# EEE3032
# Computer Vision and Pattern Recognition
# Coursework Assignment

# Visual Search of an Image Collection

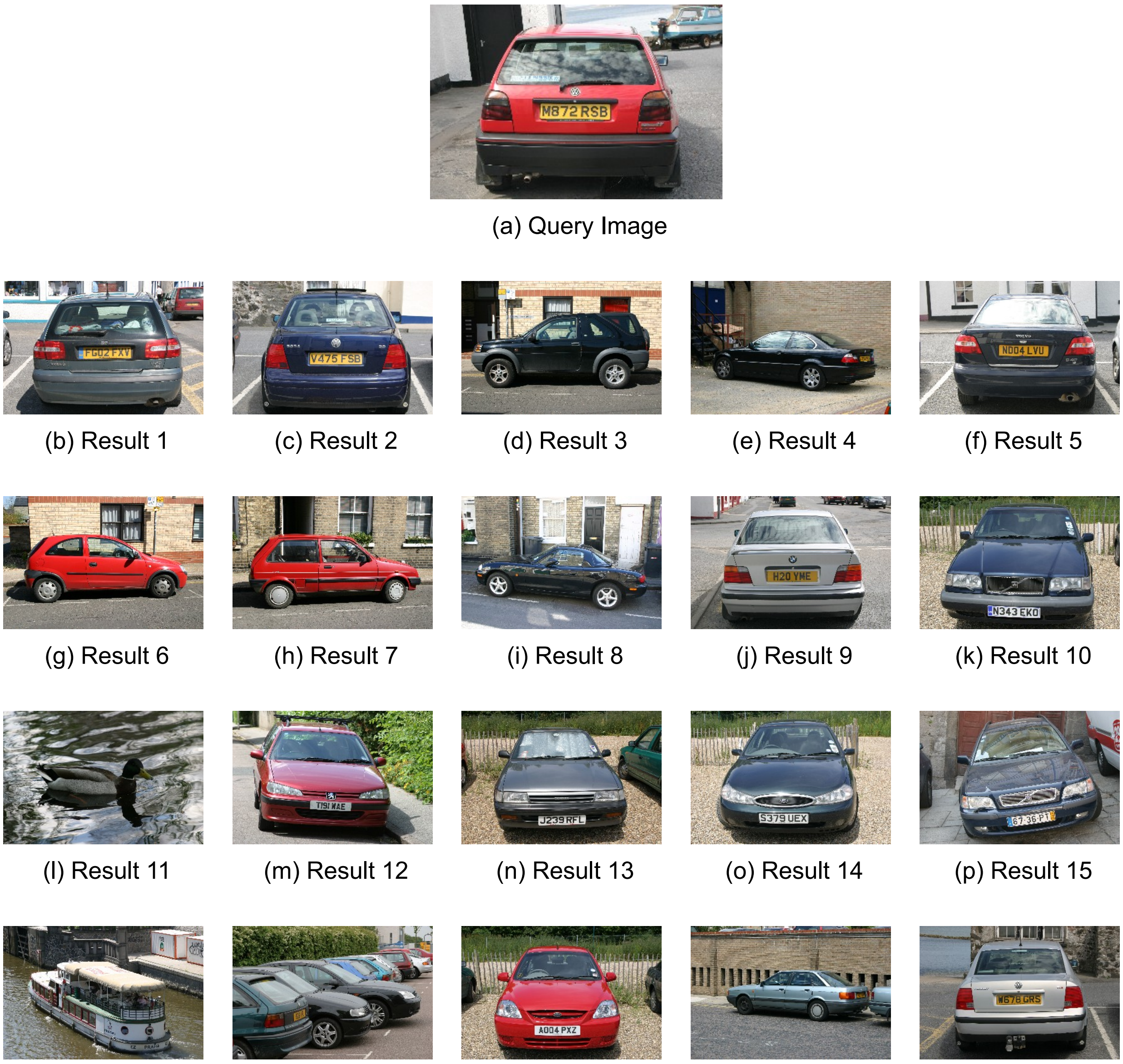


Figure 1: Image recall results for concatenated descriptors

*Joshua Tyler*

April 19, 2016

**Abstract**

This report presents the results of a visual search system implemented using MATLAB. The system is built upon a modular framework which allows tests to be run using different descriptors and distance measures.

Various distance measures have been evaluated, and results compiled for the entire database of 591 images. It has been found that a concatenated, gridded, texture and colour descriptor produces the best results, in the general case. The performance is however limited to a greater extent by the semantic gap between the low level methods used and the high level information encoded in the images.

Additional testing which evaluates different comparison techniques, as well as principal component analysis is presented in Appendices. Projection of the descriptors into a lower dimensional space, as well as use of an less computationally expensive comparator could allow for faster image recall in time critical systems.

# Contents

# 1 Introduction

Large databases of images exist in various fields. Indexing and searching these using traditional 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 similar images. 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].

For the purposes of evaluating this system, similarity will be defined using the 20 categories which the image set is sorted into. This is not ideal since the categories encode level 2 meaning, whilst the descriptors used will perform level 1 operations [3]. For this reason, it is not expected that the system will have exceptional performance.

In order to calculate the distance between descriptors, the $L_2$ norm is used. This distance measure, also known as Euclidean distance, returns the straight line distance between two points in a vector space. It is discussed in greater detail, and performance evaluated against other distance measures, in Appendix B.

## 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. In this figure shaded blocks represent functions necessary for the system's core operation, white blocks represent auxillary functions, and dashed blocks represent functions or data which may be omitted. The arrows show the transitions of data.

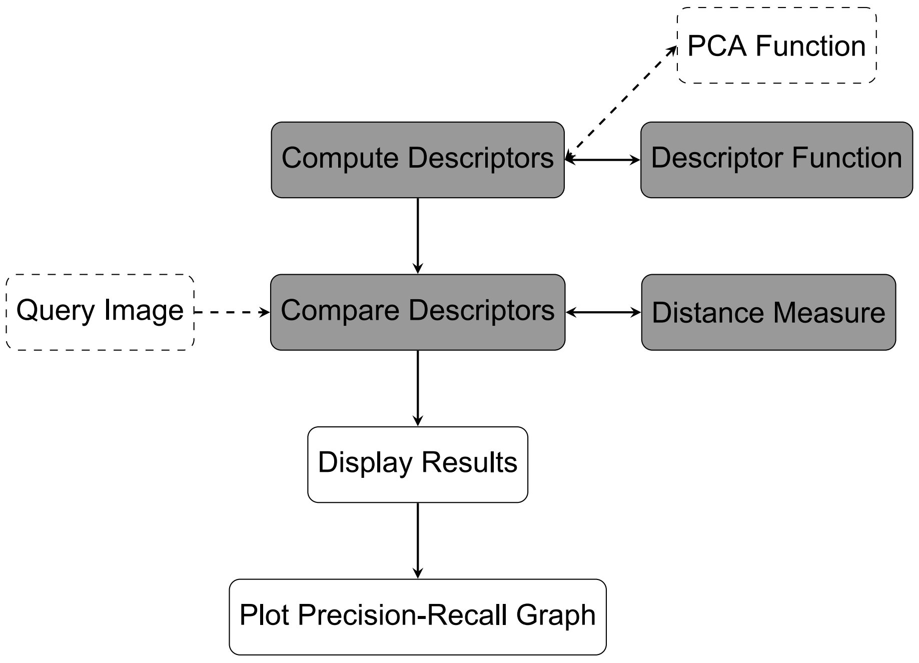


Figure 1.1: System design

# 2 Description Techniques

This Chapter discusses the different descriptor generation techniques implemented, along with the theory of their operation and the difficulties which were encountered whilst implementing them. Details of how each technique was tested is given in Appendix C.

