EEE3032
Computer Vision and Pattern Recognition
Coursework Assignment
Visual Search of an Image Collection

- Add failure case results in appendix
- Explain theory of P-R graphs
- Confusion matrix
- Explain how our categories are level 2, but algs are level 1

Joshua Tyler

April 17, 2016

Abstract

Write this at the end!

Contents

1 Introduction

Large databases of images exist in various fields. Indexing ans searching these using textual descriptors is generally not effective [1, p.657]. Visual search offers an alternative to this, by allowing a user to search an image collection using an image as the query, with the aim being to return images similar to the query. This is achieved by generating and comparing numeric, vector based, image descriptors which can be treated similar to words.

This report presents the implementation of such a system in matrix laboratory (MATLAB). Several different methodologies will be considered, and their results compared and contrasted.

In order to test the system, the Microsoft Research Cambridge Object Recognition Image Database, version 2.0 will be used. This is a freely available image database which is split into 20 rows, with each row containing similar images [2].

1.1 System Design

A common framework was used to apply generate descriptors from all of the images, compare them, and display results. This framework uses a MATLAB function handle to input both the descriptor and distance measure. This method allows the core code-body to be generic, so that in order to run a different test only the external descriptor and/or distance measure functions need to be changed.

A block diagram of the operation of the system is shown in Figure 1.1.

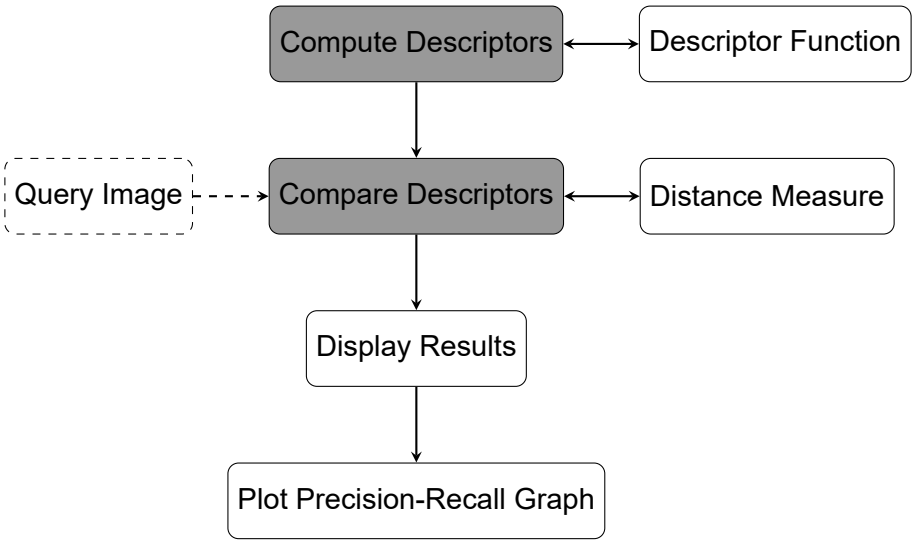


Figure 1.1: System design

1.1.1 Test Methodology

In order to test the system, a unit test has been written for all of the descriptor and distance measure functions. Each of these tests complies with MATLAB's unit testing framework, so can be executed using the `runtests` command.

Test pattern generation

Some tests require test patterns in order to test their functionality correctly. For this purpose a suite of test pattern generation programs was written to create image which contain:

- A uniform colour.
- A uniform pattern.

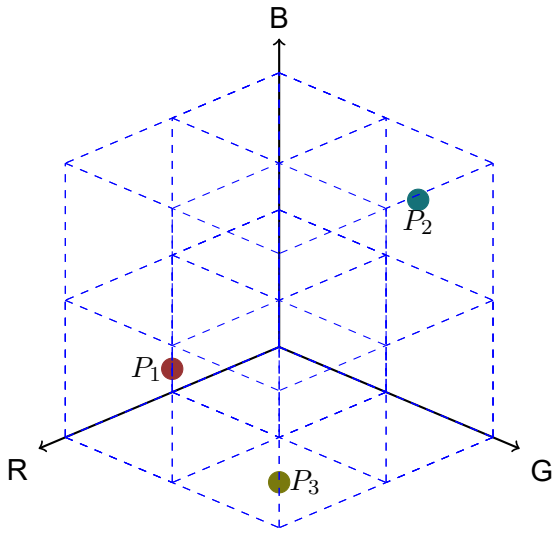


Figure 1.2: Visualisation of RGB histogram

- A combination of the above.
- A grid of several of the above images.

These test patterns can be used both internally to the unit tests, as well as to perform system level testing.

1.2 Techniques Implemented

This section discusses the visual search techniques implemented, along with the theory of their operation and the difficulties which were encountered whilst implementing them.

1.2.1 Global Colour Histogram

A global colour histogram quantises the RGB space into a series of bins. There are Q bins for each dimension, and therefore Q^3 bins for the total image. Each pixel in the image is then sorted into the appropriate bin, and the sum of the bin totals is normalised to 1. This vector histogram, of length Q^3, can therefore be used to describe the image.

The implementation of this function was relatively straightforward, since example code for the function had been provided [3]. The only modifications made to the code were to explicitly state the histogram bin edges. This was necessary to provide a robust descriptor since the provided code normalised the histogram to the range of the input. Therefore an image containing only values in one area of the RGB space, would appear similar to an image with uniform distribution.

The unit tests for this function consist of generating images of known RGB distribution, calculating the RGB histogram by hand for these images, then comparing that to the output of the function.

