Understanding Collections of Images

COS 521 Final Project Report

Steven Englehardt, Maciej Halber, Elena Sizikova

January 14, 2014

Contents

Chapter 1

Introduction

1.1 Abstract

This report explores a variety of image properties which make it possible to understand and explore collections of images. In particular, we look at how properties like color distribution, saturation, sharpness, and detail can be extracted and compared between pairs of images. To extract these features we make use of the Fast Fourier Transform (FFT) and Singular Value Decomposition (SVD). We seek to understand how the theoretical underpinnings of these two algorithms affect the way these algorithms transform images and use that to gain an understanding of the fundamental components of an image. Ultimately, we test our understanding of images by creating a concise mathematical notation (a descriptor) from image fundamentals and test how well it differentiates between a collection of images.

1.2 Motivation

There are many possible situations in which we would need to understand and compare image structure. For example, one might like to search for a location in which a photograph was taken, by looking at all the other available images, and finding the image closest to the search image. Alternatively, one may want to cluster images based on their content, and see what categories the image collection can be decomposed to. Both of these would be easier problems to solve, if the images were annotated with words; textual search is a well-solved problem. However, when the images are not labeled and the collection size is extremely large, it is impractical to label by hand. For such problems, it is important to have a model to summarize image content automatically.

Existing methods of image search by analyzing content of the image include Google Goggles and Google Image Search, both are based on similar technology [1], which checks for distinctive points, analyzes lines and textures to create a mathematical model of the image. While the exact implementation is not available, and hence we cannot begin to analyze the performance of this method, Google Image Search does not provide clustering capabilities or the possibility of analysis of an input dataset.

A more relevant study is that of Oliva and Torralba [2] which creates a descriptor, also known as the GIST descriptor, and uses perceptual dimensions (e.g. naturalness, openness, roughness, expansion, ruggedness) to classify

natural images (e.g. pictures of coasts, mountains, or cities). The author's work with a low dimensional representation of a scene is collectively known as the *Spatial Envelope*. The properties of the spatial envelope are estimated by means of Discrete Fourier Transform (DFT), Windowed Fourier transform (WFT), as well as the image's spectral properties estimates. The concise representation of an image using few well understood parameters served as an inspiration for our investigation in this project, by analysis of the image properties using FFT and SVD we tried to discover parameters that allow us to organize collections of images.

In our work we mainly focus on looking at image Color Distribution, Frequency Transform, and Singular Value Decomposition (SVD). As mentioned above, analysis of the frequency domain in images has yielded very well-performing descriptors, so naturally we decided to investigate the way such descriptors can be obtained. Similarly after seeing how well the SVD performs for image compression we have hoped to obtain a low-dimensional descriptor that could be used for image comparison.

There is a huge amount of work being done in the content-based image retrieval, as seen by the sheer amount of available surveys [12][13][14]. In general the overall approach is to compute an image descriptor based on the color, texture and shape information, and then use these vectors in to compute similarity. As seen in aforementioned surveys there is a lot of variation in terms of how the features are computed, what are the exact features we look at (which might vary depending on the application) and then how the results are aggregated and used for analysis. In our work we have used a similar approach where we compute vectors according to a set of detected features, and then use the resulting vectors for similarity comparisons. However due to time limitations we did not focus on the shape features [11], but more on the global image structure that we can understand by doing analysis of Fourier Transform and SVD. With that in mind we used a dataset distributed with [2], which contains a set of images of exterior environments. This dataset does not contain objects, which would clearly need the use of some more involved descriptors.

Chapter 2

Methods

In this project, we analyze image structure from three directions: color distribution, frequency distribution, and geometrical structure. An analysis of image color distribution is given in section 2.3. The properties of images after transformation to the frequency domain by FFT are explored in section 2.4. Images decomposed with SVD are studied in section 2.5, to discover underlying geometrical structure.

2.1 Data

The dataset that we have used for exploring image structure has been provided by Oliva and Torralba [2]. The dataset contains 2600, 256x256 color images labeled as one of 8 outdoor scene categories: coast, mountain, forest, open country, street, inside city, tall buildings and highways. The images in the dataset are varied in complexity: from very simple, planar scenes (e.g. a ground and sky) to very complex and busy images (e.g. cars, people, and buildings in a city). This dataset was chosen because it provides a full range of colors, complexities, and textures, and because image categories allow us to easily pick images with similar physical content.

2.2 Implementation

All of the analysis and implementation for the project was done in Matlab in order to take advantage of the efficient implementations of SVD, FFT, MDS, k-means clustering, k-nearest neighbors, and various image statistics. All algorithms are implemented using the standard Matlab packages unless otherwise noted. Our analysis is structured into the following three steps:

1. We perform an exploratory examination of images to see what attributes provide a good summary of a certain image property (e.g. smoothness, visual complexity, etc).
2. We use a low-dimensional representation of that attribute as an image descriptor.
3. We inspect the performance of that descriptor with respect to accurately measuring visual similarity.

In the following sections we will present our work in the same structure, with a focus on the first and second tasks. In the interest of brevity, we do not include all of the individual comparisons between descriptors that were performed to

determine the best for each desired image property. Instead, we provide a summary of the best performing descriptor for each property and show aggregate results. In the aggregate results we also compare the descriptor performance over the entire image and under a tiled implementation.

When decomposing images using SVD or working with the FFT of an image, we chose to work in grayscale. The choice to do so was motivated by the desire to explore the physical meaning of the decomposition or transformation in the context of images. Though it is possible to separate the red, green, and blue channels and work with each separately, it is difficult to attribute descriptor performance to either relative differences in colors or deeper structure of the image. To do the conversion we used matlab's built-in *rgb2gray* function, which removes hue and saturation information but preserves luminance.

2.3 Color Analysis

2.3.1 Color Spaces

In computer graphics the subset of colors which can be displayed by computer monitor is called *gamut*[6]. There are many models which aim to represent the gamut — one of the most common and well-known such models is known as the RGB model, in which color at every pixel is a combination of three intensity values, of the Red, Green, and Blue base colors, see 2.1. RGB models how devices produce colors, so our initial intuition was to look at models that model human vision more closely.

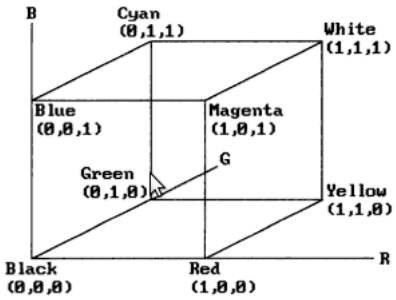
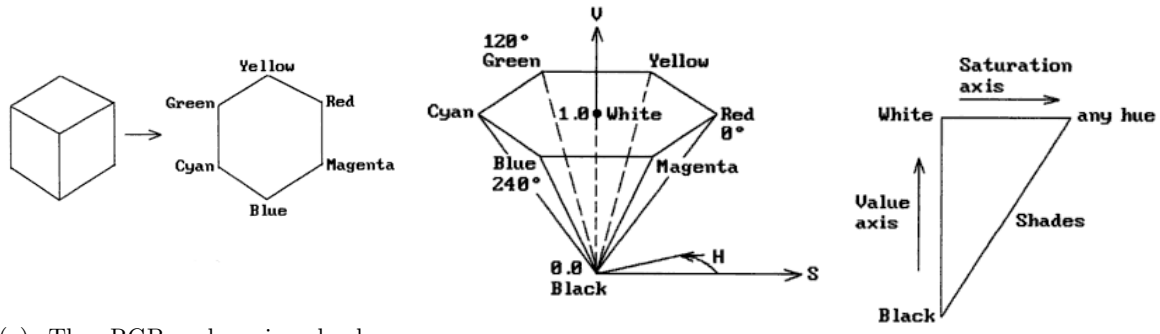


Figure 2.1: RGB Cube Model

One of such model is the Hue-Saturation-Value (HSV) model, which was designed to give more intuitive in terms of human color perception, and thus is widely used as color picker in many photo-editing and digital painting packages [6]. HSV model is a fairly straightforward transformation from the RGB model, as shown in 2.2. In particular, the hue varies from $0°$ to $360°$, and represents the color. The saturation and value are numbers in the range $[0, 1]$ which represent how far is the hue from white and black, respectively.

