UNIVERSITY OF SURREY

Visual Search of an Image Collection

EEE3032 – Computer Vision and Pattern Recognition

Georgia Blanco-Litchfield

URN: 6440647

Department of Electronic Engineering

Faculty of Engineering and Physical Sciences

University of Surrey

Abstract

This report reviews the main techniques used in a basic visual search system, including the various descriptors that can be obtained from images and several methods to compute the similarity of images to produce a sorted image collection. This is then followed by an in-depth discussion of experimental results of a developed program (written in MATLAB [1]) to visual search a dataset of 591 images in 21 categories developed by Microsoft [2].

Contents

1 Visual Search

A visual search is a task in computer vision and image processing which involves using a set (or database) of images and comparing with a chosen (query) image to measure how similar they are with the query. The images in the set are then sorted and ranked in order of similarity by the similarity score.

Visual search can use artificial intelligence (AI) on a chosen image to discern the content of the image itself and other images to return any that are related to the query.

There are many ways to decide upon how 'similar' an image is to the query; these are called image descriptor techniques.

1.1 Image Descriptor Techniques

Image descriptors are used to describe the basic features or characteristics of an image. They contain information about the image and for visual search, the distance between two descriptors shows the relationship to one another – effectively how 'similar' the image is with reference to the characteristic the descriptor evaluates. This is shown in Figure 1.

Once the image descriptors have been calculated, they can be projected into a feature space and for Visual search can be implemented through sorting the images using the value of the distance from the chosen query. The smaller the distance, the more 'similar' the image is to the query and so they are sorted in increasing distance.

The primary descriptors used in visual search are shape, texture and colour and the descriptors created should be as discriminative as possible whilst being as compact as possible.

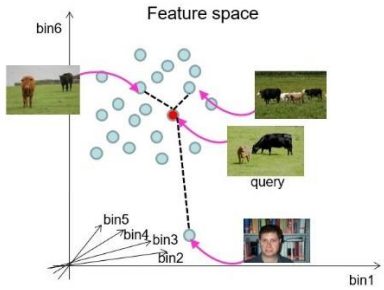


Figure 1: An example of a feature space of image descriptors and their distance to the query [3].

1.1.1 Global Colour Histogram

The global colour histogram represents the overall colour distribution of an image. It is used to detect similar images through evaluating the red, green and blue values (RGB values) of pixels in an image and plotting them onto a 3-dimensional. The plot is then subdivided into equal divisions (bins), called quantising, with points inside those bins. Once quantised, a histogram is created from the bins with a count of the number of points in each bin index. The histogram is then normalised to ensure that all images sizes are treated the same.

Suppose we are given a coloured image with RGB pixel colour values as r, g, b = [0,255] and a quantisation number of q. A Global Colour Histogram can be calculated mathematically as:

$$r' = \left[\frac{r * q}{256}\right] \quad g' = \left[\frac{g * q}{256}\right] \quad b' = \left[\frac{b * q}{256}\right] \tag{1}$$

$$bin = r' * q^2 + g' * q + b'$$

(2)

Equation (1) divides the RGB values by 256 to scale them into the range of 0 to 1 (instead of 0 to 255) and multiply it by the amount of divisions (q) specified, these values are then used to calculate which bin index they are assigned to, shown in equation (2).

However, the global colour histogram is not a very reliable measure for a descriptor as it only accounts for the overall colour distribution of the image and does not give any information about the spatial characteristics. Therefore, it is highly compact but lacks discriminative information.

1.1.2 Grid-Based Image Descriptors (Spatial Grid)

Although the Global Colour Histogram can produce visual search results, a better way to calculate descriptors is by using a Spatial Grid. The image can be divided into a grid of regular specified intervals. Each cell of the grid can then be evaluated in terms of the chosen characteristic, creating a partial image descriptor. These values can then be concatenated into a vector and used as the overall descriptor for that image. Using a spatial grid creates a higher dimensional descriptor and therefore a more discriminative visual search.

1.1.2.1 Colour

Spatial grid colour image descriptor is constructed through taking each cell in the grid and calculating it's average RGB colour, giving a mean value for red, green and blue as a partial image descriptor, shown in equation (3).

$$Cell_x = \frac{1}{W_X H_x}[r_x \quad g_x \quad b_x]$$

(3)

These values are then concatenated into a vector to obtain the complete image descriptor.

Suppose an image was gridded into 8x8, each cell would deliver a red, green and blue value. 64 cells of RGB values would give a vector of 192 values.

$$Descriptor = [R_1 \quad G_1 \quad B_1 \quad \cdots \quad R_{64} \quad G_{64} \quad B_{64}]$$

(4)

Therefore, the descriptors are higher dimensional than that of the Global Colour Histogram, improving the visual search through being more discriminative.

1.1.2.2 Texture

Spatial image descriptors can also be computed with texture. This method is largely similar to computing with colour as above, but the histogram is computed with textures/edges and becomes an Edge Orientation Histogram (EOH).

An EOH is established through grey-scaling the image and then running a convolution over it with for both x and y. The Sobel filter can be used as the edge detection filter, this filter smooths the image before detection:

$$\frac{\partial f}{\partial x} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \frac{\partial f}{\partial y} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(5)

This gives an output of two images; the differentiation of the image with respect to x and the differentiation with respect to y. Once these two images are obtained, for each pixel the magnitude and the orientation of the edges can be evaluated:

$$magnitude = \sqrt{\left(\partial f/_{\partial x}\right)^2 + \left(\partial f/_{\partial y}\right)^2} \tag{6}$$

$$\theta = atan\left(\frac{\partial f/_{\partial y}}{\partial f/_{\partial x}}\right) \tag{7}$$

The descriptor from a gridded images edge magnitudes and orientations can then be calculated. The initial stage of this process is to decide on a magnitude threshold, this is used to ensure only strong edges are evaluated for the descriptor and any noisy regions are filtered out. Each individual cell is then quantised into equal intervals (bins) between 0 and 2π and evaluated on its edge orientations for the pixels inside the cell. A frequency histogram can be built by evaluating how many pixels of a certain orientation goes into each interval. The histogram for each cell can then be normalised and concatenated into a vector to obtain the complete spatial grid texture descriptor.

Suppose an image was gridded into 8x8 and an orientation quantisation level of 8 is decided, each cell would deliver 8 edge orientation values. 64 cells of 8 values would give a vector of 512 values:

$$Descriptor = [E1_1\ E2_1\ E3_1\ E4_1\ E5_1\ E6_1\ E7_1\ E8_1 \cdots E1_{64}\ E2_{64}\ E3_{64}\ E4_{64}\ E5_{64}\ E6_{64}\ E7_{64}\ E8_{64}] \tag{8}$$

1.1.2.3 Colour and Texture

To obtain a descriptor of both colour and texture using spatial gridding methods 1.1.2.1 and 1.1.2.2 can be applied simultaneously. For each cell in the gridded image, a vector can be obtained for texture and for colour, and then these can be concatenated together.

Consider another image was gridded into 8x8 and an orientation quantisation level of 8 is decided, each cell would deliver 8 edge orientation values and average RGB values. 64 cells of 8 edge and 3 colour values would give a vector of 704 values:

$$Descriptor_{Colour} = [R_1\ \ G_1\ \ B_1\ \ \cdots\ \ R_{64}\ \ G_{64}\ \ B_{64}] \tag{9}$$

$$Descriptor_{Texture} = [E1_1\ E2_1\ E3_1\ E4_1\ E5_1\ E6_1\ E7_1\ E8_1 \cdots E1_{64}\ E2_{64}\ E3_{64}\ E4_{64}\ E5_{64}\ E6_{64}\ E7_{64}\ E8_{64}] \tag{10}$$

$$Descriptor = \begin{bmatrix} E1_1\ E2_1\ E3_1\ E4_1\ E5_1\ E6_1\ E7_1\ E8_1\ R_1\ G_1\ B_1 \\ \cdots \\ E1_{64}\ E2_{64}\ E3_{64}\ E4_{64}\ E5_{64}\ E6_{64}\ E7_{64}\ E8_{64}\ R_{64}\ G_{64}\ B_{64} \end{bmatrix} \tag{11}$$

Using Spatial gridding with colour and texture produces very discriminant descriptors but they have very high dimensionality and therefore lacks compactness.

1.2 Principle Component Analysis (PCA)

Looking at section 1.1, the more discriminative the descriptor, the less compact it is. Therefore, the feature space becomes high dimensional. This then brings the Curse of Dimensionality where the more dimensions there are in the feature space, the harder it is to spot a pattern.

PCA is a statistical approach to project a high-dimensional feature space into a lower-dimensional space using orthogonal transformation. This gives a more robust model. To compute PCA, an Eigen Model must be built from the data. A d dimensional spaces given rise to d eigenvectors and eigenvalues that are associated with it, they can be very small or even 0 if no variation in the direction of the values. Suppose you have a dataset of n 2D points, the procedures to build an Eigen Model are:

1. Arrange the points in the dataset in a matrix:

$$\boldsymbol{V} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & \cdots & x_n \\ y_1 & y_2 & y_3 & y_4 & \cdots & y_n \end{bmatrix} \tag{12}$$

2. Calculate the mean μ and subtract it from the points:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} v_i \tag{13}$$

$$\bar{v} = v - \mu \tag{14}$$

3. Calculate the covariance matrix *C* from the values obtained in equation (14):

$$C = \frac{1}{n}\bar{v}.\bar{v}^T \tag{15}$$

4. Eigenvalue decomposition (EVD) will then be computed on the Eigen Model to achieve PCA since C can be factorised in terms of eigenvector and eigenvalue [**U V**]:

$$[\boldsymbol{U} \quad \boldsymbol{V}] = eig(C) \tag{16}$$

5. Now, the model can be reduced to create a lower-dimensional feature space through computing the covariance matrix and then filtering out any space with little or no variation. The covariance matrix can be calculated using U V and the transpose of U:

$$Cov = \boldsymbol{U}\boldsymbol{V}\boldsymbol{U}^T \tag{17}$$

6. To filter the eigenvectors and values, there are two procedures:
    a. Use a threshold value to drop the null space (space with little or no variation)
    b. Examine the covariance matrix and reduce it by a chosen number.
7. Finally, once one of the above has been computed, the data can be projected into a lower-dimensional space:

$$\boldsymbol{N} = \boldsymbol{K}^{-1}\boldsymbol{M}$$

(18)

Whereby **K** is the filtered version of **U** and **M** being the original data in its higher dimensional state.

1.3 Distance Measures

Once the feature space has been created, in order to evaluate the similarity of the images to the query the distance between them is computed. Here, we will look at three different distance metrics; l_1 norm, l_2 norm and Mahalanobis distance.

1.3.1 L_1 Norm

L_2 norm, also known as the Manhattan distance is purely the sum of the absolute difference of points in each axis in a Coordinate plane. Suppose two vectors a, b where

$$\boldsymbol{a} = (a_1 \;\; a_2 \;\; a_3 \;\; \cdots \;\; a_i) \quad \boldsymbol{b} = (b_1 \;\; b_2 \;\; b_3 \;\; \cdots \;\; b_i)$$

(19)

L_1 can be express as:

$$distance(\boldsymbol{a}, \boldsymbol{b}) = \sum_{i=1}^{n} |a_i - b_i|$$

(20)

1.3.2 L_2 Norm

L_2 norm is a very common distance metric, also known as Euclidean Distance. It is defined as the straight-line difference between two points in a coordinate plane. Given two vectors **a**, **b**; L_2 norm is computed as

$$distance(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

(21)

1.3.3 Mahalanobis Distance

The Mahalanobis distance is the amount of distance a point (vector) and a distribution. It is the multivariate equivalent of L_2 norm.

In this case, it can be described as the number of standard deviation's away from the mean of distribution in each principal component axis. Once PCA is computed, the Mahalanobis distance can be calculated between any image descriptor and the Eigen model.

The formula for the Mahalanobis distance is

$$distance = \sqrt{\boldsymbol{V}^{-1}\boldsymbol{U}^{\boldsymbol{T}}(x - \mu)}$$

(22)

Where $\mathbf{V}^{-1}$ is the inverse of Eigenvalues, $\mathbf{U}^{\mathbf{T}}$ is the transpose of the eigenvectors, x is the test point in question and μ is the mean.

2 Experimental Results

The purpose for this assignment is to use a database of images develop a program that selects a 'random' query image and then searches through the dataset to deliver the top 20 images found ordered by their similarity to the chosen query. The experiments are implemented using MATLAB [1] and the Microsoft Research dataset (MSVC-v2) [2].

The chosen images to be the queries for the visual search are as follows:

Figure 2: Query images were chosen for Visual search experimental results.

These images have been chosen as they are a good representation of the category they belong to and have various dimensions.

2.1 Evaluation Methodology: Precision and Recall

To be able to evaluate the performance of the visual search implementation in this assignment, precision and recall (PR) statistics are calculated and a PR curve is plotted for a chosen amount of retrieved results.

Precision is the proportion of the retrieved results that are relevant to the query image:

$$Precision = \frac{|\{relevant\ results\} \cap \{retrieved\ results\}|}{|\{retrieved\ results\}|} \tag{23}$$

Recall is the fraction of how many of the returned results are relevant when considering the whole data set:

$$Recall = \frac{|\{relevant\ results\} \cap \{retrieved\ results\}|}{|\{relevant\ results\}|} \tag{24}$$

The visual search system can then be evaluated through a PR curve and an average precision (AP) can be calculated for a dataset of size M:

$$AP = \sum_{n=1}^{M} \frac{Precision(n) \times rel(n)}{\{relevant\ results\}} \tag{25}$$

If is relevant, then *rel(n) = 1* and *rel(n) = 0* otherwise.

The mean average precision (MAP) can then be calculated for Z images:

$$MAP = \sum_{n=1}^{Z} \frac{AP_n}{Z} \tag{26}$$

The plot and AP for the experimental results are computed in **`Precision_Recall.m`** the function takes in the distance measures of all images **`dist`**, a histogram of how many images are in each category **`hgram`** and the queries category **`qimg_cat`**. The function then

computes Equation 23 and Equation 24 for the dataset and plots the PR curve. The AP is then calculated.

2.2 Results

Due to the limitations of this report for each chosen category and query image the average precisions (AP) and the mean average precisions (MAP) will be shown in a table. Each AP is calculated to 5 decimal places and each MAP to 4 decimal places.

The visual search output for the two best and worst AP images for each category of 15 images the Precision and Recall Curves from whole dataset's visual search will be available in the Appendix.

2.2.1 Global Colour Histogram

The global colour histogram is implemented in MATLAB by the script **`ComputeRGBHistogram.m`**. The function takes in the coloured image **`img`** and quantisation level **`q`**. It then computes Equation (1) and then Equation (2) to obtain the average RGB colours of each pixel. From this, the function constructs a histogram and normalises it. This is then returned to **`cvpr_computedescriptors.m`**.