1.2.2 Texture Histogram

Texture is a concept in computer vision, but is difficult to define. Texture loosely corresponds to a repeating pattern of edge characteristics, examples of which would include grass pebbles, and hair [1, p. 194]. Texture is important because it appears to be a strong indicator of object identity.

The texture histogram function is implemented as the concatenation of two functions: an edge orientation function, and a histogram function. The edge orientation function returns a two dimensional array where each element contains a value corresponding to an estimate of the edge orientation at that location. This estimate is in the range $-\pi$ to $+\pi$, but is normalised to 0 to 1. Weak edges are set to a normalised value of 0.5, corresponding to 0 rad.

In order to perform the edge detection, and edge angle estimation, the function uses the Sobel operator [4]. The sobel operator is formed of two kernels which are convoluted with input image in order to perform edge detection in the x and y directions respectively. The kernels are defined in Equation 1.1.

$$K_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} K_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{1.1}$$

The edge magnitude and angle are calculated as described in Equations 1.2 and 1.3 respectively. It should be noted that the operates in these equations apply on an element-by-element basis, not to the array as a whole. In addition, the K_y value is inverted in order to allow the performance to match the MATLAB library function `imgradient`. This allows for easier testing. The pixels with edge magnitudes above a certain threshold are passed to the histogram generation function.

$$\text{Mag} = \sqrt{{K_x}^2 + {K_y}^2} \tag{1.2}$$

$$\theta = \text{atan2}(-K_y, K_x) \tag{1.3}$$

In order to test the edge detection function, the output is compared to the MATLAB function `imgradient`, for several test images, plus a real image. In addition system level testing has been performed with several test images.

1.2.3 Gridding and Concatenation of Features

Gridding of features allows descriptors to apply be considered locally to parts of an image. The implementation of this is fairly simple in MATLAB. The image can be split using the native array indexing of MATLAB to select a subset of the image array. The descriptor function can then be applied to each of these sub images, and the resultant descriptors concatinated into a matrix.

The gridding functon is tested by feeding a gridded sample image into the function and comparing the output to a manually calculated version of what is expected. A test descriptor function, which returns the image reshaped into a vector array is used to simplify this process.

1.2.4 Distance Measures

In order to calculate the difference between descriptors, the L2 norm is used. This distance measure returns the square root of sum of the squares of the differences between elements of a descriptor, as defined in Equation 1.4, where d_i refers to individual differences between array elements.

$$\text{Distance} = \sqrt{\Sigma({d_i}^2)} \tag{1.4}$$

The L2 norm calculation function is tested by comparing the output of the function to manually calculated values for various test vectors.

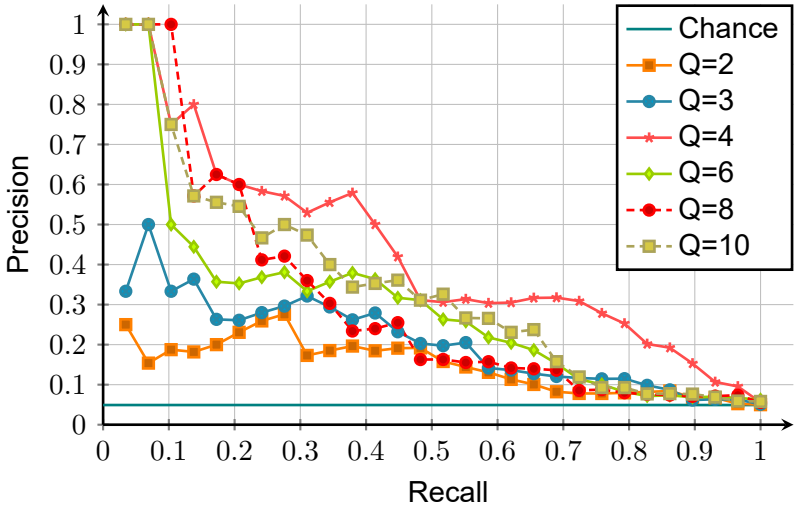
2 Experimental Results

2.1 Global Colour Histogram

The global colour histogram function is implemented as `H = vs_compute_rgb_histogram(img, Q)` where `H` is the descriptor returned, `img` is the image input, and `Q` is the quantisation level for each image dimension. The descriptor is therefore of length Q^3, as discussed in Section 1.2.1.

In order to analyse the effect of quantisation level on performance, descriptors have been generated for various values of Q, and the performance of each value analysed for each test value. Two test images have been chosen, 9_23_s and 13_1_s, and their results are displayed in Figures 2.1 and 2.2b respectively. These images have been chosen because they represent two contrasting examples of what could be presented to the algorithm. Image 9_23_s shows a sheep in a field. This image has two strongly dominant colours: the green of the grass, and the cream of the sheep's wool, and very little other colour content in the picture. On the other hand 13_1_s shows a collection of differently coloured books on a shelf. This image contains many different colours.