Another color space which we have considered in this work was the $L^*a^*b^*$ color space which also has been designed to model the human perception closely. L^* parameter control the overall lightness of the image, ranging from $L = 0$ being black, and $L = 100$ yielding white. Both a^* and b^* range from negative to positive values, where negative $a*$ yields, and positive values give colors closer to magenta. For $b*$ negative values give blue color, while positive values relate to yellow. The advantage of $L^*a^*b^*$ color space is that it better preserves color distances, based



(a) The RGB cube viewed along Black-White diagonal

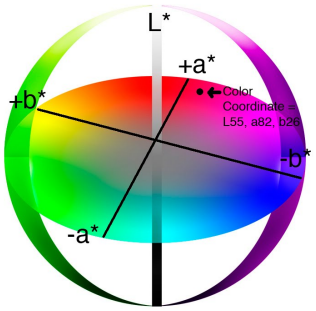
(b) Recoordination

(c) Hue-Saturation Triangle

Figure 2.2: Transformation of the RGB cube to HSV color space

on human perception, i.e. distances between colors in $L^*a^*b^*$ space are close to the distances humans would give after visual inspection. Since this color space aims to be the most complete color model that represents all colors that are perceived by human vision it was a clear choice to investigate what improvements we might get from the incorporation L*a*b* in our color descriptors, which will be introduced in next section.

Figure 2.3: Visualisation of Comission on Illumination (CIE) $L^*a^*b^*$ Color Space, also known as $L^*a^*b^*$ color space. Image obtained from [9].



2.3.2 Color-based Descriptors

The color information is widely used for image retrieval, since as a descriptor it has nice properties of being rotationally invariant, as well as not being dependent on image resolution. As mentioned in previous section, colors closely relate to how we percieve image similarity. Also similar objects will yield similar colors, especially in terms of natural objects (i.e. forest images will always be mostly green, landscapes will always show a blue sky etc.).

A very crude way of comparing images in terms of colors is to average each of the channels in specific color space, and use it as a 3-value descriptor, which will be a point in a color space. Figure 2.5 shows points plotted in $\mathbb{R}^3$. What is interesting about all the color spaces is the fact that we can easily navigate through them and have a good understanding how single parameter affects overall color appearance.

This gave us the inspiration to look for similar, simple parameters that relate to overall image structure, using FFT and SVD (see sections 2.4 and 2.5 respectively). However by a simple averaging we are not obtaining information

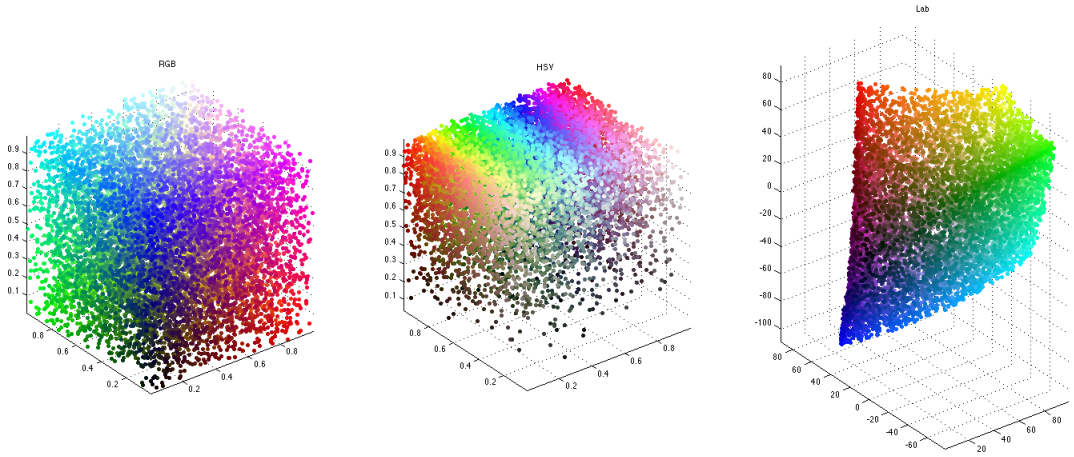


Figure 2.4: Points in $\mathbb{R}_3$, plotted according to their values in RGB, HSV and $L^*a^*b^*$ color spaces

regarding the overall color distribution. A more common way of investigating the color distribution is *normalized color histogram*. To create it we first create bins - given the number of bins n we create a set of uniformly spaced ranges $R = \{r_1, r_2, ..., r_n\}$. Now for each pixel p_i we determine its value v_i and determine into which r_k it falls to and increment value at the relating bin. After visiting all pixels p_i we normalize each bin by total number of pixels. This produces a length n vector that better describes the the overall value distribution. Above structure of course defines the procedure for grayscale images, however it is trivial to extend it to color images where we create a histogram for each of the color channels, and then concatenate resulting vectors.

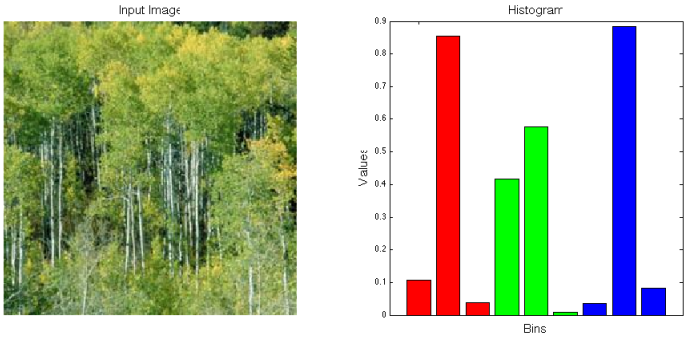


Figure 2.5: Image and its color histogram in RGB, with $n = 3$ bins per channel.

We then designed our descriptor for each color space. As noted in [7], hue is more important for color perception than saturation and lightness, and thus the HSV histograms have been always computed giving more bins to hue channel (i.e. provided that the required length of descriptor is n, the number of bins for hue would be given by $\frac{n}{2}$, while other two parameters are set to $\frac{n}{4}$. Similarly we conjecture that in terms of $L^*a^*b^*$ color space the lightness is less important than parameters describing colors, so we have also modified the number of bins for each channel accordingly ($L^* = \frac{2*n}{8}$,$a^* = \frac{3*n}{8}$, $b^* = \frac{3*n}{8}$)

To sum up our color descriptors, we have looked both at simple averaging of values as well as normalized color histograms. All these have been implemented to work with each of the color spaces discussed in section 2.3.1. In chapter 3 we discuss the overall performance of all descriptors discussed here. Also note that in literature it is

common to note that uniform quantization does not take the spatial relationship between pixels into account, and investigate the techniques to overcome this drawback [8]. However because of time limitation we have decided to settle for a simpler approach.

2.4 Fourier Transform

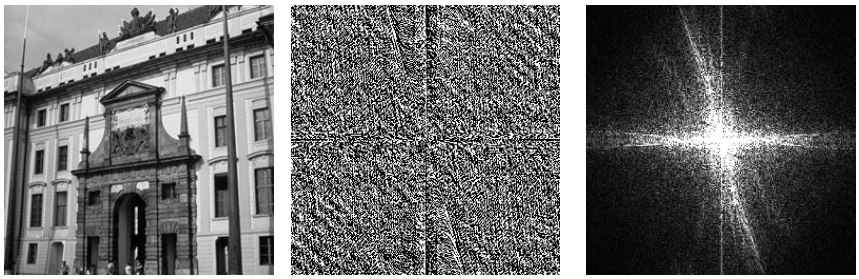
A 2-dimensional, Discrete Fourier Transform(DFT) of an grayscale image is a transformation from the spatial domain to the frequency domain. In the spatial domain, image is represented by function $f(x, y)$ on all relevant points (x, y) in $\mathbb{R}^2$. In the frequncy domain, the image is represented by a function $F(u, v)$ where u and v are frequency values. It follows that in the frequency decomposition, an image is represented by a matrix of complex values F, where $F(u, v)$ encodes the amplitude and the phase of the frequencies u and v. Mathematically, the 2-D Fourier Transform is defined as:

$$F(u, v) = \int \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+vy)} dxdy \tag{2.1}$$

For n points, we can compute the 1-D DFT efficiently in $O(n \log n)$ operations. The Matlab implements 2-D FFT in funcion *fft2*, which is extremely fast, and we had no computational time issues when basing our descriptors on these computations.

