Visual Search of an Image Collection

EEE3032 Coursework Assignment

Herman Yau

Bachelor of Engineering
Electronic Engineering with Computer Systems

Department of Electrical and Electronic Engineering
University of Surrey
United Kingdom
24th April, 2018

Abstract

This report discusses the fundamental techniques and theories behind a basic visual search system, followed by an in-depth discussion of experimental results on a object segmentation database developed by Microsoft with all implementations written in MATLAB [1].

Contents

1 Visual Search

Visual search is a perceptual task in the field of machine learning and image processing which involves active scan(s) of visual environment for specific objects or features. As we perform visual search over a database, features from each distinct data are concatenated and are plotted as a point in feature space of $\mathbb{R}^n$, where n is an arbitrarily high dimensional space, and this formulates the image descriptor. For the purpose of this coursework, we define data as images for convenience.

The purpose of visual search is to decide the similarity between certain query image and images in the database. This is commonly known as a content-based image retrieval system (CBIR). Typically, such system ranks database images in order of similarity with the query image and returns the result to the user. Notable example in practice includes sketch-based image retrieval.

In sum, the process of visual search can be briefly described as the following:

1. Compute image descriptors of all data in a certain dataset with feature extraction algorithm as defined by user. Our image descriptor is now points in feature space $\mathbb{R}^n$.
2. Feed in a query data and use the same feature extraction algorithm to represent it in the same feature space.
3. Rank data in the dataset with respect to the query data in terms of distance metrics.

The fundamental requirements of image descriptors should be as discriminative as possible while maintaining compactness(low number of dimensions in the feature space). The following sections will describe a variety of techniques implemented in this coursework. The discussion is oriented in definition and methodology of each method.

1.1 Image Descriptors

1.1.1 Global Colour Histogram

Global colour histogram is a representation of the overall colour distribution in an image, as such we define similarity of an image in terms of overall colour. A greyscale image has a colour space of $\mathbb{R}^1$, whereas a coloured image is a combination of red, green and blue(RGB) pixels and yields a colour space of $\mathbb{R}^3$. Specifically, the colour space of each image is quantised into different levels according to the pixel's numerical value and counted, which are then put into different bins in a histogram. The completed histogram is then normalised such that fair comparisons of images with different sizes can be done.

Suppose we are supplied with a coloured image with $r, g, b \in [0, 255]$, where r, g, b are RGB pixel colour values. Mathematically, global colour histogram can be calculated as

$$r'_{i,j} = \lfloor \frac{r_{i,j} * q}{256} \rfloor \qquad g'_{i,j} = \lfloor \frac{g_{i,j} * q}{256} \rfloor \qquad b'_{i,j} = \lfloor \frac{b_{i,j} * q}{256} \rfloor \tag{1}$$

$$\text{bin} = r'q^2 + g'q + b' \tag{2}$$

Equation 1 is the quantisation step where we quantise each colour into q regions so that colour features $r', g', b' \in [0, q-1]$. However, our features are still in $\mathbb{R}^3$ and we need to represent them in $\mathbb{R}^1$ in order to get image descriptor of a single point. To achieve it we use Equation 2 to transform r', g', b' into an integer value 'bin' $\in [0, q^3 - 1]$, which can be easily represented as a histogram in a feature space of $\mathbb{R}^{q^3-1}$. Now we can compare similarity between images with distance metrics.

Global colour histogram, however, is not a very reliable similarity measure. Since it accounts in the overall colour distribution in nature, this method does not retain any spatial information of images. As a result, concentrations of same colours in drastically different regions in two images are regarded as the same by global colour histogram. Thus, global colour histogram satisfies compactness but fails to be discriminative.

1.1.2 Spatial Grid: Colour

The downside of global colour histogram can be overcome quite easily by splitting an image into separate grids. Instead of computing a single point in the feature space by direct evaluation of the whole image, we crop images into 'cells' and compute features for each of them. Hence, we define similarity measure as mean colour in cells(locations), retaining both spatial and colour information. We should note that all images concerned should have equal number of cells to maintain dimensionality.

For colour spatial grids, we compute the mean RGB colours for each cell as partial image descriptor.

$$Cell_i = \frac{1}{W_i H_i} \sum_{W_i, H_i} \begin{bmatrix} r_i & g_i & b_i \end{bmatrix} \tag{3}$$

We then concatenate the cells into a single long vector to obtain the overall image descriptor.

$$Descriptor = \begin{bmatrix} r_1 & g_1 & b_1 & \dots & r_n & g_n & b_n \end{bmatrix} \tag{4}$$

This method sacrifices certain compactness in exchange for descriptor discrimination. It still does not yield excellent performance though with only mean colour computation. An improvement can be made by obtaining local colour histogram in conjunction.

1.1.3 Spatial Grid: Edge Orientation Histogram

Computing spatial image descriptor with texture is largely similar with colour as before. As with the colour variation, we compute our image descriptors for each grid and concatenate to obtain our single long vector. However, we now search for textures/edges (high frequency information) to compute edge orientation histograms(EOH).

Obtaining edges of an image is relatively simple by edge detection filter, e.g. Sobel filter. Sobel filter is defined as

$$\frac{\partial f}{\partial x} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \frac{\partial f}{\partial y} = \begin{bmatrix} 1 & 2 & -1 \\ 0 & 0 & 0 \\ 1 & -2 & -1 \end{bmatrix} \tag{5}$$

and edge strength and orientation(gradients) are defined as

$$||\nabla f|| = \sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2} \qquad \theta = \arctan \frac{\partial f / \partial x}{\partial f / \partial y} \tag{6}$$

Following a similar equation formulation in Equation 3, we may obtain our EOH image descriptor.

As with the global colour histogram, we put the edge strengths into bins according to edge orientations. Very often we enforce a certain threshold on edge strength to drop out indistinguishable edges. Typically, the orientations are quantised into 8 evenly spaced bins $\theta \in [0, 2\pi]$. All EOHs are normalised within their own cell before concatenation to maintain fair comparisons. It should be noted, however, that EOH does not excel in images with little to no textures despite its high dimensionality. The resulting system is also not scale/orientation invariant, meaning that it is sensitive to affine transformations. The chosen level of angular quantisation θ impacts the discrimination and compactness of the image descriptor as well.

1.1.4 Spatial Grid: Colour/Texture Hybrid

The colour/texture hybrid image descriptor is simply a mix of both methods, with same vector containing both information concurrently. This produces a visual search system sensitive to both colours

and textures in an image, as well as preserving spatial information. It implies however that it further suffers from the lack of compactness due to the combination. In addition, it is important to define a rational grid size; coarse grids fail to precisely cover an object which leads to poor discrimination, while fine grids yield a large number of cells which gives poor compactness.

1.2 Principal Component Analysis (PCA)

From the image descriptors discussion, we can now understand that the feature space is often of high dimension. This give rise to the problem of **curse of dimensionality**, which refers to multitudes of phenomena occurs only in high dimensional settings as well as difficulty in visualisation.

PCA is a statistical approach to project observations in a high dimensional space to lower dimension via orthogonal transformation ($\mathbb{R}^n \to \mathbb{R}^m$), such that they can be represented in a more meaningful form with redundancy removed. Typically PCA is achieved by eigenvalue decomposition (EVD) of eigenmodel.

The procedures of building an eigenmodel can be described as below:

1. Separate and arrange points in the dataset of dimension $\mathbb{R}^n$ in a matrix, such that

$$\mathbf{P} = \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_n \\ y_1 & y_2 & y_3 & \dots & y_n \end{bmatrix} \tag{7}$$

2. Calculate mean μ and subtract each point with its dimension's μ:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} = p_i \qquad \bar{p} = p - \mu \tag{8}$$

3. Calculate the covariance matrix $\mathbf{\Sigma}$ from mean-subtracted points:

$$\mathbf{\Sigma} = \frac{1}{n} p \cdot p^T \tag{9}$$

4. Since $\mathbf{\Sigma}$ can be factorised in terms of eigenvector and eigenvalue [$\mathbf{U}$ $\mathbf{V}$], we can perform eigen-decomposition of $\mathbf{\Sigma}$ to get the parameters:

$$\mathbf{\Sigma} = \mathbf{U}\mathbf{V}\mathbf{U}^T \tag{10}$$

Now we can examine our decomposed eigenmodel of the dataset with $\mu, \mathbf{U}, \mathbf{V}$ and find our interested $\mathbf{V}$ where the dataset is more expressive. We can then drop the null space by defining some threshold, meaning dimensions with little to no variation. Finally, we can project our data to lower dimensional space with most variation captured by moving to a new frame, where the basis is defined by $\mathbf{K}$, a deflated version of $\mathbf{U}$:

$$\mathbf{Q} = \mathbf{K}^{-1}\mathbf{P} \tag{11}$$

1.3 Distance Metrics

In order to deduce similarity of images in the feature space, we evaluate via computation of distances. All distance metrics are multivariate functions which maps to $\mathbb{R}^+$ and they must satisfy the triangle inequality. In this coursework, three different distance metrics are implemented, namely l_1 norm, l_2 norm and Mahalanobis distance.

1.3.1 L_1 Norm

L_1 norm is also known as Manhattan distance or taxicab distance. It is simply the sum of absolute difference of points in each coordinate axis in Cartesian plane. Given two vectors $\mathbf{p}, \mathbf{q}$ where $\mathbf{p} = \begin{pmatrix} p_1 & p_2 & p_3 & \dots & p_i \end{pmatrix}, \mathbf{q} = \begin{pmatrix} q_1 & q_2 & q_3 & \dots & q_i \end{pmatrix} \in \mathbb{R}^n$ we can express l_1 norm mathematically by

$$d_1(\mathbf{p}, \mathbf{q}) = ||\mathbf{p} - \mathbf{q}||_1 = \sum_{i=1}^{n} |p_i - q_i| \tag{12}$$

1.3.2 L_2 Norm

L_2 norm is one of the most commonly used distance metric defined as an ordinary straight line distance between two points in Cartesian plane. Given two vectors $\mathbf{p}, \mathbf{q}$, it is expressed in mathematical terms as

$$d_2(\mathbf{p}, \mathbf{q}) = ||\mathbf{p} - \mathbf{q}||_2 = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{13}$$

1.3.3 Minkowski Distance

Minkowski distance, or L_p norm, can be regarded as a generalisation of Euclidean distance with order $p \geq 1$:

$$d_p = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{\frac{1}{p}} \tag{14}$$

At $p = 1$, the distance is really L_1 norm and $p = 2$ is equivalent to L_2 norm.

1.3.4 Mahalanobis Distance

The Mahalanobis distance is a measure of distance between an observation the distribution of all data, i.e. number of standard deviation σ away from mean of distribution in each principal component axis. As such, the distance metric is obtained by the use of PCA. It is unitless and scale-invariant since the distribution of data is scaled by the principal axis' corresponding eigenvalue $\mathbf{V}$.

The formulation of Mahalanobis distance is

$$d = \sqrt{\mathbf{V}^{-1}\mathbf{U}(x - \mu)} \tag{15}$$

1.4 Bag-of-Visual-Words Retrieval

Bag of visual words (BoVW) is another highly popular, practical CBIR system. In general, a BoVW system involves the following design pipeline:

1. **Codebook construction**: Extract descriptors from images in dataset(s) using a feature extractor which are represented as points in the feature space. Ideally the feature descriptor should be invariant to affine transformations, i.e. translations, scaling, rotations, reflections. Scale invariant feature transform (SIFT) has such robustness which will be discussed later.
2. **Encoding**: Encode all the points in the feature space into visual codewords, where each codeword represents some similar features of an image (a visual word). A naive method to enforce the encoding process is by k-means clustering over all the vectors, and the center of each cluster is the visual word. We now obtain our visual codebook for the dataset.

3. **Image descriptor initialisation**: Extract descriptors for each image in the dataset(s) and compare them to the visual codebook. Assign each features from the query image to the nearest cluster using a distance metric, then construct a histogram of visual word occurrences. This is the feature vector of the image. Repeat for the query image and we can now compare the relevancy with distance metrics.

1.4.1 SIFT Descriptor

Scale invariant feature transform (SIFT) is a robust image feature generation method developed by D. Lowe [2, 3]. SIFT primarily operates by transforming an image into a large collection of feature vectors (interest points). Each feature vector is invariant to translation, scaling and rotation, and is invariant to illumination changes and projections to some degrees.

SIFT descriptor can be described with the following stages of computation:

1. **Scale-space extreme detection**: We create a stack of difference-of-Gaussian (DoG) images by convolving the original image with Gaussian filters and take differences of successive Gaussian-blurred images. By comparisons with neighbouring pixels, We can then distinguish local extremas of DoG stacks as keypoint candidates.
2. **Keypoint localisation**: There are too many possible keypoints from pure DoG detection, so we must reject unstable keypoint candidates to get accurate and representative keypoints. Techniques of achieving it includes Taylor expansion on scale-space and edge responses elimination with Hessian matrix. We have now obtained robust keypoint locations.
3. **Orientation assignment**: We assign one or more orientation to each keypoint to achieve invariance to rotation. This is done by pre-computation of gradient magnitudes and orientations.
4. **Keypoints descriptor**: Using the completed keypoint with sufficient robustness to affine transformations, we can now compute the descriptor vector over 16×16 window, which is further divided into a 4×4 grid and calculate an orientation histogram in each grid cell. The concatenated, 16×8 orientations is our $\mathbb{R}^{128}$ descriptor.

1.4.2 k-means Clustering

k-means clustering is an unsupervised learning which aims to find groups of data that can be represented with user-defined variable k[4]. It is an iterative algorithm to separate data into k clusters based on similarity in feature space.