The descriptors have been computed with 4 different quantisation levels below:

Category	**Average Precision (AP)**				
	Q=3	Q=6	Q=9	Q=18	Q=36
Books	0.23707	0.25513	0.288	0.1963	0.17362
Planes	0.29822	0.34666	0.3173	0.24815	0.14973
Cars	0.47354	0.41803	0.42392	0.38939	0.29363
Selfies	0.27473	0.34198	0.35729	0.31966	0.2673
Bikes	0.28452	0.36552	0.42621	0.44759	0.26599
Trees	0.3319	0.41983	0.41987	0.33821	0.25716
MAP	0.2692	0.3579	0.3721	0.3232	0.2346

Table 1: AP and MAP for computed global colour histogram visual search for different quantisation levels.

From Table 1 it is shown that at Q=9, the majority of the test categories performed better than that of the other quantisation levels, shown in the AP of each category and the MAP of the overall results. At the higher quantisation levels, Q = 18 and Q = 36, the pixel values are divided into high density and then the image descriptor finds it harder to differentiate unique features accurately. And at lower values such as Q=2, the RGB values can only be shown as two distinct values, which gives a poor result.

2.2.2 Spatial Grid: Colour

Colour Spatial Grids are implemented by the script **`ComputeSpatialColour.m`** with arguments of the coloured image **`img`** and the proposed grid size for the descriptors **`row`**, **`col`**. The function then splits the image into the chosen grid size and computes the mean average RGB values for each cell, in turn, creating the feature vector as shown in Equation (3). To obtain the complete feature vector each cells RGB values are concatenated horizontally (Equation (4)) using the MATLAB built-in function **`horzcat`** and returned to **`cvpr_computedescriptors.m`**.

This function is experimented with the grid sizes below:

Category	Average Precision (AP)			
	2x2	4x4	8x8	16x16
Books	0.25000	0.31568	0.22557	0.18506
Planes	0.50124	0.61782	0.66713	0.66434
Cars	0.18844	0.20543	0.20018	0.15461
Selfies	0.29062	0.17654	0.11537	0.09977
Bikes	0.25879	0.35524	0.38222	0.38892
Trees	0.35632	0.54884	0.58179	0.57429
MAP	0.3076	0.3610	0.3620	0.3445

Table 2: AP and MAP for computed colour spatial grid visual search for different grid sizes.

From Table 2 the MAP's are similar numbers for all experimented spatial grids, with gridding by 4x4 and 8x8 having the majority of the highest AP's. the best category for a colour grid search are planes, this may be because the planes in the category were of similar size, colour and on the grass with a blue sky in the background. The worst category showed to be cars as the dataset had a variety of images of different coloured cars and orientations.

2.2.3 Spatial Grid: Texture

Texture Spatial Grids using Edge Orientation Histograms are implemented in the function **`ComputeSpatialTexture.m`** with arguments of the coloured image **`img`** and the proposed grid size for the descriptors **`row`**, **`col`**.
The functional initially grey-scales the image using MATLAB's built-in function **`rgb2gray`** and then convolutes the grey-scale image with the Sobel filters (shown in Equation (5)) to obtain the edges. The gradient magnitudes and orientations are then calculated using Equation (6) and (7) respectively. Once these are calculated, the script runs through every cell in the **`row`** by **`col`** grid and obtain the magnitude and gradient image of the individual cells which are passed into another script named **`histo_compute`** which loops through every pixel using a magnitude threshold of 0.2 to ignore weak edges and orientation. The orientations are then adjusted to be in the range of [0 2π] and then binned into 8 equal sections (which can be adjusted). The bins are then checked for **`NaNs`** (values do not exist) and these values set to 0. Finally, the orientation histogram is normalised and returned to **`ComputeSpatialTexture.m`** where they are concatenated into a feature vector shown in Equation (8). Once all cells have been computed, the complete texture feature vector is returned to **`cvpr_computedescriptors.m`**.
This function is experimented with the grid sizes below:

Category	Average Precision (AP)			
	2x2	4x4	8x8	16x16
Books	0.39282	0.34390	0.27536	0.08039
Planes	0.17736	0.46211	0.2198	0.08084
Cars	0.42460	0.54849	0.28386	0.11651
Selfies	0.10735	0.09136	0.068825	0.06558
Bikes	0.14188	0.15408	0.15791	0.14503
Trees	0.80643	0.64652	0.22192	0.15037
MAP	0.3417	0.3744	0.2046	0.1065

Table 3: AP and MAP for computed texture spatial grid visual search for different grid sizes.

Table 3 shows that gridding the image by 4x4 and computing EOH on each cell outweighs the precision of the other grid sizes. This could be due to the edges in these grids being overall more predominant and expressive of the features in the image whereas in a 16x16 grid the search is unable to give good edge results. The best image for spatial texture showed to be trees, this is mainly because the trees in the data set were all of the similar leaf growth giving a similar texture.

2.2.4 Spatial Grid: Colour and Texture

The descriptor for the Colour and Texture Spatial Grid is computed in **`ComputeSpatialTextureColour.m`** and is simply the implementation of Section 2.2.2 and 2.2.3 but the final feature vector is the concatenation of Texture and average RGB values, as shown in Equation (11), for all cells in the chosen grid size. Once all cells have been computed, the complete colour and texture feature vector is returned to **`cvpr_computedescriptors.m`**.
This function is experimented with the grid sizes below:

Category	**Average Precision (AP)**			
	2x2	4x4	8x8	16x16
Books	0.44553	0.38181	0.27981	0.084249
Planes	0.50343	0.55802	0.36544	0.12036
Cars	0.49141	0.49041	0.29863	0.12834
Selfies	0.16466	0.12547	0.073242	0.067126
Bikes	0.29453	0.30714	0.29652	0.21091
Trees	0.86013	0.76425	0.25947	0.20989
MAP	0.4599	0.4379	0.2622	0.1368

Table 4: AP and MAP for computed colour texture spatial grid visual search for different grid sizes.

From Table 4, it is concluded that a 4x4 grid is the most accurate search for all categories in the evaluation of their individual AP's. Furthermore, the MAP of 2x2 calculates to be more accurate however this is due to the Trees category having a similar explanation as that of the Tree categories AP in spatial texture. A 16x16 grid is again the worst gridding for the visual search, the explanation above is also true for this descriptor technique.

2.2.5 Principle Component Analysis and Mahalanobis Distance

For PCA, the aim is to project the image descriptors into a lower-dimensional space and then compute Mahalanobis distance for all image descriptor to the query image to see whether this method obtains better performance than that of the previous experiments. PCA is implemented by the script **`image_pca`** which takes in the image descriptors **`ALLFEAT`**, computes stages 1-4 of the PCA explanation in Section 1.2. The function then sorts the Eigenvalues and vectors in descending size and deflates them by 10 (can be varied) and projects it onto a lower-dimensional space. This is then returned to **`cvpr_visualsearch.m`**. the Mahalanobis distance is then computed in the script **`Eigen_Mahalanobis.m`** this script takes in two descriptors **`F1`** and **`F2`** and the eigenvalues **`e_val`**. The function then returns the Mahalanobis distance between the two points.

PCA is observed for a Global Colour Histogram (GCH) of Q=9, a grid of 4x4 for spatial colour, a grid of 4x4 for spatial texture and a grid of 16x16 for spatial colour and texture. Table 5 includes the values before (B) and after (A) PCA was computed inside the visual search:

Category	Average Precision (AP)							
	GCH		Colour		Texture		Colour and Texture	
	B	A	B	A	B	A	B	A
Books	0.288	0.2646	0.31568	0.25113	0.34390	0.44192	0.084249	0.53942
Planes	0.3173	0.28035	0.61782	0.56411	0.46211	0.32911	0.12036	0.45751
Cars	0.42392	0.45632	0.20543	0.18695	0.54849	0.45192	0.12834	0.72267
Selfies	0.35729	0.21798	0.17654	0.12603	0.09136	0.1873	0.067126	0.2899
Bikes	0.42621	0.32489	0.35524	0.28102	0.15408	0.12322	0.21091	0.32798
Trees	0.41987	0.37378	0.54884	0.50291	0.64652	0.56543	0.20989	0.23101
MAP	0.3721	0.3197	0.3610	0.31869	0.3744	0.3498	0.1368	0.4281

Table 5: AP and MAP for computed visual search before and after PCA for different image descriptors.

From Table 5, it can be shown that the results from PCA stay largely similar to those without. However, the main difference is when computed for a very high-dimensional descriptor, a 16x16 spatial colour and texture that the results drastically improve.

2.2.6 Distance Measures: L_1 and L_2 Norm

When experimenting with the other distance measures a script for L_1 was created, called **`cvpr_compare_man.m`** and for L_2 **`cvpr_compare_euc.m`**. These scripts take in two descriptors **`F1`** and **`F2`** and compute Equation 20 in L_1 script and Equation in L_2 script.

The distance measures L_1 and L_2 Norm are observed for a Global Colour Histogram (GCH) of Q=9, a grid of 4x4 for spatial colour, a grid of 4x4 for spatial texture and a grid of 4x4 for spatial colour and texture:

Category	Average Precision (AP)							
	GCH		Colour		Texture		Colour and Texture	
	L_1	L_2	L_1	L_2	L_1	L_2	L_1	L_2
Books	0.19469	0.288	0.33658	0.31568	0.45181	0.34390	0.4762	0.38181
Planes	0.37181	0.3173	0.65528	0.61782	0.5079	0.46211	0.62927	0.55802
Cars	0.48345	0.42392	0.20991	0.20543	0.5740	0.54849	0.5100	0.49041
Selfies	0.34064	0.35729	0.17528	0.17654	0.11494	0.09136	0.16148	0.12547
Bikes	0.5494	0.42621	0.35941	0.35524	0.1463	0.15408	0.32686	0.30714
Trees	0.47084	0.41987	0.56138	0.54884	0.8458	0.64652	0.88149	0.76425
MAP	0.4018	0.3721	0.3830	0.3610	0.4401	0.3744	0.49755	0.4379

Table 6: AP and MAP for a computed visual search for different distance measures.

Table 6 shows a comparison of the AP's and MAP's of the 6 chosen query images in the data set. It is seen that the MAP of each type of descriptor is higher for the computation of L_1 Norm that that of the L_2 Norm. this could show that the L_2 Norm may not be as suitable for visual search.

3 Conclusions

A major challenge in this project was from the supplied MSVC dataset. This is due to the nature of the categorisation of the images and that some images were repeated, and some objects/features inside the images were included in more than one category. This meant that some of the results were not a true representation of a visual search.

In conclusion, a visual search system has a variety of contributions determining its accuracy and performance. These contributors can involve different descriptor techniques using quantisation and gridding, the dimensionality of the feature space and types of distance measures to decide on similarity.

4 References

[1] MathWorks, “MATLAB,” [Online]. Available: https://uk.mathworks.com/products/matlab.html.

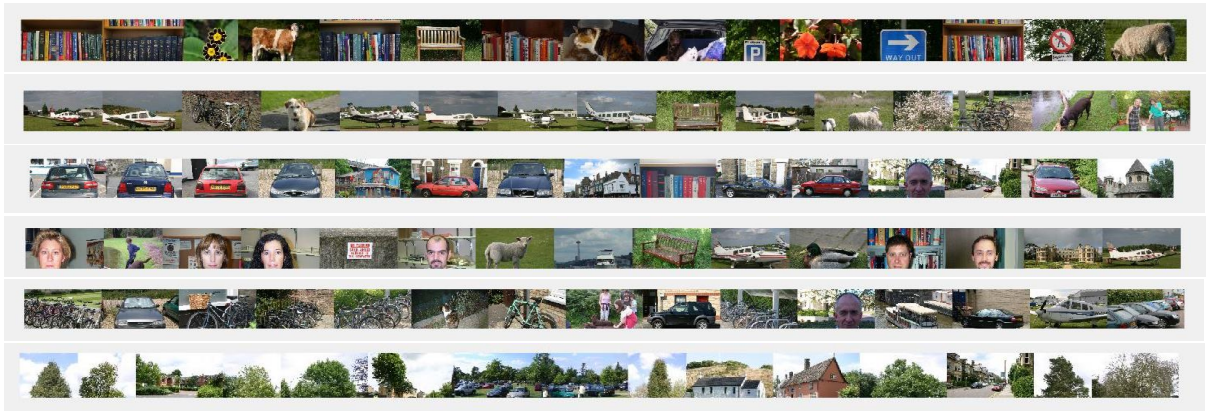
[2] Microsoft, “Download of MSVC-v2,” [Online]. Available: http://download.microsoft.com/download/3/3/9/339D8A24-47D7-412F-A1E-1A415BC48A15/msrc_objcategimagedatabase_v2.zip. [Accessed 03 11 2019].

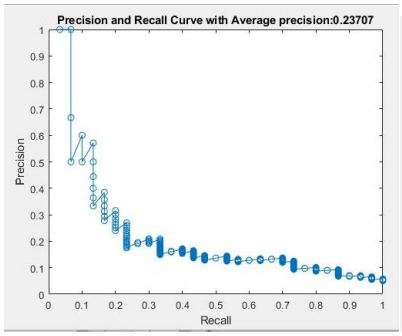
[3] P. J. Collomosse, *Lecture notes for EEE3032 – Computer Vision and Pattern,* 2019.

