Morphological Scale-Invariant Feature Transform

Gert-Jan van Ginkel

December 19, 2018

Abstract

This paper proposes a method of substituting the Scale-Invariant Feature Transform descriptors with descriptors based on morphological filters. Multiple configurations are tested for the descriptors, after which the optimal configuration is used to evaluate the new method on the COIL-100, UC Merced Land Use, and Zurich Building Image Database datasets. Not only are the descriptors generated by MorphSIFT less diverse, they also perform worse than the default SIFT descriptors. When using the descriptors in an image retrieval system that creates histograms of size k the MorphSIFT descriptors got the best performance for low values of k, while SIFT descriptors performed better with higher k. This indicates that MorphSIFT descriptors do not have the same encoding power as SIFT descriptors and ultimately have limited application.

1 Introduction

The Scale-Invariant Feature Transform is a popular method for extracting key points from images. These key points can be used for many purposes, including content-based image retrieval, image registration, and image stitching. Many attempts to derive better/faster algorithms for key-point extraction have been proposed, including ones based on mathematical morphology. The proposed method replaces the default SIFT descriptors with local descriptors based on pattern spectra. These pattern spectra are commonly used in image classification and previous work has been done to use these spectra as global [10] and local [2] descriptors.

Section 2 gives an overview of the SIFT method and Section 3 of the pattern spectra, after which Section 4 presents the newly proposed method. Section 5 describes the used datasets and Section 6 the experiments that are performed on those datasets, after which the results are described and discussed in Section 7.

2 Scale-Invariant Feature Transform

The Scale-Invariant Feature Transform (SIFT) method was first introduced by Lowe et al. in 1999 [6]. There are four stages in the SIFT method: scale space extrema detection; keypoint filtering; orientation assignment and keypoint description. All four stages are further described in their own dedicated subsection.

2.1 Scale space extrema detection

The keypoint locations are detected by finding minima and maxima in the scale space. In order to compute these locations a scale space pyramid is built with resampling between each level. To get stable locations the key points are located at regions and scales of high variation. Each level in the pyramid is smoothed using a 2D Gaussian with $\sigma = 2$. The difference of the Gaussian (DoG) is then obtained by subtracting the newly smoothed image from the previous layer. The next pyramid level is generated by resampling the previously generated layer with bi-linear interpolation with a pixel spacing of 1.5 in each direction. Using this spacing of 1.5 means that the new sample is a linear combination of the four adjacent pixels.

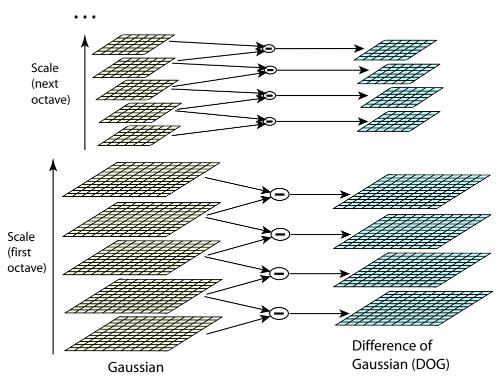


Figure 1: Scale space pyramid [5]

The minima and maxima are computed by comparing each pixel $D(x, y, \sigma)$ in the pyramid to its neighbors. The first step is comparing the pixel to the neighbors on the same level. If this pixel is indeed a minimum or maximum on the level, the check is again computed for the layer above and below while taking into account the pixel spacing of 1.5. When the pixel passes all tests it is added to the candidate list of keypoints.

2.2 Keypoint filtering

Another phase is needed to filter keypoints that result from noise or are located close to an edge. The following attributes are computed for each keypoint: location, scale, and ratio of principal curvatures.

The first step is to fit a 3D quadratic function to the local sample points which can be used to more accurately identify the local extremum. This quadratic function is approximated by the Taylor expansion of the scale space function to the quadratic term. This function is shifted in such a way that the sample point is at the origin of this approximation:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{1}$$

where $\mathbf{x} = (x, y, \sigma)$ is the offset from the keypoint. The location of the extremum $\hat{\mathbf{x}}$ is then calculated by deriving $D(\mathbf{x})$ with respect to $\mathbf{x}$ and setting it to zero:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}} \tag{2}$$

This is then approximated by using the differences of the neighboring sample points. One problem that can arise is that the local extremum is found to have an offset of more than 0.5 in any dimension. This offset implies that the true extremum is not on the sample point, but one of its neighbors. In such a case the routine is repeated for the newly found extremum and the current sample point discarded.