The algorithm works as the following:

1. **Assignment**: Pick k points in the feature space randomly as cluster centroids. Assign each observation to the nearest centre by Euclidean distance.
2. **Update**: Compute new means of all data points in each cluster, which would be the new centroids. If any of the centroids are moved, repeat the steps. The algorithm is converged if the change is negligible.

Improvements can be made to aid the convergence or find more obtimal clusters, such as k-means++ which improves the initial centroid initialisation [5].

2 Experiments

The visual search experiments are implemented and carried out using MATLAB [1] and the Microsoft Research (MSVC-v2) dataset [6]. All the implementations follows the discussion of techniques in the last major section in chronological order. As the MSVC-v2 dataset is originally designed for image segmentation purpose, image labelling is not accurate for visual search and some images may appear in multiple classes. To avoid ambiguities and allow legitimate performance evaluations, we select query images with distinctive features as shown below. All evaluation uses L_2 norm unless otherwise specified. Due to limited spaces, this section will only include a table summarises average precisions for each experiment. PR curves and results visualisation are available in appendix.

Figure 1: List of images that are used in experiment evaluations. Most of them have their unique categories within the MSVC-v2 dataset, thus more suitable for visual search.

2.1 Evaluation Methodology

In order to evaluate the performance of the implemented visual search systems, precision-recall (PR) statistics are computed and plotted as a PR curve using MATLAB built-in function. In information retrieval context, precision and recall are defined as a function of set of relevant results and set of retrieved results.

Precision is the fraction of relevant results with respect to the query and all retrieved items:

$$\text{precision} = \frac{\{\text{relevant results}\} \cap \{\text{retrieved results}\}}{\{\text{retrieved results}\}} \tag{16}$$

Recall is the fraction of relevant results with respect to the query and all relevant results:

$$\text{recall} = \frac{\{\text{relevant results}\} \cap \{\text{retrieved results}\}}{\{\text{relevant results}\}} \tag{17}$$

Then we can evaluate a retrieval system by plotting the PR curve and calculating the average precision (AP):

$$AP = \sum_{n=1}^{M} \frac{P(n)rel(n)}{\{\text{relevant results}\}} \tag{18}$$

where n is the rank index of result, M is the total number of retrieved results, $rel(.) = 1$ if n is relevant, otherwise $rel(.) = 0$.

Mean AP is calculated as the average AP of the six images:

$$MAP = \sum_{i=1}^{6} \frac{AP_i}{6} \tag{19}$$

2.2 Results

2.2.1 Global Colour Histogram

Global colour histogram is implemented in MATLAB by the script `extractRGBHistogram.m`. This function takes in an unmodified coloured image `img` and quantisation level `Q`, bins the quantised RGB

pixel values, computes and normalises the histogram and returns it to `cvpr_computedescriptor.m`. Here we experiment with 4 different quantisation levels, $Q \in \{2, 4, 8, 16\}$.

Category	Average Precision			
	$Q = 2$	$Q = 4$	$Q = 8$	$Q = 16$
Planes	0.0333	**0.05**	0.0381	0.0333
Books	0.0789	**0.164**	0.152	0.111
Cars	0.142	**0.150**	0.0712	0.0474
Bikes	0.0767	0.05	0.136	**0.196**
Flowers	0.109	0.106	**0.111**	0.0860
Chairs	0.0333	0.0333	0.0333	**0.0389**
MAP	0.0789	**0.0922**	0.0903	0.0854

Table 1: Average precisions for global colour histogram visual search.

From Table 1, we can see that at $Q = 4$, most of our test categories performs better than other quantisation levels. At $Q = 2$, the low quantisation level means that the RGB pixel values can only be divided into 2 distinct values which diminishes the discrimination power. At $Q = 8$ and $Q = 16$, the high quantisation level has the opposite effect as we can observe the APs are progressively deteriorating; the pixel values are divided in such high density that the image descriptor cannot distinguish unique features properly.

2.2.2 Spatial Grids: Colour

Colour spatial grids are implemented by the script `extractColour.m`. The function script takes in three arguments, `img, rowGrids, colGrids` which is the image, the number of grids horizontally and vertically. Two for loops are used to execute operations on image cropping and grid descriptor calculation: the size of each grid cell is calculated and stored into a separate matrix variable. The mean red, green and blue colours are then calculated and concatenated into the feature vector. These steps are repeated for each iteration until all grids have been exhausted, where the final obtained feature vector for the image is passed to `cvpr_computedescriptors.m`. Here we experiment with 4 different numbers of grid cells, 2×2, 4×4, 8×8, 16×16.

Category	Average Precision			
	2×2	4×4	8×8	16×16
Planes	0.226	0.390	**0.462**	0.424
Books	0.164	0.171	**0.189**	0.182
Cars	0.0667	0.1	**0.0938**	0.0870
Bikes	0.0407	0.0333	**0.0456**	0.0385
Flowers	0.0313	0.0938	**0.111**	0.0938
Chairs	0.114	**0.162**	0.152	0.141
MAP	0.154	0.158	**0.176**	0.161

Table 2: Average precisions for colour spatial grids visual searches.

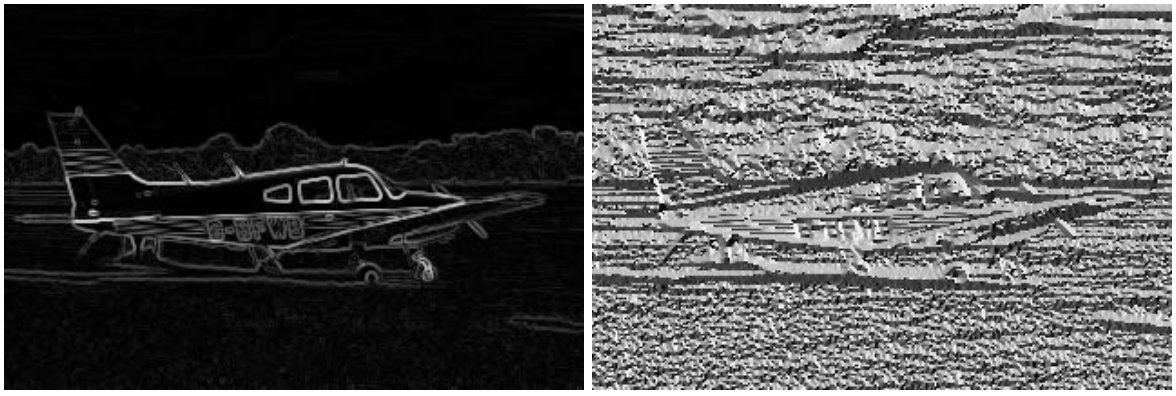
Table 2 shows the results for the experiment, where 8×8 grids give the best results. The result can be explained twofold. The first can be explained with a similar reason given for global colour histogram; the second is that the same features may not be found spatially but in other grids, which impacts the ability to return relevant result in some degrees.

2.2.3 Spatial Grid: Edge Orientation Histogram

Edge orientation histogram spatial grids is implemented by the script `extractEOH.m`. The script accepts three arguments, `img, rowGrids, colGrids, Q` which is the image, the number of grids horizontally and vertically, and the number of quantisation levels. The grid cells extraction is exactly the same as colour grids.

The script defines Sobel filters as previously explained in Equation 5. It first converts `img` into greyscale first, then convolves it with the Sobel filters using `imfilter` to obtain the edges. The gradient magnitudes and orientations are then calculated with Equation 6, with magnitude thresholded at 0.15 to reject weak edges and orientation $+\pi$ to adjust range from $[-\pi, \pi]$ to $[0, 2\pi]$. For each grid cell, the script finds the indices of all non-zero magnitudes and extract the orientations for those indices. The extracted orientations are then binned according to `Q`. If there are any `NaN`s, i.e. value does not exist, all the bins are set to 0. Finally, the binned values are normalised and concatenated with the feature vector. The process is repeated until all grid cells are exhausted.

Again, we experiment with 4 different numbers of grid cells, 2×2, 4×4, 8×8, 16×16. The best result is picked from the results of grid cells and it is tested with different angular quantisation levels $\theta \in \{2, 4, 8, 16\}$ to examine the effect.



(a) Edge magnitude of an image. (b) Edge orientations visualised as an image.

Figure 2: Edge magnitude and orientation visualisation of an image by `extractEOH.m`.

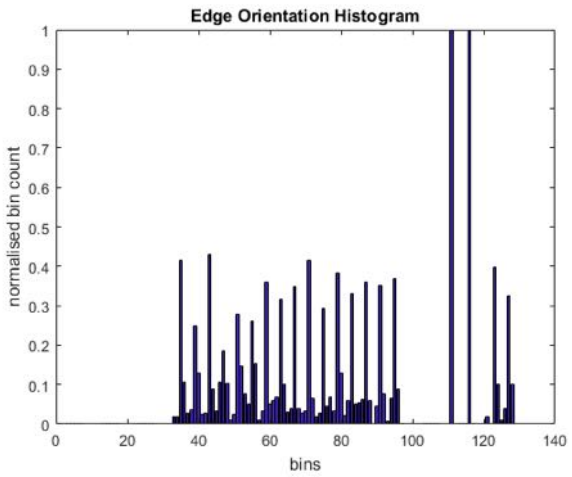


Figure 3: An example image descriptor of EOH representing an image with 4×4 number of grid cells and $\theta = 8$.

Category	Average Precision			
	2×2	4×4	8×8	16×16
Planes	**0.308**	0.151	0.118	0.187
Books	0.182	**0.276**	0.126	0.122
Cars	0.146	**0.241**	0.0516	0.0333
Bikes	0.147	**0.219**	0.0795	0.0996
Flowers	**0.226**	0.111	0.103	0.0721
Chairs	0.188	**0.284**	0.219	0.118
MAP	0.200	**0.214**	0.116	0.105

Table 3: Average precisions for EOH visual searches with different numbers of grid cells.

Category	Average Precision			
	$\theta = 2$	$\theta = 4$	$\theta = 8$	$\theta = 16$
Planes	0.08	0.142	**0.151**	0.141
Books	0.0333	0.204	**0.276**	0.26
Cars	0.0661	0.151	0.241	**0.284**
Bikes	0.133	0.199	**0.219**	0.213
Flowers	0.0313	**0.127**	0.111	0.0907
Chairs	0.195	0.248	**0.284**	0.248
MAP	0.0898	0.179	**0.214**	0.206

Table 4: Average precisions for EOH visual searches for grid cells number 4×4 with different quantisation levels.

The phenomenon observed in Table 3 can largely be explained with the same reasons as colour grid cells. It is worth mentioning that EOH performs better at 4×4 grids instead, probably due to edges' greater expressiveness of image features than mean colours.

As for different quantisation levels, we can easily see that $Q = 8$ almost entirely outperforms other lower quantisation levels, indicating that image descriptors computed with low quantisation levels is not sufficiently discriminative. It is interesting to see similar performance at $Q = 16$, and this shows that sacrificing compactness to include more quantisation levels does not necessarily yield better visual search performance.

2.2.4 Spatial Grid: Colour & EOH

This is simply the concatenation of colour and EOH feature vectors. It is implemented by the script `extractEOHColour.m`. We experiment with 4 different numbers of grid cell, 2×2, 4×4, 8×8, 16×16.

Category	Average Precision			
	2×2	4×4	8×8	16×16
Planes	**0.462**	0.173	0.198	0.219
Books	**0.253**	0.251	0.133	0.142
Cars	**0.233**	0.220	0.0467	0.0333
Bikes	0.177	**0.249**	0.131	0.111
Flowers	**0.115**	0.0678	0.0703	0.0521
Chairs	0.243	**0.267**	0.224	0.149
MAP	**0.247**	0.205	0.134	0.118

Table 5: Average precisions for colour and EOH spatial grids concatenation visual searches.

Immediately from Table 5 we can observe that the system is performing surprisingly well at $Q = 2$. The reason behind it may be due to concatenation of colour and EOH feature vectors directly; this gives equal weightings to both features and may impact distances of points in feature space as one may overpower the other. A possible improvement would be multiplying the feature vectors with a constant to give more emphasis on certain feature vectors.

2.2.5 Principal Component Analysis (PCA)

PCA is implemented by the script `cvpr_pca.m`. For PCA, we aim to project image descriptors to a lower dimensional space and use Mahalanobis distance as an alternative distance metric to see if we can obtain a better or comparable performance with previous experiments. PCA is implemented by the script `cvpr_pca.m` and Mahalanobis distance is implemented in `cvpr_compare_Mahalanobis.m`, both of them are implemented as discussed in previous section. For global colour histogram, we use the results from $Q = 4$ as the baseline. For spatial grids, we use results from 4×4 grids cells and EOH bins $\theta = 8$ as the baseline. We set the PCA such that 95% of the original image descriptors.

Category	Average Precision			
	Global	Colour Grids	EOH Grids	Combined Grids
Planes	*0.055*/0.05	*0.332*/0.390	*0.191*/0.151	*0.216*/0.173
Books	*0.161*/0.164	*0.207*/0.171	*0.190*/0.276	*0.195*/0.251
Cars	*0.0767*/0.150	*0.0810*/0.1	*0.110*/0.241	*0.0972*/0.220
Bikes	*0.177*/0.05	*0.0333*/0.0333	*0.162*/0.219	*0.176*/0.249
Flowers	*0.141*/0.106	*0.0402*/0.0938	*0.266*/0.111	*0.104*/0.0678
Chairs	*0.0333*/0.0333	*0.140*/0.162	*0.154*/0.284	*0.179*/0.267
MAP	*0.107*/0.0922	*0.139*/0.158	*0.179*/0.214	*0.161*/0.205

Table 6: Average precisions before and after PCA and Mahalanobis distance computation. Italic text shows results after.

