haar basis will provide the most economical representation for our image indexing scheme and scoring algorithm.

### 2.1.2 Wavelet Indexing Scheme

We pre-compute signatures of each image in our database as follows: First, we apply a 2D Haar basis decomposition to our image. Jacobs et al. show that we need only store the 60 coefficients of largest absolute magnitude [1]. Hence, we discard the rest. Since our image signal is actually composed of 3 channels (RGB), we must actually apply this transform to each channel separately. But since distance in RGB space does not accurately describe human perception, we choose to transform our images into the YIQ basis, where the Y coefficient (the most important one) corresponds to the “black or whiteness” of an image. Studies have shown that human perception is much more sensitive to changes in Y than to I or Q [1].

### 2.1.3 Wavelet Scoring Algorithm

After pre-processing an image, we store only the signatures calculated. When a user performs a query, we calculate the signature of the painted image and compare it to all the signatures in the database with a simple scoring algorithm that is based on the quantization of the wavelet coefficients (to  $\pm 1$ ) and assigning importance to certain coefficients via a set of weights [1].

## 2.2 Overview of Color Histograms

As simple and beautiful as the theory of wavelets is, it does not give the full solution to the problem of finding similarity between two

images. It has been shown that, although wavelet decomposition methods can extract accurate shape/texture information from an image, they fare poorly when faced with linear transformation of an image (i.e., scaled, rotated, or translated versions of the original image). In other words, if the user’s query draws a red blob in the upper left hand corner and the blob is actually in the right hand corner of the real image, wavelet methods may not catch the image from the database. Similar examples for rotations and scaling show poor performance in those cases as well. However, there is another method of comparison that is based on color histograms that treats each image as a 3D distribution in YIQ-space. Since YIQ-space is independent of location in an image (i.e., x,y) we do not have to worry about linear transformations of an image, because both the original image and a transformation of it should have similar YIQ distributions. Of course it would be computationally expensive to measure the YIQ distance between two images if we allow the range to have 256 values as is normally done.

### 2.2.1 The Quantization of YIQ Space

Instead, we quantize the YIQ color space further, taking into account the colors that are used the most by people in queries. After much training and testing, we decided upon a fixed palette of 24 colors. Essentially, what we’ve done is partition YIQ space into 24 equivalence classes (not necessarily equal in size). A YIQ point (y, i, q) is in a certain equivalence class E if and only if the corresponding palette color is the *closest* palette color in terms of YIQ distance.

### 2.2.2 Color Histogram Indexing Scheme

With our quantized color space of only 24 colors, we store only the frequencies of these colors in an image. We normalize by dividing by the total number of pixels in the image so as to allow valid comparison to other images

of possibly different dimensions. We perform this step of calculating color histogram signatures in the pre-processing phase of database construction. If new images are added, others in the database are not affected.

### 2.2.3 Color Histogram Scoring Algorithm

Scoring for color histograms is rather simple; we simply sum the absolute differences between palette color frequencies and sort them from lowest to highest.

## 2.3 Hybrid Metrics

Although color histograms provide a solution to the problem of queries being linear transformations of the real image, they suffer from poor performance in other categories, namely shape and texture information. But this is exactly the wavelet method's strong suite. This fundamental symmetry poses the question: What if we could combine the two methods?

In developing a hybrid metric that uses information from both the wavelet and color histogram methods, we have encountered the problem of how exactly to meld the two metrics in order to get a good overall metric. This problem, not local to just image processing, it is a fundamental problem of conflicting information: if one method says this image is a good match for the query and another says it is not, which do we trust? For this problem, we have tried several hybrid solutions.

### 2.3.1 Naïve

The naïve method of combination is truly simple: run each method on the query image and return the top  $m$  images from each so that  $2m$  images fit on the screen.

### 2.3.2 Average Rank

Another, perhaps naïve, method is to ignore the absolute differences in scores between images and assign two relative ranks to each image, one for each method. Then, the overall rank is simply the average of the two; weighting can be done if one method is trusted more than another.

### 2.3.3 Color-Proportional Influence

This scheme takes advantage of the observation that user queries with more distinct colors should place more influence on color histograms and less on wavelets since (presumably) the user is trying harder to match certain colors more than shape or texture. However, we chose not to test this method since our user queries tended to be limited to a few colors (usually less than 4).