2.4.1 Amplitude and Phase Analysis

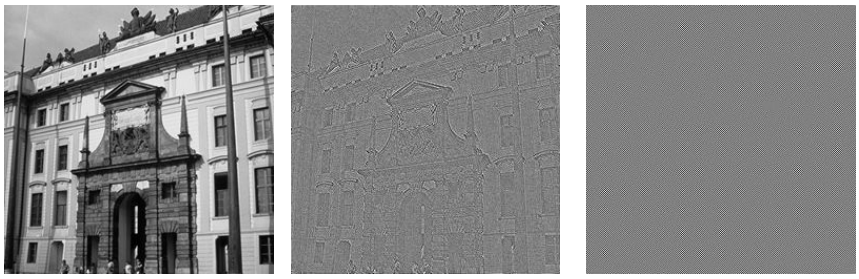
Consider the decomposition of an image of a building in a city from the GIST dataset into its power(amplitude) and phase spectra.



(a) Original image (b) Phase $\Phi(x, y)$ (c) Amplitude $A(x, y)$

Figure 2.6: Phase and power spectra of building image

As Oliva and al. write in [2], the phase image represents local properties of the image. It contains information relative to the form and the position of image components. This can be further understood by taking the true phase values of the image, and setting all the amplitude values to 1 (effectively taking out all amplitude variation and flattening the image), or taking true amplitude values and randomizing the phase:



(a) Original image (b) Flat amplitude (c) Randomized phase

Figure 2.7: Analysis of Contributions of both Phase and Amplitude to an Image

Note that a reconstructed image in which the amplitude information was not preserved retains the information about the edges and outlines in the original image. However, after [2], the amplitude image (power spectra) informs us on global properties of the image structure, telling us about the orientation, smoothness, lenght and strenght of the edges that appear in the image. Two images that are similar might have different local information in terms of contours appearing in them, however they should have similar power spectra. This insight motivates our decision to investigate the amplitude image for our descriptor, since our goal is do capture the global properties of the image.

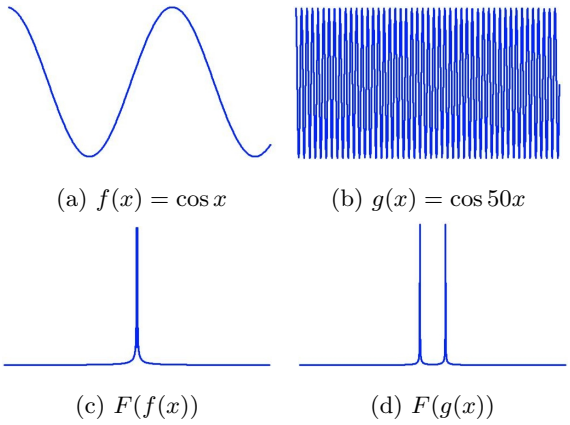
2.4.2 Localization fact

To understand the properties of the 2-D Fourier Transform more deeply, let us first analyze the 1-D variant first. Consider the following functions, and their representations in the Fourier domain (2.8). A plot of $f(x) = \cos x$ is not very well localized in the spatial domain, that is, we can inutitively state that function is streteched along the x-axis. In more mathematical terms we can say that a small change in the function domain δ_x relates to small change δ_y in the function image. However, when we analyze same function in the Fourier domain, we notice that its representation is extremly packed — for $F(cos(x))$, the representation is just a single peak. Conversely, $g(x) = \cos 50x$ is well localized in the spatial domain, i.e. function can be thought of being squeezed along x-axis. Again, more strict definition would be that small change in the function domain δ_x, relates to dramatic changeδ_y, i.e. $\delta_y \gg \delta_x$. However, in the Fourier domain, we observe inverted behaviour - the peaks of the $F(g(x))$ spread apart away from the zero frequency. This simple fact regarding the behavior of Fourier Transfrom suggested us a way of how to describes images in the Fourier domain, analyzing its amplitude image $A(x, y)$. Images containing a lot of high frequency periodicy (buildings, trees) will have more contribution from these (high) frequencies, while images showing open landscapes will mostly consist of low frequencies. Similarly to our approach with color descriptors, we can try to generate bins of different frequencies, and then normalize them to get overall contribution from each frequency range.

To test the above analysis we have created a simple, single-value descriptor, that was just looking at the contribution from the low frequencies - the descriptor $d(I)$ is simply given by :

$$d(I) = \sum_{x^2+y^2 \in r_0} A(x, y)$$

where $r_0 = \{0, \omega_1\}$ is a range of frequencies from zero to some small frequency ω_1. Idea is again that images with lots of high frequency information, will have less contribution from low frequencies, and thus returning smaller value



(a) $f(x) = \cos x$ (b) $g(x) = \cos 50x$

(c) $F(f(x))$ (d) $F(g(x))$

Figure 2.8: Example of localization properties of different functions in the spatial and Fourier domains. Images localized in the spatial domain are not localized in the Fourier domain, and vice versa.

$d(I)$. We sorted the images according the increasing contribution of low frequencies 2.9. The images should be read as one line that starts at top-left corner and ends in bottom-right.

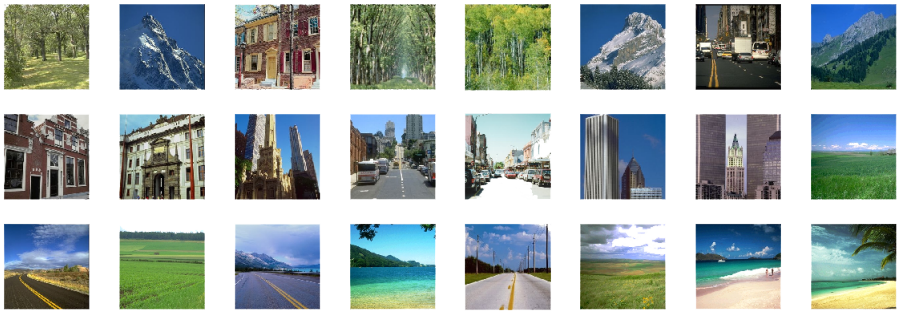


Figure 2.9: Ordering of a subset of the dataset by increasing contribution of low frequencies

One can see that the images on the left side of the spectrum, with a lot of contribution from high frequencies and not so much from low frequencies have many small details: they show leaves, rock incisions on the mountain, and fine building facade. In comparison, the images at the right side of the spectrum are images of open country, roads, and beaches. These images are simple, in the sense that they have a dominant horizon line and not so much small detail. It follows that these images are described mostly by low frequencies, and not by high frequencies. Notice that this analysis discards a lot of information about the distribution of frequency contribution.

2.4.3 Frequency-Based Descriptor

Given that simple descriptor described in the previous section allowed us to create quite meaningful ordering of images we decided to extend it to give us better information regarding the overall frequency distribution, similarly to our color descriptors. The said extension, simply consider a set of disjoint, uniformly spaced, frequency ranges $\{r_0, r_1, ..., r_n\}$, see fig. 2.10.