Finally, the value $D(\hat{\mathbf{x}})$ can be used to filter against extrema with a low contrast. This value can be computed using the following equation:

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}} \tag{3}$$

The value $|D(\hat{\mathbf{x}})|$ can then be used to filter out low contrast keypoints. The threshold value recommended by Lowe et al. is 0.03.

2.3 Eliminating edge responses

The next step in filtering the keypoints is eliminating edge responses. The DoG inherently includes edges, but noise can influence the location of the keypoint along the edge. The fluctuations in the location along the edge make these keypoints unstable, so they need to be filtered out. An edge has a large principal curvature in one direction while having a small one in the orthogonal direction. As such, we can use this property to filter out keypoints located along an edge. The principal curvature can be computed by the following Hessian matrix:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{4}$$

This Hessian matrix can be approximated by taking the differences of neighboring sample points. The principal curvature corresponds to the ratio of the eigenvalues of $\mathbf{H}$, which means we can avoid having to calculate the eigenvalues of $\mathbf{H}$. As such, let α be the eigenvalue with the largest magnitude and β the eigenvalue with the smallest magnitude. These can be used to express the trace and determinant of the matrix in terms of the eigenvalues α and β:

$$\begin{aligned}\mathrm{Tr}(\mathbf{H}) &= D_{xx} + D_{yy} = \alpha + \beta \\ \mathrm{Det}(\mathbf{H}) &= D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta\end{aligned}$$

Let r be the ratio between the largest and smallest eigenvalue, such that $\alpha = r\beta$. Substituting this definition into the formula yields us the following:

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{5}$$

In order to check whether the ratio of two eigenvalues of less than r we only need to evaluate the following equation:

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r} \tag{6}$$

This is a cheap computation and avoids having to calculate the eigenvalues α and β, while still yielding the desired result. The threshold used by Lowe et al. is $r = 10$.

2.4 Orientation assignment

The final step in the keypoint localization is the orientation assignment. Incorporating this orientation means that the subsequent steps can perform on a consistently oriented keypoint, resulting in rotation invariance. The scale s of the keypoint is used to select the properly scaled image L from the Gaussian pyramid, such that the calculations are performed invariant to scale. The magnitude $m(x, y)$ and orientation $\theta(x, y)$ of each pixel in the neighborhood are defined as follows:

$$m(x,y) = \sqrt{m_h(x,y)^2 + m_v(x,y)^2} \tag{7}$$

$$\theta(x,y) = \tan^{-1}\frac{m_v(x,y)}{m_h(x,y)} \tag{8}$$

where

$$\begin{aligned}m_h(x,y) &= L(x+1,y) - L(x-1,y) \\ m_v(x,y) &= L(x,y+1) - L(x,y-1)\end{aligned}$$

Next, a histogram with 36 bins of the orientations is created, weighted by the magnitudes and multiplied by a Gaussian window with $\sigma = 1.5s$. The keypoint is assigned the dominant orientation of the histogram. Another keypoint with the same scale and position is added if there is another orientation with a weight that is at least 80% of the weight of the dominant orientation.

2.5 Descriptors

The last step in SIFT is creating keypoint descriptors, which can then be used to describe the image. First of all, the gradient orientation and magnitude are calculated for all the pixels in the neighborhood of the keypoint. The orientation of these gradients are relative to the orientation of the keypoint, maintaining rotation invariance.

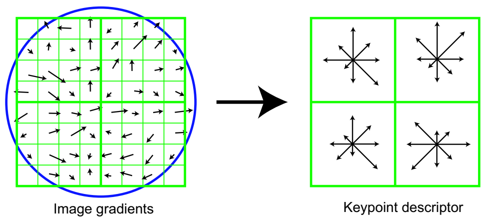


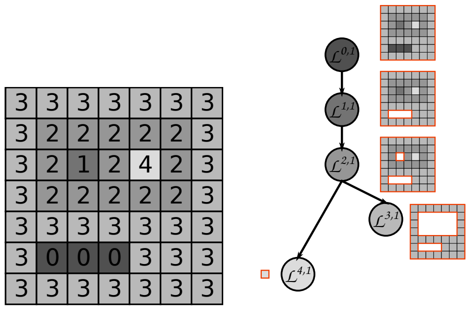
Figure 2: Descriptor computation for SIFT. The blue circle denotes the Gaussian window [5].