### 2.3.4 Relative Proportionality Influence (WHASSUP)

Sparked by the observation that when a certain method returns a relatively high score, it is usually accurate and we would like to trust it more than the other method, we want to give relative importance to each method based on its given score on an image. In other words, given two standardized, normalized scores  $s_1$  and  $s_2$  for an image  $I$ , the calculated overall score is  $s = s_1(s_1/(s_1 + s_2)) + s_2(s_2/(s_1 + s_2))$ . This simplifies to  $s = (s_1^2 + s_2^2)/(s_1 + s_2)$ . Some test cases show that  $s = 1$  when *either* score equals 1 (or both), as we desire. The scores are each standardized by assuming a normal distribution of scores (Central Limit Theorem of Statistics with  $n = 1000$  images).

## 3. Implementation

To test WHASSUPP, we needed to acquire a set of test images, set up a general-purpose database, and develop a client application to paint the queries and display the results. In order to be consistent with previous studies, we used the same image set that was used by

WBIIS [2], freely available from ftp://db.stanford.edu/pub/wangz/image.var y.jpg.tar. For the database, we acquired the latest release of MySQL for Windows (3.23.27). Finally, our client was developed in VisualCafe 4.1 Standard Edition, now free from www.webgain.com. All development and testing was done on a Pentium II 400 with 192 MB RAM running Windows98 SE.

### 3.1 Images

The WBIIS sample image database consists of 10,000 assorted JPEG images in 5 different dimensions: 128x85, 128x96, 128x128, 96x128, and 85x128 pixels [2]. Image content ranges from simple patterns to paintings to complex nature scenes. When an image with canvas size other than 128x128 was analyzed, it was centered on a 128x128 canvas and padded with black pixels on either side. Thus, our image database implementation is consistent with that used by Jacobs et al [1].

### 3.2 MySQL

The choice to use MySQL was not an easy one. We knew that we were going have to store 180 data points for each wavelet decomposition and 24 data points for each color histogram and we were not convinced that MySQL would scale well. However, we also knew that most high-end commercial solutions would be unnecessary (and unattainable) for our project. We considered developing our own system using a freely available embedded database, such as SleepyCat, but we wanted to be able to harness the power and simplicity of SQL for data retrieval. (As you may recall, one of our design goals for WHASSUPP was that it would be implemented on top of a general database system.)

Another consideration we had in choosing a database involved the issue of interfacing it with our query client. Though many commercially available databases have JDBC-

ODBC bridges, MySQL was the only one we could find that had a fully native (Level 4) and free Java driver.<sup>3</sup> Thus, MySQL became the logical choice.

#### 3.2.1 Storing Image Information

The Images table is depicted below. Notice that each image is assigned a unique ImageID. In order to keep our database lean and mean, we merely stored an image's path or URL. We also chose to store the average YIQ colors here.

Field Name	Field Type
ImageID	INT UNSIGNED AUTO_INCREMENT PRIMARY KEY NOT NULL
Location	VARCHAR(255)
Yavg	DOUBLE
Iavg	DOUBLE
Qavg	DOUBLE

#### 3.2.2 Storing Quantized Wavelet Coefficients

In an effort to allow MySQL to scale more gracefully, wavelet coefficient data is separated into six tables, two for each color channel (YPos, YNeg, IPos, INeg, QPos, QNeg). Each table contains the following fields:

Field Name	Field Type
UniqueID	INT UNSIGNED AUTO_INCREMENT PRIMARY KEY NOT NULL
ImageID	INT UNSIGNED NOT NULL
Xcoor	INT UNSIGNED NOT NULL
YCoor	INT UNSIGNED NOT NULL
	INDEX (XCoor, YCoor)

<sup>3</sup> <http://www.worldserver.com/mm.mysql/>

After an image is decomposed and the largest 60 coefficients of each color channel are quantized, they are merely added to the appropriate positive or negative table.

The purpose of indexing based on X and Y coordinates is so that if the database ever becomes too large to be loaded into Java data structures, images matching the query at a particular (X, Y) value can still be quickly retrieved.