(a) Query image

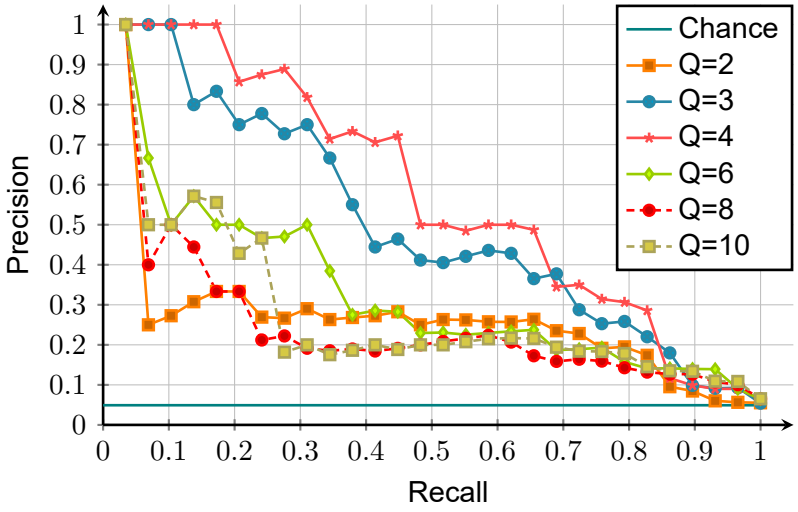


(b) Resultant graph

Figure 2.1: Global colour histogram results for query image 9_23_s

The results shown in Figures 2.1 and 2.2b match expectations suggesting that a very low quantisation value does not deliver good results, because there are not enough bins to differentiate between the images. In addition very high values, such as $Q = 10$ do not work well because there is two many possible bins, and as such nominally similar colours will be sorted into different bins, and so will not be considered similar by the distance measure function. Both test images suggest that a quantisation level of $Q = 4$, yielding 64 bins, gives the best result, since the line joining these data points is closest to the top right hand corner.

(a) Query image



(b) Resultant graph

Figure 2.2: Global colour histogram results for query image 13_1_s

2.2 Gridding

Gridding is implemented as the function `F = vs_grid(img, h_level, v_level, compute_function)`, where `img` is the image input, `h_level` is the horizontal quantisation level, `v_level` is the vertical quantisation level and `compute_function` is the descriptor generation function. This configuration therefore gives us two independent variables, `h_level` and `v_level`.

However, since most of the images in the dataset have a quantisation level of approximately 1.5:1, it makes sense to set the vertical quantisation level to 1.5 times the horizontal quantisation level. Therefore the overall quantisation level, Q can be defined, where `h_level = ceil(1.5*Q)`, and `v_level = Q`.

In a similar manner to that of Section 2.1 the optimum quantisation factor will be determined by computing descriptors for various values of Q and then, using two test images, determining which has best results. For this test, the value of Q for the global colour histogram which prodced the best results ($Q = 4$) will be used. The same test images as used in Section 2.1 have also been maintained in order to allow a performance comparison with the non-gridded image.

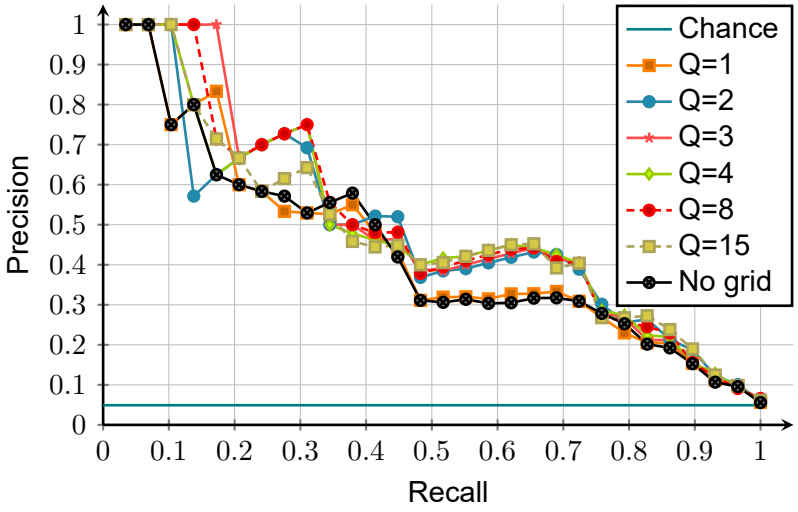
The same effect of increasing Quantisation factor as experienced with the global colour histogram is expected. Increasing the value should increase performance up to a maximum, at this point the performance should decrease with increasing quantisation factor. One reason for this is that tighter grids mean that small movements in the object could potentially result in the image being quantised in a very different way, meaning visually similar images would be distant from each other in the quantised space.

Figures 2.3 and 2.4 show the results of the gridding process.

Figure 2.3 shows little variation across gridding quantisation levels, this is likely due to the fact that the image has only two dominant colours, so all grids have similar colour histograms. Gridding does in the general case show slight performance improvement with this image however. Notably increasing prevision values for higher quantisation levels.

Figure 2.4 on the other hand shows great variation in the result with grid size. Increasing the grid quantisation level to 2 greatly increases the performance up to a recall of around 0.4, and a grid quantisation level of 1 also increases the performance by a large amount also. In addition to this, much higher quantisation steps, such as 8 and 15 show greatly decreased performance. This is due to reasons mentioned previously.