These gradients are then aggregated into 4×4 regions by the means of an orientation histogram with 8 bins. The magnitudes of the gradients are multiplied by a Gaussian window centered on the keypoint, such that gradients close to the center contribute more to this histogram. Figure 2 shows the keypoint descriptor for an 8×8 patch, which results in a 2×2 descriptor array. Note that SIFT uses a 16×16 patch with a 4×4 descriptor array, resulting in 128-dimensional feature descriptor.

3 Pattern Spectra

3.1 Max-tree

The first step to extracting pattern spectra is to encode the image as a tree. One example of such a tree is the max-tree. The nodes in the max-tree correspond to connected components with the same or higher gray level k, such that $\mathcal{L}^k = \{p \in I | f(p) \geq k\}$, where $f(p)$ is the gray value for pixel p in the image I. Another indexing component i is required because there could be multiple peak components, such that the i-th peak with value k is denoted by $\mathcal{L}^{k,i}$. These are nested for decreasing values of k. Constructing the full tree results in a max-tree such that it contains nodes that correspond to each peak component.



(a) (b)

Figure 3: An image (a) with the computed max-tree (b). Taken from [2].

Figure 3 shows an example of an image with the associated max-tree. Note that the tree splits after node $\mathcal{L}^{2,1}$ into two peaks $\mathcal{L}^{4,1}$ and $\mathcal{L}^{3,1}$. If, for example, the region $\mathcal{L}^{4,1}$ had a value of 3 it would be indexed as node $\mathcal{L}^{3,2}$. Additionally, another tree is required to perform operations on the dark areas of the image. A min-tree is used instead of a max-tree, such that $\mathcal{L}^k = \{p \in I | f(p) \leq k\}$. This min-tree can also be built by building a max-tree of the inverted image instead, thus simplifying the entire process.

3.2 Attributes

Attributes can be computed on the nodes of the max-tree to describe the node. Two of these attributes are discussed and used: area and the corrected noncompactness (CNC).

Area The area of a node is the simplest attribute to compute, as it is simply defined as the number of pixels that belong to that node. The area is shape and rotation invariant, making it useful for a descriptor.

Corrected noncompactness (CNC) The CNC is an attribute which describes the shape of a node. The shape is described by an elongation measure, defined as follows:

$$CNC(\mathcal{L}^{k,i}) = 2\pi \left(\frac{I(\mathcal{L}^{k,i})}{A(\mathcal{L}^{k,i})^2} + \frac{1}{6A(\mathcal{L}^{k,i})} \right) \tag{9}$$

where $I(\mathcal{L}^{k,i})$ is the moment of inertia of node $\mathcal{L}^{k,i}$, which in turn corresponds to the corrected first moment invariant of Hu [4]. The CNC attribute is 1 when the node is a perfect circle and grows as the shape of the node approaches a line. This attribute is translation, rotation and scale invariant, making it a very useful attribute for describing a node in the tree.

3.3 Granulometry

The final step in computing pattern spectra is aggregating the attributes for each node in the max-tree into a single pattern spectrum.

The size spectrum of an image can be computed by measuring the amount of detail removed in a size granulometry, resulting in a 1D histogram containing the sum of gray levels for each size class. Using the max-tree provides an advantage, since this histogram can be computed in a single pass on the max-tree instead of filtering the image repeatedly. The shape spectrum can be computed in a similar manner, but using the shape attribute instead. By computing both the size and shape spectrum we can combine these into a single 2D size-shape spectrum.

4 MorphSIFT

MorphSIFT aims to combine the robustness of keypoint location and scale space keypoint description of SIFT and the fast and accurate feature description of pattern spectra. Instead of computing the pattern spectra of keypoints from the raw image it first extracts a patch from the DoG pyramid, after which the pattern spectrum is computed from that patch. It allows for a lot of flexibility, such as using various kinds of attributes on nodes and combinations thereof.