### 3.2.3 Storing Color Histograms

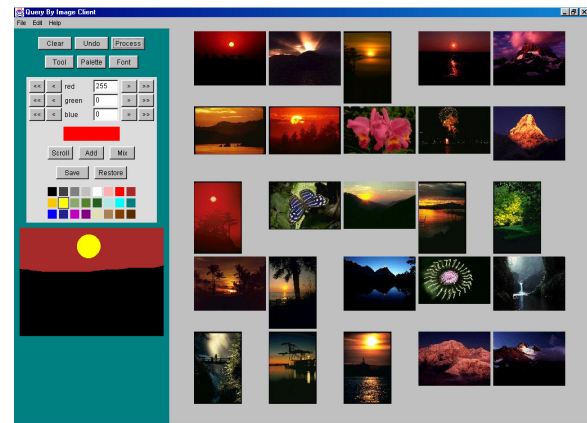
Two tables are used to store color histogram data. The Colors table contains the mappings from our 32x32x32 bucketed YIQ cube to our fixed palette. It is generated by a separate utility we wrote to simply determine the closest palette color for each bucket. The data from the colors table is loaded into a static data structure every time the client is launched. On our test system, this data takes no more than four seconds to load.

Color frequency distributions for each image are stored in the Hist table, depicted below. When calculating the L1 difference between a query and all target images, all data from this table must be examined. Since there are only 24 data points per image, and since each is just an integer, we feel that it is quite reasonable to load the entire table into memory at client execution time. This results in an amazing performance boost over a simple sequential scan of the table.

Field Name	Field Type
HistID	INT UNSIGNED AUTO_INCREMENT PRIMARY KEY NOT NULL
Color	INT UNSIGNED NOT NULL
Rank	INT UNSIGNED NOT NULL

## 3.3 Java Client

For ease of GUI building, we used Java to develop a simple client application for painting queries, comparing them to the target images, and scoring results. The Query By Image Client, pictured below, has two main visual components: the scribble pad, and results pane.



*Query by Image Client. Lower left shows the user-painted query and results pane is on the right. Hybrid method results shown.*

### 3.3.1 The Scribble Pad

The scribble pad is where a user paints their query. It is based on a Java applet developed by Brian Prentice.<sup>4</sup> We modified it to use our fixed palette and to allow a user to choose their canvas shape.

A number of standard drawing tools are available for composing a sketch. The user merely needs to choose a tool, a line width, and a color to begin their painting. They can add as many colors as they wish and have unlimited levels of undo.

When the user is finished with their drawing, they press the process button to start a query.

### 3.3.2 The Results Pane

<sup>4</sup> Source code freely available at <http://linuxenvy.com/bprentice/Paint/Document.html>

The results pane is where the images returned by the chosen scoring algorithm are displayed. Only the top 25 results are shown.

## 4. Experiments

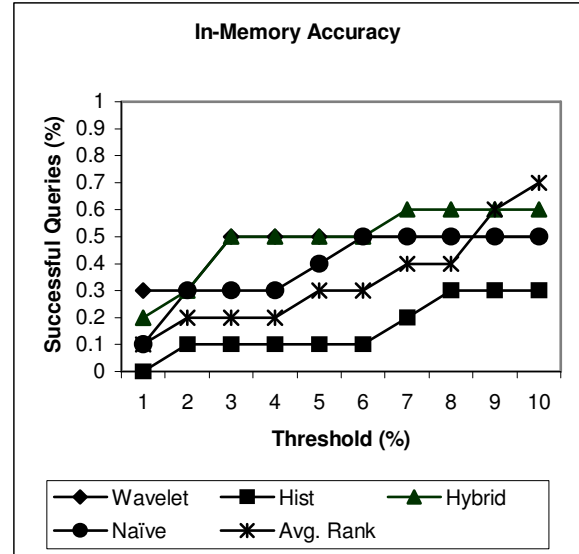
To evaluate the accuracy of our new method, we performed the following tests, which are similar to those devised by Jacobs et al [1]:

- An “in-memory” test, where a user was briefly presented with a target image and then asked to paint it.
- A “no-memory” test, where a user could look at the target image while painting it.
- Linear transformation tests, where the query image is a scaled, rotated, or translated version of the target image.

The size of our test database was 1000 images. For each test, we recorded the rank of the target image in the result set and the execution time of the query. Five querying metrics were compared: wavelet only, color histogram only, hybrid (WHASSUPP), naïve hybrid, and average rank.