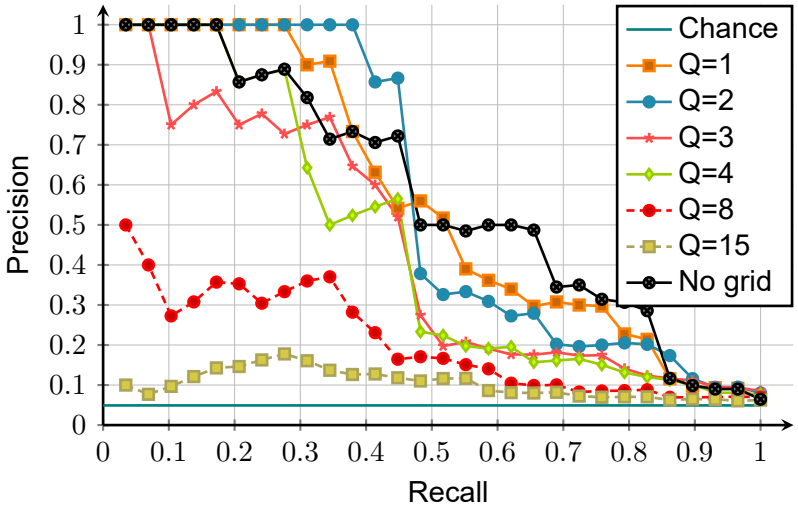
(a) Query image



(b) Resultant graph

Figure 2.3: Gridded global colour histogram results for query image 9_23_s

(a) Query image



(b) Resultant graph

Figure 2.4: Gridded global colour histogram results for query image 13_1_s

2.3 Texture Histogram

The edge orientation function is implemented as `F = vs_edge_detect(img, compute_function, strength)` where `F` is the descriptor returned, `img` is the image input, `compute_function` is the histogram generation function, and `strength` is the minimum edge strength to consider (range 0–1). The histogram generation function is `H = vs_compute_histogram(img, Q)` where `H` is the histogram returned, `img` is the image input and `Q` is the quantisation factor. When `vs_compute_histogram` is used as the `compute_function` in `vs_edge_detect`, the system can produce an edge orientation histogram, with two independent parameters. These parameters determine the minimum strength edge to consider, as well as the quantisation level of the histogram. These parameters will be referred to as E and Q respectively.

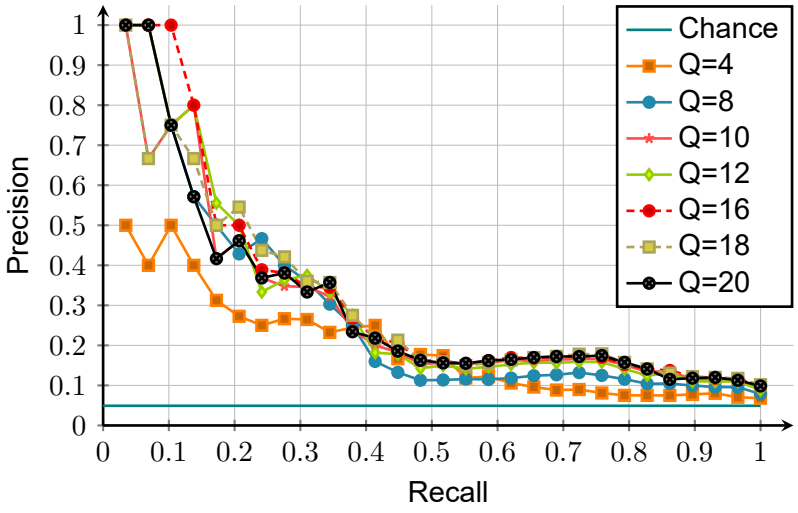
Additionally, since texture is an inherently local feature, the image will be gridded. The descriptor function handle is therefore : `@(x)vs_grid(x,3,2, @(x)vs_edge_detect(x, @(x)vs_compute_histogram(x,Q),` The grid quantisation factor of $Q = 2$ is used, since this was determined to be the most effective for gridding the RGB histogram in Section 2.2.

Evaluating the effect of changing both E and Q together is computationally expesive, and the results are hard to visualise, since a 3 axis plot would be required. Therefore the two variables will

be evaluated separately.

Initially the effect of changing the histogram quantisation level, Q, will be investigated using a constant E of 0. This will leave all edges in the image for the consideration of the histogram

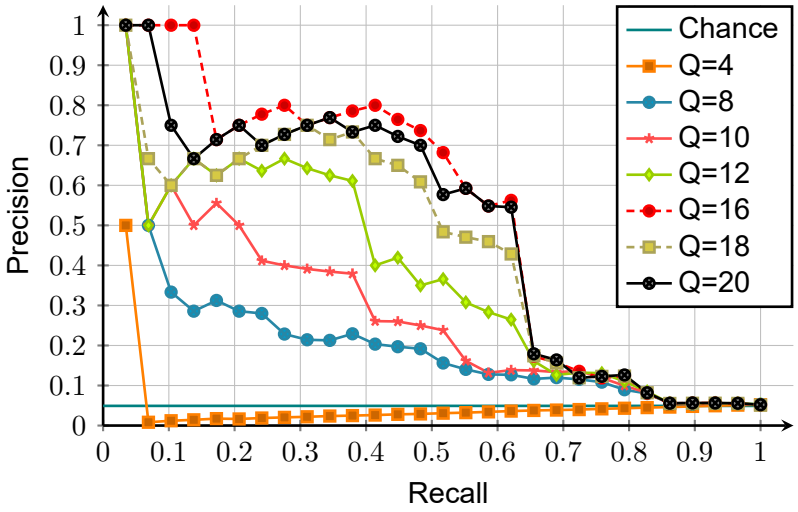
(a) Query image



(b) Resultant graph

Figure 2.5: Gridded texture histogram results for query image 9_23_s, with $E = 0$