Keypoints in the image are found in the exact same manner to SIFT. A scale space is built from the image, after which the method delineated in Section 2 finds the locations of the keypoints in the scale space. No additional filtering on the keypoints is performed.

4.1 Patch extraction

Patches are extracted from the difference of Gausssian (DoG) pyramid around the keypoint located by SIFT. All keypoints are found on a specific location (x, y, σ) in the pyramid, such that a square patch can be extracted on that location.

There are two pyramids that one can choose from: the Gaussian or DoG pyramid. Note that while the found keypoints are located on the DoG pyramid they could still be mapped to the Gaussian pyramid. Patches extracted from the Gaussian pyramid, however, do not contain the same information as the DoG pyramid. In fact, the DoG pyramid contains more information as it is a linear combination of two layers on the Gaussian pyramid. Some experiments were performed extracting patches from the Gaussian pyramid, but these consistently scored worse than extracting patches from the DoG pyramid.

Additionally, the orientation computed by SIFT could be used to rotate the patch before computing the pattern spectrum. Again, empirical experimentation showed little improvement, since the pattern spectra used are inherently rotation-invariant. The orientation could still be useful, though, since they could enable the use of non-rotation invariant attributes.

4.1.1 Masking

An additional mask can be used to restrict the computation of the pattern spectrum to a particular area. One example of such a mask is a circular mask, which would match better with the way the Gaussian pyramid is computed – the Gaussian function on a plane also has a circular shape. This implicates that the value on the keypoint was influenced by a circular area around the point, such that one would expect that the circular area would be the most descriptive of that point.

4.2 Feature description

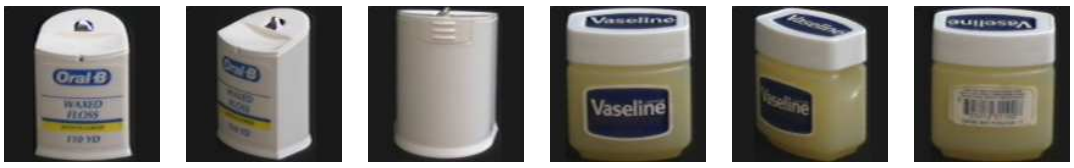
Two max-trees are built with the extracted patch E and the inverted variant E^{-1} respectively. These max-trees are then used to compute attributes of the max-tree and build a pattern spectrum. Combining the pattern spectra $PS(E)$ of E and $PS(E^{-1})$ results in a combined feature vector $(PS(E), PS(E^{-1}))$.

5 Datasets

Experiments were performed on three different datasets: COIL-100 [8], the UC Merced Land Use dataset [11] and the Zurich Building Image Database [9]. These datasets contain very different types of images and different retrieval goals.

COIL-100 The COIL-100 dataset contains 7200 images of 100 objects. The object is put on a rotating platform and photos are taken at every 5 degree interval, such that there are 72 photos of every object. All images have a 128×128 resolution. Only ten images at fixed angles were used for the training and testing. These ten images are taken from angles $0, 5, 15, 30, 45, 50, 65, 100, 145, 180$, in a similar fashion to Bakar et al. [1]. Figure 4 shows two different objects of the dataset, each from angle $0°, 45°$ and $180°$. These images on average contain 44.81 features.

UC Merced Land Use The UC Merced Land Use dataset contains 2100 256×256 aerial



(a) (b) (c) (d) (e) (f)

Figure 4: Two objects from the COIL-100 database with angles 0°, 45° and 180°.

images of 21 different classes of land use from the U.S. Geological Survey (USGS) National Map, such that there are 100 images of every class. Figure 5 shows images from the harbor and airplane categories. The average number of features per image is 677.8, although there is a huge variety per image class as described by Yang et al. [12].

Zurich Building Image Database The Zurich Building Image Database [9] (ZuBuD) contains 1000 images of 200 buildings in Zurich, such that there are 5 photos from different perspectives of any building. Figure 6 shows photos of two buildings, each taken from a different viewpoint. On average the images contain 1705 features.

6 Experiments

This section will first describe the experiments performed to obtain the final pattern spectra configurations used by the rest of the experiments. Next, this configuration is further elaborated upon in Section 6.2, after which the retrieval process is described for the second experiment.