### 4.1 In-Memory

The graph below shows the data we collected from 10 trials of the in-memory test.



Using the same metric as Jacobs et al, we calculated the percentage of queries (Y-axis) that returned the target image within the top X percent of the query result set (X-axis) [1].

From the graph we see that our hybrid algorithm edges out the wavelet method and outperforms the color histogram method in terms of accuracy. Average rank shows mediocre performance for the most part, but ends up having the top queries with the target image in the top 10% of the result set.

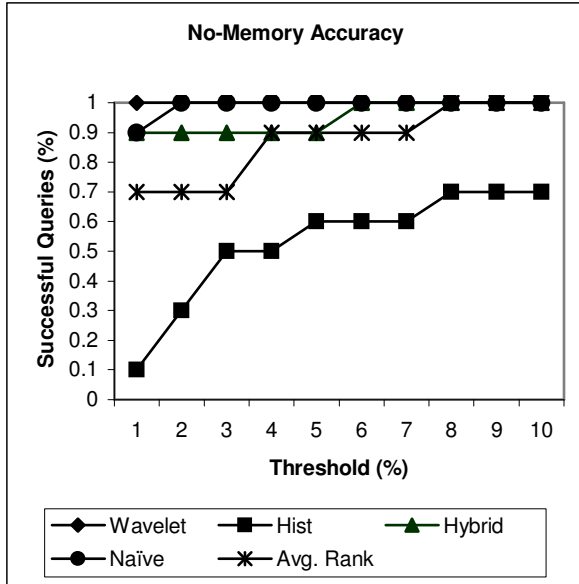
Target image rank statistics for each scoring method are shown in the below below.

Scoring Method	Mean Target Rank	Standard Deviation	Confidence Interval (90%)
Wavelet	124.0	162.5	6.5
Hist.	324.4	281.8	11.2
Hybrid	115.8	136.9	5.4
Naïve	165.8	169.7	6.7
Avg. Rank	117.7	116.3	4.6

*Target image rank statistics for in-memory trials*

### 4.2 No-Memory

The graph below shows the data we collected from 10 trials of the no-memory test.



In contrast to the in-memory tests, the no-memory tests show the superiority of wavelets for undistorted sketches. Color histograms showed some improvement from the in-memory trials, but were not nearly as accurate as any other method.

Target image rank statistics for each scoring method are shown in the table below.

Scoring Method	Mean Target Rank	Standard Deviation	Confidence Interval (90%)
Wavelet	2.8	2.8	0.1
Hist.	148.1	197.0	7.8
Hybrid	8.0	16.3	0.7
Naïve	5.6	5.6	0.2
Avg. Rank	15.9	25.0	1.0

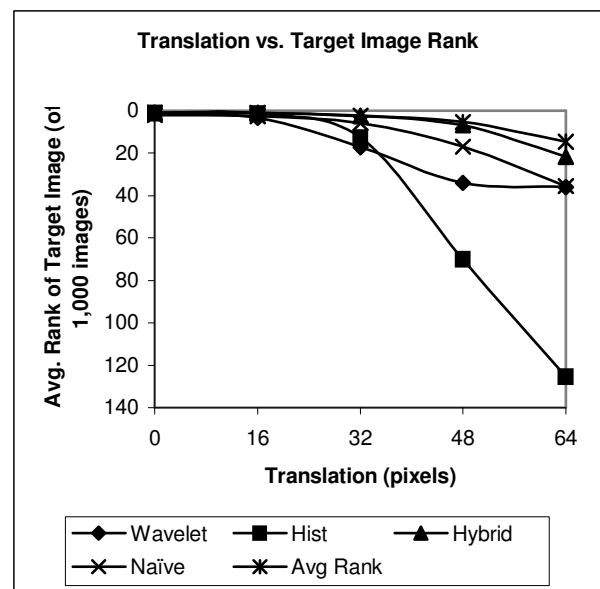
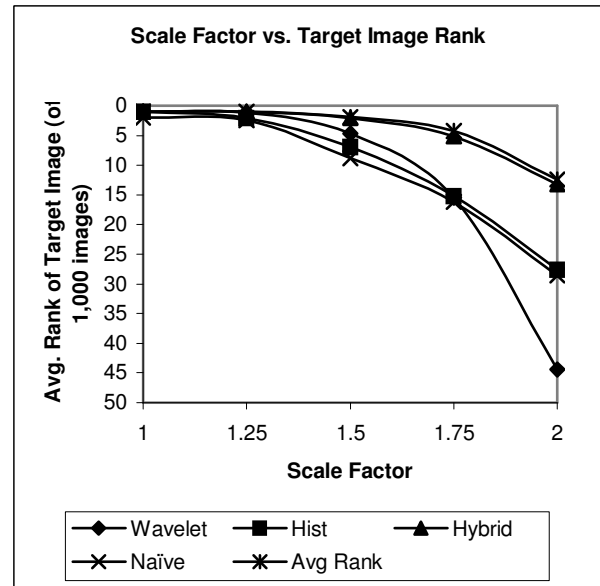
*Target image rank statistics for no-memory trials*