From Table 6, we can see that the APs largely stays similar to before performing PCA. This indicates that PCA successfully reduced the dimension of image descriptors with Mahalanobis distance accounts in the sparsity of data, therefore achieving relatively satisfactory visual search performance. However, some deterioration can be observed. This is likely due to the inevitable information loss with the PCA algorithm.

2.2.6 L_1 Norm

We experiment with the L_1 norm and compare the performance as oppose to L_2 norm. This is implemented in the script `cvpr_compare.m` with p specifies the norm to be used. Likewise, we select $Q = 4$ for global colour histogram and 4×4 grids cells and EOH $\theta = 8$ as the baseline.

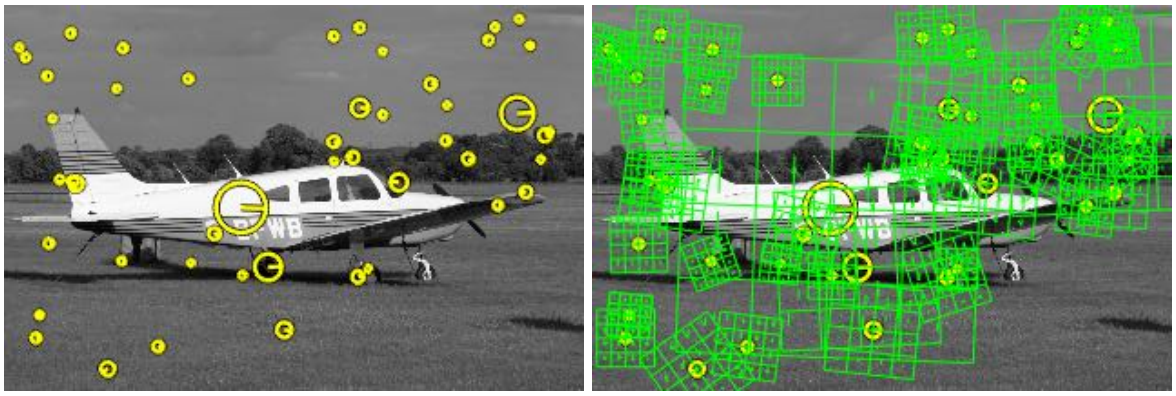
Category	Average Precision			
	Global	Colour Grids	EOH Grids	Combined Grids
Planes	*0.0429*/0.05	*0.421*/0.390	*0.272*/0.151	*0.340*/0.173
Books	*0.135*/0.164	*0.165*/0.171	*0.337*/0.276	*0.372*/0.251
Cars	*0.144*/0.150	*0.136*/0.1	*0.256*/0.241	*0.312*/0.220
Bikes	*0.08*/0.05	*0.0389*/0.0333	*0.233*/0.219	*0.271*/0.249
Flowers	*0.106*/0.106	*0.112*/0.0938	*0.125*/0.111	*0.108*/0.0678
Chairs	*0.0333*/0.0333	*0.166*/0.162	*0.345*/0.284	*0.317*/0.267
MAP	*0.0902*/0.0922	*0.173*/0.158	*0.261*/0.214	*0.287*/0.205

Table 7: Average precisions with L_1 and L_2 norm. Italic text shows result of L_1 norm.

From 7, we can see that L_1 norm outperforms L_2 norm by a significant margin, especially in EOH/colour grid cells. Such result arises since L_2 distance finds the unique shortest distance between two points which lacks sparsity and robustness in high dimension. Therefore, feature outliers can give negative impacts to visual search which is shown above. L_1 norm, however, produces multiple solutions and innately enforces sparsity and detects outliers better than L_2. In sum, this shows that L_2 norm may not be the optimal distance metric for visual search.

2.2.7 Bag-of-Visual-Words and SIFT

BoVW and SIFT are primarily implemented with VLFEAT open source library[7] with two helper scripts `sift.m` and `cvpr_sifthistogram.m` to aid computation. `sift.m` accepts an image `img` and call the function `vl_sift` to obtain SIFT frames and SIFT descriptors for the image. All the descriptors obtained from `vl_sift` are concatenated in `cvpr_sift_computedescriptors.m` and compute the codebook using `vl_kmeans`. By default, `vl_kmeans` initialises cluster centroids by picking k points randomly. The dataset is then scanned again by `cvpr_sifthistogram.m` where each SIFT descriptor is compared with the codebook and assigned to the nearest cluster by L_2 norm.



(a) Selected SIFT keypoints. (b) Selected SIFT descriptors.

Figure 4: Visualisation of SIFT keypoints and descriptors respectively.

We experiment with different number of clusters: $k \in \{1000, 2500, 5000, 10000\}$ with L_1 and L_2 norm.

Category	Average Precision			
	$k = 1000$	$k = 2500$	$k = 5000$	$k = 10000$
Planes/L_1	0.245	0.280	**0.310**	0.306
Books/L_1	0.411	0.457	**0.462**	0.460
Cars/L_1	**0.0889**	0.0556	0.0622	0.0333
Bikes/L_1	0.218	0.199	0.239	**0.252**
Flowers/L_1	**0.291**	0.233	0.215	0.216
Chairs/L_1	0.117	0.127	0.148	**0.185**
MAP	0.228	0.235	0.241	**0.242**

Table 8: Average precisions for SIFT with different k using L_1 norm.

Category	Average Precision			
	$k = 1000$	$k = 2500$	$k = 5000$	$k = 10000$
Planes/L_2	**0.0822**	0.0544	0.0333	0.0333
Books/L_2	**0.322**	0.2	0.113	0.1
Cars/L_2	0.0333	0.0333	0.0333	0.0333
Bikes/L_2	**0.0972**	0.0519	0.0333	0.0333
Flowers/L_2	0.228	**0.245**	0.203	0.224
Chairs/L_2	**0.129**	0.1	0.0767	0.0667
MAP	**0.149**	0.114	0.0821	0.0818

Table 9: Average precisions for SIFT with different k using L_2 norm.

The result from Table 8 and 9 shows that using SIFT descriptor yields more consistent and favourable APs compared to previous methods with L_1 norm (with the exception of cars). This gives supporting evidences for Table 7 that L_1 norm is a better distance metric for visual search, with L_2 norm's result significant worse than the former. In addition, it is found that the size of all keypoints concatenated together is 128×148968 which is gigantic and adds computational complexity. Also, the number of cluster equals to the dimension of image descriptor for each image which is large as well. This means that we have sacrificed compactness in return for discrimination of features when compared to simpler methods such as EOH and EOH/colour hybrid which largely offers the same PRs. Points of improvement include dimensionality reduction via PCA and thresholding SIFT keypoint detection.

3 Conclusions

A major source of difficulty comes from the nature of the MSVC dataset. The dataset is originally designed to aid researches related to object segmentation, which is partitioning of digital image into individual meaningful segments. As such, the original labelling of the dataset is inaccurate and unsuited for image retrieval tasks.

In conclusion, visual search involves a plethora of techniques in terms of representation of features, features comparisons and dimensionality reduction as shown in the experimental results. In particular, the results show that there are a wide range of tweakable parameters, e.g. number of grid cells in spatial grid-based systems, quantisation levels, number of clusters, etc., each and every one of them can contribute an impact to the performance of CBIR system.

References

[1] MATLAB, *version 9.2.0.538062 (R2017a)*. Natick, Massachusetts: The MathWorks Inc., 2017.

[2] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, (Washington, DC, USA), pp. 1150–, IEEE Computer Society, 1999.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov 2004.

[4] J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *JSTOR: Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979.

[5] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, (Philadelphia, PA, USA), pp. 1027–1035, Society for Industrial and Applied Mathematics, 2007.

[6] M. R. Cambridge, *MSRC Object Category Image Database v2*. Cambridge, United Kingdom: Microsoft Research Ltd., 2005.

[7] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms." `http://www.vlfeat.org/`, 2008.