(a) Query image



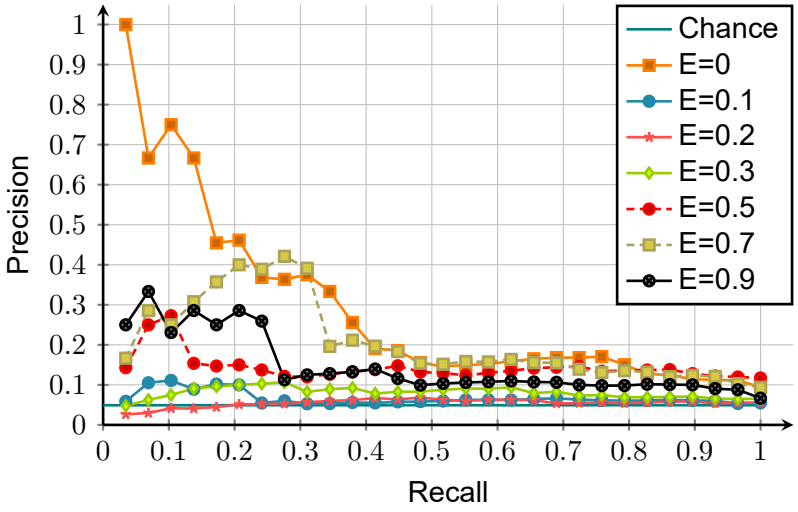
(b) Resultant graph

Figure 2.6: Gridded texture histogram results for query image 13_1_s, with $E = 0$

Figure 2.5 shows that for the sheep image, there is very little difference in performance with quantisation levels greater than approximately 4. Figure 2.6 shows that the books, on the other hand, performance is dependant on quantisation level. Higher quantisation levels continue to improve performance, up to approximately $Q = 16$, after this point performance decreases.

Figures 2.7 and 2.8 show the result of varying E, whilst keeping Q at a constant value of 16. The results are interesting. For the bookshelves, increasing E above zero improves performance up to approximately $E = 0.3$, however, for the sheep increasing E above 0 greatly hinders performance. This is likely due to the face that whilst the books have lots of strong edges (at the edges of the books), which will remain with increasing values of E, the sheep image likely lacks these strong edges, and so increasing E removes almost all of the relevant edges from the generated descriptor.

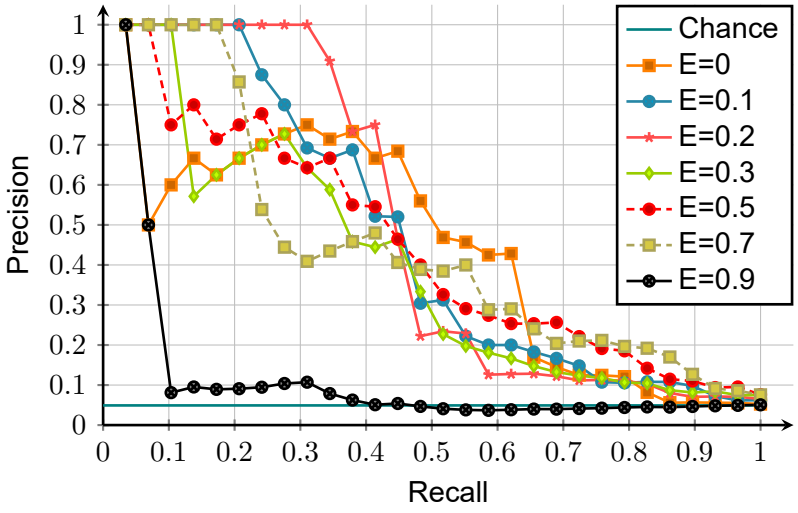
(a) Query image



(b) Resultant graph

Figure 2.7: Gridded texture histogram results for query image 9_23_s, with $Q = 16$

(a) Query image



(b) Resultant graph

Figure 2.8: Gridded texture histogram results for query image 13_1_s, with $Q = 16$

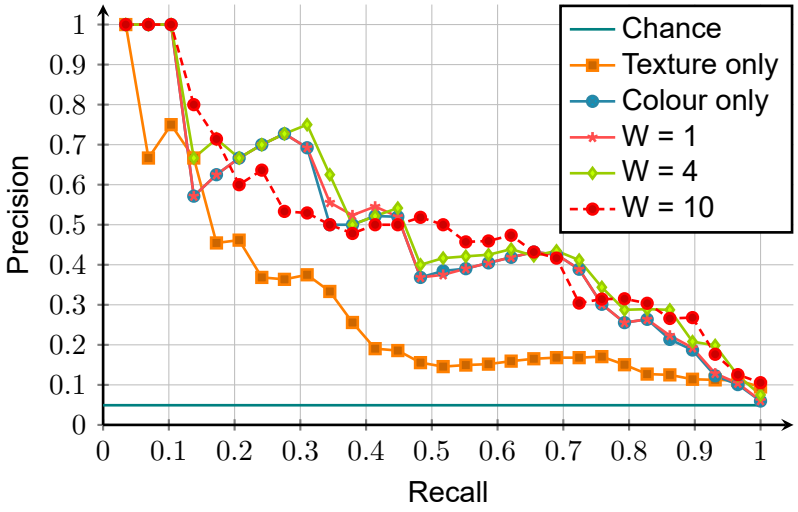
2.4 Concatenation of Descriptors

Sections 2.2 and 2.3 demonstrate the effectiveness of gridded colour histogram and texture descriptors. It stands to reason therefore that these descriptors could be combined to produce a descriptor of even greater effectiveness. This is possible in matlab using the native array indexing interface, but it is clearer to use the `horzcat` function.

A combined descriptor can therefore be invoked using the function handle `@(x)horzcat( texture_func(x) .* M, color_func(x))` where `texture_func(x)` and `color_func(x)` are function handles to the texture and colour descriptor functions respectively, where their single parameter, `x` is the input image. The only other parameter in this function is `M`. `M` is a multiplier in order to adjust the weighting between the texture and colour descriptors. When `M` is 1 the two descriptors are equally weighted, however since the colour descriptor is 4 times as long as the texture descriptor (a 64 bin histogram, vs and 8 bit histogram), it is effectively weighted more highly since there are many more dimensions in which values can be distant. Whilst weighting of $M = 4$ should therefore make up for this difference in descriptor length, a different value may provide better results.