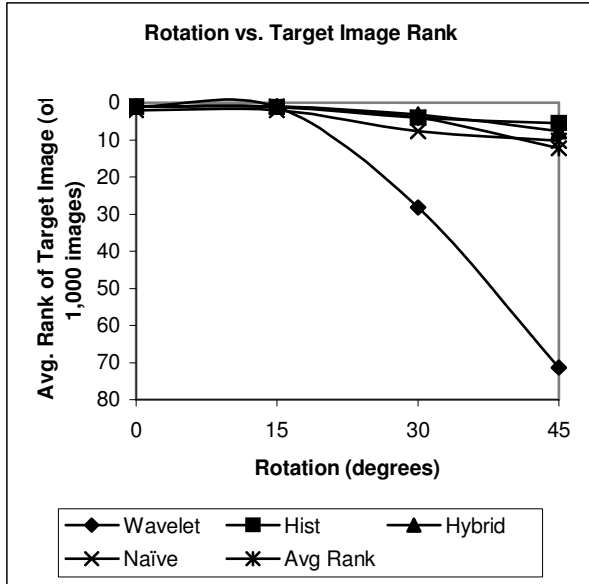
### 4.3 Linear Transformations

To test the robustness of our algorithm with respect to distortions of the query image, we performed three types of tests modeled on those done by Jacobs et al [1]. Ten images were randomly selected from the database. In the first test, each query image was scaled by a factor  $s$  ranging from 1 to 2 in increments of 0.25. In the second test, each query image was translated by a distance  $d$  ranging from 0

to  $\max(\text{width}, \text{height})/2$  pixels in increments of 16 pixels. In the third test, each query image was rotated by an angle  $r$  ranging from 0 to 45 degrees in increments of 15 degrees.

The graphs below show the results of these three tests. The average rank of the target image (Y-axis) is plotted against the severity of the chosen distortion (X-axis).





We have decided to use “inverted” axes in our graphs so that the direction of degradation is down, and thus more intuitive. Keeping this in mind, methods that are robust to distortion will stay relatively horizontal, falling less than other methods.

The scaling tests show that the hybrid and average rank methods are significantly more stable than the rest; wavelets do the worst, as expected. The translation tests, however, are somewhat counterintuitive, with color histograms exhibiting the worst performance. We conjecture that this unexpected observation is due to our method of testing robustness in which we pad any “blank” regions resulting from translation with the average color of the image. Although we do this in an attempt to reduce distortion (adding more of the average color will not change the average color of the query image), we believe that images with multimodal color distributions are problematic. Unlike unimodal distributions, in which the mean and the mode are usually approximately equal, Multi-modal distributions can have several modes, and thus increasing the frequency of the mean “bucket” can severely distort a color histogram. Indeed, of our test images, we found that those with unimodal color