Our descriptor is then a vector $\vec{d}(I)$, of n-values, where each value is simple averaging of frequency values in

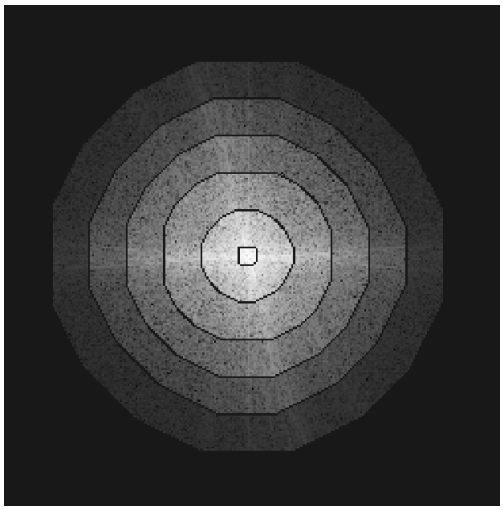


Figure 2.10: Frequency Descripor Visualization

specific bin, that is then normalized by $N = \sum_x, yA(x, y)$, the averaging of all values in an image. This way, each element d_i give us percentage contribution from each frequency range. We will discuss the performance in chapter 3.

2.5 Singular Value Decomposition

A deeper understanding of Singular Value Decomposition (SVD) in the context of images allows the creation of a descriptor that captures overall image complexity in a relatively small descriptor length. SVD is a factorization of any real or complex 2-dimensional matrix. Since images will always be represented by real matrices, we ignore the complex case in our analysis.

Consider an $m \times n$ matrix A. The singular values of A correspond to the non-zero square roots of the eigenvalues from AA^T and A^TA. The matrix AA^T is spanned by the row space of A, and the matrix A^TA is spanned by the column space of A [3]. The row space and column space being the set of all linear combinations of row vectors and column vectors of A, respectively.

SVD conveniently decomposes A, separating out singular values, row space eigenvectors, and column space eigenvectors into three different matrices. The SVD of A is defined as:

$$A_{m\times n} = U_{m\times m} S_{m\times n} V^T_{n\times n}$$

where S is a diagonal $m \times n$ matrix with the singular values of A on the main diagonal (in decreasing order). The columns of U are the eigenvectors of AA^T and the columns of V are the eigenvectors of A^TA [3]. U and V are known as the left and right singular vectors of A, respectively.

The rank of a matrix is informally defined as a measure of the "nondegenerateness" of system of linear equations encoded by that matrix, or more formally is equal to the size of the row space or column space of the matrix. Linearly independent singular vectors correspond to non-degenerate singular values. The rank of a matrix is thus equal to the number of non-degenerate singular values of the matrix, and is also equal to the number of linearly independent vectors in U or in V. If all singular values are non-degenerate, A is a full rank matrix and the SVD of A is unique.

2.5.1 SVD and Compression

It is useful to think of each singular value in S as scaling the row and column singular vectors from U and V to generate an 'eigenimage' [4], or the contribution of that specific singular value to the overall image. This construction is what enables image compression through a low-rank matrix approximation. Let $\tilde{A}$ represent the k-approximation of A, where $rank(\tilde{A}) = k$. Thus:

$$\tilde{A} = U(:, 1:k)\tilde{S}V(:, 1:k)^T$$

(a) Original image (b) Rank 1 to 4 from top left to bottom right

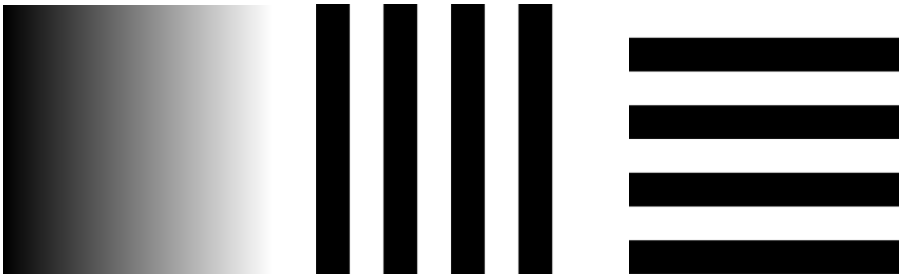
Figure 2.11: Rank k-approximations of an image using SVD

The performance of SVD in image compression shows that there is a significant amount of visual information in low rank approximations of images. Our desire is to capture this low rank information in a concise descriptor for use in image retrieval. An examination of both the singular values and the singular vectors follows.

2.5.2 Singular Values

Image retrieval using all singular values of a matrix has been shown to perform better than a simple distance metric, such as the Mahalanobis distance, Manhattan distance, or Euclidean distance [5]. Several factors contribute to this performance, but relate to the fact that the distribution of singular values in S is connected to the rank of A, the image matrix.

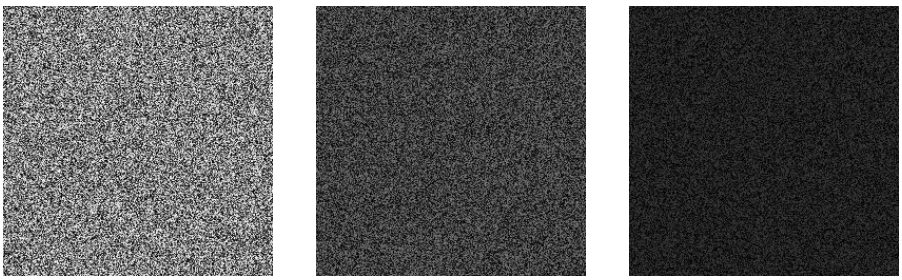
The alignment of an images' strongest color contours to either the horizontal or vertical axis allow it to be fully captured in the left or right singular vectors of an image and thus requires few singular values. The means the image's matrix representation is of lower rank and thus loses less information by ignoring the smallest singular values. In an extreme example, the three images in Figure 2.12 can be fully represented using only the top singular value since they are all rank 1 matrices.



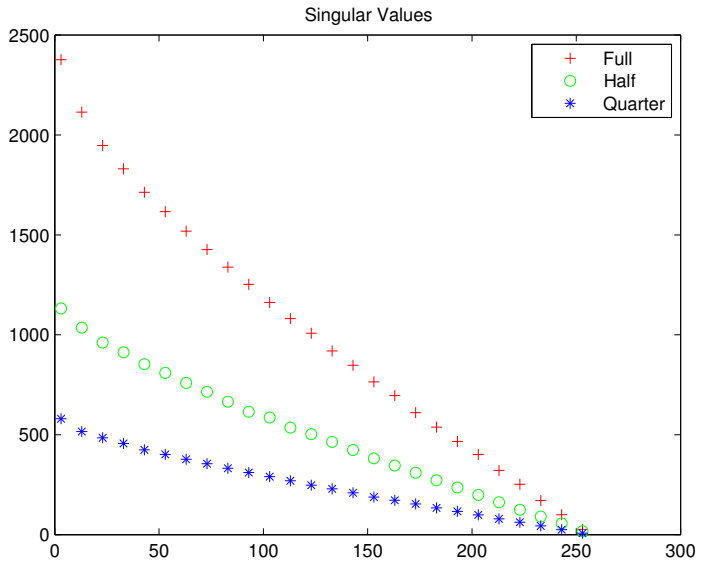
(a) $S(1,1) = 37726$ (b) $S(1,1) = 46041$ (c) $S(1,1) = 46041$

Figure 2.12: Rank 1 images and their top singular value

On the opposite end of the spectrum, consider the randomly generated images in Figure 2.13. These three images are full rank and require all 256 singular values for a full reconstruction. The plot of the singular values from these images reveals that the range of intensity in the image directly correlates to the magnitude of the singular values needed, but has no correlation to the number of values needed.