Appendix

Global Colour Histogram Visual Results

$Q = 2$

Figure 5: Global Colour Histogram: $Q = 2$

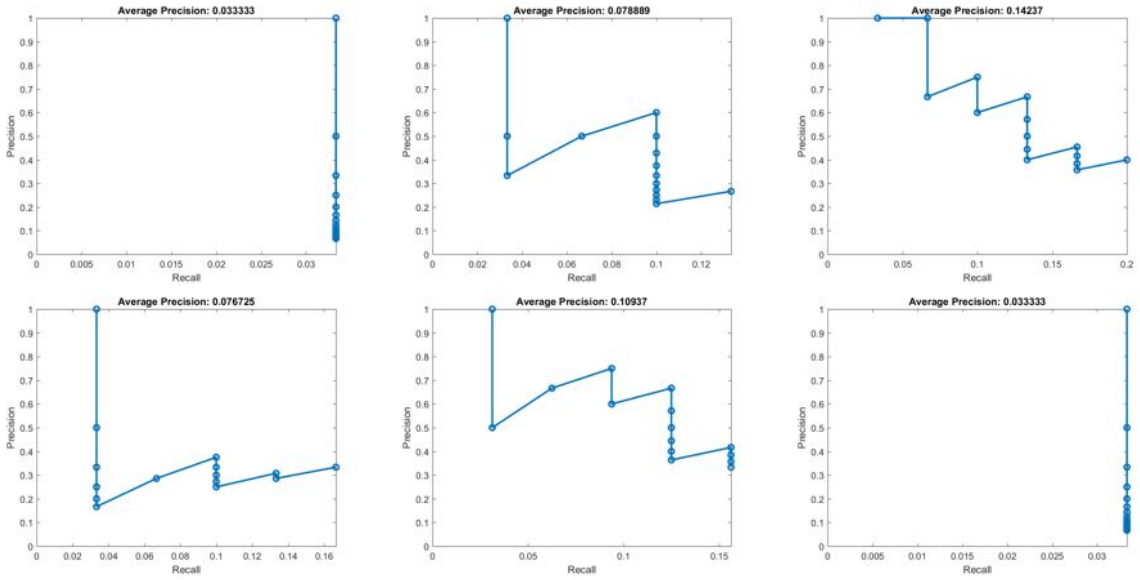


Figure 6: Global Colour Histogram PR Curves: $Q = 2$

$Q = 4$

Figure 7: Global Colour Histogram: $Q = 4$

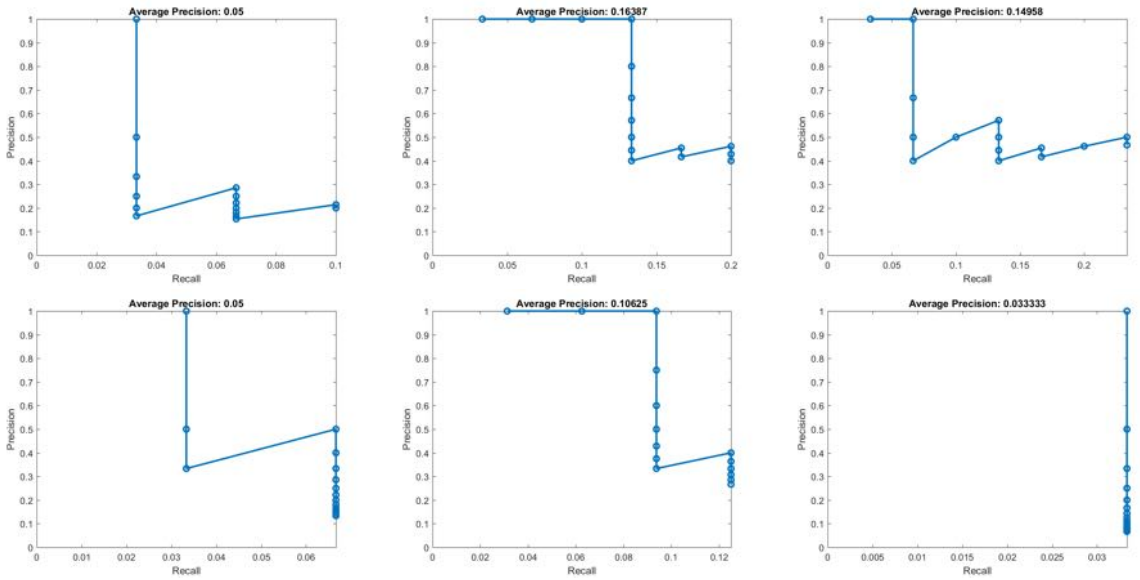


Figure 8: Global Colour Histogram PR Curves: $Q = 4$

$Q = 8$

Figure 9: Global Colour Histogram: $Q = 8$

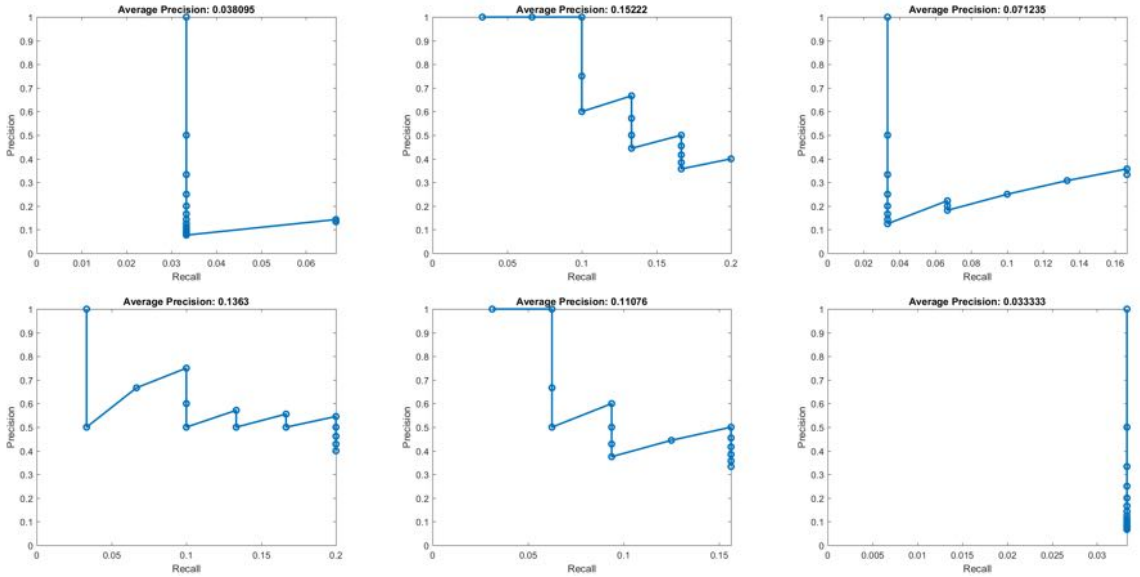


Figure 10: Global Colour Histogram PR Curves: $Q = 8$

$Q = 16$

Figure 11: Global Colour Histogram: $Q = 16$

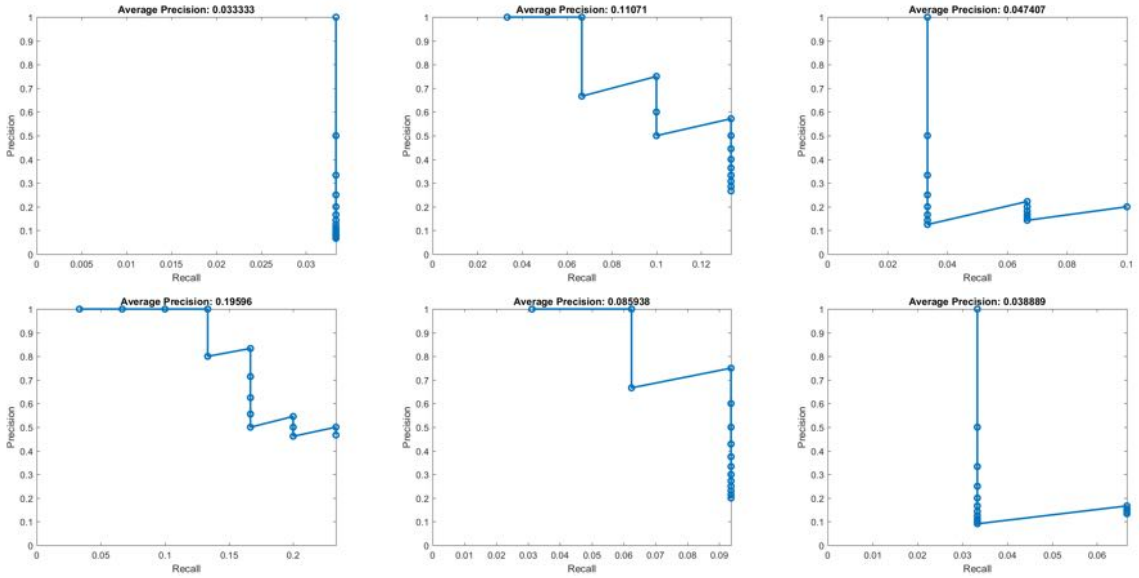


Figure 12: Global Colour Histogram PR Curves: $Q = 16$

Colour Spatial Grids Visual Results

2×2

Figure 13: Colour Spatial Grids: 2×2

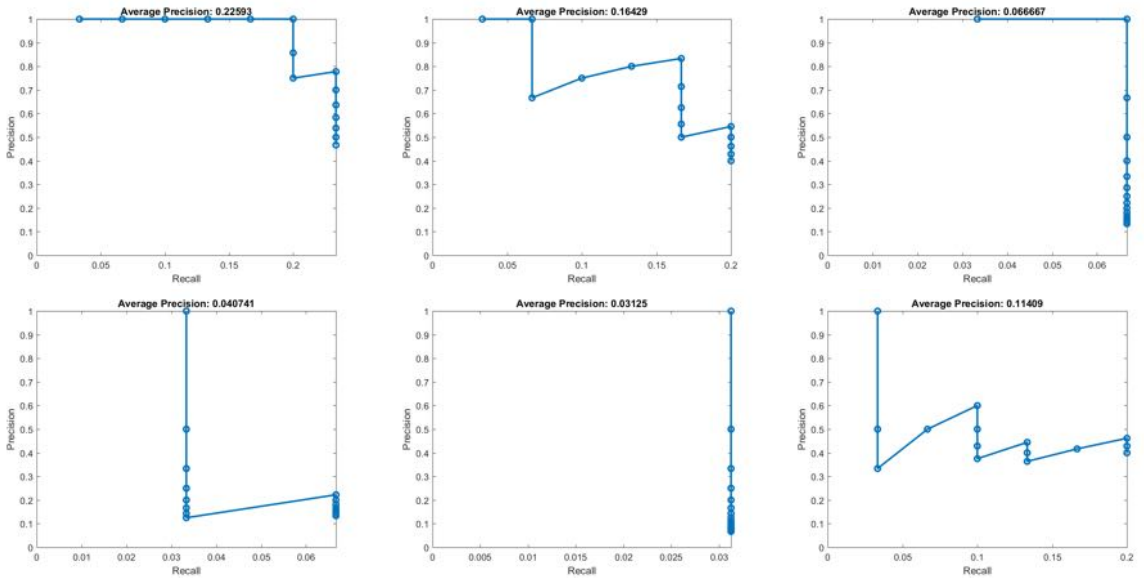


Figure 14: Colour Spatial Grids PR Curves: 2×2

4×4

Figure 15: Colour Spatial Grids: 4×4

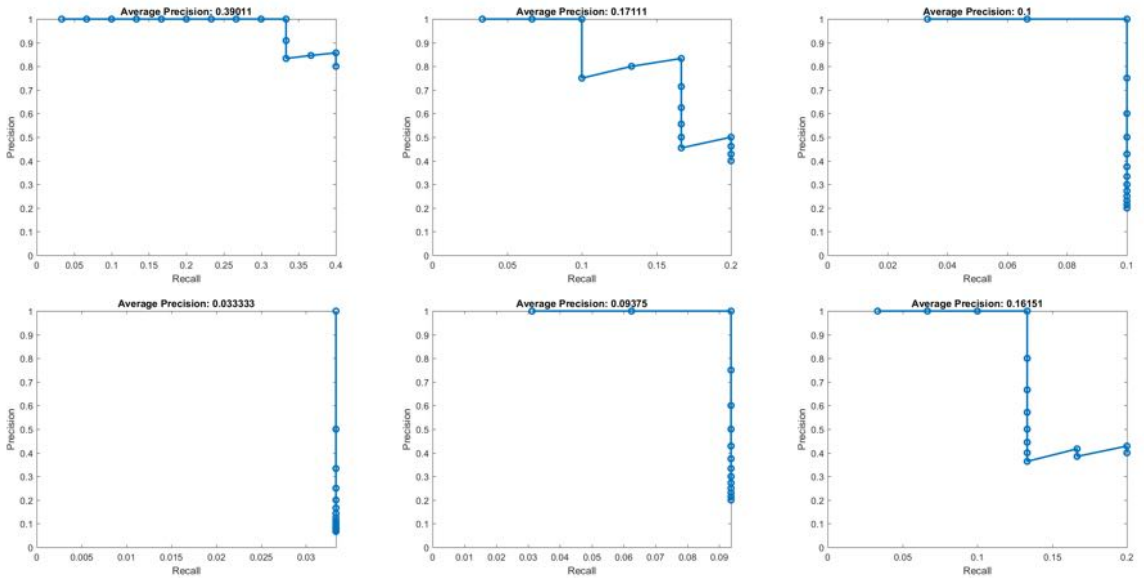


Figure 16: Colour Spatial Grids PR Curves: 4×4

8×8

Figure 17: Colour Spatial Grids: 8×8

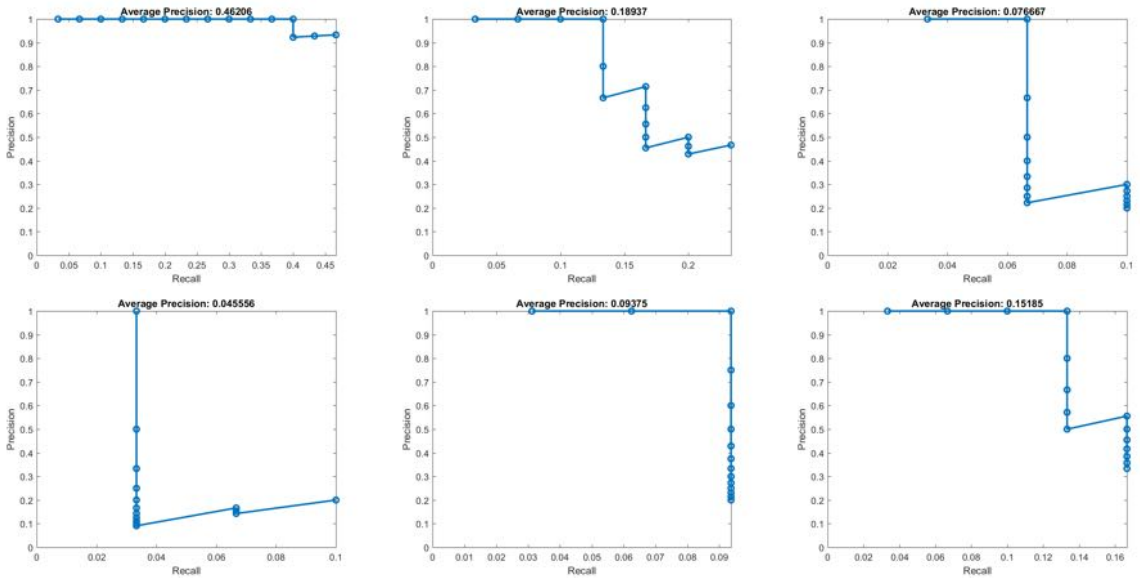