6.1 Pattern spectra configuration on the COIL-100 dataset

In order to explore the effect of various pattern spectra configurations a simple test was created to evaluate these. The choice for the computationally cheap test results from the fact that many iterations were required to test the many different types of pattern spectra. These tests were performed on the COIL-100 dataset, since their small size allowed for quick prototyping.

First of all, a random 80/20 split is made on the images such that there are 800 training images and 200 testing images. In order to classify a test image we first consider all features of that image and match these features to their closest feature in the training data by means of a 1-nn classifier. This feature belongs to a particular class of images, such that all features in the test image can cast a vote for this class. The final classification for this test image is obtained by taking the most occurring feature classification of that image. The final classification for this image is obtained by taking the most occuring feature classification of that image. Finally, the percentage of correctly classified images is taken as the result. This process is repeated 10 times with different training and testing splits, after which all results are averaged to obtain the final performance of the descriptors.

Three different sizes for the patch extraction were tested: 12×12, 16×16 and 24×24. In addition the circular mask was enabled or disabled. Finally, the binning range for the CNC attribute was varied with $[1.0, 2.0]$, $[1.0, 3.0]$ and $[1.0, 4.0]$.

6.2 Pattern spectra configuration

Similar settings to Bosilj et al. [3] were used for the pattern spectra to maintain a proper comparison [10] among the different methods. Some settings, however, were adjusted according to experimentation on the COIL-100 dataset as described in Section 6.1.

The size of the patch was chosen to be 16×16, similar to the size of the feature descriptors used in SIFT. Both the area (A) and the corrected non-compactness (CNC) are used as attributes of the pattern spectrum. This

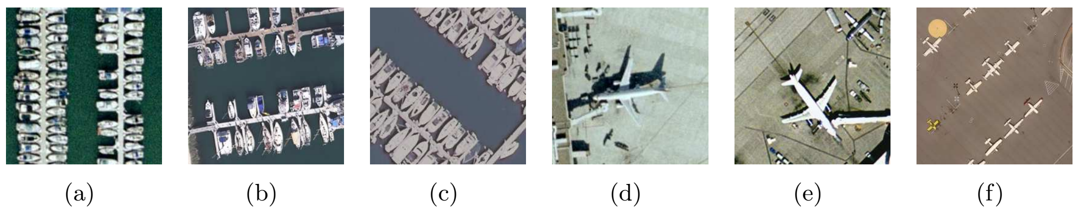


Figure 5: Three samples from the harbor and airplane categories from the UC Merced database.

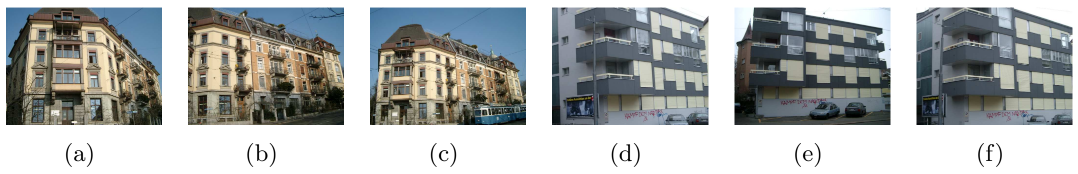


Figure 6: Two buildings with three images each from the ZuBuD dataset.

2-dimensional spectrum of dimensions 10×6 with range $(1, 16^2$ for area and $[1.0, 2.0]$ for the CNC. This results in a 60-dimensional feature vector for a single tree, summing up to a 120-dimensional feature vector for the entire patch.

These settings are used for all further experiments on the COIL-100, UC Merced and ZuBuD dataset.

6.3 Retrieval process

An image retrieval process similar to the one used by Yang et al. [12] was used to compare the performance of the SIFT descriptor with the pattern spectra descriptors.

The first step in image retrieval is to generate the training features for all the training images. Next, k codebooks are generated by computing k clusters from all generated features using k-means. Note that all features are pooled together regardless of the corresponding class. All training features are matched against their closest cluster centroids and a histogram is created for all training images, such that every training image is described by a histogram with k bins. Three different kinds of normalization can be applied to the histograms: none, L1 and L2 normalization. Finally, multiple similarity measures are used to compute the similarity of a query image q with feature vector f^q to a training image m with feature vector f^m. The following similarity measures are used and compared: 1) Manhattan distance (L1); 2) Euclidean distance (L2); 3) cosine similarity; 4) inner product and 5) intersection i.e. the sum of pointwise minimum of the feature vector.