## 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. A diagrammatic representation of this technique is shown in Figure 2.1. This Figure shows the RGB space quantised with a quantisation level of $Q = 2$, yielding $2^2 = 4$ bins. $P_1$, $P_2$, and $P_3$ represent points in the RGB space which will be quantised to their containing boxes.

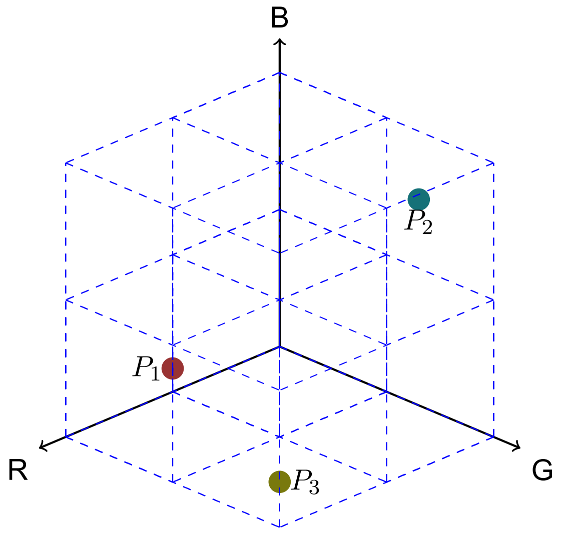


Figure 2.1: Visualisation of RGB histogram

The implementation of this function was relatively straightforward, since example code for the function had been provided [4]. 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 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 only variable parameter is therefore $Q$.

## 2.2 Gridding

Gridding of features allows descriptors to apply 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, and the descriptor function can then be applied to each of these sub images. The resultant descriptors are then concatenated into a matrix.

Gridding is implemented as `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 two independent variables, `h_level` and `v_level`.

However, since most of the images in the dataset have an aspect ratio 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`.

## 2.3 Edge Orientation Histogram

Texture is an important 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].

One method of creating a texture based descriptor is to use an edge orientation histogram. This works as follows:

1. Edges are detected in the image.
2. An estimation of the orientation of each edge is calculated.
3. A histogram is created of the edge orientations.

This process is implemented using two functions: an edge orientation function, and a histogram generation function. The edge orientation function returns a two dimensional matrix 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 Sobel operator is used [5]. The Sobel operator is formed of two kernels which are convolved with input image in order to perform edge detection in the x and y directions respectively. The kernels are defined in Equation 2.1.

$$K_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \; K_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.1}$$

The edge magnitude and angle are calculated as described in Equations 2.2 and 2.3 respectively. It should be noted that the operators in these equations apply on an element-by-element basis, not to the matrices 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.

$$\text{Mag} = \sqrt{{K_x}^2 + {K_y}^2} \tag{2.2}$$

$$\theta = \text{atan2}(-K_y, K_x) \tag{2.3}$$

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, v_level, h_level, @(x)vs_edge_detect(x, @(x)vs_compute_histogram(x,Q),E))`.

## 2.4 Concatenation of Descriptors

Descriptors focusing upon an individual aspect of an image, such as colour or texture have already been discussed, as well as how the can be applied to multiple sections of an image rather than globally.

It stands to reason therefore that multiple descriptors could be combined to produce a descriptor of even greater effectiveness, such as a combination of an RGB histogram and a texture histogram. 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 `@(x)horzcat( func_1(x) .* M, func_2(x))` where `texture_func(x)` and `color_func(x)` are function handles for two descriptor functions with a single parameter, `x` for the input image. The only other parameter in this function is `M`. `M` defines a weighting factor between the two descriptors, and can be used to give a greater weighting to one over the other. This works by using a multiplier value to either spread all of the values in the descriptor further apart, or bring them closer together.

# 3 Experimental Results

In order to evaluate the performance of each descriptor, two test images have been chosen from the data set: 9_23_s and 13_1_s. 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, as well as two regions of largely uniform texture: the sheep's body, and the grass. Image 13_1_s strongly contrasts this. This image shows a collection of differently coloured books on a shelf. The image therefore contains a large variety of colours distributed around the image. In addition to this the transitions between the books, as well as the text on the spine of the books means that a large amount of varied texture is distributed across the image.

The fact that these images contrast in many different ways means that they should be ideal to evaluate the performance of each descriptor in a fair way, as well as determine the optimum parameters for each descriptor function. The performance of the descriptors across the wider images in the dataset will then be presented in Section 3.

In order to evaluate performance, a precision-recall characteristic is plotted for each test. This characteristic is a plot showing the proportion of the returned results which have been relevant (precision), vs the number of relevant results returned (recall). The ideal precision-recall characteristic would be a horizontal line at precision = 1, a system which produced this characteristic would return all the relevant images, and none of the irrelevant images. Most practical systems produce a characteristic which slopes from the top left to the bottom right, this occurs because systems first return the results they are most sure about. These results are likely to be correct, and so precision will be high, but since not many results have yet been returned, recall will be low. As the system returns an increasing number of results, an increasing number of irrelevant results will be returned also, causing precision to decrease as recall increases.

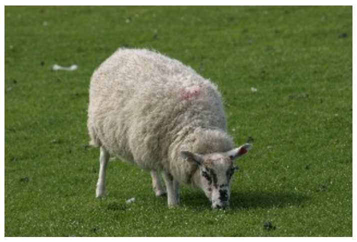
## 3.1 Global Colour Histogram

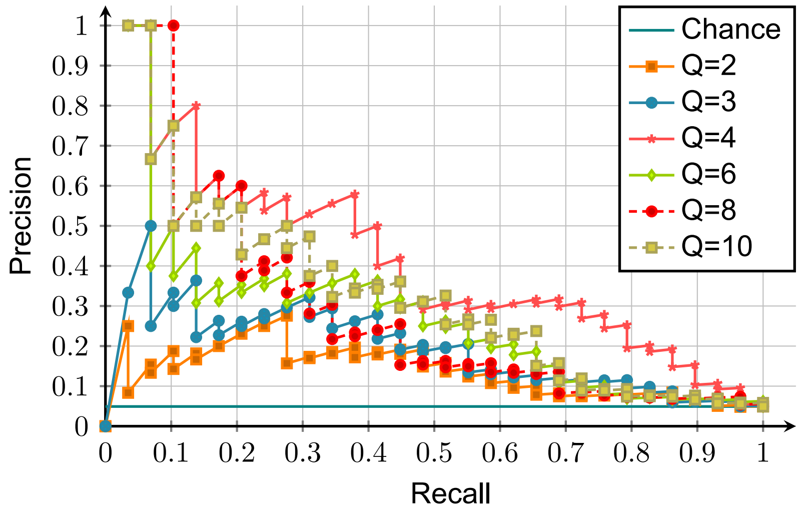
The global colour descriptor has one parameter the quantisation level, $Q$, which produces a descriptor of length $Q^3$, as discussed in Section 2.1.