Figure 18: Colour Spatial Grids PR Curves: 8×8

16×16

Figure 19: Colour Spatial Grids: 16×16

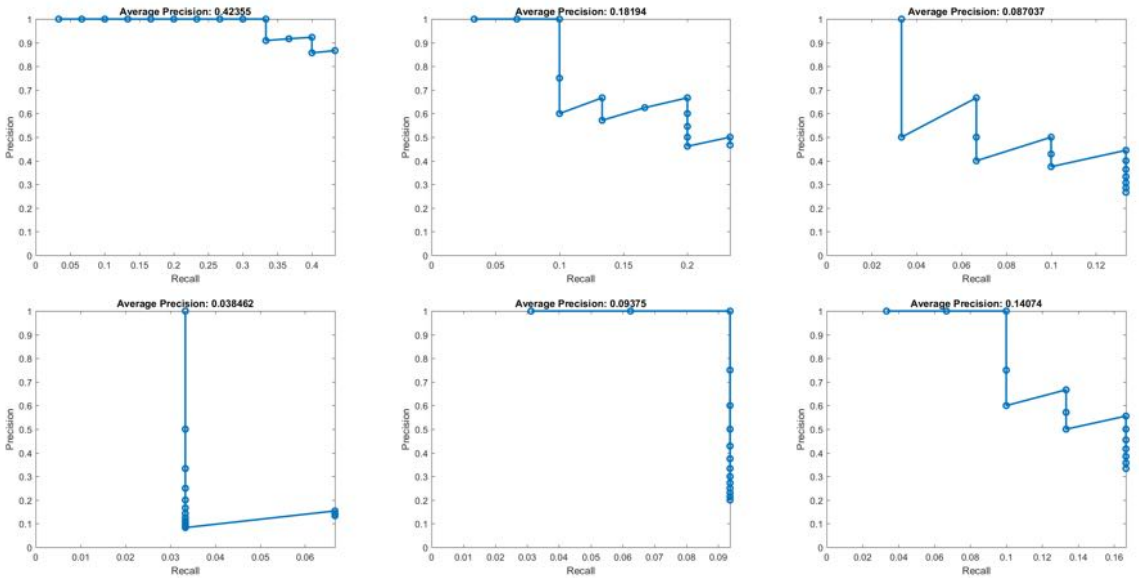


Figure 20: Colour Spatial Grids PR Curves: 16×16

EOH Spatial Grids Visual Results

$\theta = 2$

Figure 21: EOH Spatial Grids: $\theta = 2$

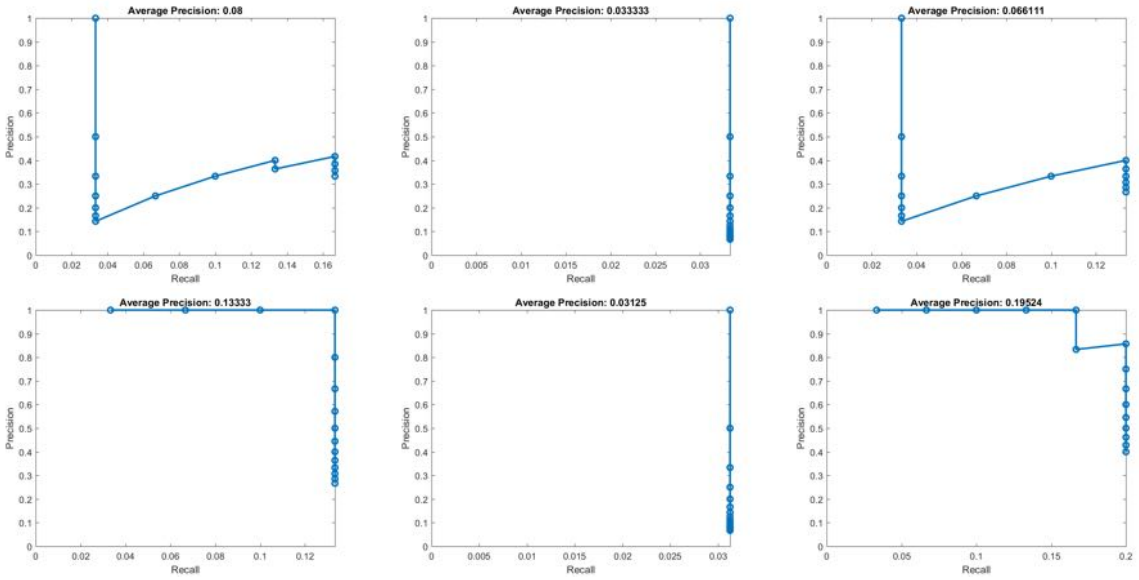


Figure 22: EOH PR Curves: $\theta = 2$

$\theta = 4$

Figure 23: EOH Spatial Grids: $\theta = 4$

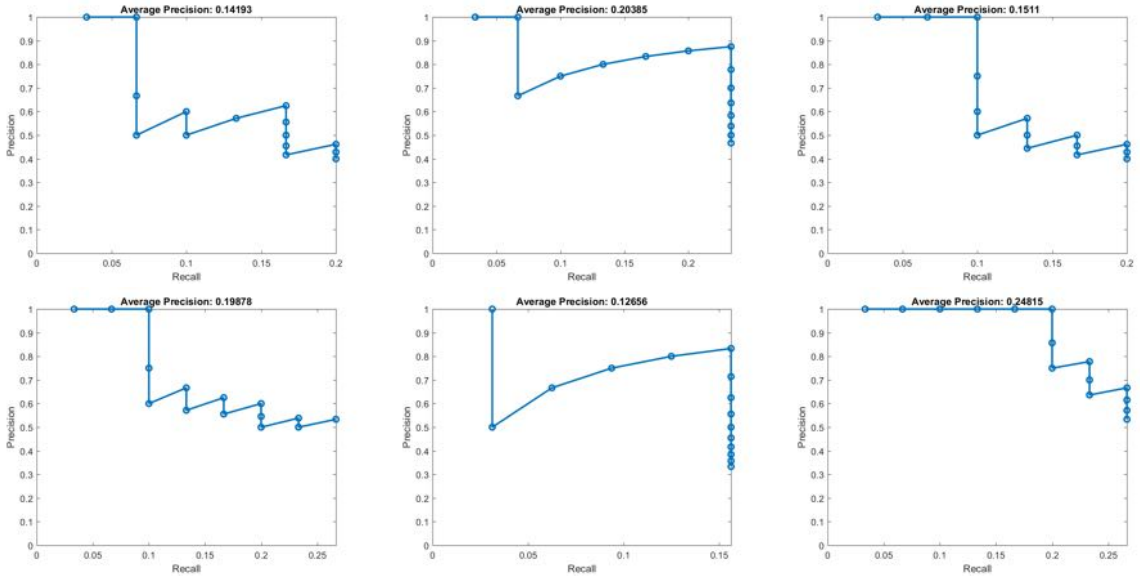


Figure 24: EOH PR Curves: $\theta = 4$

$\theta = 8$

Figure 25: EOH Spatial Grids: $\theta = 6$

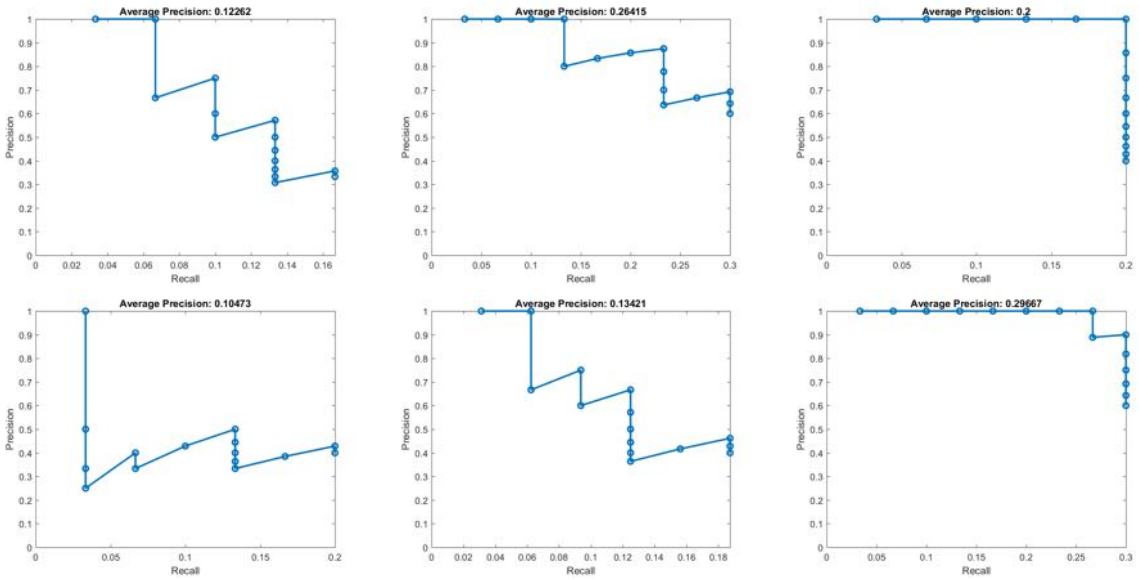


Figure 26: EOH PR Curves: $\theta = 6$

$\theta = 8$

Figure 27: EOH Spatial Grids: $\theta = 8$

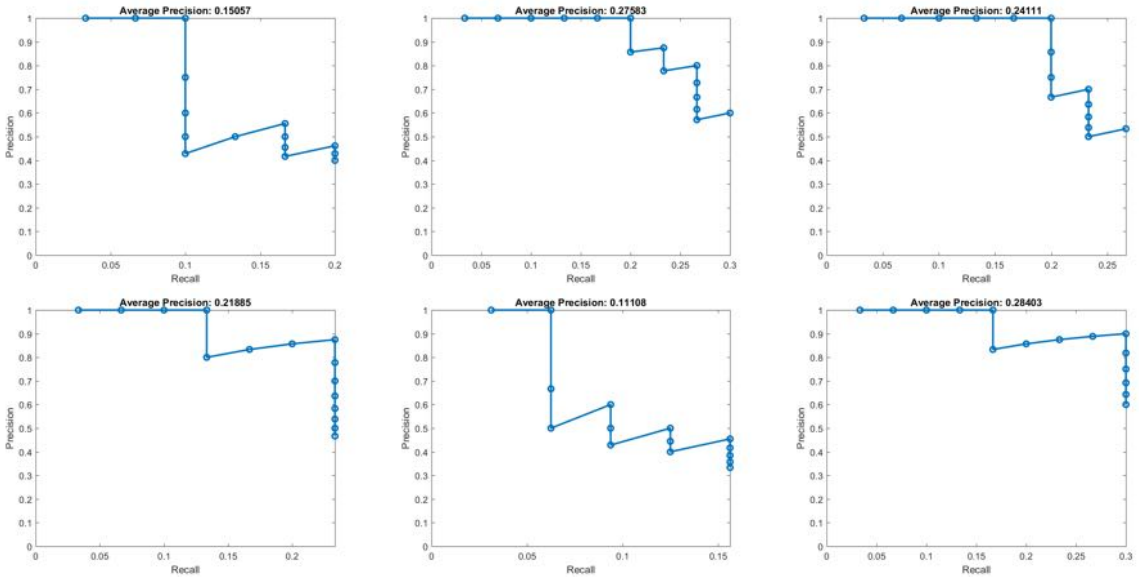


Figure 28: EOH PR Curves: $\theta = 8$

Combined Spatial Grids Visual Results

2×2

Figure 29: Combined Spatial Grids: 2×2

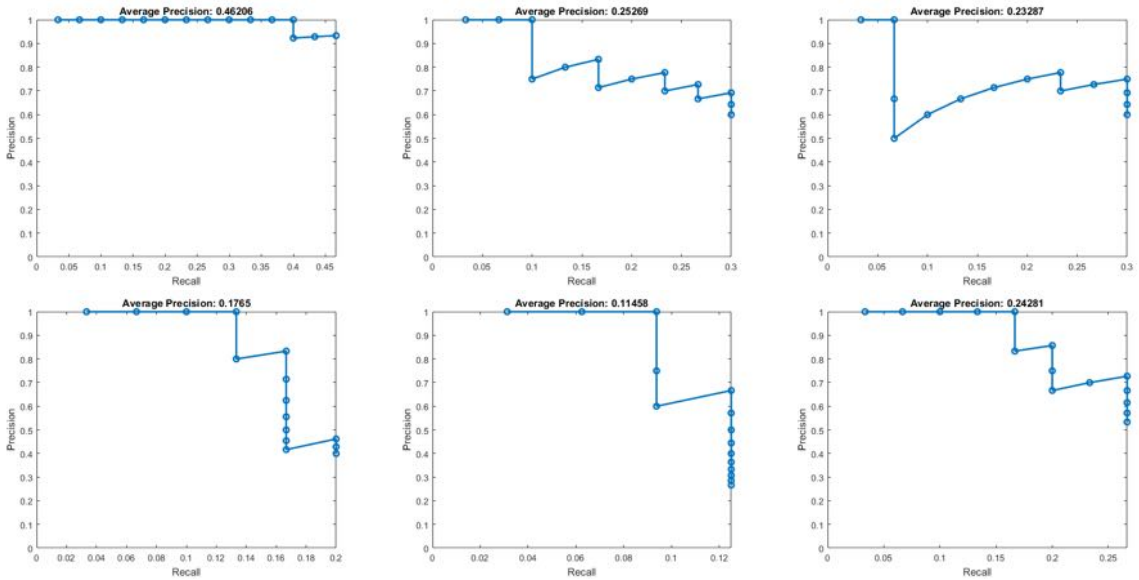


Figure 30: Combined Spatial Grids PR Curves: 2×2

4×4

Figure 31: Combined Spatial Grids: 4×4