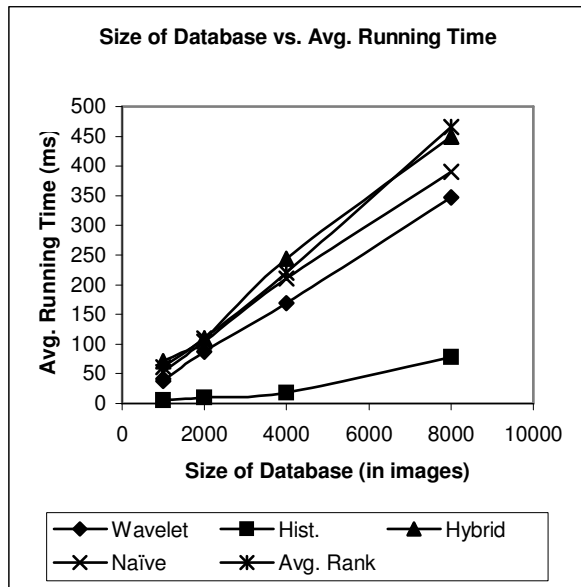
distributions resulted in better target image ranks than those with multimodal distributions. Nonetheless, our hybrid scheme performs among the best, only slightly less robust than average rank. In our rotation test, we discovered that nearly all methods performed approximately the same except for wavelets, which did very poorly. The method of color histograms only slightly edged out the hybrid method. The aforementioned problem of unimodal vs. multimodal distributions was less prevalent in the rotation trials, due to (1) less image padding and (2) less loss of crucial image details. (The center of an image usually has more details than the corners. In the translation trials, the center was moved from the canvas, whereas in rotation trials, only corners disappeared.)

The tables below give target image rank statistics for only the “worst-case” distortions (i.e., scale factor of 2, translation of 64 pixels, or rotation of 45 degrees).

Scoring Method	Mean Target Rank	Standard Deviation	Confidence Interval (90%)
<b>ScaleFactor=2</b>			
Wavelet	44.4	99.2	3.9
Hist.	27.6	29.7	1.2
Hybrid	13.2	18.2	0.7
Naive	28.6	38.0	1.5
Avg. Rank	12.4	18.5	0.7
<b>Translate=64</b>			
Wavelet	35.9	43.7	1.7
Hist.	125.4	155.4	6.2
Hybrid	21.7	19.5	0.8
Naive	35.6	35.3	1.4
Avg. Rank	14.6	11.2	0.4
<b>Rotation=45</b>			
Wavelet	71.4	156.2	6.2
Hist.	5.5	6.2	0.2
Hybrid	7.6	17.1	0.7
Naive	10.2	12.7	0.5
Avg. Rank	12.2	31.2	1.3

#### 4.4 Performance Analysis

To test the performance of each scoring algorithm, we ran 10 queries on databases of 1000, 2000, 4000, and 8000 images. The query images we used were the same 10 that had been utilized for the linear transformation tests. All tests were performed on a Pentium II 400 with 192MB RAM running Win98SE. The graph below shows our results.



As expected, there is a linear increase in the running time of each scoring algorithm. (Color Histograms gives the appearance of being exponential only because the scoring algorithm ran too quickly for the Java clock function to record a time on many of the smaller database trials.)

## 5. Discussion and Future Work

Given the data from our testing, we conclude that the hybrid method is indeed more effective than either scoring method alone. The concept of weighting each of the scores by their relative proportionality worked rather well because each method does well in different situations. By giving each method importance based on how high the image score is allows us the freedom of having asymmetric rankings between both methods. For example, if the wavelet method returns a

ranking of 15 for an image and color histogram 351, our experiments show that the hybrid's ranking is *always* significantly closer to the higher score, say 22. An important result derived from this is that if both methods do well, in many instances hybrid will do even *better*, returning a rank greater than what is calculated from either method alone.

Perhaps the most shocking result, however, was the impressive performance of the average rank scheme. It surpassed our expectations and in some cases exceeded the performance of our hybrid scheme. Overall, the two schemes are comparable in terms of robustness to distortion and overall accuracy, and thus further testing will be needed to differentiate the strengths and weaknesses of the two.

We were initially disappointed by the performance and accuracy of the color histogram method alone, and even more so since it directly affected the effectiveness of the hybrid method. We believe that poor performance was mainly due to (1) the initial use of RGB space in color histograms and (2) the introduction of too many shades of gray onto the fixed palette of colors. Although we used YIQ in the wavelets method, we did not use it initially for color histograms because of the incorrect assumption that a fixed palette of colors would make image comparison very easy and accurate. Indeed, we discovered that human perception really does NOT correspond well to Euclidean distance in RGB space. Hence, we transformed all our RGB-dependent code to the YIQ-space, where the Y component, as in the wavelet method, refers to the black-whiteness of an image (i.e., the center diagonal from (0,0,0) to (1,1,1) in RGB color space). The accuracy of the color histogram method improved drastically and thus buoyed the effectiveness of the hybrid method.