In order to analyse the effect of quantisation level on performance, results will be generated for various values of $Q$, and the performance of each image analysed for each test value. The expected result is that higher values of $Q$ will produce better results, up to a certain point, after which performance will decrease. This should happen because at very high quantisation levels values, there are too many possible bins, and as such nominally similar colours will be sorted into different bins. Since each bin is a different dimension in the descriptor, they will not therefore be considered similar by the distance measure function.

The results shown in Figures 3.1 and 3.2 match expectations suggesting that a very low quantisation value does not deliver good results, yet performance also drops off at higher quantisation levels. 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.

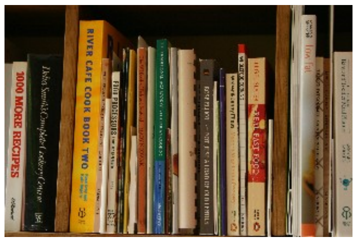
The results also show that query image 13_1_s performs better than 9_23_s. The reason for this is likely due to the fact that 13_1_s has a much more varied colour distribution, allowing it to have a much more unique histogram signature. This means that similar images can be better matched. In contrast to this, the green field will dominate the histogram of image 9_23_s meaning the system may confuse it with other images containing a green background, such as those of other farm animals in a field.

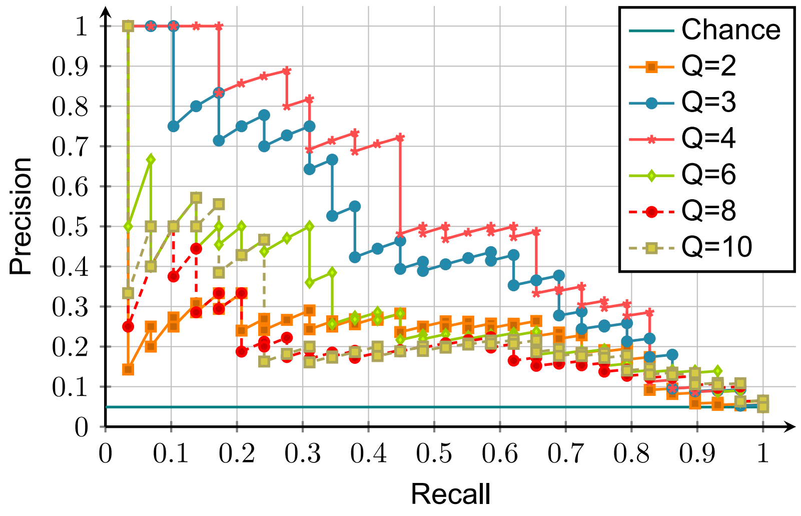

(a) Query image



(b) Resultant graph

Figure 3.1: Global colour histogram results for query image 9_23_s


(a) Query image



(b) Resultant graph

Figure 3.2: Global colour histogram results for query image 13_1_s

## 3.2 Gridding

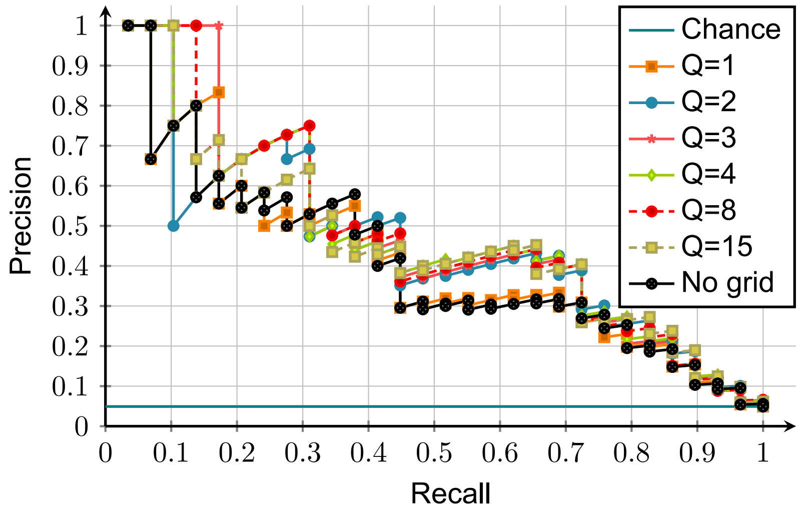
As discussed in Section 2.2, the independent parameters for the gridding function can be reduced to a single quantisation factor $Q$. In a similar manner to that of Section 3.1 the optimum quantisation factor will be determined by computing descriptors for various values of $Q$ and then, using the two test images, determining which has best results.

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 3.3 and 3.4 show the results of the gridding process.

Figure 3.3 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 precision values for higher recall levels.

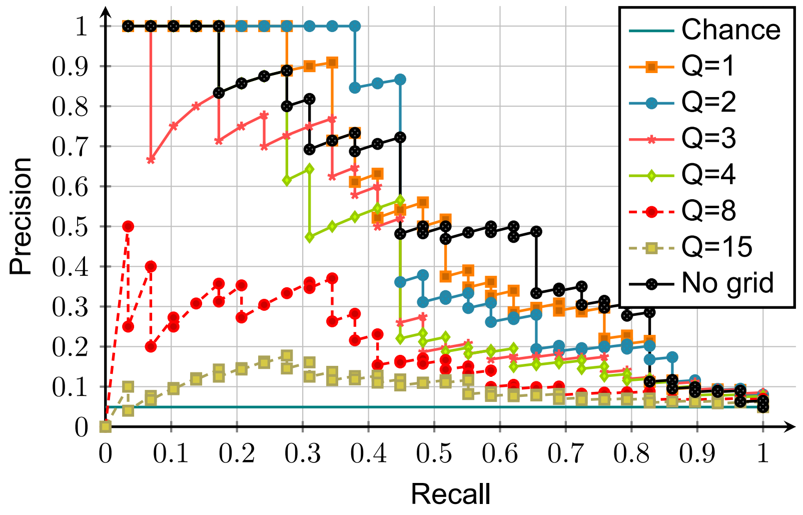
(a) Query image



(b) Resultant graph

Figure 3.3: Gridded global colour histogram results for query image 9_23_s

(a) Query image



(b) Resultant graph

Figure 3.4: Gridded global colour histogram results for query image 13_1_s

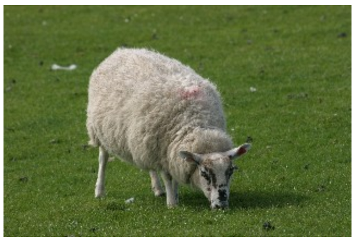
Figure 3.4 on the other hand shows great variation in the result with grid size. Increasing the grid quantisation level to 2 greatly increases the performance up to a recall of around 0.4, and a grid quantisation level of 1 also increases the performance by a large amount also. In addition to this, much higher quantisation steps, such as 8 and 15 show greatly decreased performance.

The optimum quantisation factor is therefore $Q = 2$, yielding `h_level = 3`, and `v_level = 2` using the formulae defined in Section 2.2.