(a) Full: $0 - 255$ (b) Half: $0 - 127$ (c) Quarter: $0 - 63$



(d) Plot of singular values (1st value omitted for scaling)

Figure 2.13: (a-c): Random images with given intensity range (d): Singular values of (a-c)

Figure 2.12 shows very simple rank 1 images that only require the 1st singular value, whereas Figure 2.13 shows

that full rank images requires all singular values. To get a better understanding of the required number of singular values we look at what happens when there is symmetry in the image. Figure 2.14 shows the plot of singular values from several random images which span the full intensity range from $0-255$, but have symmetry in various directions. In all images, the symmetry is about a central axis, so it covers half of the image.

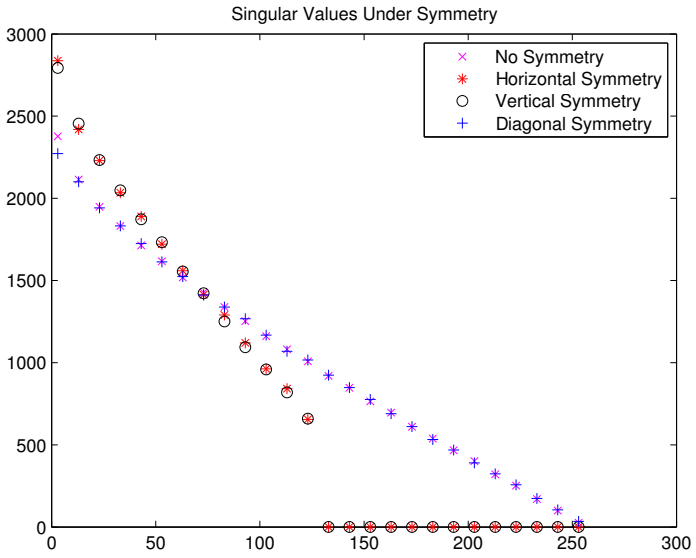
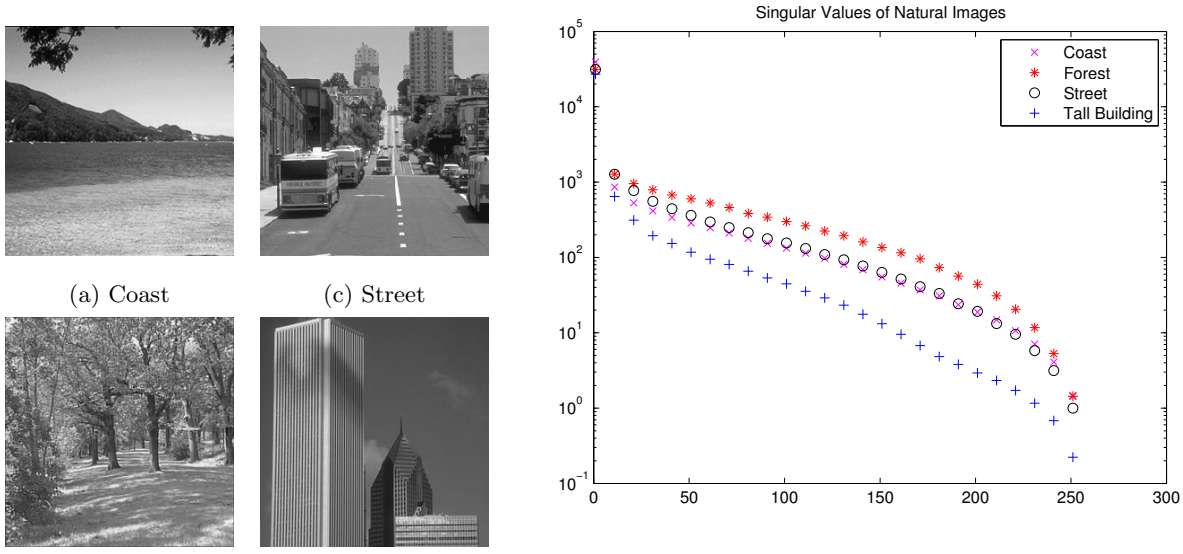


Figure 2.14: Plot of singular values under symmetry (1st value omitted for scaling)

Figure 2.14 shows that the amount of symmetry within an image can greatly reduce the number of singular values required for reconstruction (in this case by half, since half of the image was repeated). The direction of symmetry along the horizontal or vertical axis has very little affect on the distribution of singular values. However, an image which has symmetry off-axis at a $45°$ angle will still have linearly independent vectors in U and V as the image is still full rank. Despite the fact that half of the image is repeated, the distribution of singular values is nearly identical to the image with no symmetry. It should be noted that $45°$ is chosen because it represents the worst-case off-axis example. Angles above and below will experience some level of symmetry in either the horizontal or vertical direction.



(a) Coast (c) Street

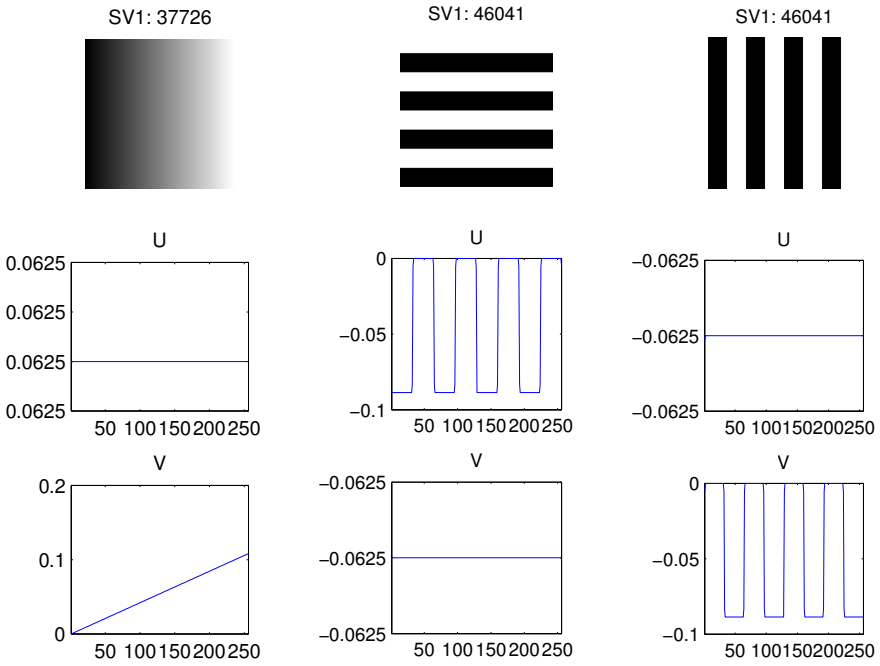
(b) Forest (d) Building (e) Plot of singular values (log-scale)

Figure 2.15: Examination of singular values for natural image classes

Figure 2.15 shows a sample of the distribution of singular values for natural images which vary quite a bit in terms of complexity and repeated structure within the image. Images with high repeated structure aligned to the horizontal (a,c) or vertical (d) direction decay more quickly than images with less repeated structure (b). Despite being similar in the repeated structure of the image, the coast and street images are quite different otherwise but have a very similar distribution of singular values. The detail provided by the respective eigenvectors will be helpful in creating a descriptor which differentiates these two images.

2.5.3 Singular Vectors

The left and right singular vectors, given by U and V are vectors in the row and column space of the image A. Since the singular values in S are ordered from highest to lowest, the vectors of U and V are conveniently ordered from those with the highest contribution to those with the lowest. In order to get a better understanding of the singular vectors we examine the same low-rank images from Figure 2.12.



(a) Gradient (b) Vertical (c) Horizontal

Figure 2.16: 1st singular value composite image (top), left singular vector (mid), and right singular vector (bottom)

Figure 2.16 shows the first composite image that results from the multiplication of the first singular value by the first singular vectors. Since these images are rank 1, all other singular values are 0 and are not shown. Variation of color in the horizontal direction is caused solely by singular vectors from V and variation in the vertical direction is caused by singular vectors from U. For low-rank matrices, the use of singular vectors as descriptors would preform well and would provide additional information above singular values (orientation of color change). However, similar to the problems summarized in Figure 2.14, off-axis color change leads to much different results.