These different measures are then combined with the various normalization schemes to study whether the optimal configuration for the SIFT descriptors is also the optimal configuration for the pattern spectra descriptors.

Choice of k for codebooks Finally, multiple values of k were chosen for the different datasets. The UC Merced and ZuBuD datasets contain many more keypoints per image, while the COIL-100 dataset has a limited number of keypoints. For the COIL-100 dataset the following values of k are used: 50, 100, 500, 1000, 2000, 4000, 8000. The same values are used for the ZuBuD and UC Merced dataset, except that the following higher values are incorporated as well: 10000, 15000, 20000.

6.4 Evaluation metric

All datasets were evaluated with the same metric, the average normalized modified retrieval rank (ANMRR) [7]. This metric has been widely used as a retrieval performance measure in the MPEG-7 standard and not only measures whether a retrieval system predicts a class correctly, it also incorporates the ranking of the correct results. This metric has already been used extensively to evaluate SIFT on the UC Merced dataset [11, 12], as well as evaluating global and local pattern spectra [3].

Rank First of all, consider a query image q, the rank of the k-th ground truth image in the retrieval process, determined by the distance metric, as $Rank(k)$ and the number of ground truth images of q as $NG(q)$. The modified rank $Rank^*(k)$ for this image is then defined as follows:

$$Rank^*(k) = \begin{cases} Rank(k) & \text{if } Rank(k) \leq K(q) \\ 1.25K(q) & \text{if } Rank(k) > K(q) \end{cases}$$

where $K(q) > NG(q)$ is the number of ranks that are considered relevant for retrieval. This number $K(q)$ is typically chosen as $2NG(q)$. The modified rank essentially bounds $Rank(k)$ to a maximum of $1.25K(q)$ while implementing a penalty for incorrectly classified items. Using this modified rank we can define an average rank (AVR) for this query image q:

$$AVR(q) = \frac{1}{NG(q)} \sum_{k=1}^{NG(q)} Rank^*(q)$$

Finally, the ANMRR is defined as:

$$\text{ANMRR} = \frac{1}{NQ} \sum_{q=1}^{NQ} \frac{AVR(q) - 0.5(1 + NG(q))}{1.25K - 0.5(1 + NG(q))}$$

where NQ is the number of query images. The value of ANMRR ranges from 0 to 1, where lower values indicate better retrieval performance.

6.5 Training and validation

The model is trained by taking a random 80/20 split of the data as training/validation set. 10 different splits are created for every run configuration and the results are averaged to produce the final performance metric of a test.

6.6 Implementation

MorphSIFT was implemented in C/C++ by using the provided SIFT implementation of the library OpenCV 3.4.1, which extracts the keypoints, their locations in the scale space and SIFT descriptors. This scale space is then used to extract the patches as described in Section 4.1. The pattern spectra are computed using a C library.

Finally, a Python program performs the image retrieval process and computes the final ANMRR values.

7 Results and discussion

The results obtained by the experiments in the previous section are described in this section. First of all, the results on the various pattern spectra configurations are described, after which the averaged descriptors are shown and discussed. Finally the results on the effect of codebook size and similarity measures are shown on the different datasets.

7.1 Pattern spectra configurations

Table 1 shows the results of various combinations of pattern spectra configurations. First of all, it is clear that extracting a larger patch results in a higher accuracy. Additionally, choosing a narrower range for the CNC attribute results in a better accuracy as well. The choice for using a mask or not is not as straightforward though, as using a mask for patch sizes of 12 and 16 results in a lower accuracy, while using such a mask for a patch size of 24 results in slightly higher accuracy.