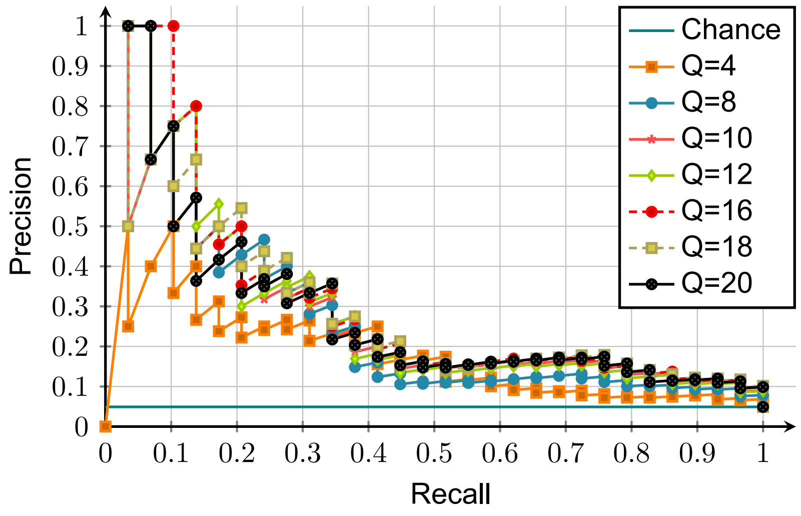
## 3.3 Edge Orientation Histogram

As discussed in Section 2.3, the gridded edge orientation histogram implementation has two variable parameters, $E$ and $Q$ (setting the grid quantisation factors to the optimised values from Section 3.2). Evaluating the effect of changing both together is computationally expensive, 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. The expected result is similar to increasing the quantisation level of the RGB histogram – increasing $Q$ will increase performance up to a point, but increasing past that point will result in poorer results

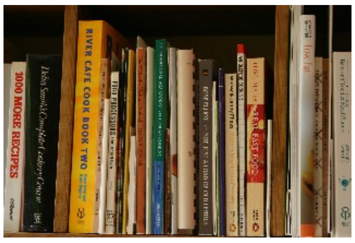


(a) Query image

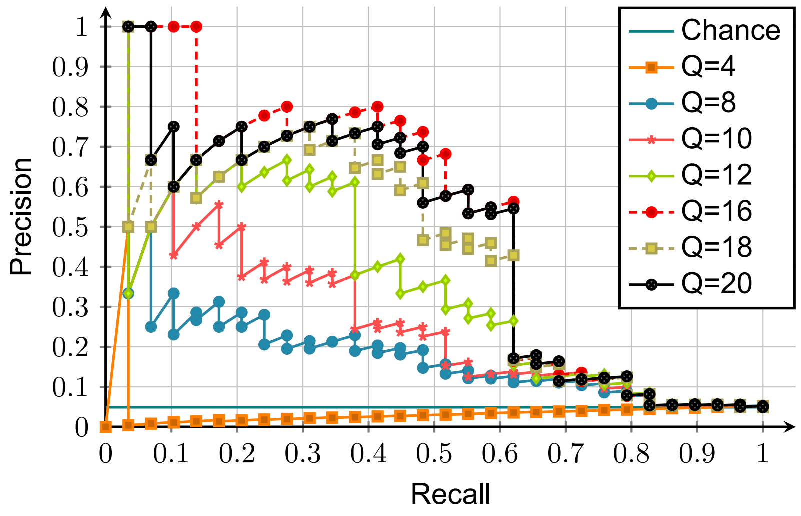


(b) Resultant graph

Figure 3.5: Gridded texture histogram results for query image 9_23_s, with $E = 0$



(a) Query image

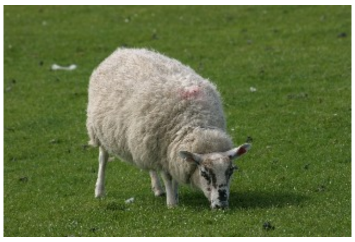


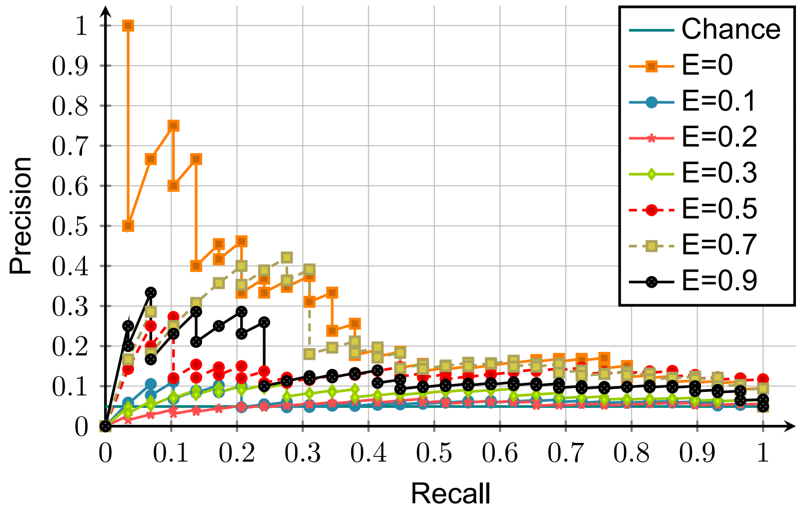
(b) Resultant graph

Figure 3.6: Gridded texture histogram results for query image 13_1_s, with $E = 0$

Figure 3.5 shows that for the sheep image, there is very little difference in performance with quantisation levels greater than approximately 4. Figure 3.6 shows that for the books, on the other hand, performance is strongly dependant on quantisation level. Higher quantisation levels continue to improve performance, up to approximately $Q = 16$, after this point performance decreases.

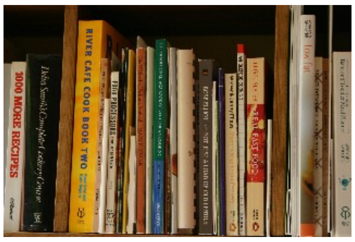
Figures 3.7 and 3.8 show the result of varying $E$, whilst keeping $Q$ at a constant value of $16$. The results are interesting. For the bookshelves, increasing $E$ above $0$ improves performance up to approximately $E = 0.3$, however, for the sheep increasing $E$ above $0$ greatly hinders performance. This is likely due to the fact 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.