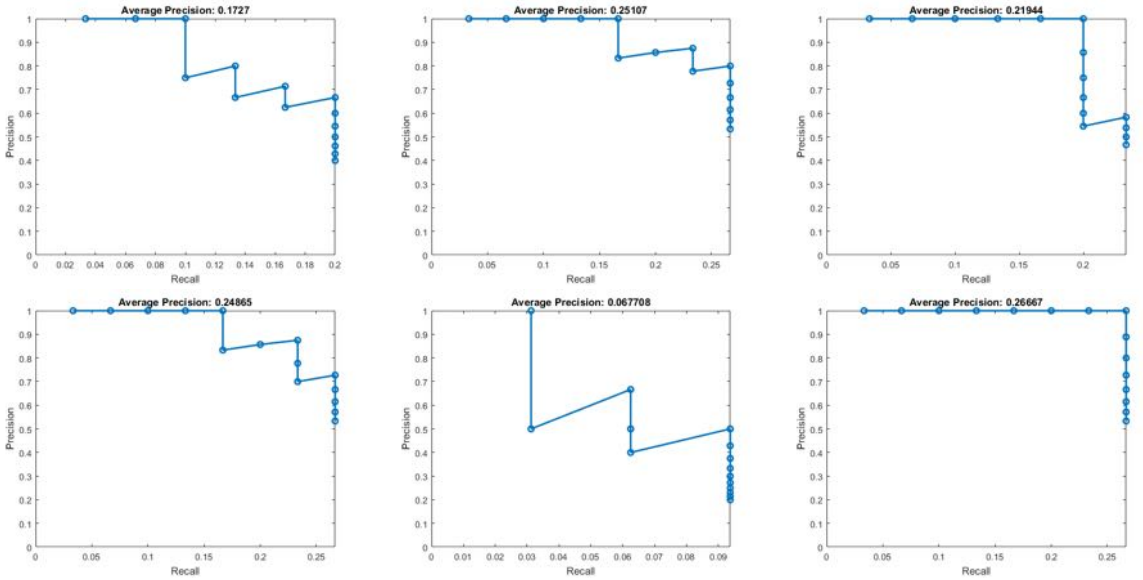


Figure 32: Combined Spatial Grids PR Curves: 4×4

8×8

Figure 33: Combined Spatial Grids: 8×8

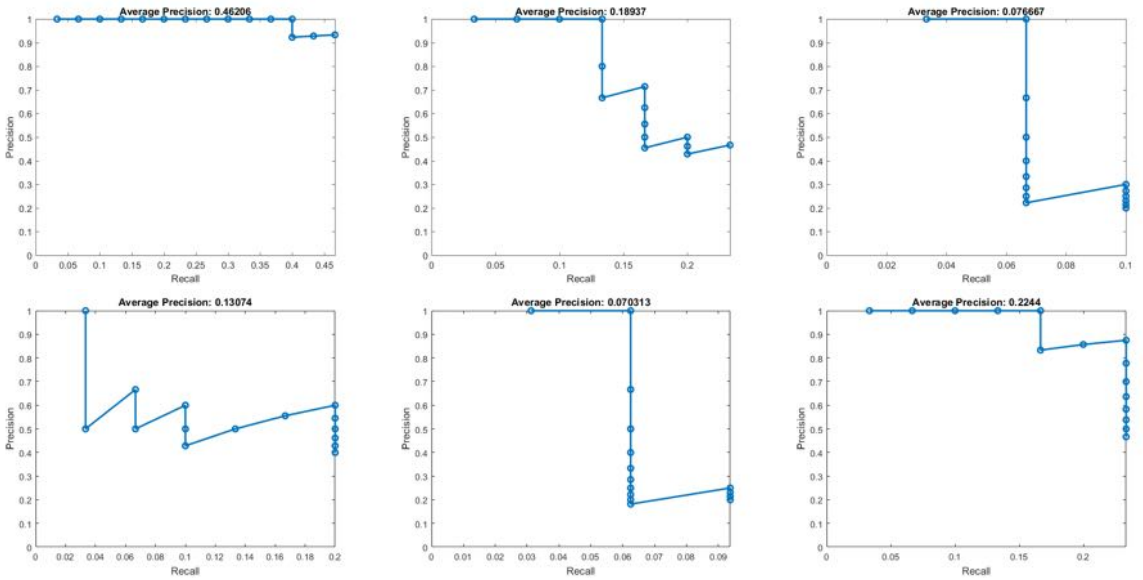


Figure 34: Combined Spatial Grids PR Curves: 8×8

16×16

Figure 35: Combined Spatial Grids: 16×16

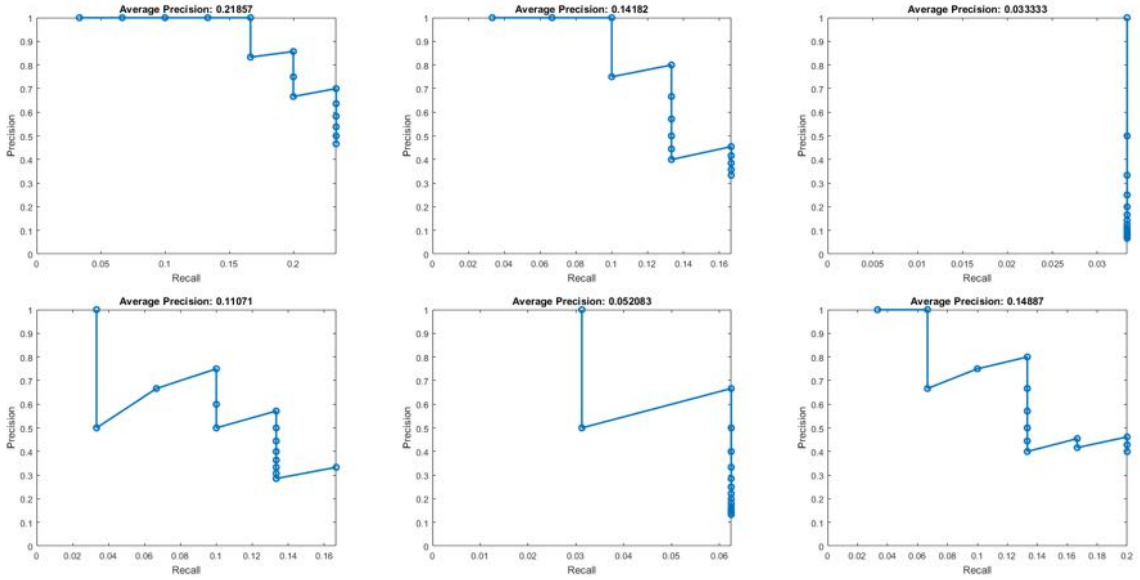


Figure 36: Combined Spatial Grids PR Curves: 16×16

PCA Visual Results

Global Colour Histogram

Figure 37: PCA with Global Colour Histogram

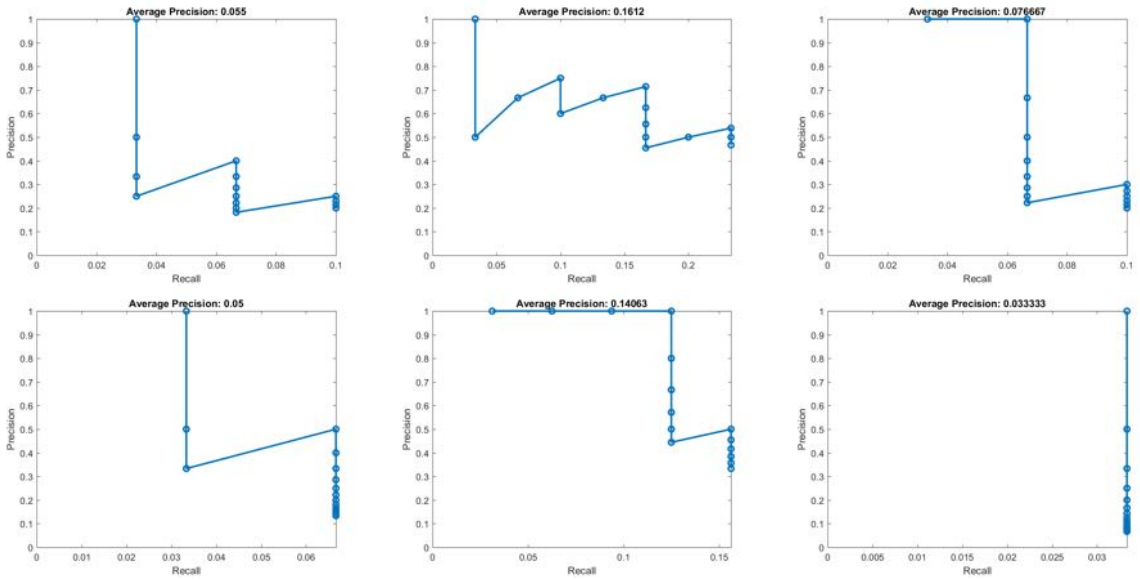


Figure 38: PCA with Global Colour Histogram PR Curves

Colour Spatial Grids

Figure 39: PCA with Colour Sptial Grids

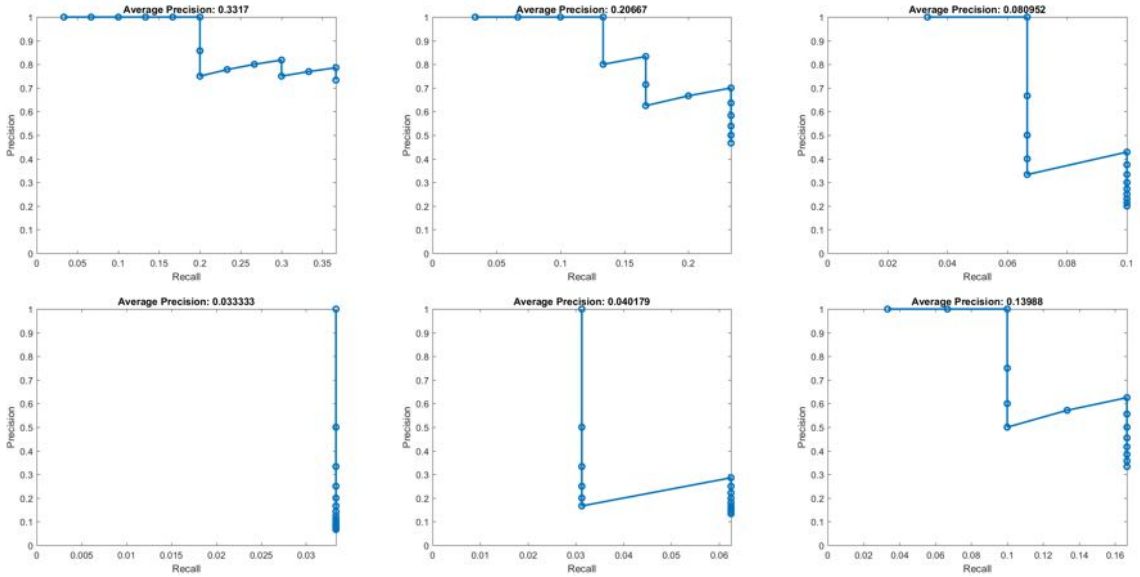


Figure 40: PCA with Colour Spatial Grids PR Curves

EOH Spatial Grids

Figure 41: PCA with EOH Spatial Grids

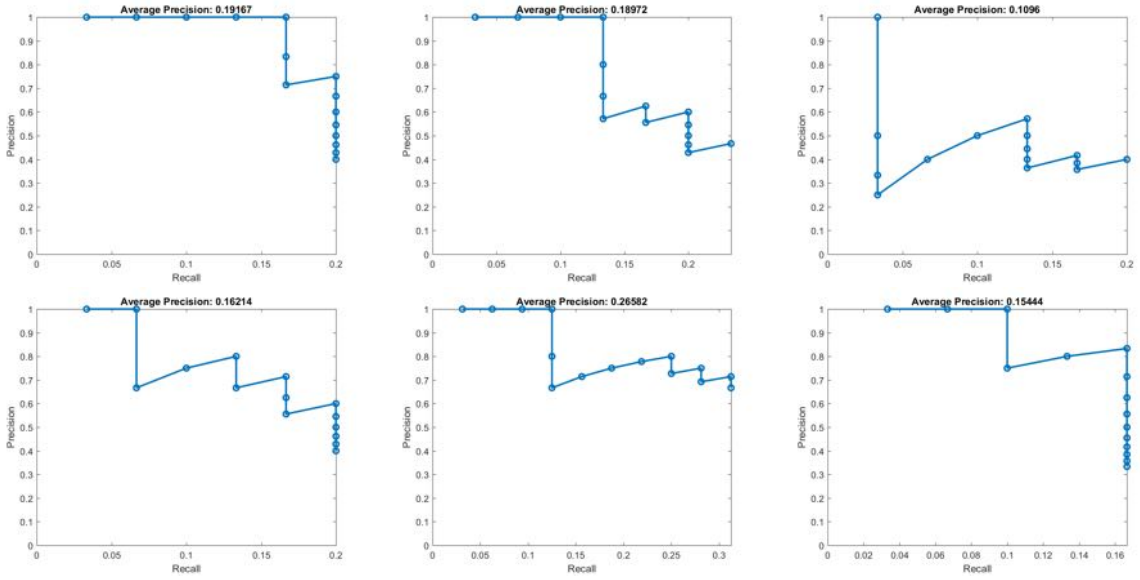


Figure 42: PCA with EOH Spatial Grids PR Curves

Combined Spatial Grids

Figure 43: PCA with Combined Spatial Grids

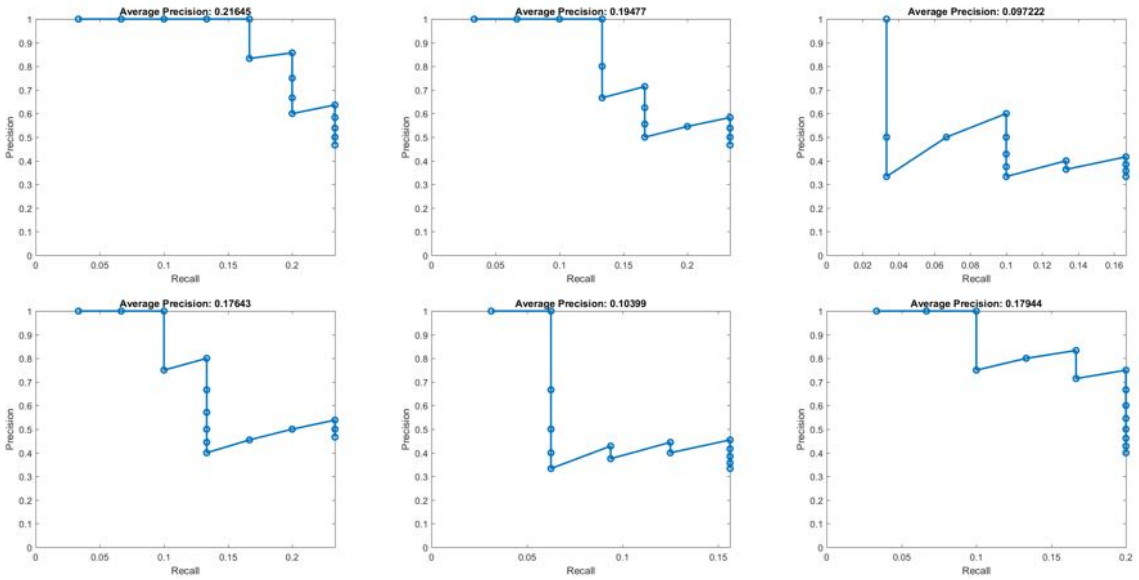


Figure 44: PCA with Combined Spatial Grids PR Curves

L_1 Visual Results