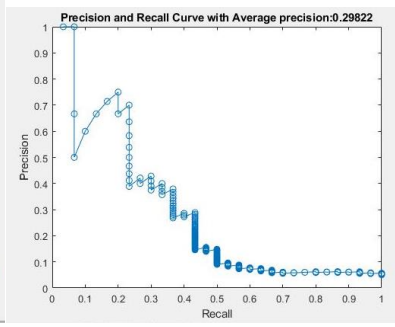
5 Appendix

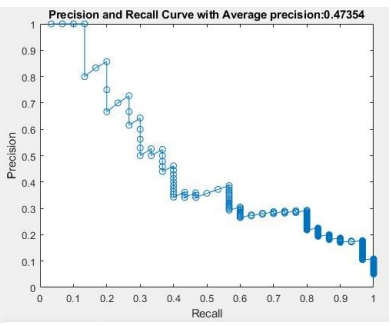
5.1 Global Colour Histogram

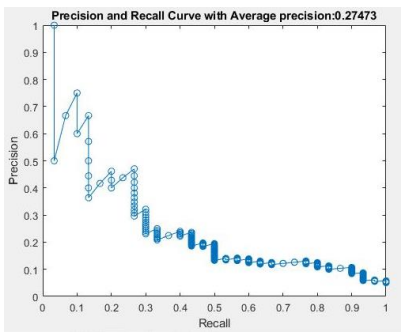
5.1.1 Quantisation Level = 3

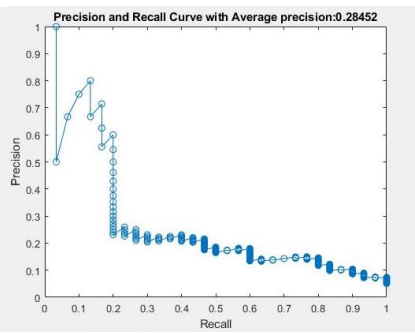


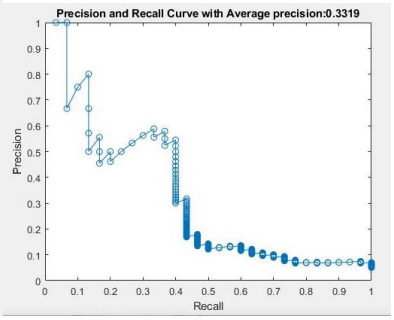




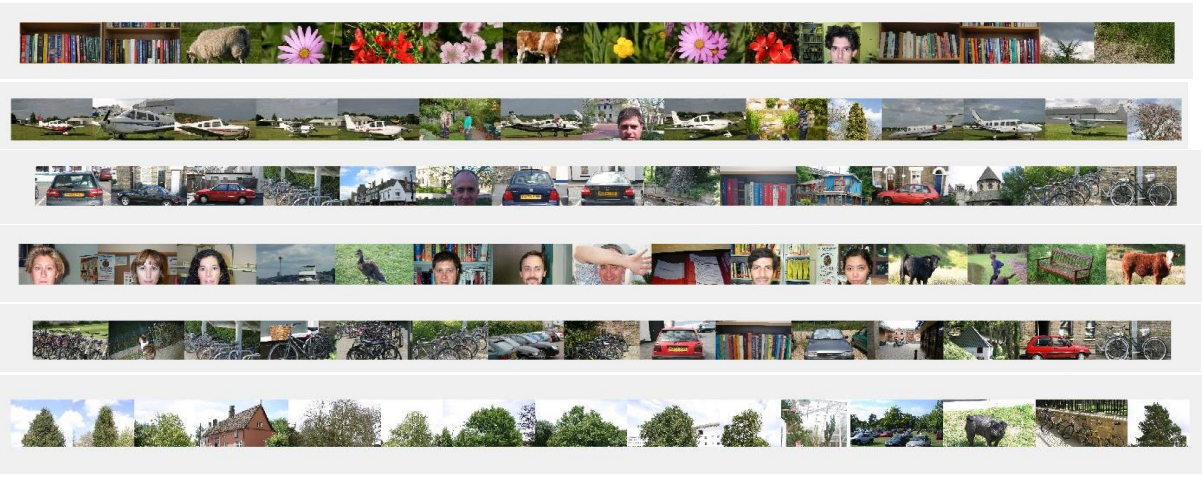


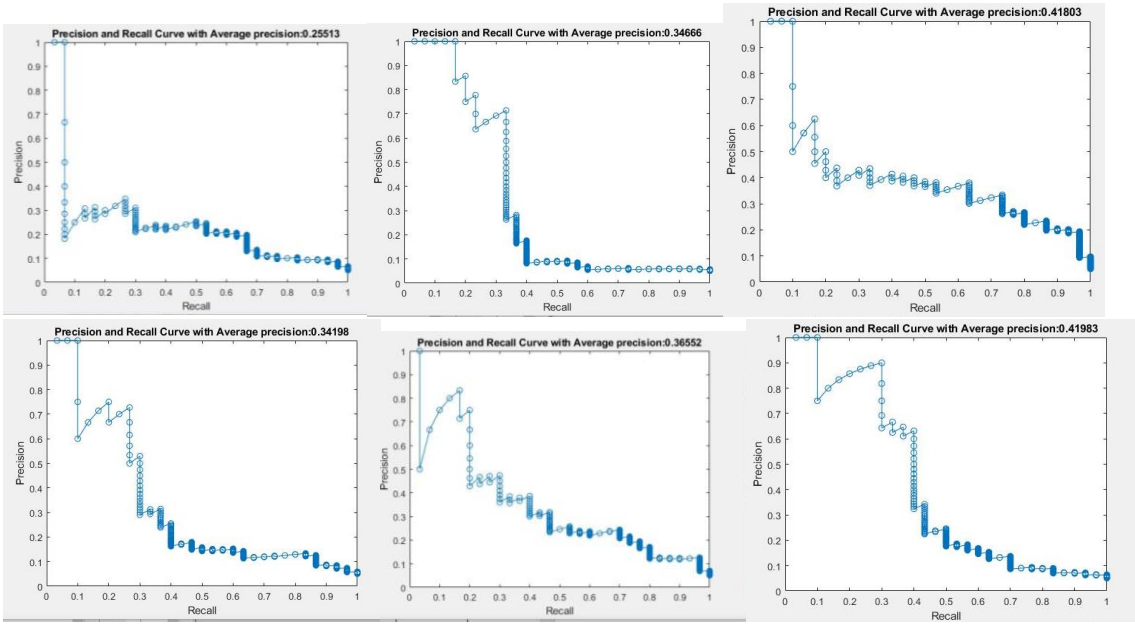




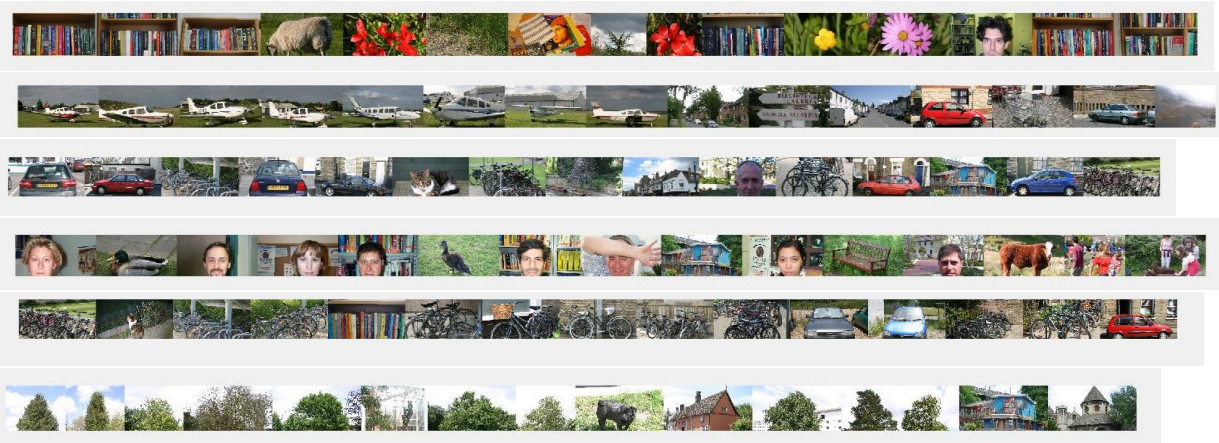


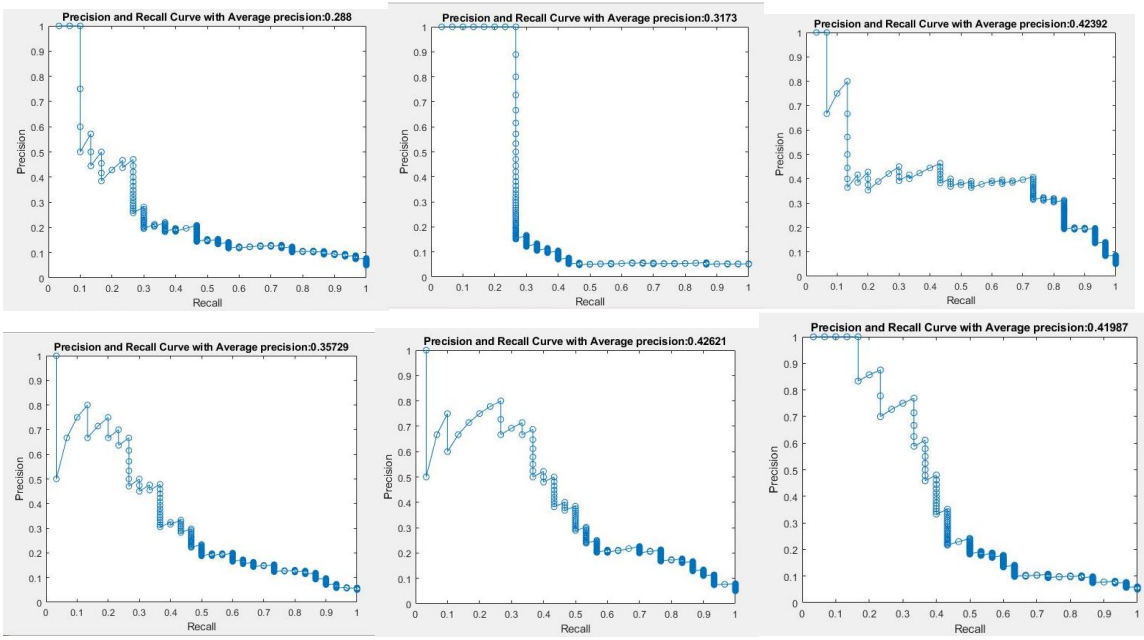
5.1.2 Quantisation Level = 6



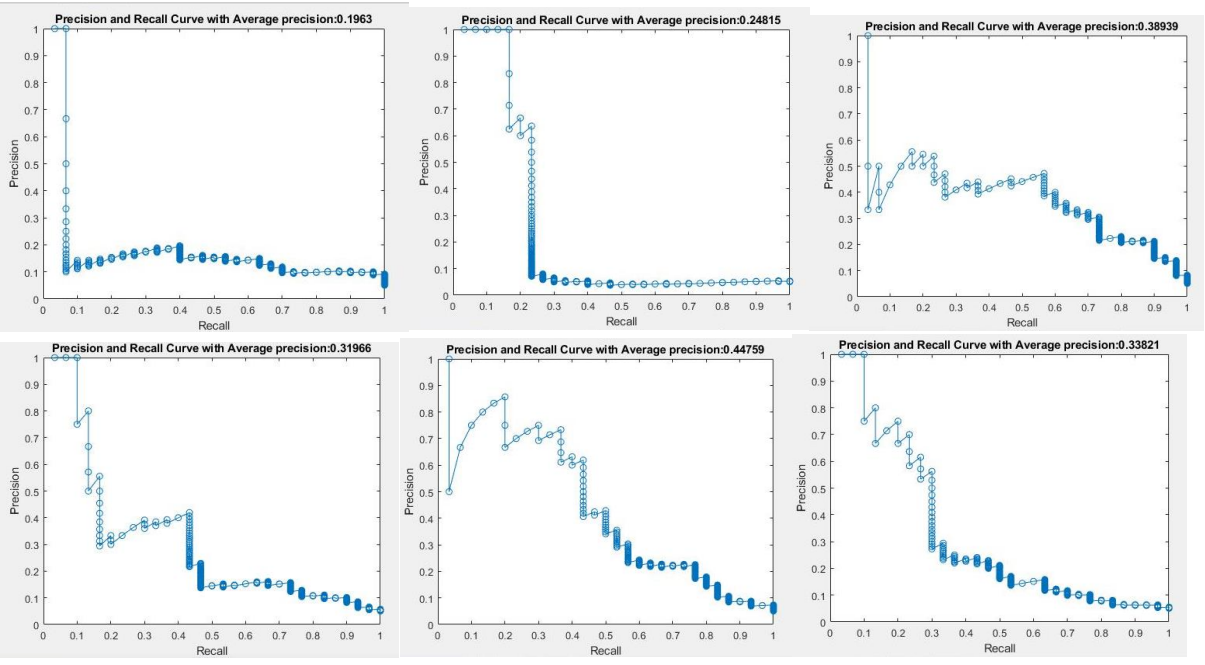


5.1.3 Quantisation Level = 9

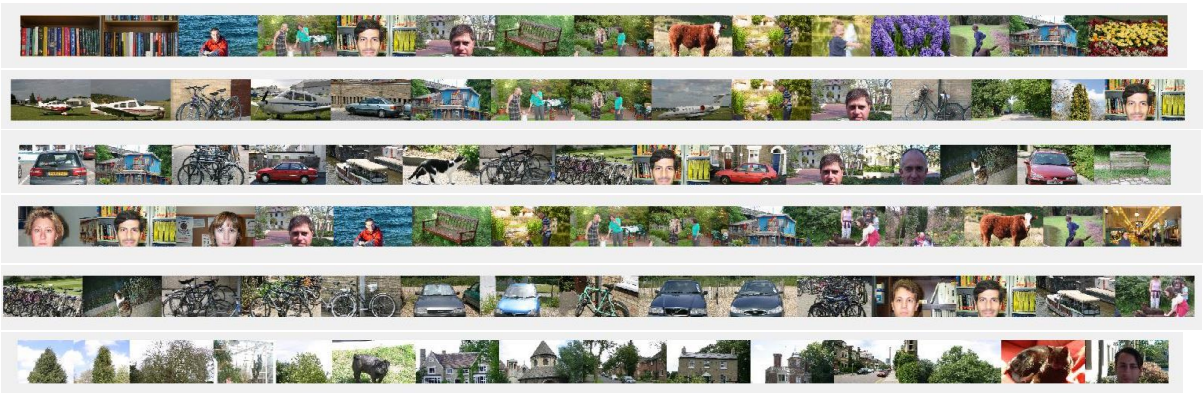


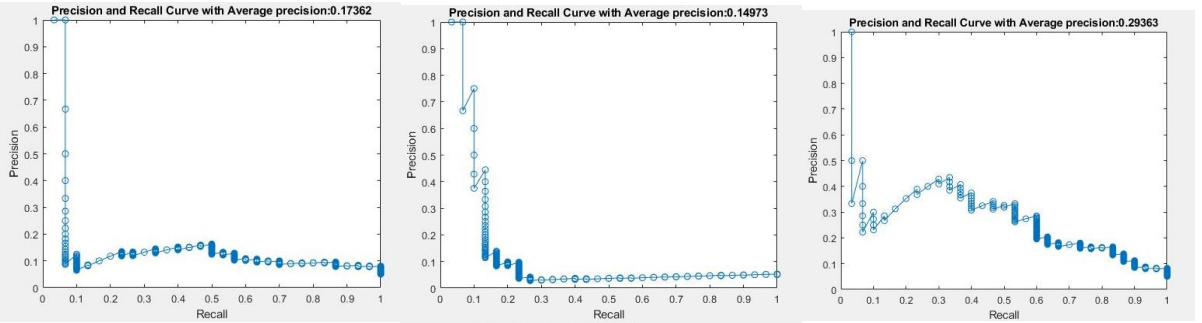


5.1.4 Quantisation Level = 18



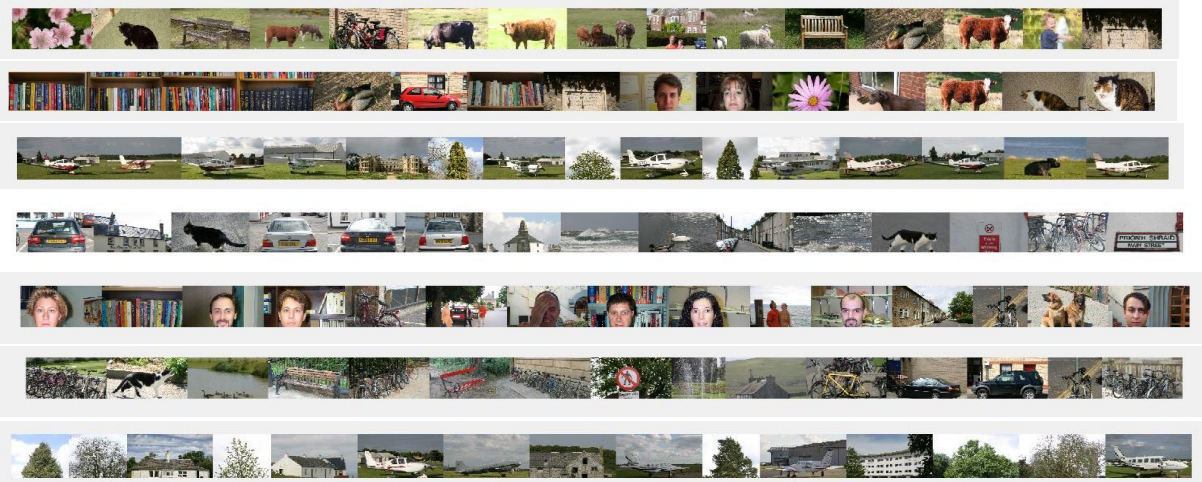
5.1.5 Quantisation Level = 32

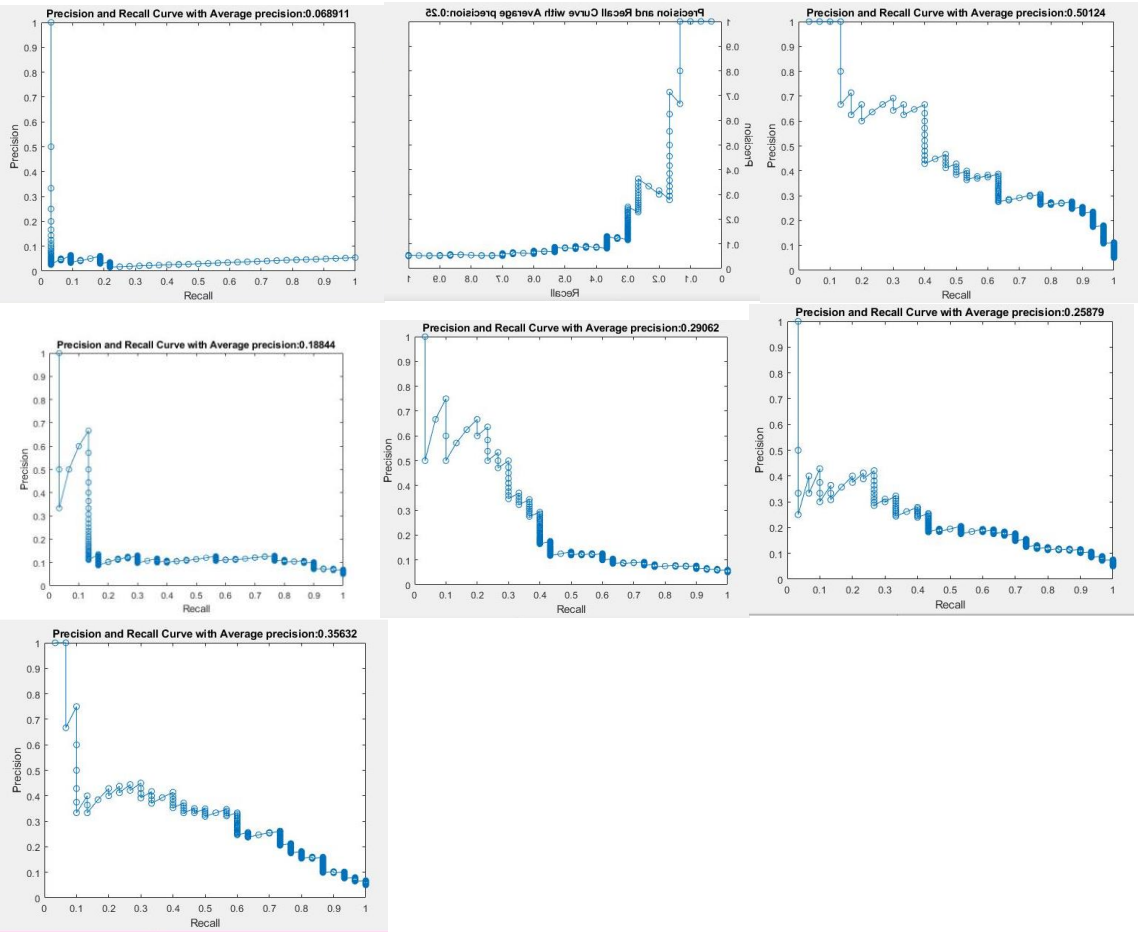




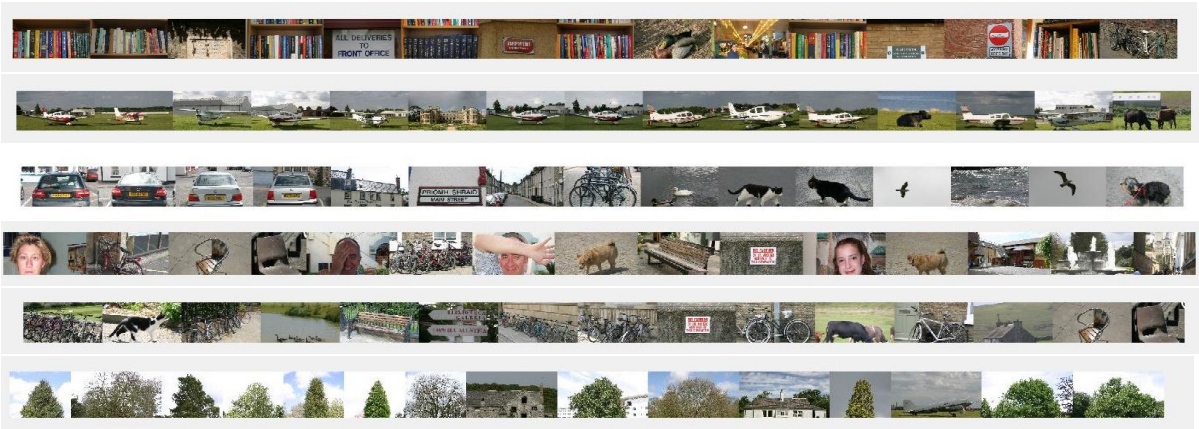
5.2 Spatial Grid: Colour

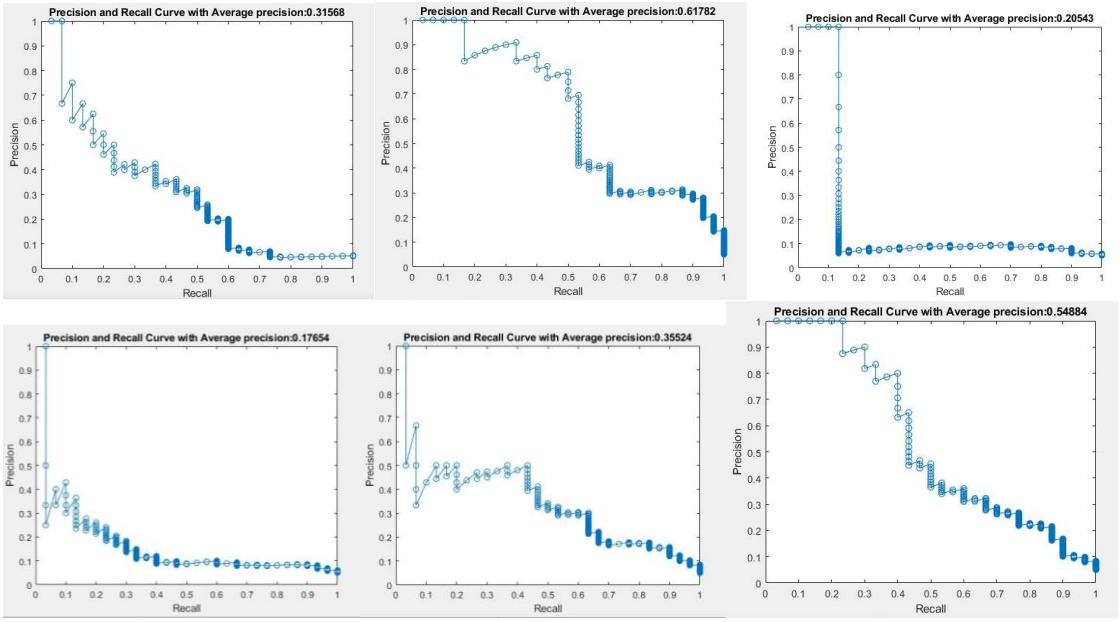