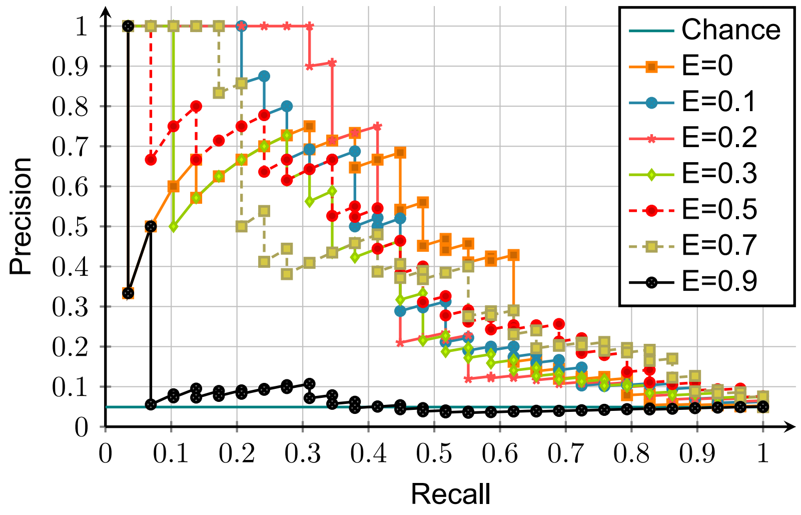

(a) Query image



(b) Resultant graph

Figure 3.7: Gridded texture histogram results for query image 9_23_s, with $Q = 16$


(a) Query image



(b) Resultant graph

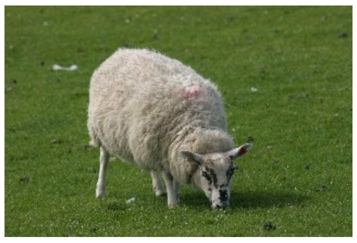
Figure 3.8: Gridded texture histogram results for query image 13_1_s, with $Q = 16$

## 3.4 Concatenation of Descriptors

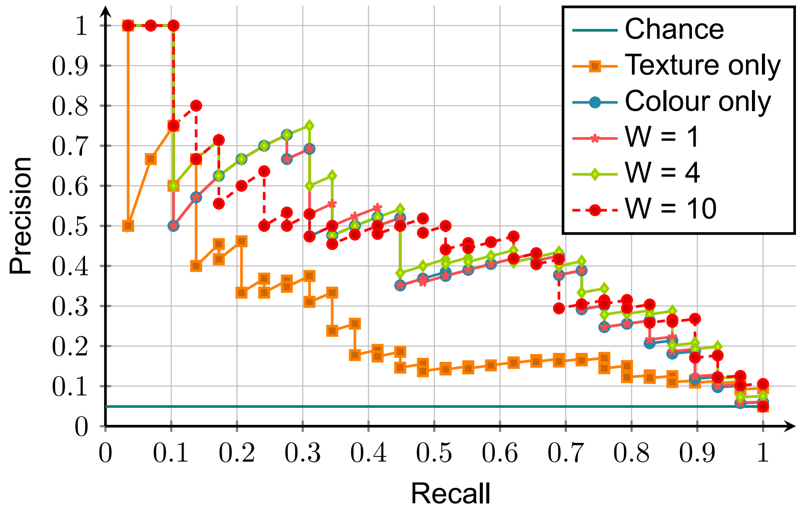
Sections 3.2 and 3.3 demonstrate the effectiveness of gridded colour histogram and texture descriptors. It stands to reason therefore that these descriptors could be combined to produce a descriptor of even greater effectiveness. This is possible using the methodology described in Section 2.4.

In order to combine the gridded texture and gridded global colour histogram results, the parameters of these functions which were found to be optimal in Sections 3.2 and 3.3 will be used. The only free parameter of the concatenation function is therefore $M$. $M$ is a multiplier that is applied to the texture function 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 bin 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 3.9 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 texture plot, leading to the colour plot dominating the results.

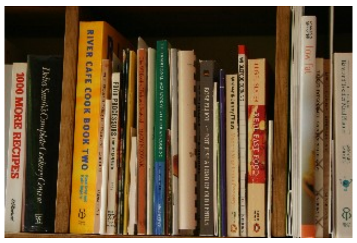


(a) Query image

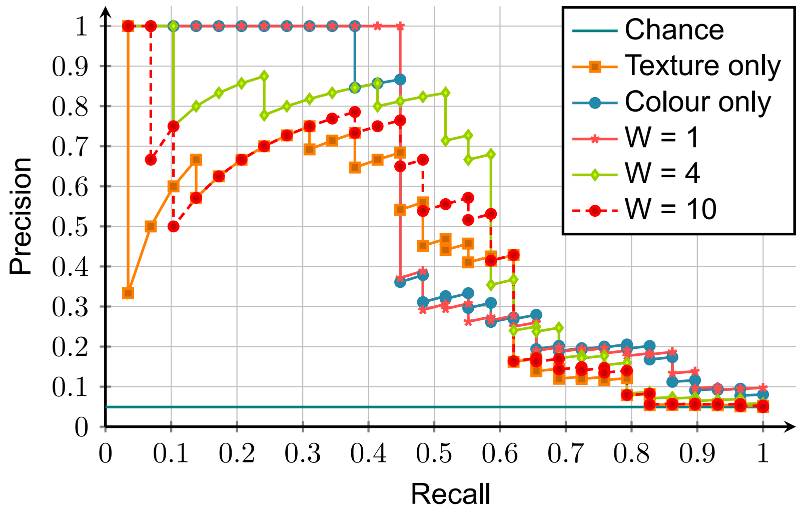


(b) Resultant graph

Figure 3.9: Concatenated descriptor results for query image 9_23_s



(a) Query image



(b) Resultant graph

Figure 3.10: Concatenated descriptor results for query image 13_1_s

Figure 3.10 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 are discriminative. The results show that at 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 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.

## 3.5 Overall Results

Following the characterisation each algorithm, discussed in the previous parts of this section, it has been possible to test the performance of each descriptor against the entire database of images. This is achieved by calculating mean average precision (MAP). MAP is a single numeric measure for determining the performance of a descriptor across a dataset. Is is the mean of average precision (AP), which is defined in Equation 3.1. In this equation P$(i)$ represents the precision at that value of

$i$, and Rel$(i)$ is $1$ if the result is relevant, or $0$ if it is not.

$$\text{AP} = \frac{\sum_{i=1}^{n} (\text{P}(i) \times \text{Rel}(i))}{\sum_{i=1}^{n} \text{Rel}(i)} \tag{3.1}$$

Table 3.1 tabulates the MAP statistics for all images in the dataset, both within each category, as well as globally. These results show MAP across the entire dataset, so a lot of the numbers are low. The reason for this is that whilst the images have similar Level 2 content, some of the images are very different in terms of colour and texture content. In order to account for this, Table 3.2 tabulates MAP statistics for the first 10 relevant recalled images. This data removes the effect of the aforementioned 'outlier' images in each category, and better demonstrates how well each type of image content responds to each descriptor.

The results in Table 3.2 show that the image sets respond well to some descriptors, and some respond poorly. One example of this is category 4, where the MAP is 0.71 for the edge orientation histogram, but 0.25 and 0.44 for the global colour histogram and gridded global colour histogram. This can be explained by looking at the type of image category 4 is. Category 4 contains images of aeroplanes. Almost all of these images have a smooth aeroplane body in the middle of the image, with a clear sky at the top, and plain grass at the bottom. This distinctive texture signature is what allows this high AP figure. Other image sets which respond particularly well to the edge orientation histogram are categories 7 and 13 respectively. Category 7 contains cars, and category 13 books on shelves, both of these have distinct texture signatures.

In general, most of the categories responded better to the gridded colour histogram than the global colour histogram. This is expected since encoding the spatial aspect of the colour data should allow for a more discriminative descriptor. A notable example which does not is category 18. Category 18 is a series of images which all contain water, however the water is in a different location in almost every image, this kind of spatial variance cannot be encoded in the gridded descriptor, so it performs poorly.