Figure 2.9 shows that the concatenated descriptor remains very close to the plot of only the RGB histogram for all values of W plotted. This implies that all of the descriptors are very close in the

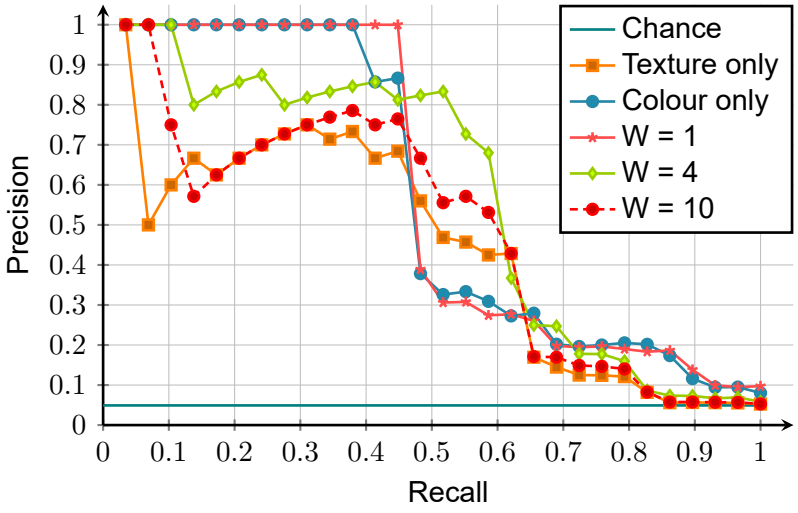
(a) Query image



(b) Resultant graph

Figure 2.9: Concatenated descriptor results for query image 9_23_s

(a) Query image



(b) Resultant graph

Figure 2.10: Concatenated descriptor results for query image 13_1_s

texture plot, leading to the colour plot dominating the results.

Figure 2.10 on the other hand shows that the concatenated descriptor does vary between the two plots with differing values of W. This implies that both descriptors have a large amount of differentiation. The results show that a value of $W = 1$, the results are close to the RGB histogram, but at $W = 10$, the results are closer to the edge orientation histogram. A value of $W = 4$ provides a compromise between the two as expected. The results are poorer than the RGB histogram only for this particular query image, but this is unlikely to hold true for all query images since some images will likely be described by colour well, and others by texture well.

3 Conclusion

Bibliography

[1] D. Forsyth and J. Ponce, *Computer Vision. A modern approach*, 2nd ed. Pearson, 2012.

[2] A. Criminisi, “Microsoft Research Cambridge Object Recognition Image Database, version 2.0,” http://research.microsoft.com/en-us/projects/objectclassrecognition/, 2004, [Online; accessed 09-April-2016].

[3] J. Collomosse, “Lecture notes for EEE3032 – Computer Vision and Pattern Recognition,” 2016.

[4] I. Sobel, “History and Definition of the so-called “Sobel Operator”,” https://www.researchgate.net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator, 2015, [Online; accessed 10-April-2016].

[5] I. Gradshteyn and I. Ryzhik, *Table of Integrals, Series, and Products*, 7th ed. Elsevier, 2007.

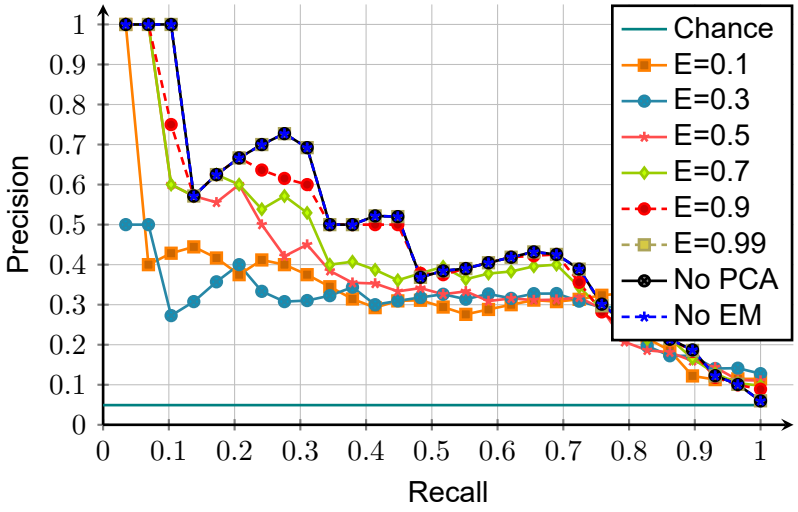
A Principal Component Analysis

Descriptors should be as discriminative as possible, whilst also being as compact as possible. This means that each descriptor should encode encode as much information allowing it to be differentiated from other images as possible, whilst also containing the minimum amount of data allowing it to do this. The descriptors investigated in the main body of the report focus on creating discriminative descriptors, but do not place as much importance on generating compact descriptors.It is possible that many of the descriptors contain information which is common to all images, and therefore acts as wasted data.

priniciple component analysis (PCA) analyses these descriptors and, via construction of an eigen-model, allows for them to be projected into a lower dimensional space. This discards the 'wasted' data in the descriptors.

Figures A.1 and A.2 show the results of projecting the descriptors into a lower dimensional space. The E value refers to the proportion of total eigenvalue energy to preserve. Eigenvalue energy is equal to the sum of all of the Eigenvalues which make up the descriptor. Therefore, for $E = 0.5$, the sum of the eigenvalues of the dimensions which the data is projected into is greater than or equal to half of the total sum of the eigenvalues.