5.2.1 2x2 Grid



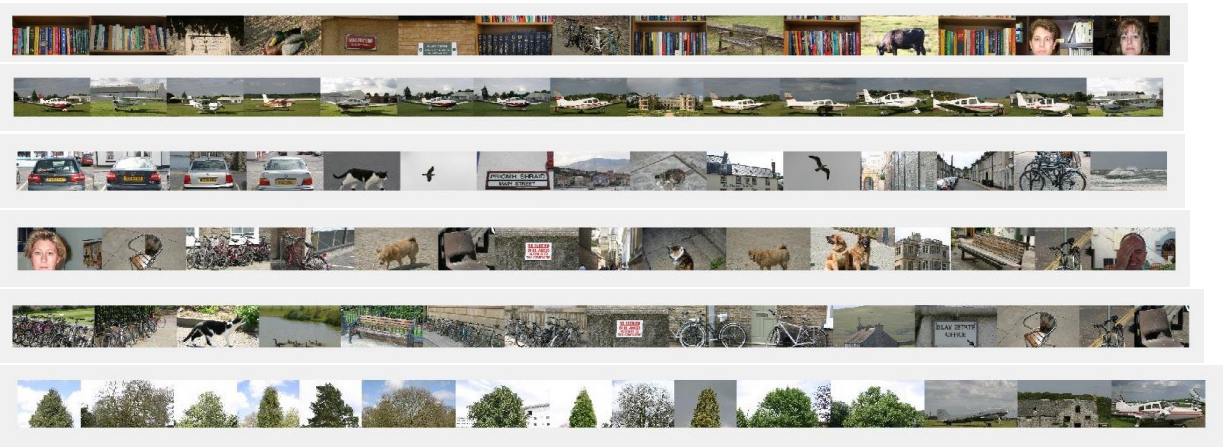

Precision and Recall Curve with Average precision:0.068911
Precision and Recall Curve with Average precision:0.50124
Precision and Recall Curve with Average precision:0.18844
Precision and Recall Curve with Average precision:0.29062
Precision and Recall Curve with Average precision:0.25879
Precision and Recall Curve with Average precision:0.35632
Precision
Recall

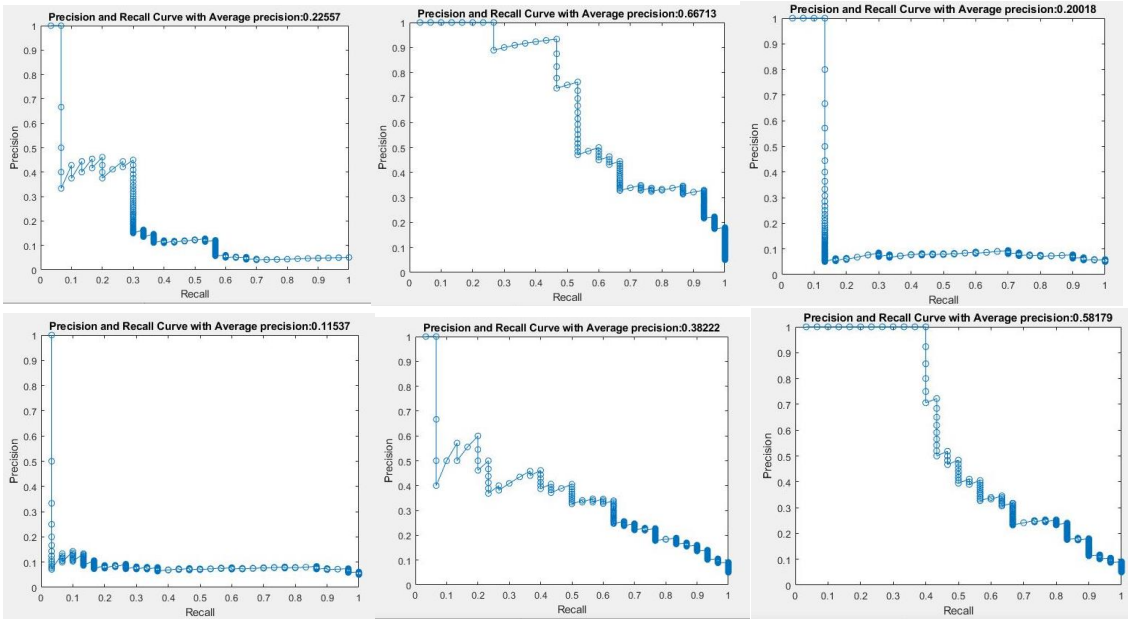
5.2.2 4x4 Grid



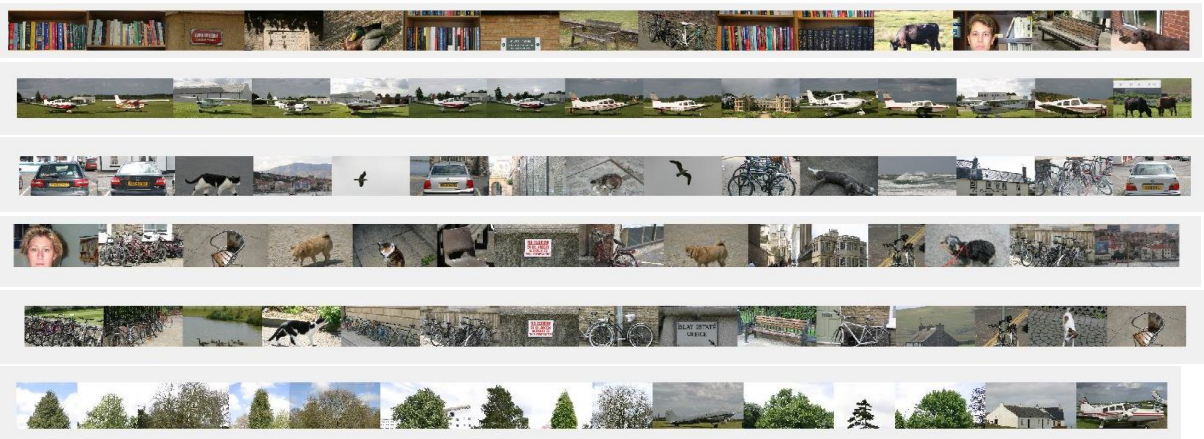

Precision and Recall Curve with Average precision:0.31568
Precision and Recall Curve with Average precision:0.61782
Precision and Recall Curve with Average precision:0.20543
Precision and Recall Curve with Average precision:0.17654
Precision and Recall Curve with Average precision:0.35524
Precision and Recall Curve with Average precision:0.54884
Precision
Recall