In terms of execution time, we were pleased with the performance of our hybrid scheme.

Java is not well suited for computationally intensive applications, and yet, even for a database of 8000 images, all queries ran in under 1/2 second. Initially, we were afraid that our algorithm might run unbearably slow. However, once we preloaded histogram data into memory, execution times plummeted.

## 5.1 Future Work

**Hybrid vs. Average Rank** – due to the surprising effectiveness of the average rank scheme, more testing is needed to distinguish between them.

**Database tuning** – we would like to try our scheme on other commercial systems to see if there is any performance boost.

**Dynamic databases** – right now, all data from the database is loaded into memory at client execution time. In a commercial implementation of WHASSUPP, it might be desirable to have a system by which data can be refreshed without restarting the program or incurring a huge performance hit.

**Region-based color histograms / dynamic programming** – WALRUS achieved a high level of success with wavelets by using dynamic programming to compute region-based signatures of an image [4]. It would be interesting to see if we could improve the performance of our color histogram metric by computing regional signatures rather than just a single signature for the entire image.

**Daubechies wavelets** – Wang et al showed that using Daubechies wavelets as the basis for wavelet decomposition often achieves higher accuracy than the Haar basis [2]. However, they compare solely real images, not user painted queries, which have much coarser resolution. It would be interesting to see how Daubechies wavelets would perform on these cases and if some combination of the two bases is possible.

**Palette tuning** – Fixing the palette to 24 colors resulted in tremendous performance and accuracy increases for the color histogram method. We would like to try and find the optimal 24-color palette. Perhaps the best palette can be derived from the images in the database.

## 5.2 Conclusion

In general, when searching a multi-dimensional space, algorithms tend to become exponential and computationally intractable. The most widely used solution is dimensional reduction, which is effectively what our indexing scheme does. We reduce an infinite-dimensional space of wavelet coefficients to 60, and then furthermore we quantized these 60 coefficients to +1, -1. Similarly, we partition YIQ space into a finite number of unequal “chunks”. And as a final step, we apply weights to the wavelet coefficients giving greater emphasis on those that describe lower resolution images, based on the observation that user-painted queries are indeed very coarse in detail. In summary, our solution to the multi-dimensional indexing problem is to find the right basis and apply a set of weights that emphasize the most important coefficients (these are specific to the problem at hand and are found by observation, the general case is intractable, of course).

We set out to develop a system that would solve the problems inherent in wavelet decomposition and color histogram methods. Results indicate that WHASSUPP can be that system. Our hybrid-scoring scheme maintains the accuracy of wavelet methods while providing the robustness to distortion that wavelet methods lack. WHASSUPP is thus a successful unification of two disparate paradigms of image retrieval.

## References

- [1] C. E. Jacobs, A. Finkelstein, D. H. Salesin, Fast Multiresolution Image Querying, *Proceedings of SIGGRAPH 95, in Computer Graphics Proceedings, Annual Conference Series*, pp. 277-286, August 1995.
- [2] James Z. Wang, Gio Wiederhold, Oscar Firschein, Sha Xin Wei, "Content-based image indexing and searching using Daubechies' wavelets," *International Journal of Digital Libraries(IJODL)*, 1(4):311-328, Springer-Verlag, 1998.
- [3] James Z. Wang, Gio Wiederhold, Oscar Firschein, "System for screening objectionable images using Daubechies' wavelets and color histograms," *Interactive Distributed Multimedia Systems and Telecommunication Services*, Ralf Steinmetz and Lars C. Wolf (eds.), Proceedings of IDMS, 20-30, Springer-Verlag LNCS 1309, ACM, Darmstadt, Germany, September 1997.
- [4] A. Natsev, R. Rastogi, K. Shim. WALRUS: A Similarity Retrieval Algorithm for Image Databases. Technical report, Bell Laboratories, Murray Hill, 1998.
- [5] "Wavelet Methods for Computer Graphics," S. J. Gortler, PhD thesis, Princeton University
- [6] Graps, A.L.; An Introduction to Wavelets, *IEEE Computational Sciences and Engineering*, Volume 2, Number 2, Summer 1995, pp 50-61.