(a) Query image



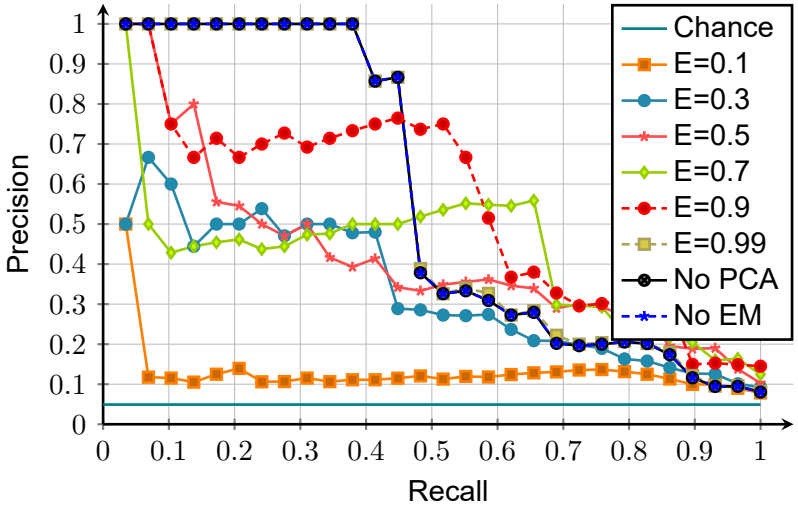
(b) Resultant graph

Figure A.1: Gridded RGB histogram, using PCA for query image 9_23_s

Both Figures A.1 and A.2 show the expected result that the higher the value of E, the better the performance of the distance measure. The 'No PCA' plot shows the result of generating the eigenmodel and projecting the descriptors into this space, but without reducing the dimensionality of the descriptor. This plot is identical to the 'No EM' plot, which is the results obtained without generating the eigenmodel, and projecting the descriptors into that space. This is the expected result since, if all dimensions are maintained, the data has not been changed, and the Euclidean distance between descriptors in the two spaces should be identical.

Table A.1 shows the number of dimensions in the resultant descriptors when performing PCA on a gridded global colour histogram at different energy values. The table shows that even for an energy value of 0.99, i.e. 99 % of the total eigenvalue engery, the descriptor dimensions can be reduced from 384 dimensions to 104. This shows that there are many dimensions with almost no variation in the original descriptor which can be removed without reducing performance by a large amount. This is demonstrated in the test data, where the plot for $E = 0.99$ is almost identical to the plot for No PCA. This reduces the size of the descriptor to only 26% of it's size without PCA. Reducing the value of E even further to 0.9 means that the descriptor is only 8.9% of its original size, whilst still maintaining good performance.

(a) Query image



(b) Resultant graph

Figure A.2: Gridded RGB histogram, using PCA for query image13_1_s

Table A.1: PCA Dimension reduction for Gridded RGB Histogram

Energy	Descriptor dimensions	Proportion of total dimensions
0.10	1	0.3%
0.20	2	0.5%
0.30	3	0.8%
0.40	4	1.0%
0.50	5	1.3%
0.60	7	1.8%
0.70	11	2.9%
0.80	18	4.7%
0.90	34	8.9%
0.95	52	13.5%
0.99	101	26.3%
1.00	384	100.0%

B Distance Measures

The results presented in the body of this report use the L2 Norm, otherwise known as Euclidean distance, to measure the distance between descriptors. However despite this being the most common choice for distance measure, other distance measures do exist, these will be compared in this Appendix.

The distance measured investigated are:

1. L_1 norm (Manhattan distance)
2. L_2 norm (Euclidean distance)
3. L_2 norm squared
4. L_∞ norm (Chebyshev distance)
5. Mahalanobis distance

A vector norm, $|\mathbf{x}|$ refers to a scaler property that, in perhaps an abstract way, describes the length of a vector, $\mathbf{x}$. Mathematically this is defined as shown in Equation B.1 [5, p. 1081].

$$\begin{aligned} |\mathbf{x}| &> 0 \quad \text{when } \mathbf{x} \neq \mathbf{0} \\ |\mathbf{x}| &= 0 \quad \text{when } \mathbf{x} = \mathbf{0} \\ |k\mathbf{x}| &= |k||\mathbf{x}| \quad \text{for any scalar } k \\ |\mathbf{x}+\mathbf{y}| &\geq |\mathbf{x}| + |\mathbf{y}| \end{aligned} \tag{B.1}$$

The L_1, L_2 and L_3 norms describe this distance in different ways. The L_2 norm is the most common norm because it describes the point to point distance in Euclidean geometry. It is defined in Equation B.2 [5, p. 1081].

$$|\mathbf{x}|_2 = \sqrt{\sum_{r=1}^{n} |x_r|^2} \quad \text{where } \mathbf{x} = [x_1, x_2, ..., x_n] \tag{B.2}$$

The L_1 norm describes distance in a slightly different way. It is the sum of the magnitudes of the vector components. This is the reason it is given the name 'Manhattan' or 'city block' distance, because in a two dimensional vector space, the L_2 norm would be the hypotenuse of a right angle triangle joining two node L_1 norm would be the sum of the other two sides, analogous to waling along city blocks to go between two points. It is defined in Equation B.3 [5, p. 1081].

$$|\mathbf{x}|_1 = \sum_{r=1}^{n} |x_r| \quad \text{where } \mathbf{x} = [x_1, x_2, ..., x_n] \tag{B.3}$$