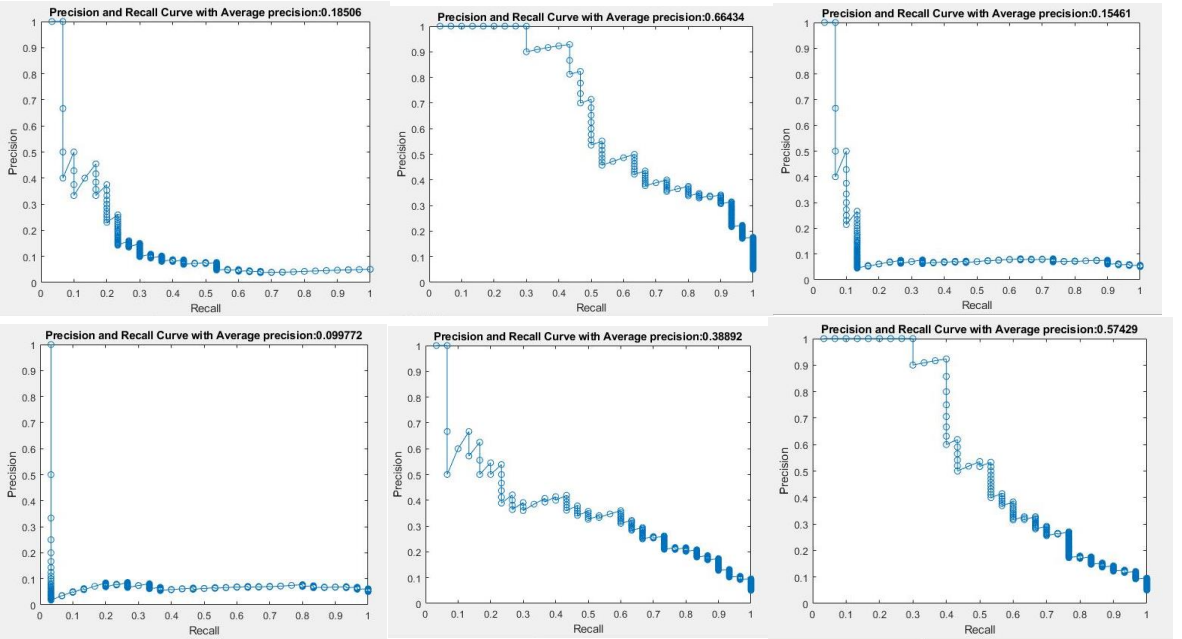
5.2.3 8x8 Grid




Precision and Recall Curve with Average precision:0.22557
Precision and Recall Curve with Average precision:0.66713
Precision and Recall Curve with Average precision:0.20018
Precision and Recall Curve with Average precision:0.11537
Precision and Recall Curve with Average precision:0.38222
Precision and Recall Curve with Average precision:0.58179
Precision
Recall

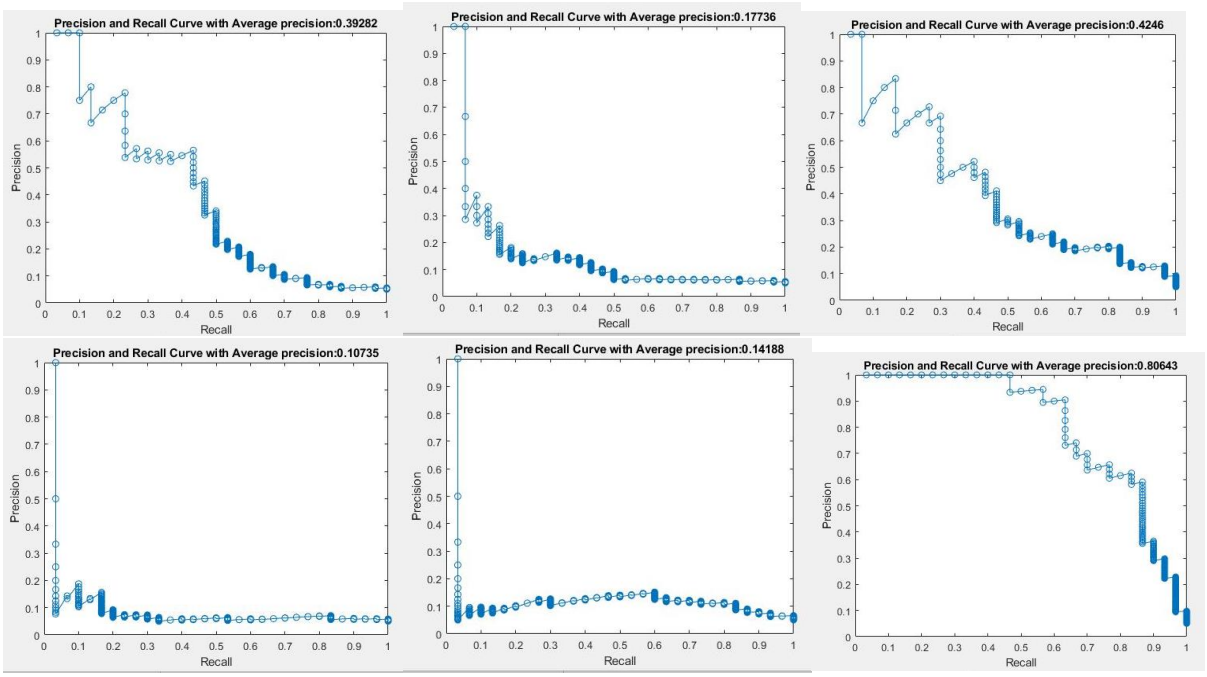
5.2.4 16x16 Grid



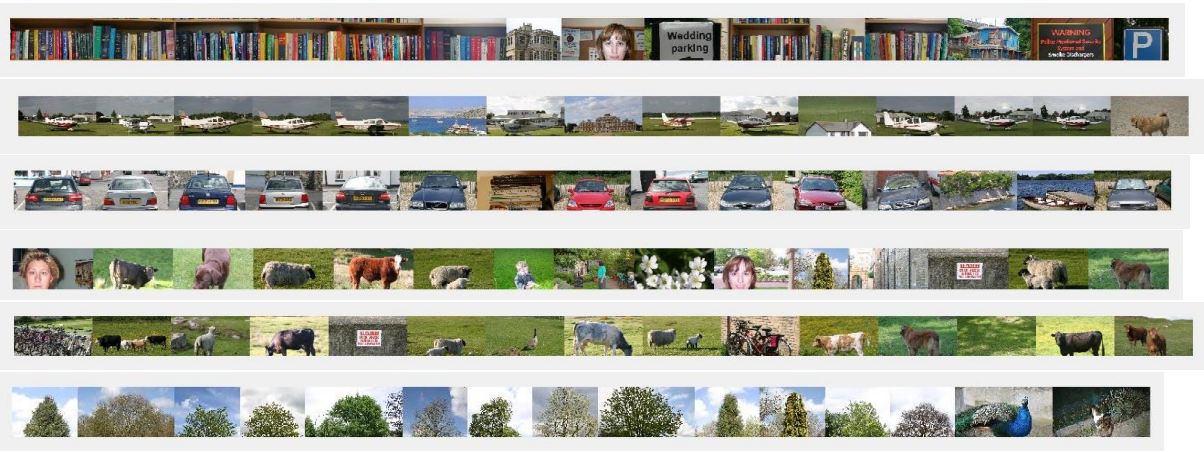


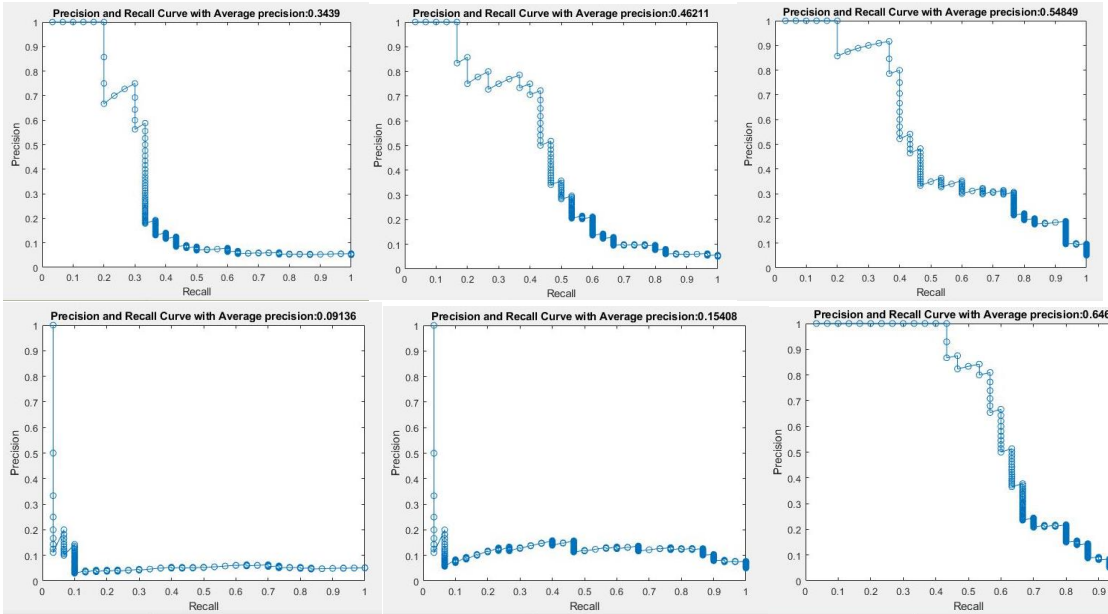
5.3 Spatial Grid: Texture

5.3.1 2x2 Grid


Precision and Recall Curve with Average precision:0.39282
Precision and Recall Curve with Average precision:0.17736
Precision and Recall Curve with Average precision:0.4246
Precision and Recall Curve with Average precision:0.10735
Precision and Recall Curve with Average precision:0.14188
Precision and Recall Curve with Average precision:0.80643
Precision
Recall

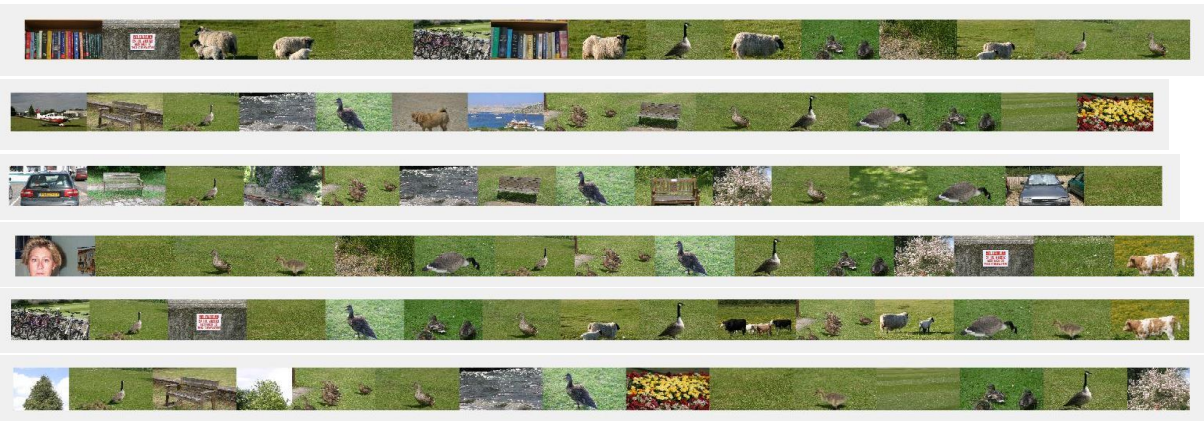
5.3.2 4x4 Grid

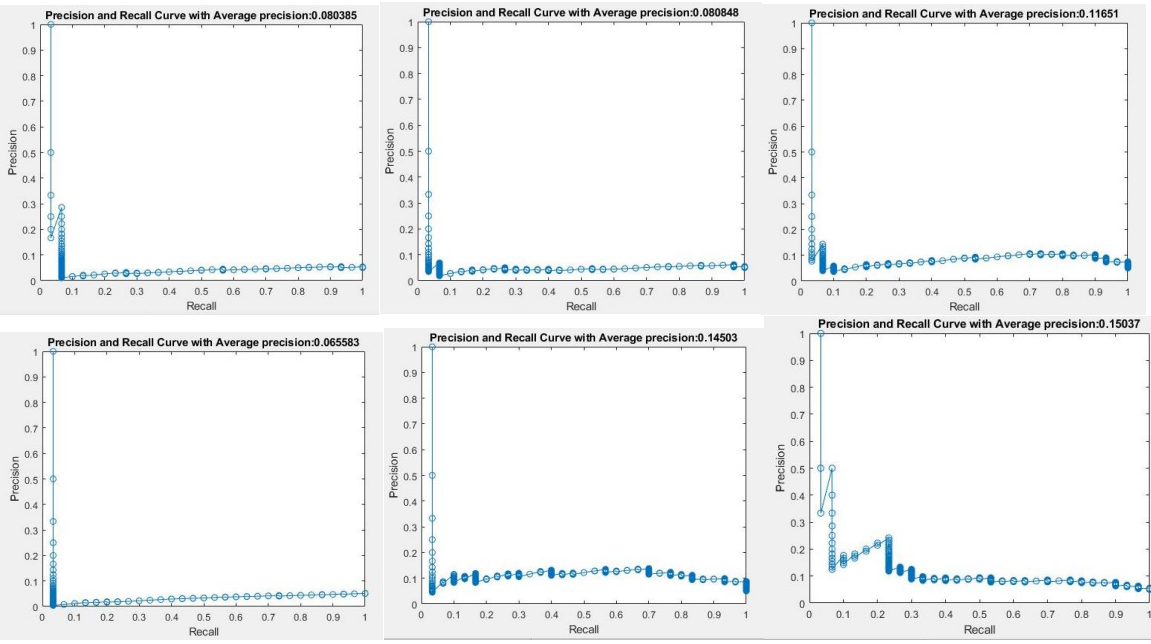

Wedding
parking
P


Precision and Recall Curve with Average precision:0.3439
Precision and Recall Curve with Average precision:0.46211
Precision and Recall Curve with Average precision:0.54849
Precision and Recall Curve with Average precision:0.09136
Precision and Recall Curve with Average precision:0.15408
Precision and Recall Curve with Average precision:0.64652
Precision
Recall