Two categories which responded particularly well to the colour descriptor are categories 2 and 13. These are images of trees, mostly on a background of a washed-out sky, and bookshelves of colourful books respectively. Both images likely work well with this descriptor because they contain a variety of colours which are both common across most of the images, and distinctive in the image set itself.

In general, concatenation of colour and texture descriptors improved performance over the individual descriptors. The cases where they did not are cases where one descriptor had a much greater performance over the other, such as category 4, which responded particularly well to the texture descriptor and category 2 which responded particularly well to the colour descriptors. One example of a category which showed great improvement with descriptor concatenation are categories 7 and 10. These categories contain differently coloured cars in different orientations and differently coloured flowers, with varying shapes. Concatenation helps with the cars because the variance of colour across the set means that the similarly coloured cars strongly match each other in a colour histogram, but all others are distant, whereas all of the cars are likely moderate matches with the edge orientation histogram. The flowers have a similar effect.

One category which performed particularly poorly in all cases is category 16. This category contains photographs of dogs, of various sizes and colours, all taken from different camera angles and on different backgrounds. This is an example of Level 2 concepts which do not translate well at all to Level 1 attempts at interpretation.

Table 3.1: MAP statistics for all methods and categories

| Image category | Global colour histogram | Gridded colour histogram | Gridded edge orienatation histogram | Concatenated descriptors |
|---:|---:|---:|---:|---:|
| 1 | 0.16 | 0.16 | 0.15 | 0.17 |
| 2 | 0.30 | 0.45 | 0.17 | 0.33 |
| 3 | 0.09 | 0.11 | 0.08 | 0.11 |
| 4 | 0.14 | 0.23 | 0.47 | 0.35 |
| 5 | 0.12 | 0.13 | 0.17 | 0.17 |
| 6 | 0.18 | 0.14 | 0.17 | 0.20 |
| 7 | 0.22 | 0.31 | 0.37 | 0.52 |
| 8 | 0.15 | 0.20 | 0.13 | 0.33 |
| 9 | 0.26 | 0.29 | 0.26 | 0.34 |
| 10 | 0.16 | 0.19 | 0.11 | 0.25 |
| 11 | 0.07 | 0.09 | 0.10 | 0.11 |
| 12 | 0.08 | 0.09 | 0.15 | 0.10 |
| 13 | 0.38 | 0.38 | 0.21 | 0.29 |
| 14 | 0.08 | 0.10 | 0.16 | 0.13 |
| 15 | 0.10 | 0.11 | 0.13 | 0.15 |
| 16 | 0.08 | 0.08 | 0.08 | 0.08 |
| 17 | 0.21 | 0.23 | 0.16 | 0.22 |
| 18 | 0.10 | 0.08 | 0.16 | 0.11 |
| 19 | 0.10 | 0.10 | 0.07 | 0.11 |
| 20 | 0.18 | 0.13 | 0.24 | 0.20 |
| **Global** | **0.16** | **0.18** | **0.18** | **0.21** |

Table 3.2: MAP statistics for first 10 images in each set, over all methods and categories

| Image category | Global colour histogram | Gridded colour histogram | Gridded edge orienatation histogram | Concatenated descriptors |
|---:|---:|---:|---:|---:|
| 1 | 0.30 | 0.30 | 0.25 | 0.30 |
| 2 | 0.50 | 0.71 | 0.36 | 0.64 |
| 3 | 0.12 | 0.19 | 0.13 | 0.21 |
| 4 | 0.25 | 0.44 | 0.71 | 0.61 |
| 5 | 0.24 | 0.24 | 0.27 | 0.31 |
| 6 | 0.28 | 0.25 | 0.30 | 0.34 |
| 7 | 0.33 | 0.46 | 0.57 | 0.71 |
| 8 | 0.24 | 0.34 | 0.19 | 0.56 |
| 9 | 0.41 | 0.43 | 0.39 | 0.48 |
| 10 | 0.26 | 0.31 | 0.14 | 0.46 |
| 11 | 0.11 | 0.16 | 0.16 | 0.20 |
| 12 | 0.14 | 0.17 | 0.31 | 0.21 |
| 13 | 0.65 | 0.69 | 0.42 | 0.55 |
| 14 | 0.13 | 0.18 | 0.28 | 0.23 |
| 15 | 0.16 | 0.20 | 0.19 | 0.26 |
| 16 | 0.11 | 0.11 | 0.10 | 0.12 |
| 17 | 0.33 | 0.40 | 0.30 | 0.40 |
| 18 | 0.17 | 0.14 | 0.29 | 0.19 |
| 19 | 0.17 | 0.16 | 0.10 | 0.18 |
| 20 | 0.29 | 0.22 | 0.36 | 0.32 |
| **Global** | **0.26** | **0.31** | **0.29** | **0.36** |

# 4 Conclusion

This report presents several methods of generating descriptors for images, and uses test data to set their adjustable parameters, the results of which is presented in Section 3. The overall results show that the MATLAB system is capable of performing with a reasonable MAP over the first 10 images in each set. However performance does suffer when considered over the whole dataset. Overall the concatenated descriptors perform best over the entire dataset with a MAP of 0.21, or 0.36 if only the first 10 images are considered. A global colour histogram performs most poorly overall with a MAP of 0.16 over the entire dataset, or 0.26 if only the first 10 images are considered.

The Appendices present further results which extend the investigation in areas other than investigation of different descriptors. Appendix A investigates principal component analysis (PCA) and shows how descriptors can be made much more compact without impact on performance. The investigated descriptor, a gridded colour histogram, was able to be reduced to 26% of it's original size without noticeable performance impact.

Appendix B investigates various distance measures which can be used to compare descriptors and tests their relative performance. The $L_2$ norm, which has been used throughout the remainder of this investigation was found to be the strongest overall performer.

In addition Appendix C presents the test methodology for verification of the functionality of the descriptor generation functions.

To conclude the system presented by the report is a modular and extensible visual search system, with characterised performance, and scope to be extended beyond its current functionality.

# 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. P. Eakins, “Techniques for image retrieval,” *Library and Information Briefings 85, British Library and South Bank University*, 1998.

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

[5] 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].

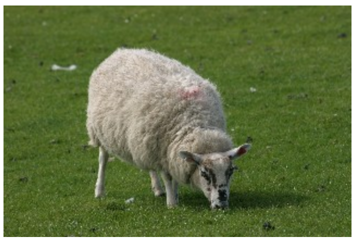
[6] I. Gradshteyn and I. Ryzhik, *Table of Integrals, Series, and Products*, 7th ed. Elsevier, 2007.