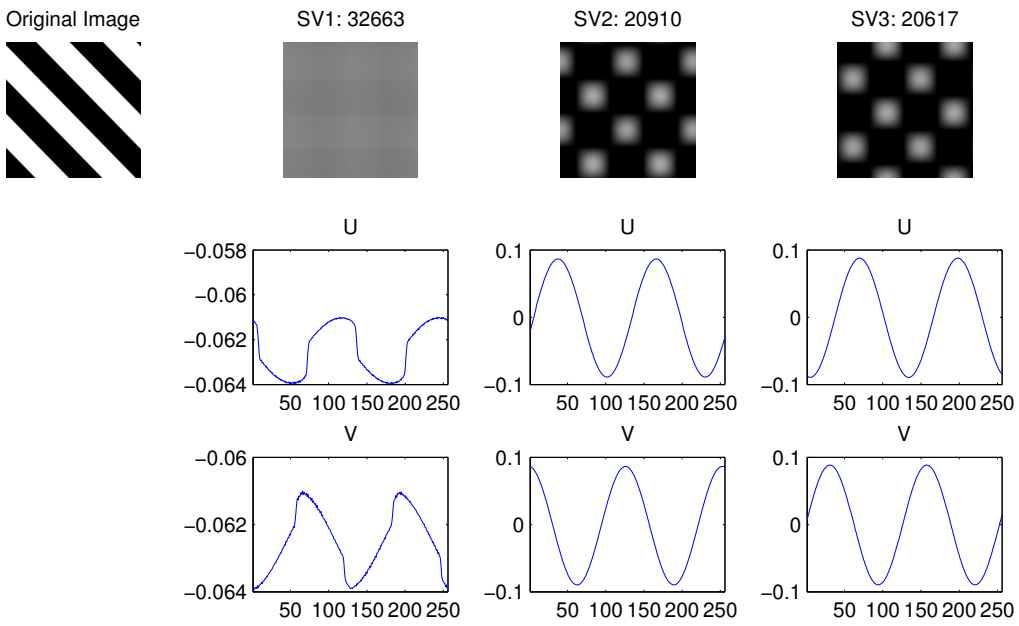
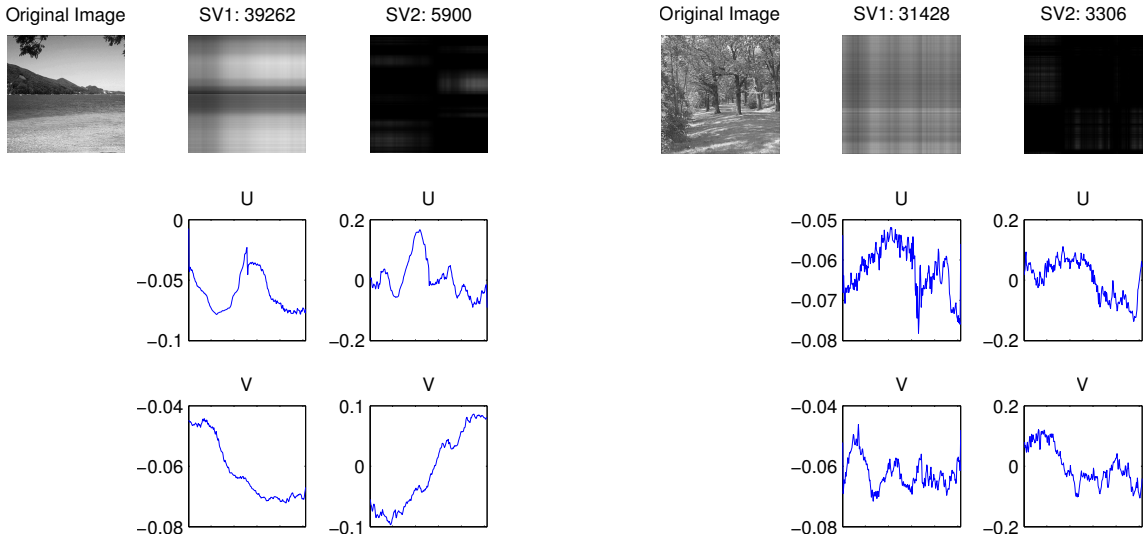


Figure 2.17: singular value composite image (top), left singular vector (mid), and right singular vector (bottom)

Figure 2.17 shows an example of worst case off-axis color change. Since color change does not align with either axis, the matrix is full rank and the decomposition requires all singular values. This case shows why a direct use of singular vectors to compare images (or singular values) is difficult. Although this image is arguable similar to the horizontal and vertical line images in Figure 2.16, it has a much different decomposition.



(a) Coast

(b) Forest

Figure 2.18: A comparison between top singular vectors and values of natural images

Figure 2.18 shows the top contributing singular vectors for a pair of natural images. The benefit of using singular vectors over singular values is again made clear here. Although the distribution of singular values is rather close between the two images (Figure 2.15), the singular vectors capture an image's local variation as noise in their contribution. The coast has smooth transitions of color while the forest has sharp and frequent differences. This property gives an overall estimation of the complexity of structure in the image.

2.5.4 A meaningful descriptor

The distribution of singular values has been shown to provide a rough estimation of the amount of symmetry and repeated structure in an image, while doing little to capture more complex local structures. Singular vectors improve upon this to capture local variation as noise in the values of the vectors that contribute the most to an image's reconstruction. Both singular values and singular vectors are skewed by the fact that the decomposition depends heavily on image structure being aligned to the row bases or column bases.

To account for the effects of axis alignment, we simply take an image and rotate it by $45°$ and $-45°$ and concatenate descriptors for both the rotated and non-rotated images. The motivation for this construction being that if there is structure aligned along a line in the image (axis or off-axis), then a $45°$ rotation will bring that closer to (or further from) an axis. Images with no linear structure should see little change from rotation.

To capture local noise in singular vectors we decide to take the FFT of the singular vectors rather than use them directly. This will capture the high frequency components of a noisy vector (such as the forest in 2.18) as well as the low frequency changes of a coastline or street.

Chapter 3

Analysis

3.1 Overview of best performing descriptors

In this section we provide a quick discussion on the descriptors that we have decided to combine together and use as an aggregate descriptor to differentiate between images. The aggregated descriptor $d_a(I)$ is simply a concatenation of normalized descriptor vectors $\{\hat{d_1}(I), \hat{d_2}(I), \hat{d_3}(I)\}$. We are aware of the fact that each descriptor $d_i(I)$ might have a different importance thus in our testing stage we have tried to discover a proper weighting to each component. In sections **??** and **??** we show the performance of aggregated descriptor on our dataset.

@Steve - Once you're done with the performance, check if vector lengths below agree with the ones used in visualization :)

3.1.1 Color-Based Descriptor

In our analysis we have not found the particular space performing much better than any other. One reason for that might be that, referring to the figure 2.5, the HSV and $L^*a^*b^*$ color spaces are non-linear, while our comparison metric was euclidean distance. When comparing colors in these spaces a more sophisticated techniques than just stacking the histograms for each channel [10]. Due to time constraints we have not implemented any of these methods and thus we have simply settled on comparing basic image statistics μ, σ in $L^*a^*b^*$ color space, that is having our descriptor length $|d_1(I)| = 6$. As soon as we use windowed version of this descriptor it was found sufficient to differentiate images of different colors in our database. However we note in chapter 4, that these clearly should be improved in any future work.

3.1.2 Frequency-Based Descriptor

Our frequency descriptor, that looks at the contributions from various bands of frequencies 2.4.3 have been found to differentiate images of different appearance very well 2.9. In final version we have decided to use 32 frequency bands, giving us a descriptor of length $|d_2(I)| = 32$

3.1.3 SVD-Based Descriptor

@Steve - This might be a bit of a bs here, please check

As discussed in section 2.5.4 the SVD based descriptor looks at frequencies of first singular vectors of 3 versions of input image, one being rotated by $45°$ and second rotated by $-45°$. Thus in total we consider 6 singular vectors $\vec{u}_0, \vec{v}_0, \vec{u}_{45}, \vec{v}_{45}, , \vec{u}_{-45}, \vec{v}_{-45}$, and look at their frequency transform, preserving top 30 frequencies. Thus in total we have descriptor $d_3(I)$ of length $|d_3(I)| = 180$.