It is important to note that SIFT uses a 16×16 area around the keypoint to compute the descriptor, so extracting larger patches would not yield a fair comparison. As such, a patch size of 16×16 is used for the remaining experiments.

n_{patch}	**Mask**	**CNC range**	**Accuracy**
12	None	$[1.0, 2.0]$	0.8465
		$[1.0, 3.0]$	0.8235
		$[1.0, 4.0]$	0.7830
	Circle	$[1.0, 2.0]$	0.8000
		$[1.0, 3.0]$	0.7695
		$[1.0, 4.0]$	0.7470
16	None	$[1.0, 2.0]$	0.8925
		$[1.0, 3.0]$	0.8640
		$[1.0, 4.0]$	0.8665
	Circle	$[1.0, 2.0]$	0.8890
		$[1.0, 3.0]$	0.8635
		$[1.0, 4.0]$	0.8510
24	None	$[1.0, 2.0]$	0.9110
		$[1.0, 3.0]$	0.9070
		$[1.0, 4.0]$	0.8850
	Circle	$[1.0, 2.0]$	0.9250
		$[1.0, 3.0]$	0.9140
		$[1.0, 4.0]$	0.8925

Table 1: Results of various pattern spectra configurations on the COIL-100 dataset.

7.2 Descriptors

The descriptors of both methods are tested and evaluated to gain more understanding of the difference in performance of both methods.

Figure 7 shows the averaged descriptors for SIFT and MorphSIFT, split on the three different datasets used. Both methods display a very different distribution of peaks and valleys, with the SIFT method being more spread out, while the MorphSIFT descriptors appear to contain periodic repetitions. This periodicity is a direct consequence of the way in which the descriptors are formed. The descriptors can be separated into two segments: the descriptors for dark areas and for bright areas. Both segments can each be further separated into 6 different segments. These smaller segments represent the 6 different bins to which the CNC is mapped and also show a similar pattern for the different bins of CNC. This disparity in spread among the histograms could indicate that the descriptors for MorphSIFT are much less diverse and might be unable to encode more information than the default SIFT descriptors.

7.3 Overall results

Table 2 shows the overall best performance of each method on the various datasets. It is clear that SIFT does much better than MorphSIFT overall, with wide ranges of ANMRR on the various datasets. Note that MorphSIFT got the best ANMRR for a very low number of codebooks, whereas SIFT performed better when using a higher number of codebooks. This is especially apparent when comparing the results for the ZuBuD dataset, which contains many descriptors, where the difference in best ANMRR is very large.

7.4 Codebook size and similarity measure

Figures 8, 9 and 10 show the effect of various combinations of distance measure, codebook size and normalization. The first thing to note on all these results is that all results for the intersection, inner and cosine distance measures stay constant for all normalization measures. Secondly, the results for the inner and intersection distance measures usually overlap.

All three datasets have a slightly different response of the ANMRR curve with the various parameters. There are, however, many similarities in the responses for common parameters. Similar to what Yang et al. [12] described, most of the results fall into two categories: one where the ANMRR decreases with the number of codebooks and one where the ANMRR increases with the number of codebooks. When comparing the ANMRR of SIFT and MorphSIFT for similar settings on a dataset we see that SIFT performs much better overall. More interestingly there are many cases where the curve for SIFT decreases with number of codebooks, while the curve for MorphSIFT shows the opposite behavior. This

Dataset	Method	Norm	Dist	Codebooks	ANMRR
COIL	SIFT	L1	Inner	8000	0.3254843
	MorphSIFT	L1	Inner	8000	0.5678468
UC Merced	SIFT	L1	L1	15000	0.6002196
	MorphSIFT	L2	L1	100	0.6565865
ZuBuD	SIFT	L1	L1	20000	0.1224121
	MorphSIFT	None	L2	500	0.4093080

Table 2: Overall results of SIFT and MorphSIFT.