# A Principal Component Analysis

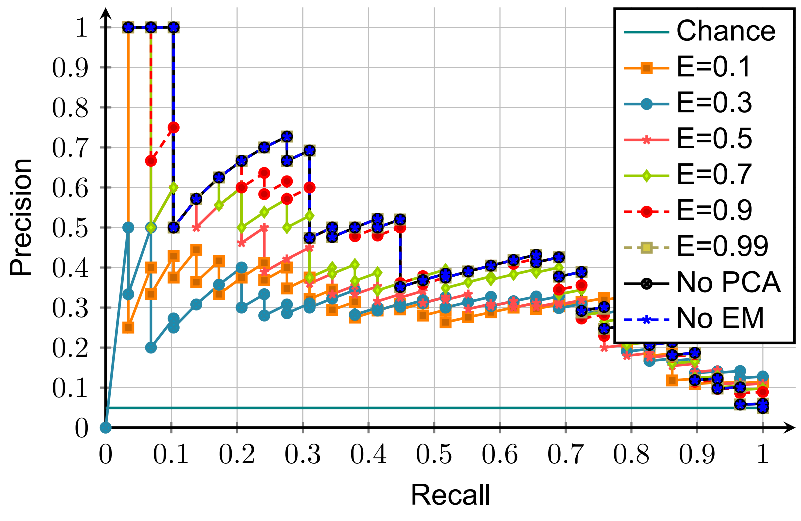
Descriptors should be as discriminative as possible, whilst also being as compact as possible. This means that each descriptor should encode encode as much information allowing it to be differentiated from other images as possible, whilst also containing the minimum amount of data allowing it to do this. 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.

PCA analyses these descriptors and, via construction of an eigenmodel, 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.

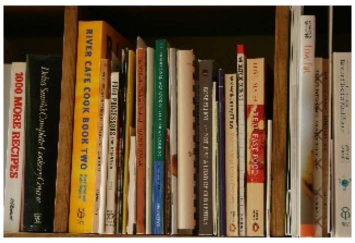


(a) Query image

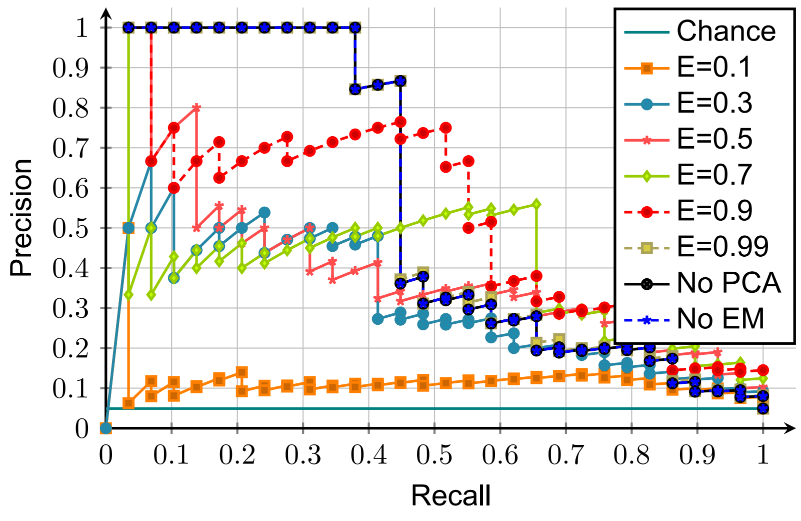


(b) Resultant graph

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



(a) Query image



(b) Resultant graph

Figure A.2: Gridded RGB histogram, using PCA for query image13_1_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 projecting the descriptors into the eigenmodel 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 $E = 0.99$, 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 space 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.

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. $L_\infty$ 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 [6, 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 [6, 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 nodes, whilst the $L_1$ norm would be the sum of the other two sides, analogous to walking along city blocks to go between two points. It is defined in Equation B.3 [6, 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 $L_\infty$ norm describes distance as the magnitude of the largest component of the vector. This therefore ignores the magnitudes of all smaller components. It is defined in Equation B.4 [6, p.1081].

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

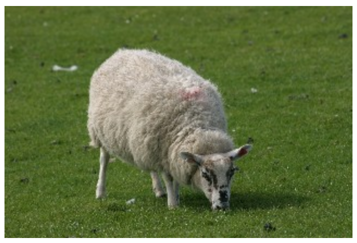
These norms can be summarised in one equation which can define the $L_p$ norm for any $p$. However using the $L_p$ norm for values other than $p = 1, 2, \infty$ is not common.

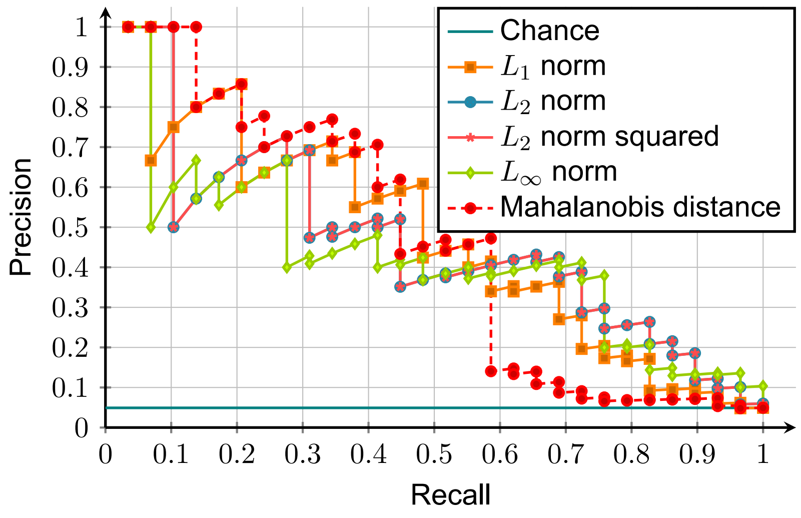
$$|\mathbf{x}|_p = \sqrt[p]{\sum_{r=1}^{n} |x_r|^p} \quad \text{where } \mathbf{x} = [x_1, x_2, ..., x_n] \tag{B.5}$$

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 sum 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 presented. 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 smaller contribution than the same linear 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.

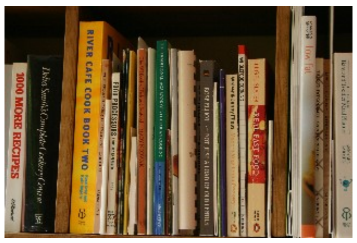
$$|\mathbf{x}|_2 = \sqrt{\sum_{r=1}^{n} \frac{|x_r|^2}{v_r}} \quad \text{where } \mathbf{x} = [x_1, x_2, ..., x_n] \tag{B.6}$$

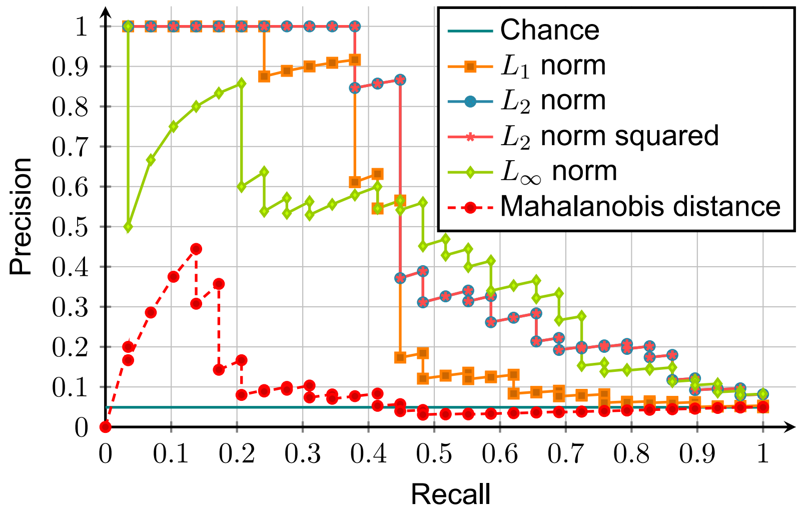

(a) Query image



(b) Resultant graph

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


(a) Query image



(b) Resultant graph

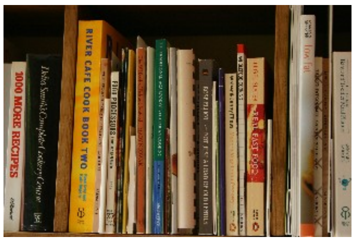
Figure B.2: Gridded RGB histogram results for query image 13_1_s