The L_∞ norm describes distance as the magnitude as the largest component of the vector. This therefore ignores the magnitudes of all smaller components. It is defined in Equation B.4 [5, p. 1081].

$$|\mathbf{x}|_\infty = \max_i |x_i| \quad \text{where } \mathbf{x} = [x_1, x_2, ..., x_n] \tag{B.4}$$

These norms can be summarised in one equation which can define the L_p norm for any p.

$$|\mathbf{x}|_p = \sqrt[p]{\sum_{r=1}^{n} |x_r|^p} \quad \text{where } \mathbf{x} = [x_1, x_2, ..., x_n] \tag{B.5}$$

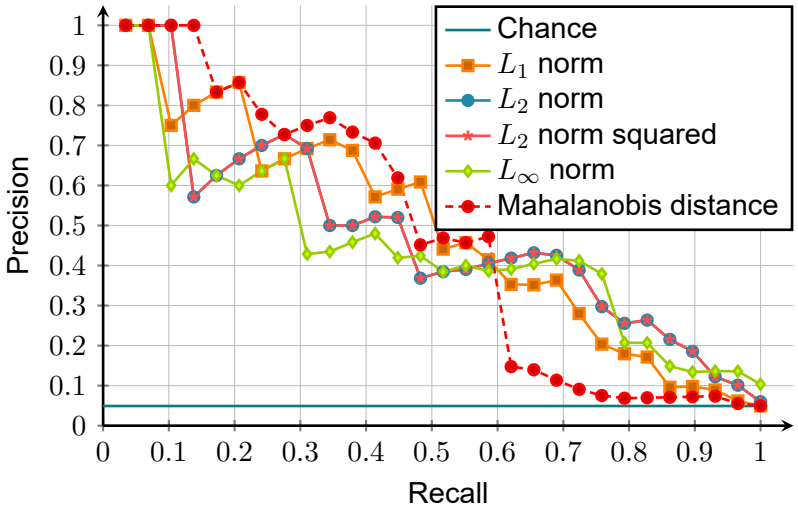
The only other distance measures under consideration are the L_2 norm squared, and the Mahalanobis distance. The L_2 norm squared is equal to the L_2 norm definition, but only the sums of the squared components is considered, this sum is not square rooted. This measure therefore places an

increasing weight on larger components, as they will contribute a larger amount to the sum compared to their contribution to the square root of the number.

The Mahalanobis distance is slightly different to the other distance measures here, as instead of considering distance as a function of the differences between different vector dimensions in a standard Euclidean space, the Mahalanobis distance instead first calculates the spread of the data in each dimension, and then measures difference in terms of how many standard deviations each linear difference represents. This can provide a normalising factor to the difference function because a difference in a dimension with a large spread of data will represent a much larger contribution than the same difference in a dimension with a small spread. This effect can be thought of as the distance function fitting itself to the shape of the data. The Mahalanobis distance is implemented in the system by weighting the Euclidean distance with the eigenvalue for that dimension. This is shown mathematically in Equation B.6, where v_r represents the eigenvalue for each dimension.

$$|\mathbf{x}|_2 = \sqrt{\sum_{r=1}^{n} \frac{|x_r|^2}{v_r}} \quad \text{where } \mathbf{x} = [x_1, x_2, ..., x_n] \tag{B.6}$$

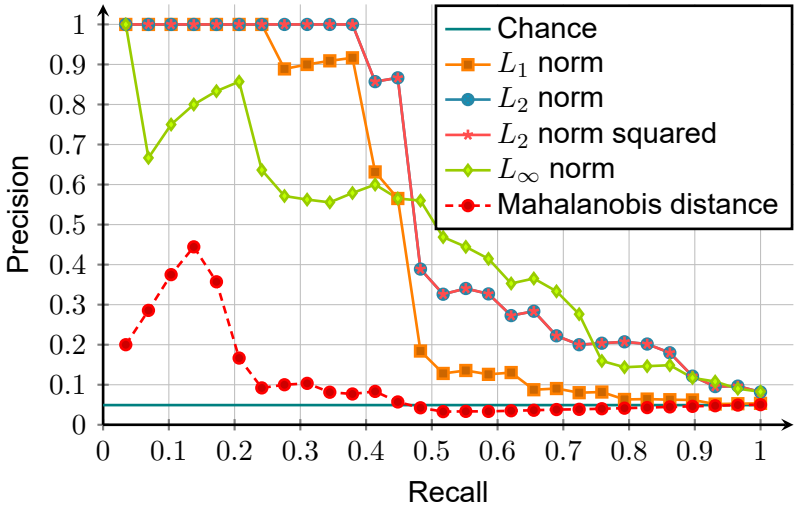
(a) Query image



(b) Resultant graph

Figure B.1: Gridded RGB histogram descriptor results for query image 9_23_s

(a) Query image



(b) Resultant graph

Figure B.2: Gridded RGB histogram descriptor results for query image 13_1_s

Figures B.1 and B.2 show the results of applying different distance measures to two test images. In both cases the L_2 norm squared has performed exactly as well as the L_2 norm. This is an interesting result, it implies that, in the test data, no matrix dimension difference magnitude had a much larger magnitude than the others.

The results for the other measures showed much greater difference in the books. It can be seen that whilst the L_1 and L_2 norm showed generally similar performance, the L_∞ norm showed significantly poorer performance. This confirms that only considering the most significant vector dimension is not sufficient to accurately determine differences between the images. The L_2 norm also outperformed the L_1 norm in the average case for the books. The image of the sheep however showed similar performance between distance measures.