Global Colour Histogram

Figure 45: L_1 Norm: Global Colour Histogram

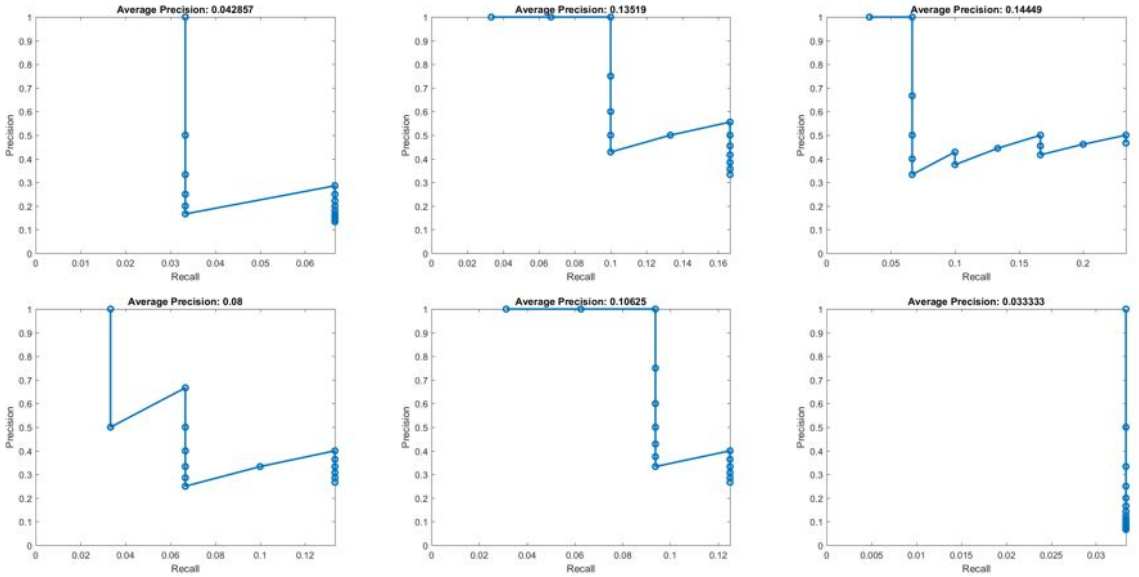


Figure 46: L_1 Norm: Global Colour Histogram PR Curves

Colour Spatial Grids

Figure 47: L_1 Norm: Colour Spatial Grids

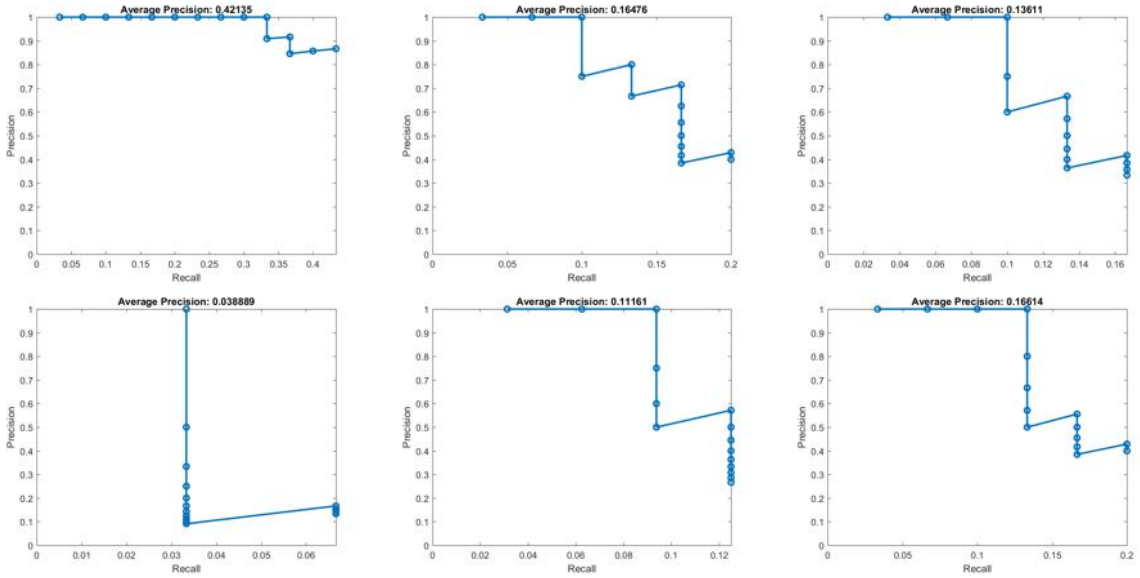


Figure 48: L_1 Norm: Colour Spatial Grids PR Curves

EOH Spatial Grids

Figure 49: L_1 Norm: EOH Spatial Grids

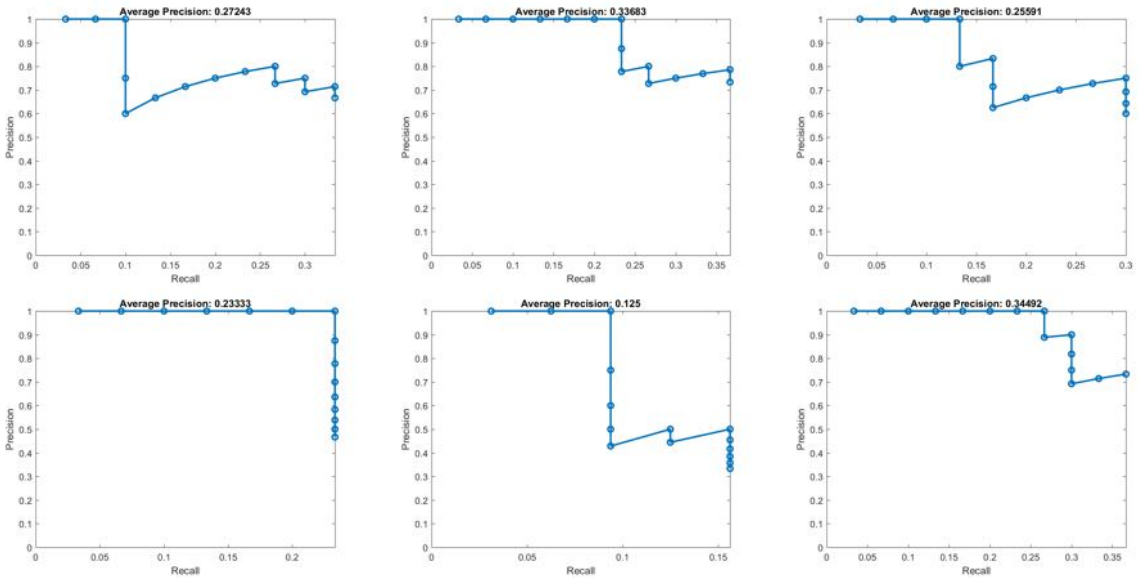


Figure 50: L_1 Norm: EOH Spatial Grids PR Curves

Combined Spatial Grids

Figure 51: L_1 Norm: Combined Spatial Grids

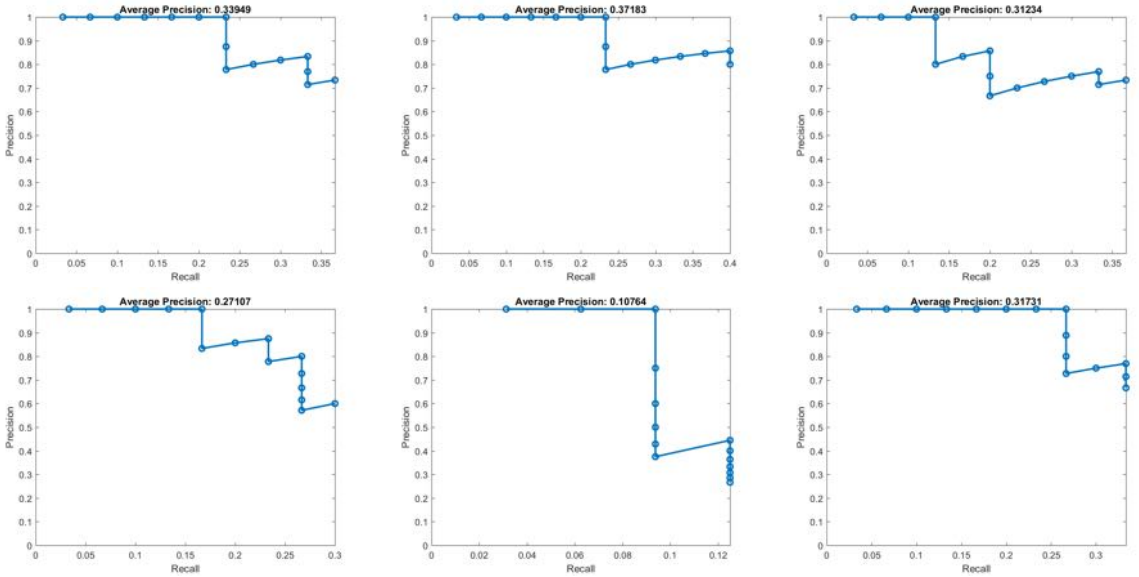


Figure 52: L_1 Norm: Combined Spatial Grids PR Curves

BoVW and SIFT, L_1 Norm

$k = 1000$

Figure 53: SIFT Cluster Size with L_1 Norm, $k = 1000$

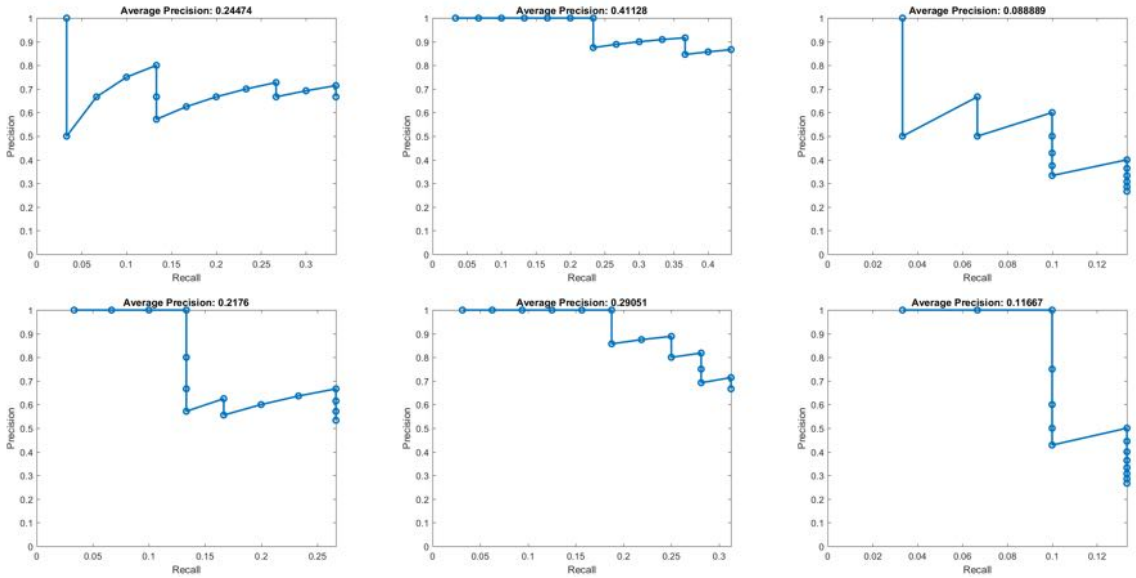


Figure 54: SIFT Cluster Size with L_1 Norm, $k = 1000$ PR Curves

$k = 2500$

Figure 55: SIFT Cluster Size with L_1 Norm, $k = 2500$

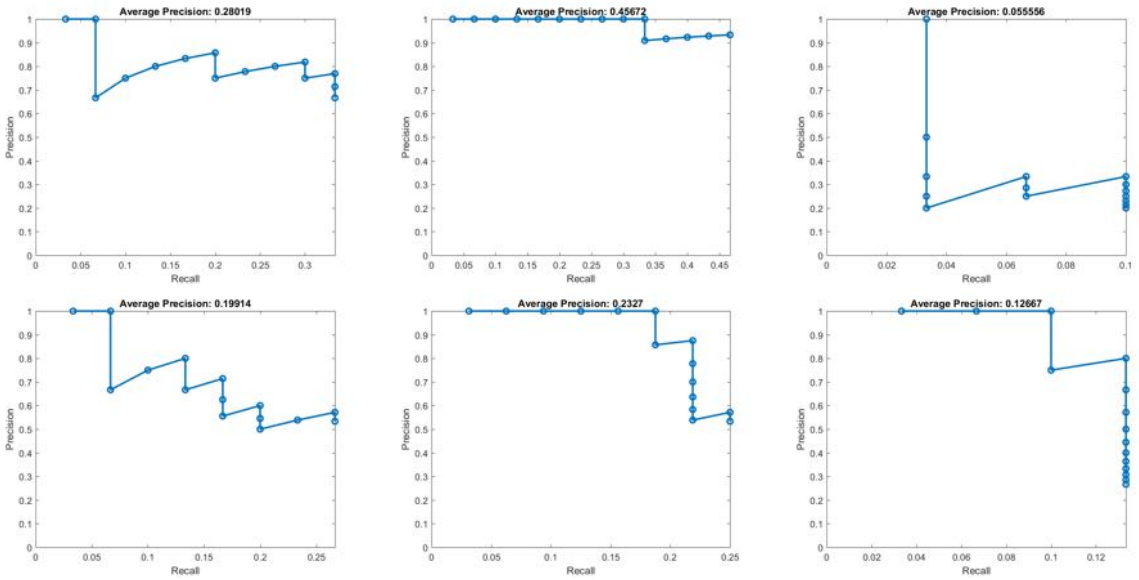


Figure 56: SIFT Cluster Size with L_1 Norm, $k = 2500$ PR Curves

$k = 5000$