3.2 Windowed descriptors

For natural images computing the descriptor over the whole image might led to two images with different appearance be placed in our space quite close by, especially for our color based descriptor. Thus we decided, similar to [2], compute a windowed version of our descriptors, where we divide our images into n by m grid (in our implementation we use $n = m = 4$), and compute our descriptor for each of the cells. Then we concatenate the descriptor from each cell to create a new windowed descriptor of length $|d_w(I)| = n \cdot m \cdot |d_a(I)|$. This change improved our results, as shown in section **??**, which will discuss the performance of non-windowed version and in section **??**

3.3 Collection understanding methods

In order to judge descriptor performance we used a confusion matrix as well as unmodified Matlab implementations of the following algorithms: multidimensional scaling, k-means clustering, and k-nearest neighbor searching. A confusion matrix is a simple visualization of pairwise distances between images of different categories. For each of the Matlab algorithms the motivation for using it is given, along with a short description of the version used.

Multidimensional scaling is an algorithm used for visualization high-dimensional pairwise euclidean distances in N-dimensional space. The embedding attempts to preserve some measure of input pairwise distance in a lower dimension. For our purposes, we chose to use Matlab's classic (metric based) multidimensional scaling algorithm for embedding into two dimensional space. This visualization provides an idea of what properties of an image a descriptor is differentiating by. For example, if the spread of images seems to correlate only to image color then we know that the descriptor is heavily biased towards differences in color.

K-means clustering attempts to partition a set of objects into k clusters in which each object belongs to the cluster with the nearest mean. We cluster by the squared euclidean distances between our descriptors. This clustering allows us to quickly see if a descriptor groups images from the same category or similar visual appearance together.

K-nearest neighbors is an algorithm which takes a test object and compares it to objects in a given feature space to return an output based on the k-nearest neighbors within that space. In our case we build the feature space with all the image descriptors and then query into it to return the index of the k-nearest images within that space. Retrieving images visually similar to a query image is the ultimate goal of our descriptor design, so this visualization gives us an idea of how well our descriptors ultimately perform.

3.4 Non-Windowed Performance

In this section we show and discuss the results of our descriptors in terms or k-nearest neighbor query output. As title indicates in this section we will discuss the results using the non windowed version of our descriptors, and in the following sections we will show the same queries using windowed variant.

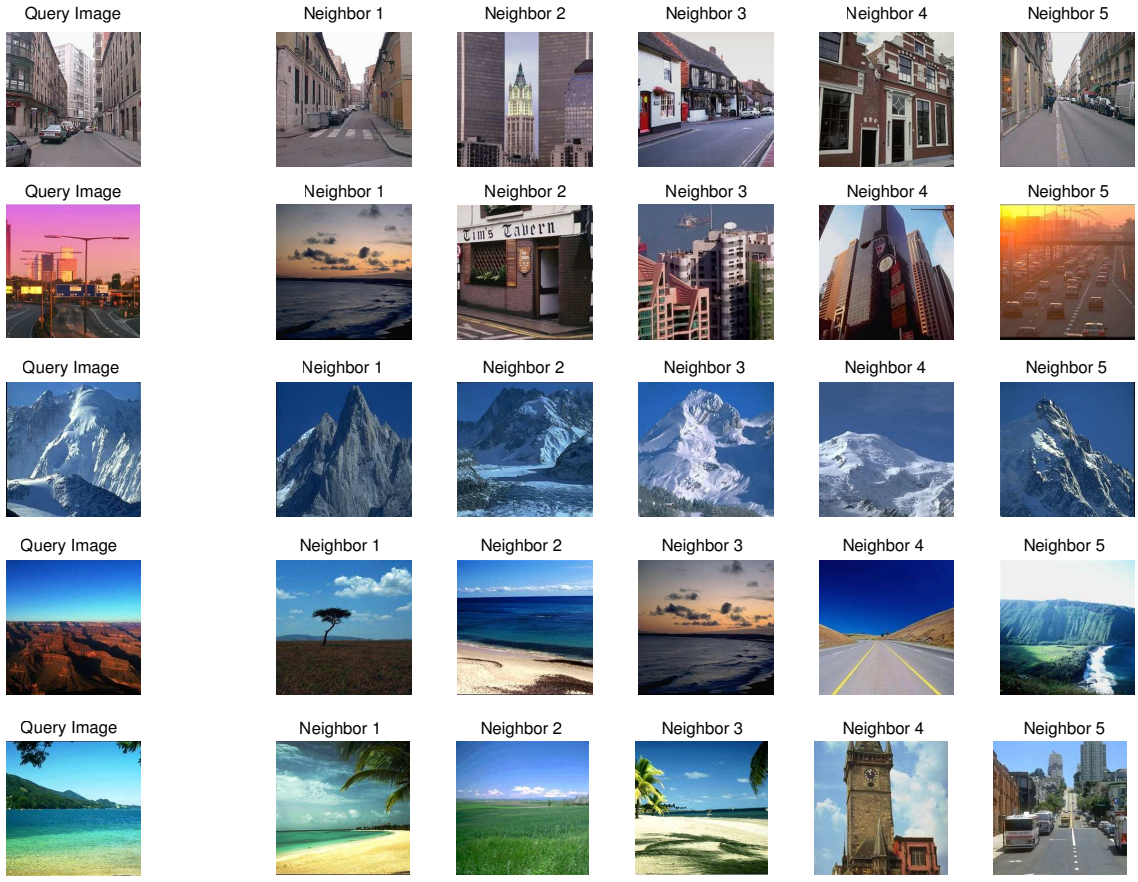


Figure 3.1: A sample of query images and the top 5 neighbors using our best-performing descriptor

Going from the top row of figure 3.1 we can see that for the query image, most of the neighbors are selected correctly also showing a town scenes from down the street. The only exception being the second neighbor, however notice that it does have very close structure (vertical buildings on the sides of the image) and color distribution.

Second row is clear fail case - one reason might be that our database does not really contain much of images with similar color distribution (the sunset sky). Other then that the closest neighbor can be said to have similar geometric structure to the query, with a ground plane, and gradient on the sky. Other images might have been chosen since they have similar frequency transforms with a strong contributions form low frequencies, as well very strong high frequency periodicity (neighbor 2 window crate, neighbors 3 and 4 facades)

Third row show mountains, and all of our results show similar images this is caused by the fact that all mountain images have similar frequency structure, with a lot of complicated texture, as well as similar color distribution, all

having generally blue hint to them.

Fourth row is again somewhat of a fail case. The neighbor 1 can be considered as similar image, but other three show different scenes. Arguably the structure and color is similar for neighbors 2-4, and thus have been selected as neighbors. The last neighbor appearance is less obvious, and is most likely caused by similar horizontal periodicity in both images.

Last row queries for a beach pictures, and two of the neighbors are correct. The second neighbor is having somewhat similar structure, and due to the fact that sea in query image is turquoise, we have got similar picture in structure, however showing a field of grass (turquoise is closer to green than blue). Last two neighbors are clear fail cases, with only somewhat similar color distribution.