5.3.3 8x8 Grid

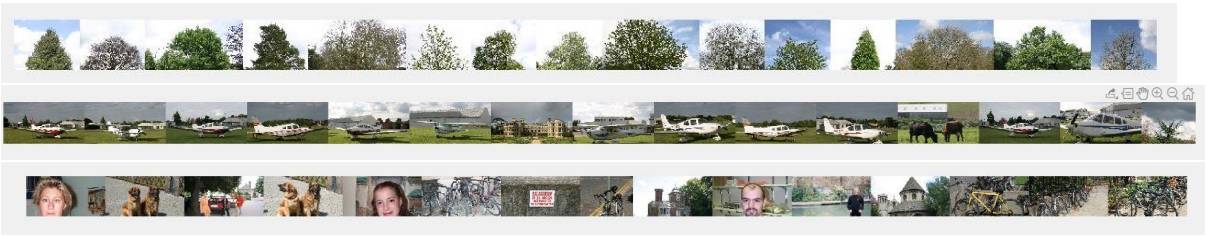
5.3.4 16x16 Grid

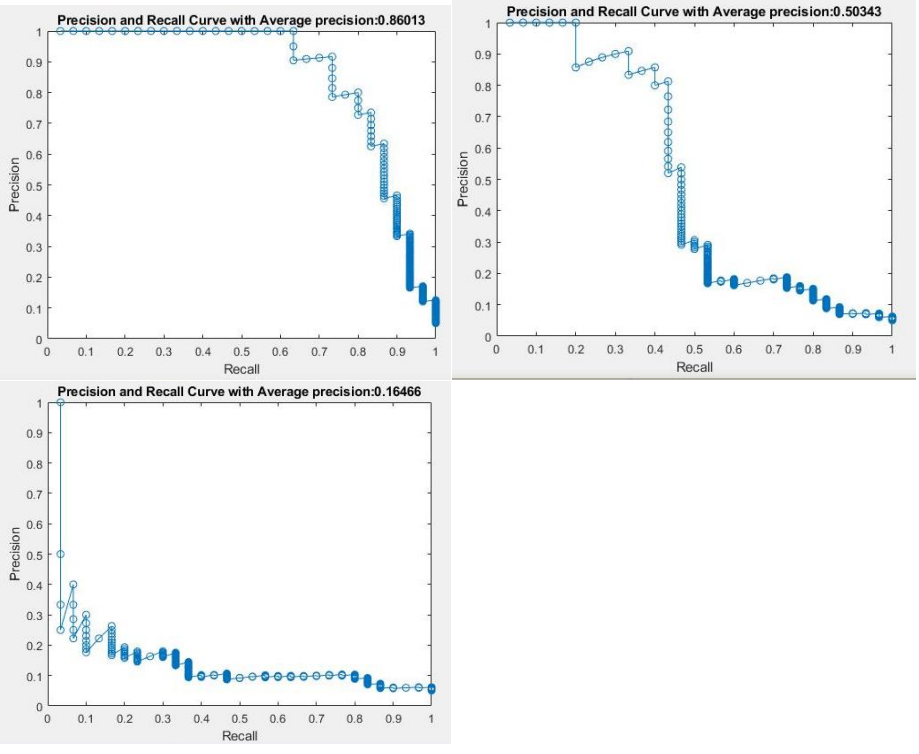




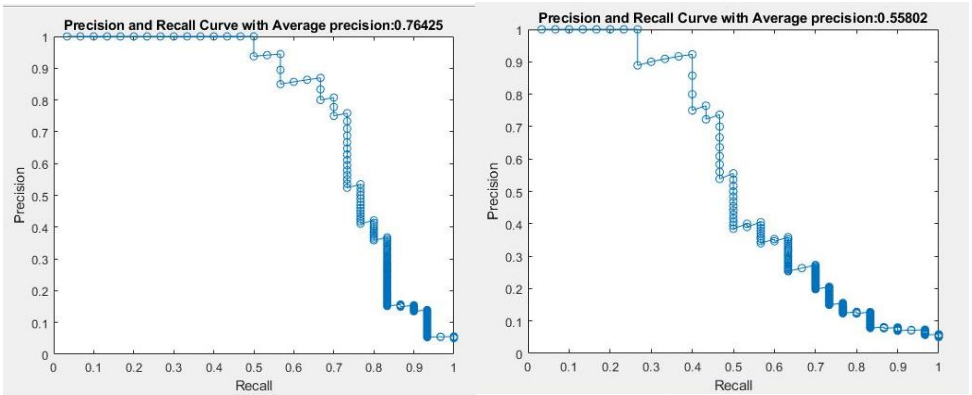
5.4 Spatial Grid: Colour and Texture

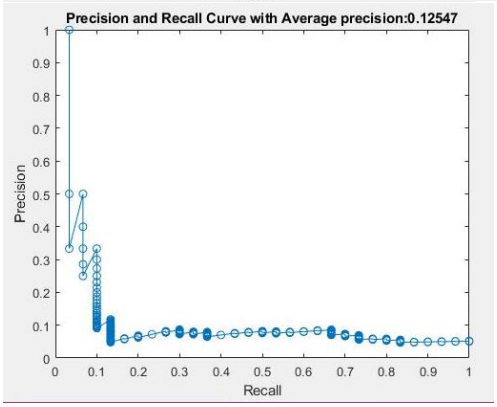
5.4.1 2x2 Grid



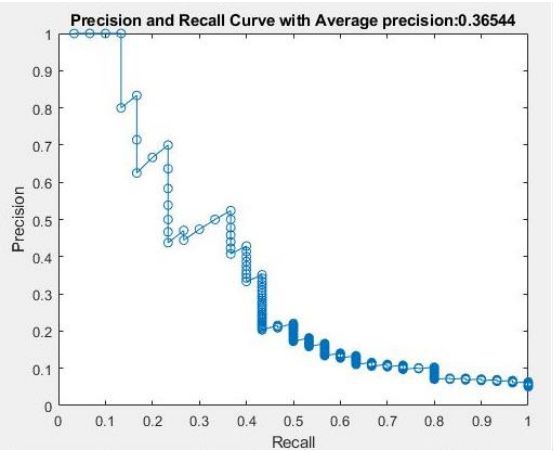


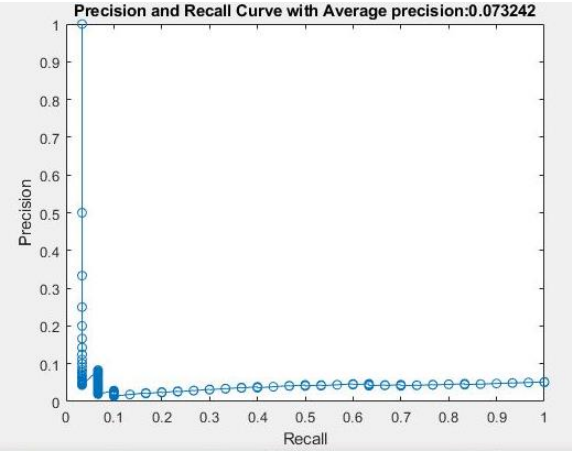
5.4.2 4x4 Grid

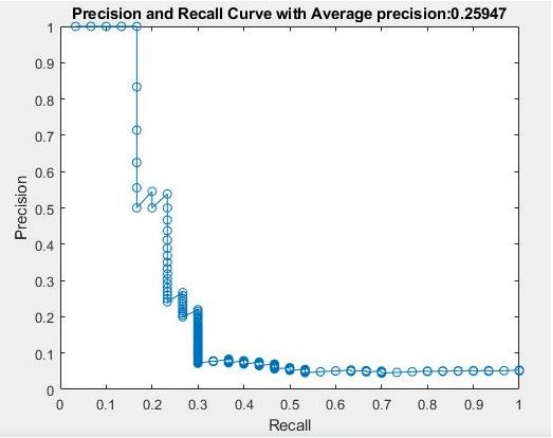




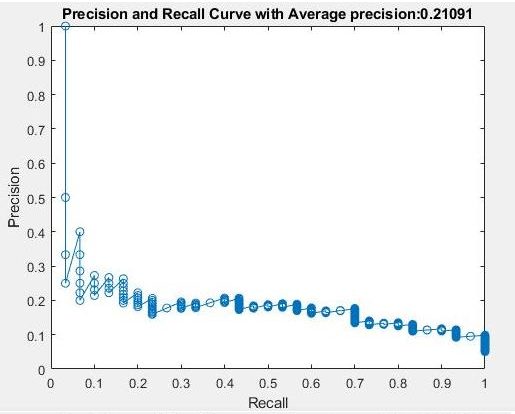
5.4.3 8x8 Grid

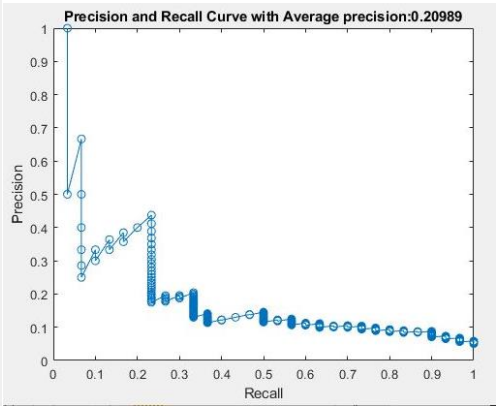


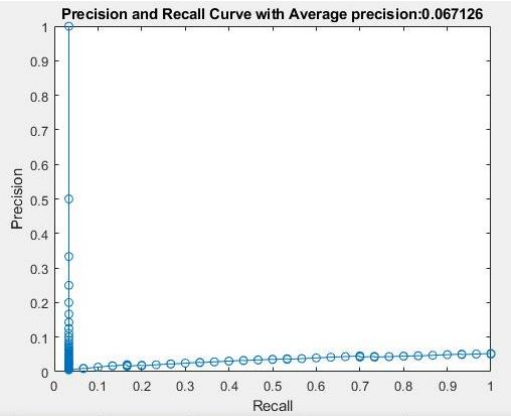




5.4.4 16x16 Grid

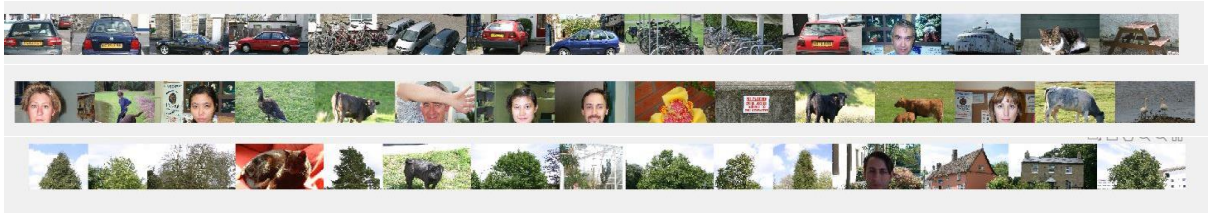


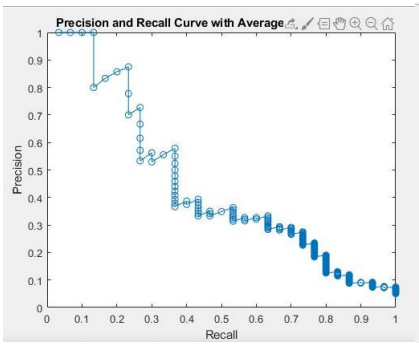


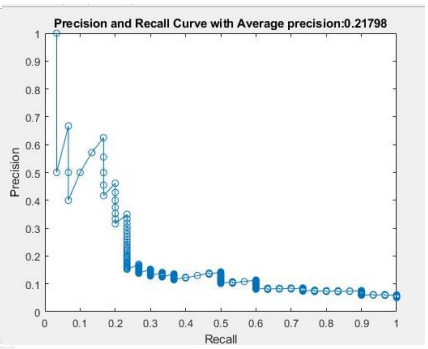