Figure 57: SIFT Cluster Size with L_1 Norm, $k = 5000$

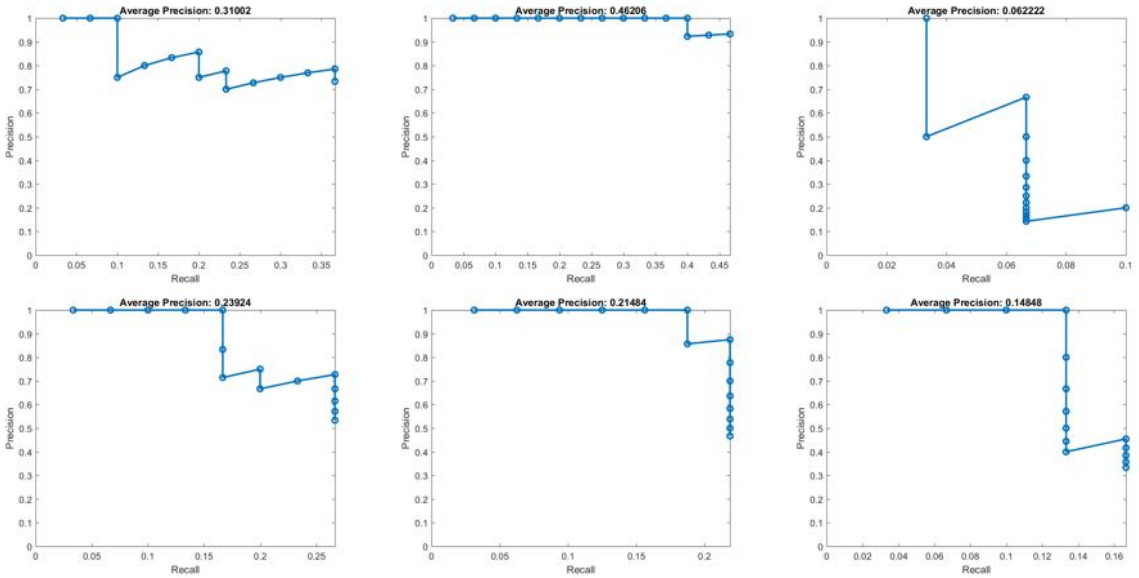


Figure 58: SIFT Cluster Size with L_1 Norm, $k = 5000$

$k = 10000$

Figure 59: SIFT Cluster Size with L_1 Norm, $k = 10000$

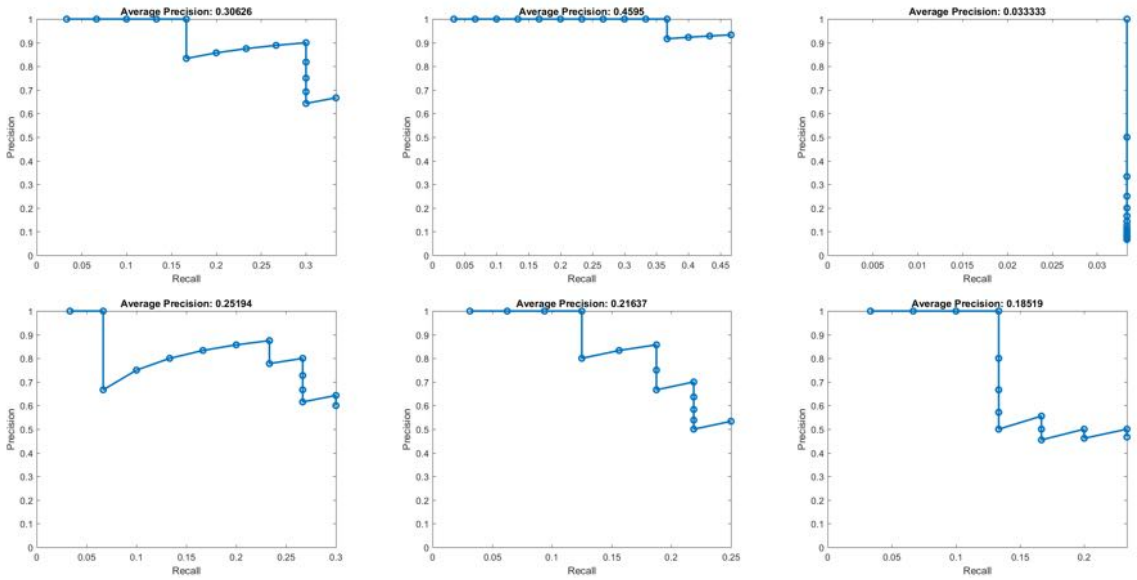


Figure 60: SIFT Cluster Size with L_1 Norm, $k = 10000$

BoVW and SIFT, L_2 Norm

$k = 1000$

Figure 61: SIFT Cluster Size with L_2 Norm, $k = 1000$

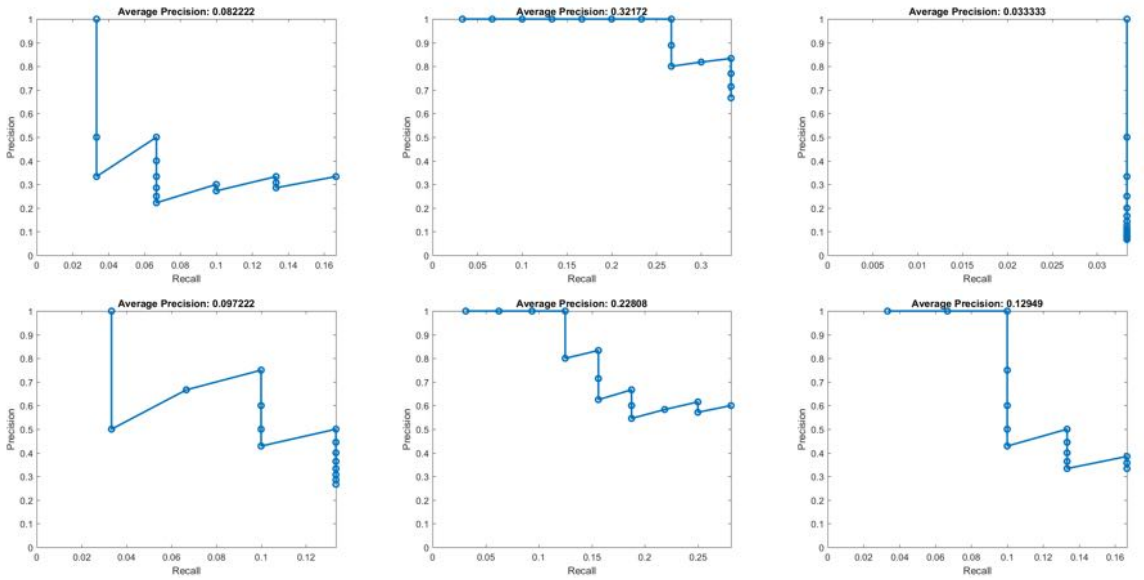


Figure 62: SIFT Cluster Size with L_2 Norm, $k = 1000$ PR Curves

$k = 2500$

Figure 63: SIFT Cluster Size with L_2 Norm, $k = 2500$

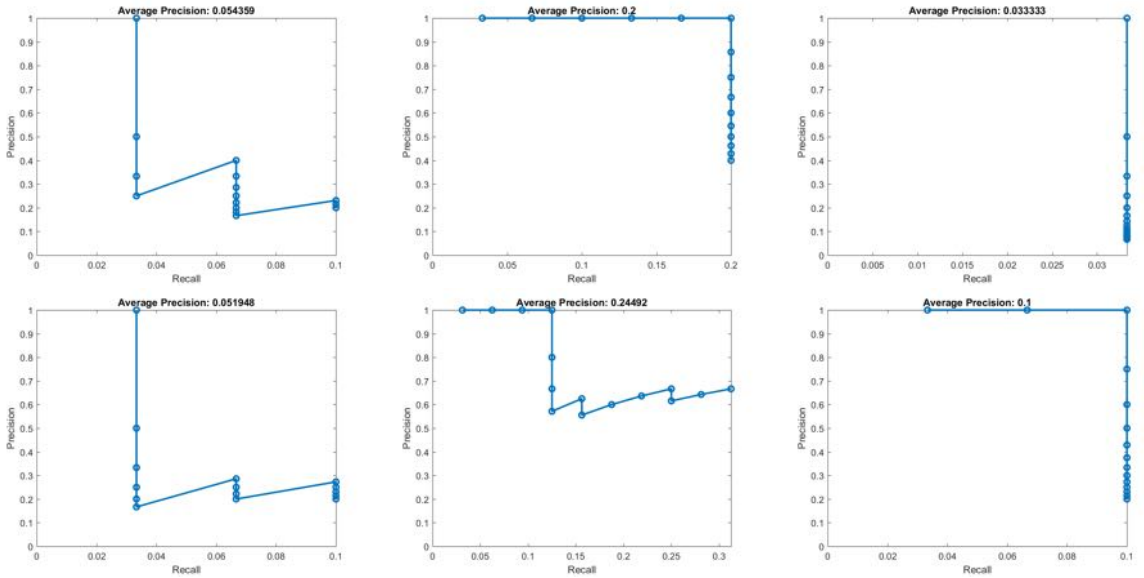


Figure 64: SIFT Cluster Size with L_2 Norm, $k = 2500$ PR Curves

$k = 5000$

Figure 65: SIFT Cluster Size with L_2 Norm, $k = 5000$

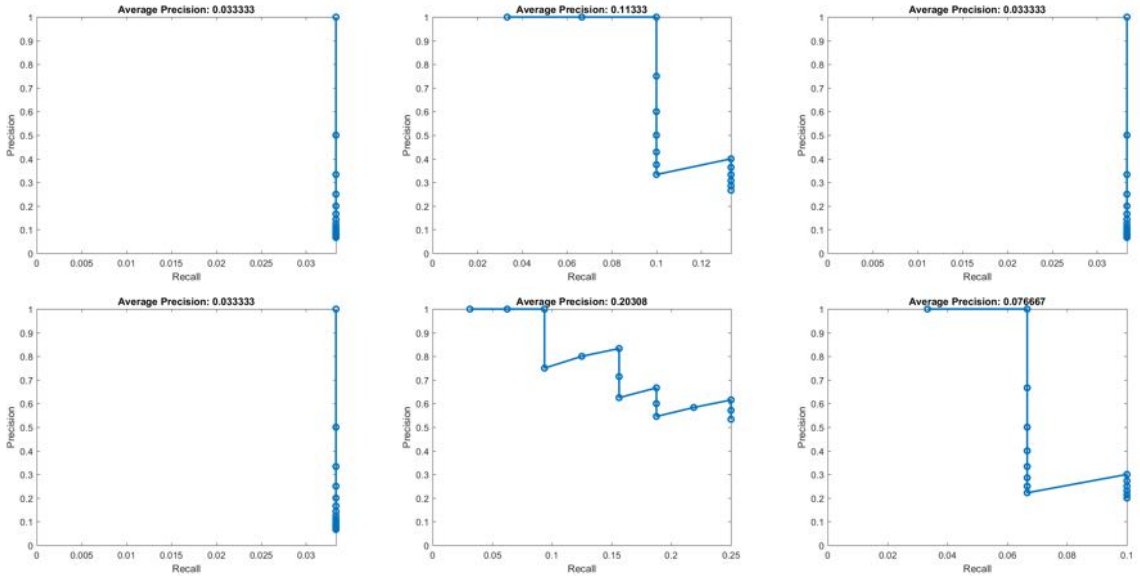


Figure 66: SIFT Cluster Size with L_2 Norm, $k = 5000$

$k = 10000$

Figure 67: SIFT Cluster Size with L_2 Norm, $k = 10000$

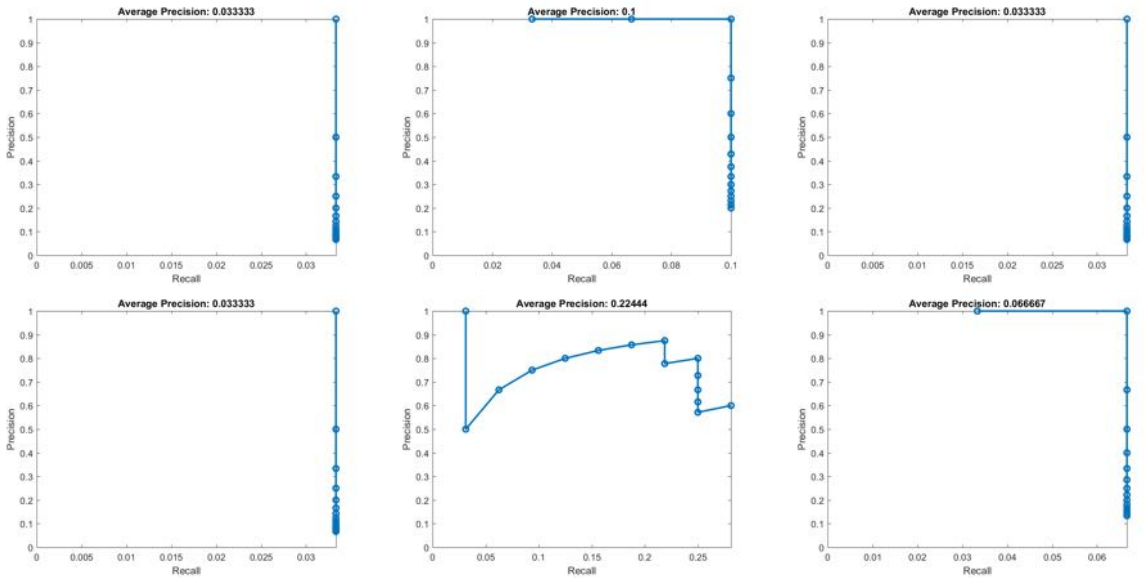


Figure 68: SIFT Cluster Size with L_2 Norm, $k = 10000$