3.5 Windowed Performance

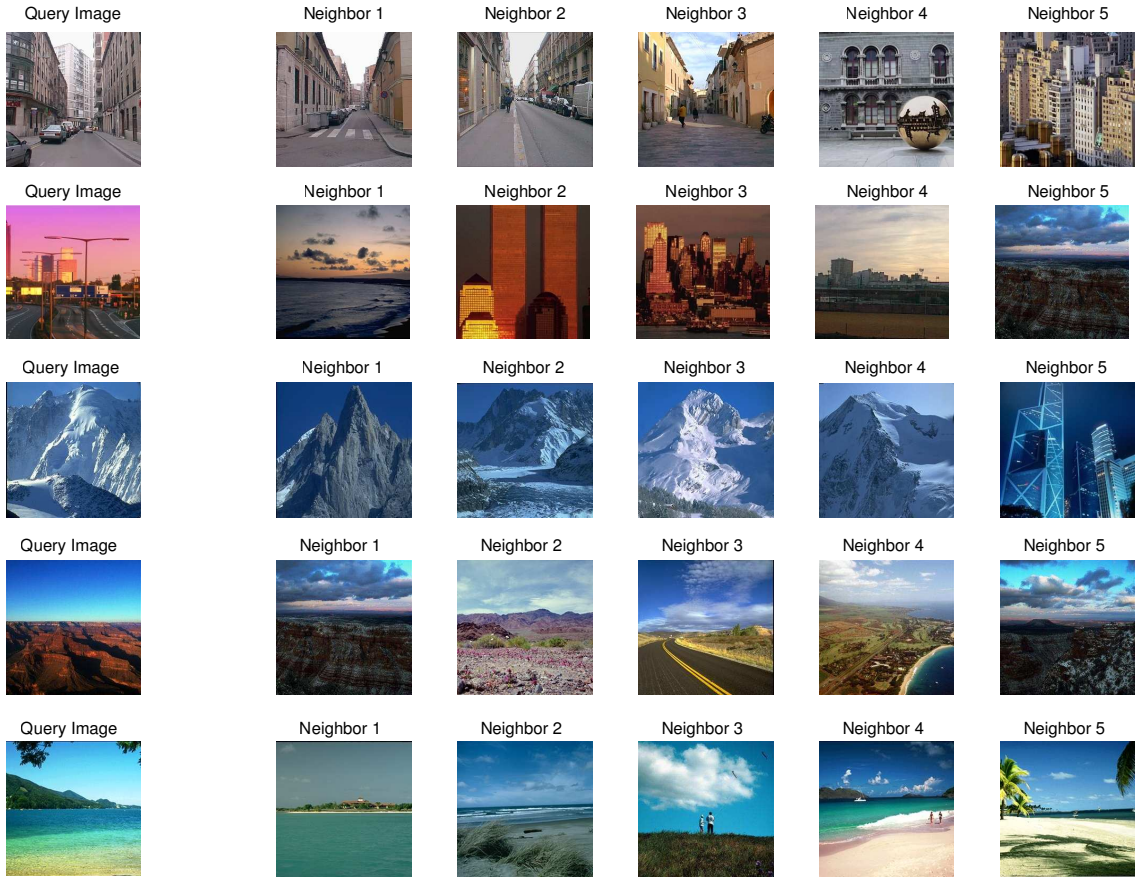


Figure 3.2: A sample of query images and the top 5 neighbors using our windowed descriptor

Again going from the top of the figure we will discuss the querying results for each row, just as in previous section.

We also provide a results of k-means clustering with $k = 6$ and MDS based visualization of our space in the appendix.

Chapter 4

Conclusions and Further Work

Work done during this project was of an exploratory nature, where we have investigated what cues for image understanding can be obtained by analyzing the outputs of FFT and SVD algorithms. We have found that these methods do provide us with useful information when analyzing natural images, and can be easily used in generating meaningful low level descriptors, as shown in previous chapter. Fourier domain informs us about the overall image texture and roughness. Similarly our analysis of Singular Value Decomposition shows that the singular vectors can tell us about the image geometry, and could be used for matching images of similar orientation. Our results show embedding that successfully cluster images of similar nature quite well, however they are clearly not competitive in terms of performance with state of the art algorithms.

In this work we have been working under the assumption that parameters that control image appearance and are of equal importance. Those are clearly wrong and more involved machine learning techniques should be used to learn about behavior of the space of natural images. Also, as mentioned previously, SVD and FT seem to give us similar understanding of an image, with discovering images of simpler versus more complex structure. It would be interesting to investigate the relationship between SVD and Fourier Transform in terms of natural images.

In this is project we have not really focused on the color investigation very deeply, and have settled on comparing images based on euclidean distance between the color values. However HSV and $L^*a^*b^*$ color spaces are non-linear, thus euclidean metric is not giving us the true distance in those spaces. Further investigation of the non-linear distance metrics in HSV and $L^*a^*b^*$ spaces would most likely yield better results.

Our data base provided pictures from 8 categories, and provided labels were describing these categories. However some images from one category might appear very similar to image in other category (i.e. *tall buildings* and *inside city* categories). With that in mind we have decided not to measure performance using the provided labels, but rather be more exploratory, since our focus was to develop understanding the factors that would cluster visually similar pictures togheter. Thus our conclusions are based on the exploration of the embeddings provided by MDS, k-means clustering, and confusion matrices. In any future investigation the database would have to be relabeled to create a subsets of visually similar images (for example by asking users of Amazon Mechanical Turk), and use these labels. Also our dataset has been limited to outdoor categories. It would be interesting to measure the performance in terms of database that also captures indoor environments, as well as pictures of man-made objects.

Bibliography

[1] Johanna Wright, *Search by Text, Voice, or Image*. Inside Search: Official Google Search Blog, 2011.

[2] Aude Oliva, Antonio Torralba, *Modeling the Shape of the Scene: a Holistic Representation of the Spatial Envelope*. International Journal of Computer Vision, Vol. 42(3): 145-175, 2001.

[3] Ientilucci, Emmett J, *Using the singular value decomposition*. Chester F. Carlson Center for Imaging Science, Rochester Institute of Technology, 2003.

[4] Andrews, Harry C. and Patterson, C., III, *Singular Value Decomposition (SVD) Image Coding*. IEEE Transactions on Communications, Vol. 24(4): 425-432, 1976.

[5] Jie-xian Zeng; Dong-ge Bi; Xiang Fu, *A Matching Method Based on SVD for Image Retrieval*. Measuring Technology and Mechatronics Automation, 2009. ICMTMA '09. International Conference on, Vol. 1: 396-398, 11-12 April 2009

[6] Agoston, Max K. *Computer Graphics and Geometric Modelling: Mathematics*. Springer; 2005 edition, ISBN:1852338172

[7] D. Androutsos and K. N. Plataniotis and A. N. Venetsanopoulos, *A novel vector-based approach to color image retrieval using a vector angular-based distance measure*, Computer Vision and Image Understanding, Vol. 75 : 46-58, 1999

[8] Sangoh Jeong, Chee Sun Won, Robert M. Gray *Image retrieval using color histograms generated by Gauss mixture vector quantization*, Computer Vision and Image Understanding Vol 94: 44-66, 2004

[9] Molina, Luis. *LAB Color Space (translated from Spanish)*. 2010 Blogpost: `http://sobrecolores.blogspot.com/2010/03/modo-de-color-lab.html`

[10] Han, Ju and Ma, Kai-Kuang. *Fuzzy Color Histogram and Its Use in Color Image Retrieval*, Trans. Img. Proc., Vol. 11 : 944 – 952, Aug. 2002.

[11] David G. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, Vol : 60, 2 : 91-110, 2004

[12] Shishir K. Shandilya and Nidhi Singhai, *A Survey On: Content Based Image Retrieval Systems*, International Journal of Computer Applications, Vol. 4, 2: 22-26, July, 2010

[13] Ritendra Datta and Dhiraj Joshi and Jia Li and James Z. Wang, *Image retrieval: ideas, influences, and trends of the new age*, ACM COMPUTING SURVEYS, 2008

[14] Liu, Ying and Zhang, Dengsheng and Lu, Guojun and Ma, Wei-Ying, A Survey of Content-based Image Retrieval with High-level Semantics, Pattern Recogn., Vol. 40, 1 : 262 - 282, January, 2007

4.1 Appendix