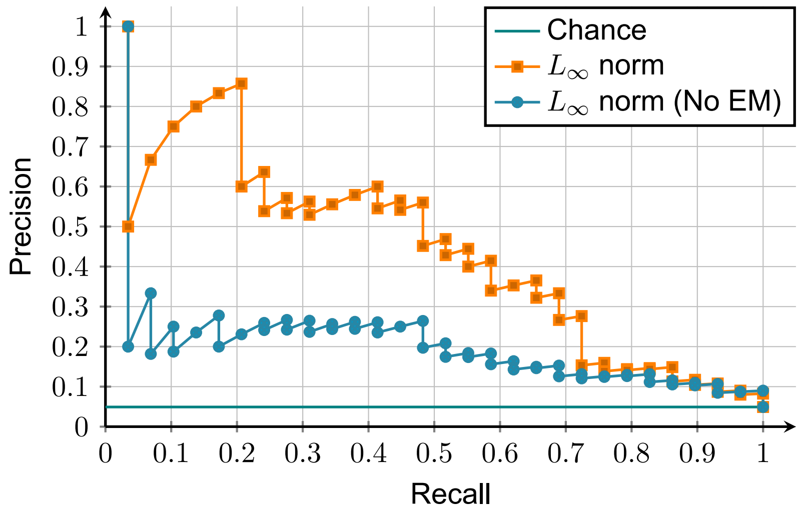
Figures B.1 and B.2 show the results of applying different distance measures to two test images. In both cases the $L_2$ norm squared has performed exactly as well as the $L_2$ norm. This is an interesting result, it implies that, in the test data, no matrix dimension difference magnitude had a much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

The results for the $L_1$ norm and $L_\infty$ norm showed much greater difference in the test using image 13_1_s than 9_23_s. It can be seen that whilst the $L_1$ and $L_2$ norm showed generally similar performance, the $L_\infty$ 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.

The Mahalanobis distance performed very poorly for image 13_1_s, which is a surprising result. This implies that it representing difference in terms of the standard deviations moved, rather than linear distance, does not translate well into the output of the descriptor.

It is worth noting that Figures B.1 and B.2 were performed on descriptors which had had an eigenmodel generated from them, and then had the descriptors projected into that space. This was performed in order to allow the Mahalanobis distance to be compared also. However one effect this had is that it artificially increased the performance of the $L_\infty$ norm. The $L_\infty$ norm only compares the largest dimension of difference between two points, therefore when the descriptors are projected into the eigenmodel space, the direction of largest variation is aligned with the first dimension of the descriptor. This gives the $L_\infty$ norm will often use this dimension of largest variation as its differentiator, leading to artificially higher results than if the descriptors were naturally aligned with the dimensions. This is illustrated in Figurefig:dist-meas-nopca-books.


(a) Query image



(b) Resultant graph

Figure B.3: Gridded RGB histogram results for query image 13_1_s

# C Test Methodology

In order to test the system, a unit test has been written for all of the descriptor and distance measure functions, as well as several other key functions. Each of these tests complies with MATLAB's unit testing framework, so can be executed using the `runtests` command. A description of the tests for each function is detailed in Table C.1.

Table C.1: List of tests implemented

| **Function Tested** | **Tests implemented** |
|---|---|
| `vs_compute_histogram` | • Two tests against manually calculated data sets. |
| `vs_compute_rgb_histogram` | • Four tests against test images with known colour distributions, against manually calcualated histograms. |
| `vs_construct_eigenmodel` | • Test eigenmodel dimensions and against Eigen_Build.<br>• Test projected descriptors against Eigen_Project.<br>• Test PCA applied eigenmodel against Eigen_Deflate.<br>• Test PCA applied projected descriptors against Eigen_Project. |
| `vs_edge_detect` | • Test operation against library function with three test images.<br>• Test operation against library function for a sample image. |
| `vs_grid` | • Test 1x1 grid output using computation function that returns entire image.<br>• Test 2x2 grid output using computation function that returns entire image. |
| `vs_L1_norm` | • Test output of zero vector against itself.<br>• Test output of random vector against itself.<br>• Test two manually calculated results.<br>• Test against library function for random vectors. |
| `vs_L_Inf_norm` | • Test output of zero vector against itself.<br>• Test output of random vector against itself.<br>• Test two manually calculated results.<br>• Test against library function for random vectors. |
| `vs_L_2_norm` | • Test output of zero vector against itself.<br>• Test output of random vector against itself.<br>• Test two manually calculated results.<br>• Test against library function for random vectors. |
| `vs_L_2_norm_squared` | • Test output of zero vector against itself.<br>• Test output of random vector against itself.<br>• Test two manually calculated results.<br>• Test against library function for random vectors. |
| `vs_mahalanobis_distance` | • Test output of zero vector against itself.<br>• Test output of random vector against itself.<br>• Test three manually calculated results. |

## C.1 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:

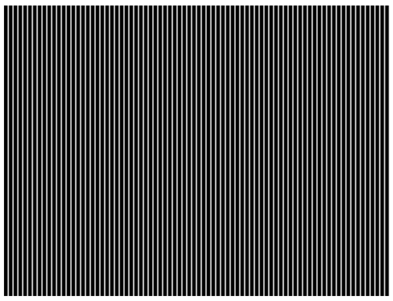
- A uniform colour.
- A uniform pattern.
- A combination of the above.

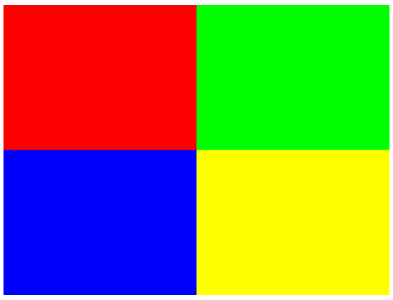
- A grid of several of the above images.

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

A sample of the test images is shown in Figure C.1

(a) A uniform colour test image


(b) A uniform pattern test image


(c) A grid of uniform colour images

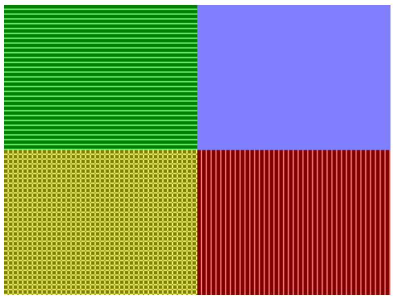

(d) A grid of transposed colour and pattern test images

Figure C.1: Example test images