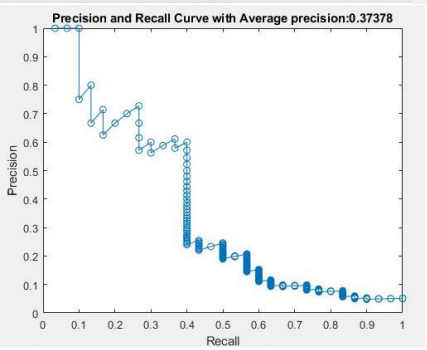
5.5 PCA and Mahalanobis Distance

5.5.1 Global Colour Histogram

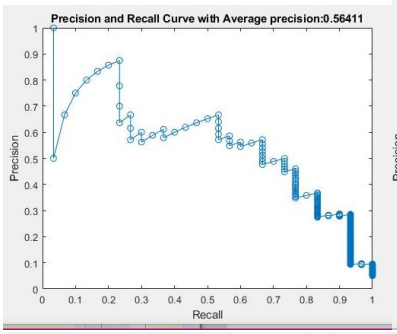


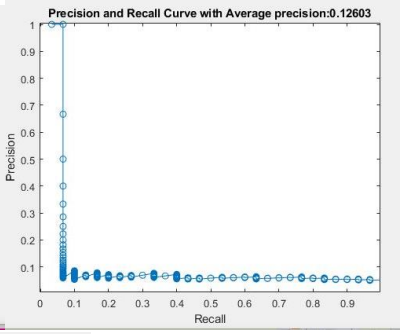


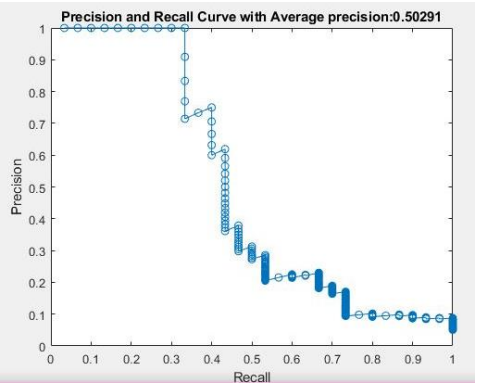




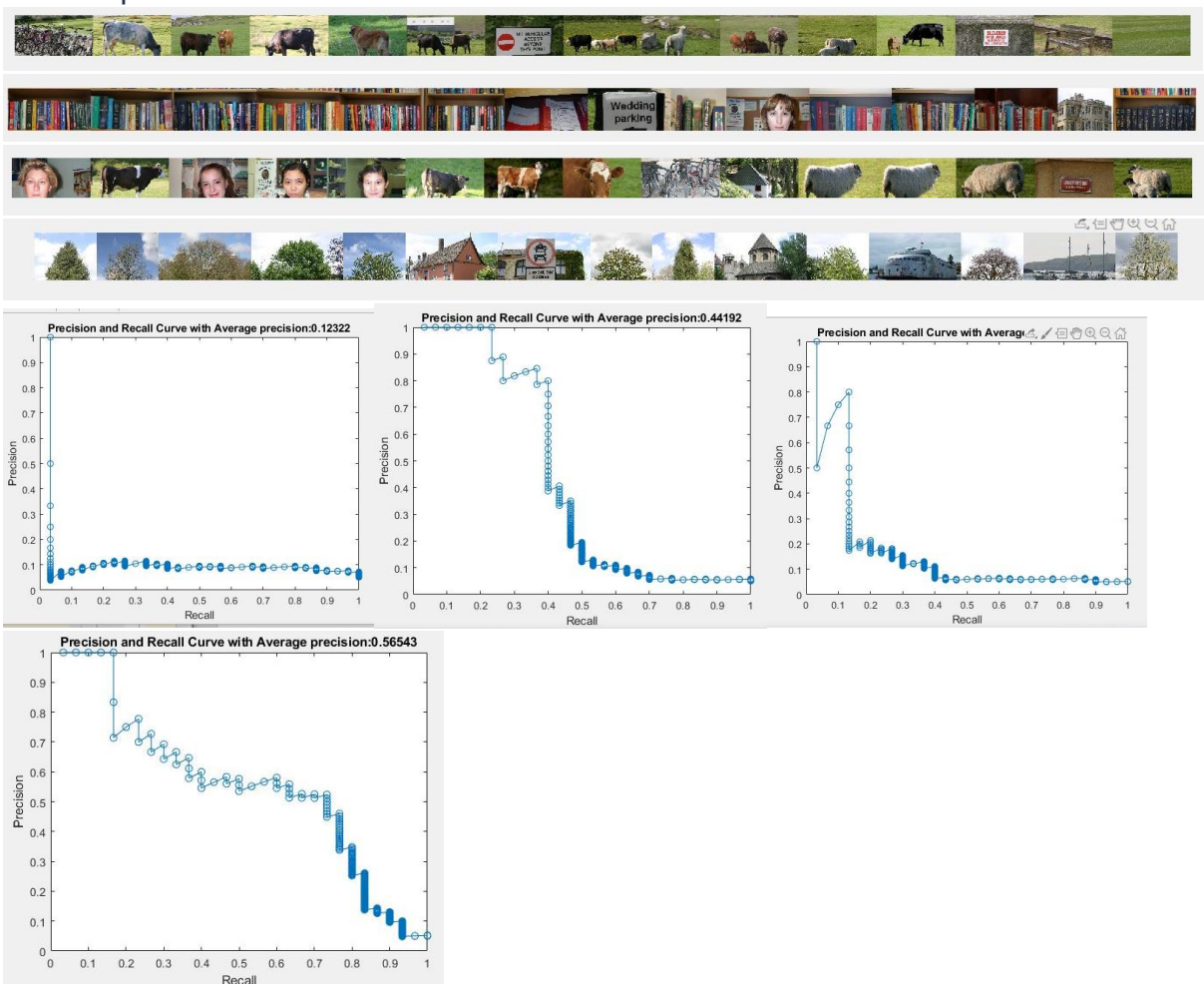
5.5.2 Spatial Grid: Colour



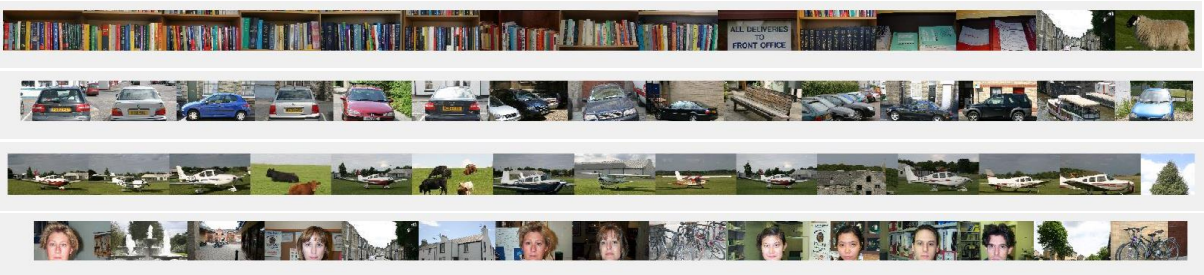


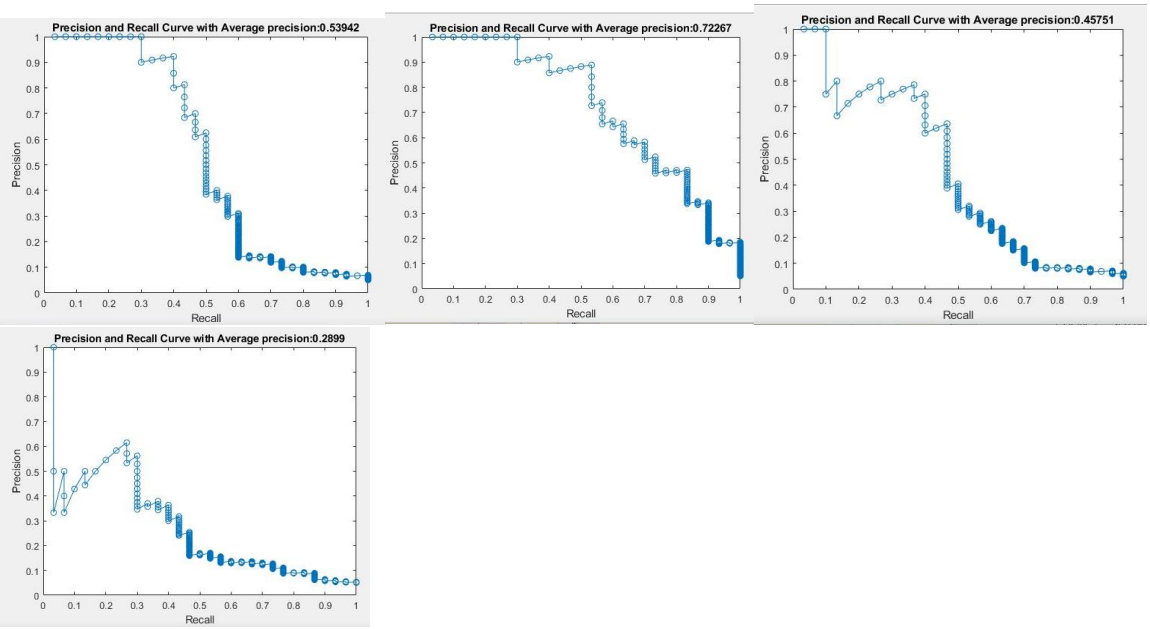


5.5.3 Spatial Grid: Texture



5.5.4 Spatial Grid: Colour and Texture

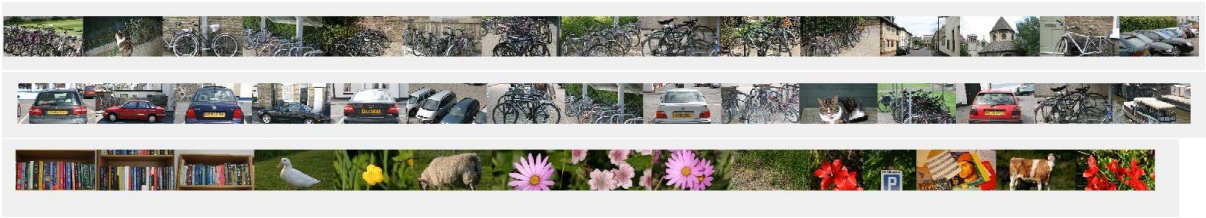


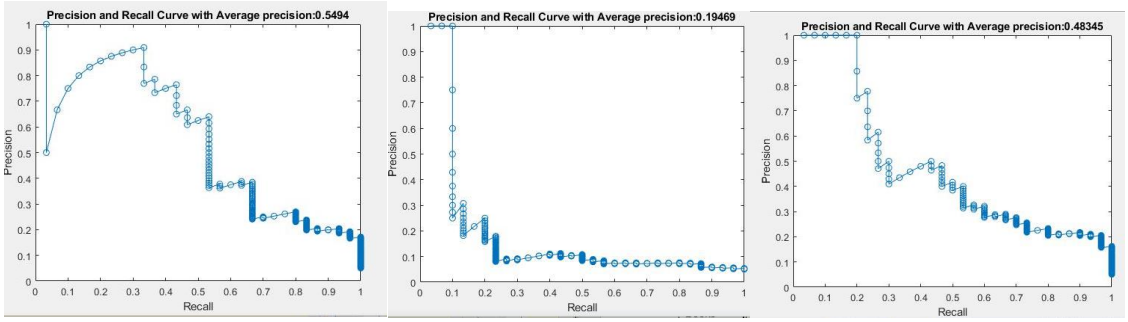


5.6 Distance Measures

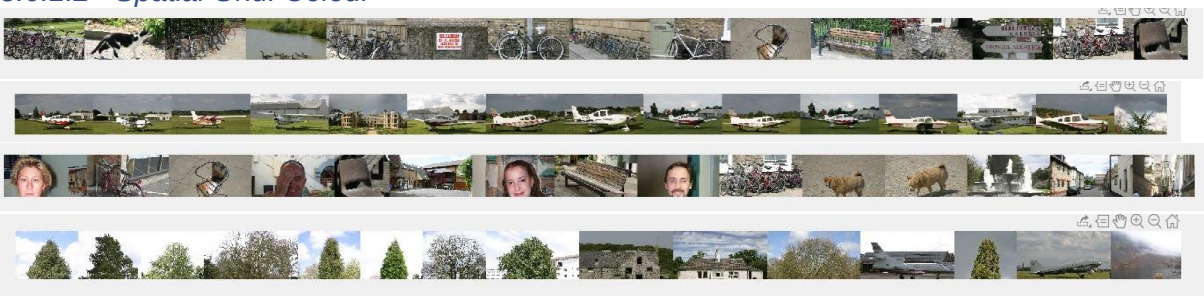
5.6.1 L_1

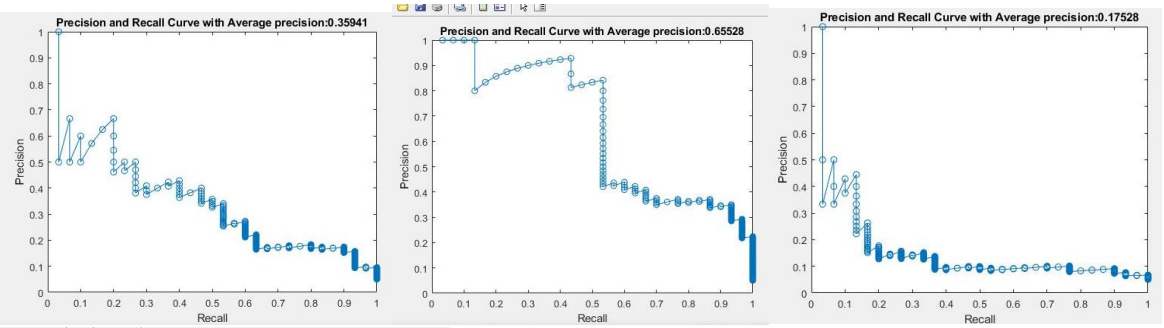
5.6.1.1 Global Colour Histogram

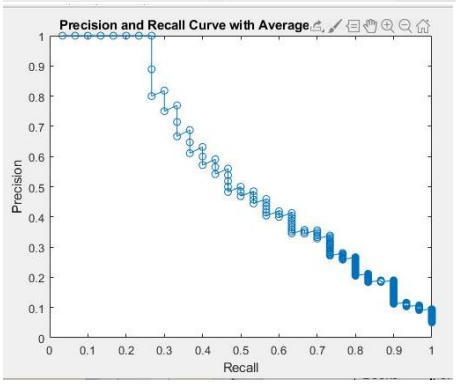




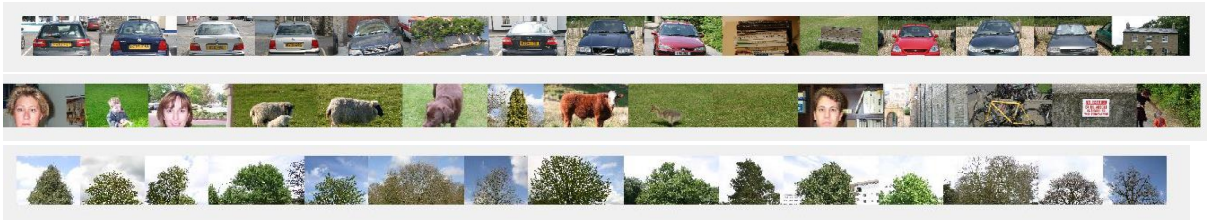
5.6.1.2 Spatial Grid: Colour

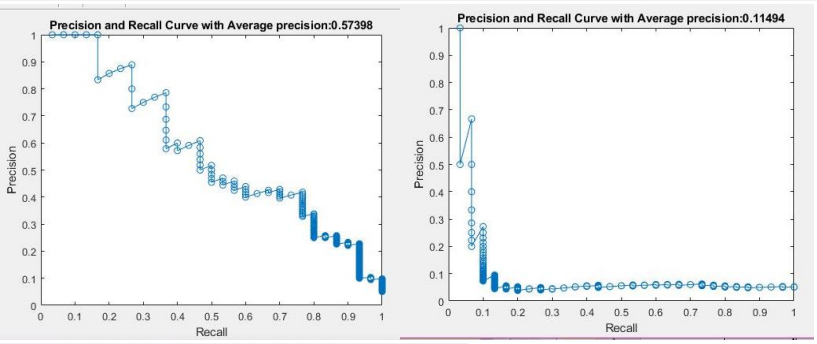


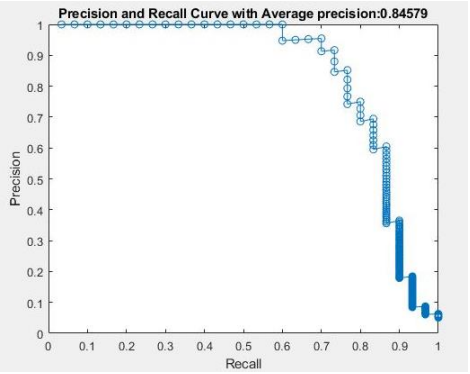




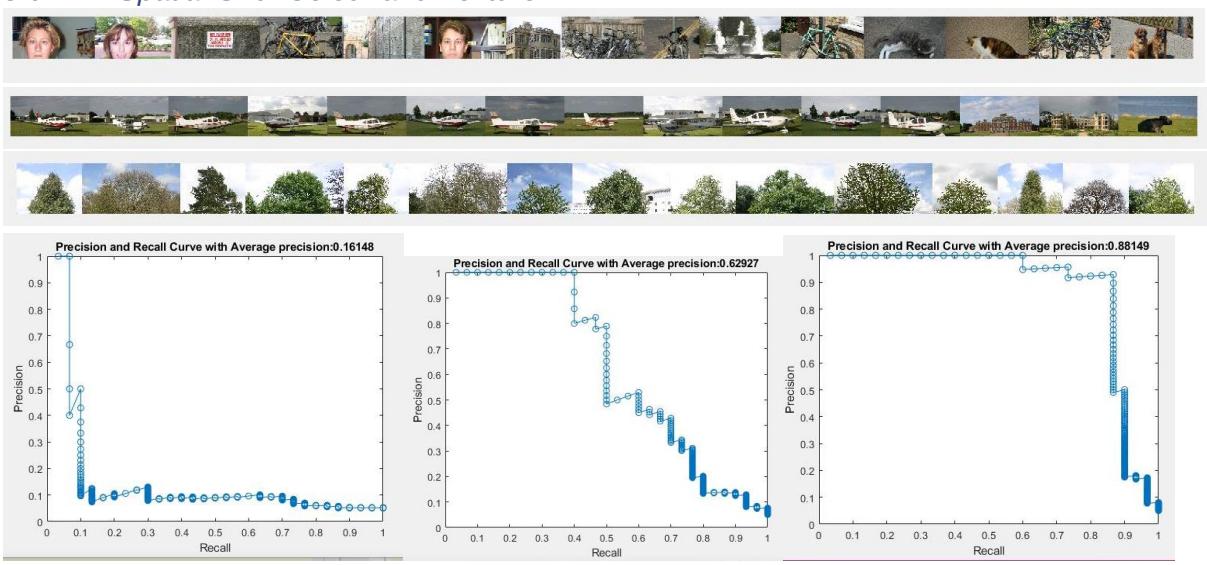
5.6.1.3 Spatial Grid: Texture





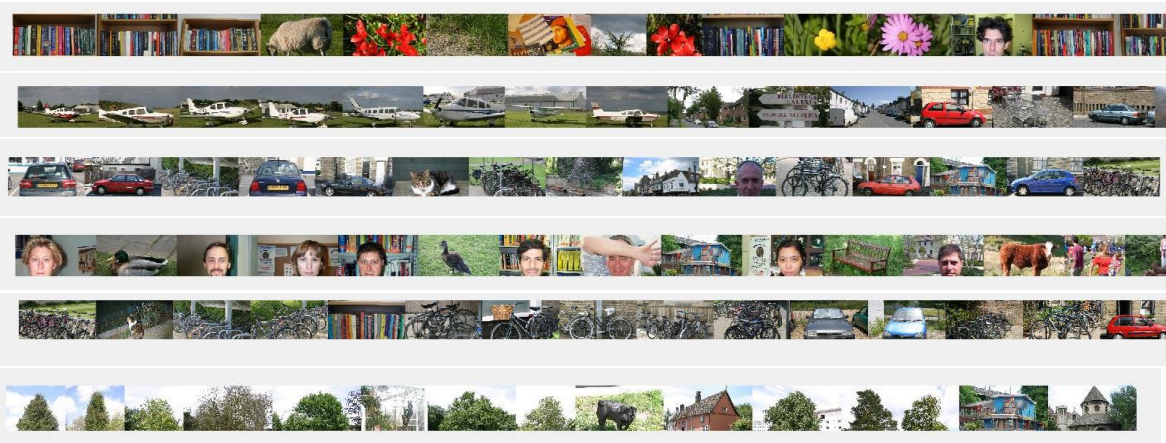


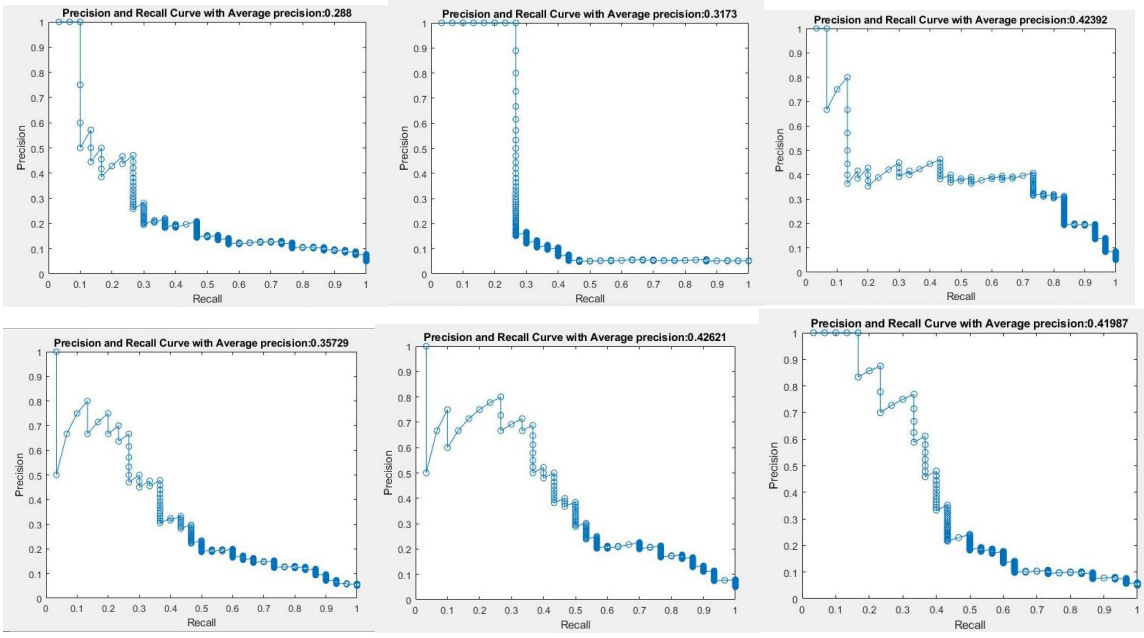
5.6.1.4 Spatial Grid: Colour and Texture



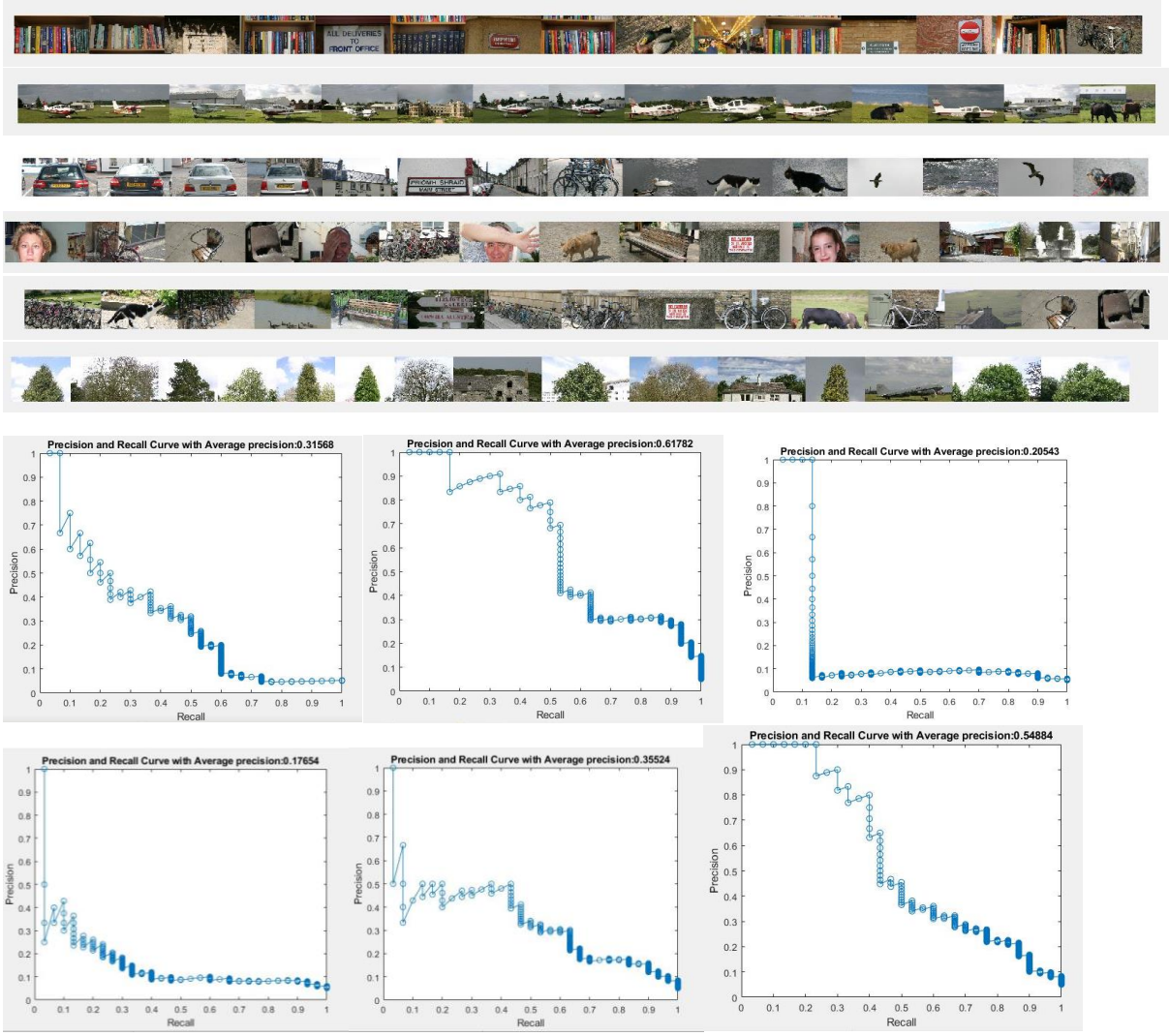
5.6.2 L_2

5.6.2.1 Global Colour Histogram

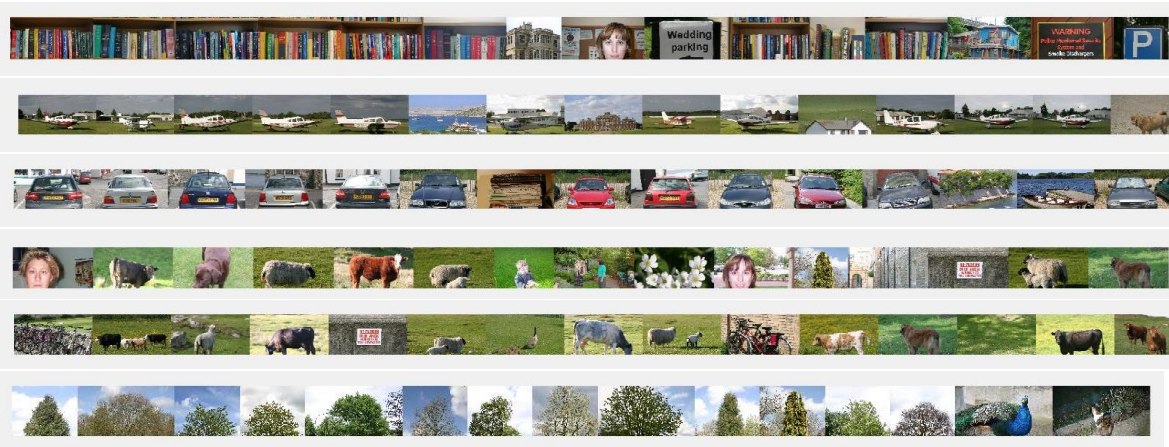


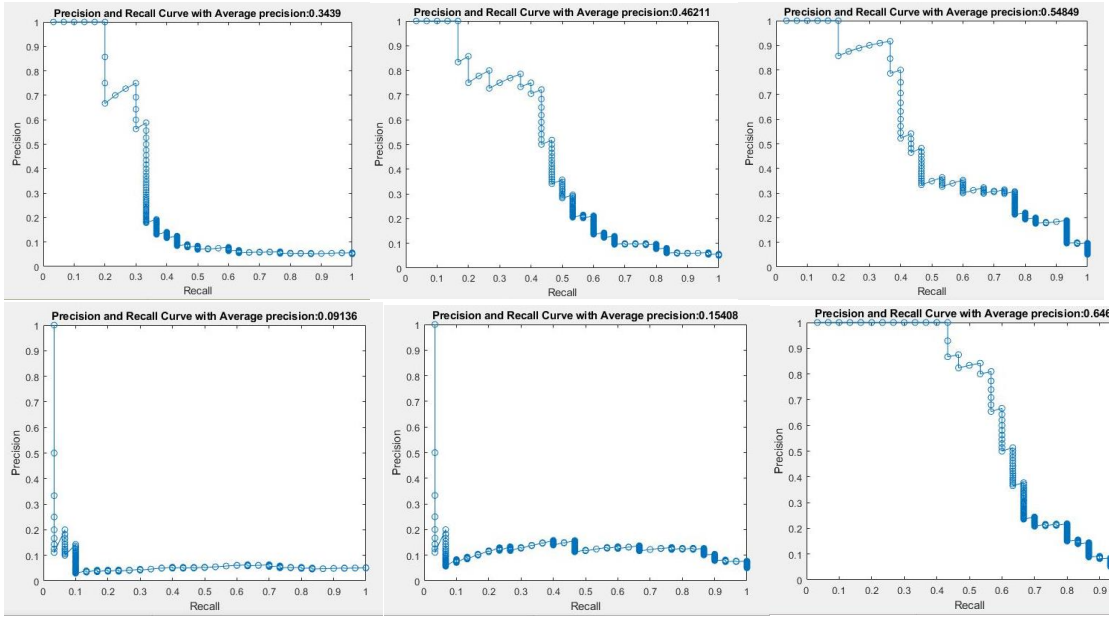


5.6.2.2 Spatial Grid: Colour



5.6.2.3 Spatial Grid: Texture





5.6.2.4 Spatial Grid: Colour and Texture