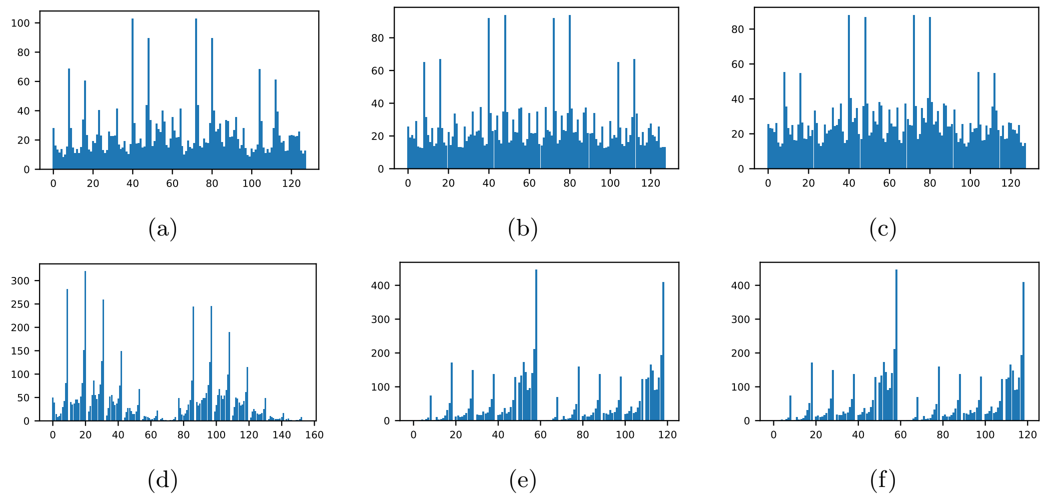


Figure 7: Averaged feature vector for SIFT (a,b,c) and MorphSIFT (d,e,f). Results for the COIL-100 (a,d), ZuBuD (b,e) and UC Merced (c,f) datasets are shown separately.

indicates that the descriptors for MorphSIFT do not scale as well for higher number of k, nor do they possess the same descriptive power as the SIFT descriptors.

COIL-100 dataset: Figure 8 shows the results in ANMRR for the different configurations for both SIFT and MorphSIFT. The overall results show that MorphSIFT performs significantly worse than SIFT across the board. This disparity in performance is already apparent at low number of codebooks, where MorphSIFT scores roughly 0.1 ANMRR higher than SIFT. At higher number of codebooks this disparity only continues to grow for the majority of the measures, with the sole exception of the intersection distance measure. While one might assume that the intersection distance metric together with a high number of codebooks is the correct way to approach this dataset it most likely is a result of overfitting. With just 44813 features in total and 8000 codebooks only 5.6 features are used for one codebook on average.

UC Merced dataset: The results for UC Merced show much worse overall retrieval performance compared to the COIL-100 dataset, regardless of method used. The general trend for both SIFT and MorphSIFT is very similar, except that, just like the COIL-100 dataset, the retrieval performance for MorphSIFT worsens with higher number of codebooks while SIFT performs better with more codebooks. For low number of codebooks, however, this disparity is still very small but it increases quickly as k grows.

Interestingly, the results for this dataset do not fully correspond to the results found by Yang et al. [12] on the same dataset. While most of the image retrieval process is the same, there is a major difference in the way the descriptors are trained. Yang et al. used randomly sampled keypoints from the U.S. Geological Survey (USGS) National Map to generate the k codebooks, whereas we generated the codebooks by randomly sampling keypoints from the training data. This results in an overlap of the training and validation images and could account for the difference in ANMRR.

There is, however, another big difference, as the curves for the various similarity measures do not have the same shape, with the curves for the inner and intersection measures being very different in particular. The ANMRR stays close to 1 in our results, whereas the ANMRR is much closer to 0.7 in the results from Yang et al. Multiple attempts were made to make these results match up, but the results differed consistently. It is currently unknown as to why these results differ this much.

Finally, note that while both the results for COIL-100 and ZuBuD show a difference of more than 0.2 for the best ANMRR this effect is not as big for this dataset. One possible cause could be that this dataset is inherently harder to classify.

ZuBuD dataset: There is a huge overall difference between the two methods on this dataset across all normalization options, as can be seen in Figure 10. The results for MorphSIFT show an interesting pattern where the ANMRR for some parameters peak around 4000 after which it decreases again. This pattern repeats itself for multiple combinations of normalization and distance metrics.

Effect of descriptor variety on ANMRR One explanation for the results of ANMRR of MorphSIFT can be found in the averaged descriptors as described in Section 7.2. The averaged descriptors as shown in Figure 7 are vastly dissimilar, with the descriptors for MorphSIFT being much less varied than its counterpart. Using a descriptor that is less expressive due to the lack of variety can often result in overfitting, a pattern many of the ANMRR curves show. The accuracy is best when using a very low number of k, after which the optimal value has been reached and increasing k leads to too many wildly different descriptors that do not generalize well to the validation set.

8 Conclusion

This paper has explored the idea of substituting keypoint descriptors based on morphological filters for SIFT descriptors, while maintaining the SIFT keypoint location method. Various configurations for these pattern spectra were tested and compared to each other on three different datasets, after which the effect of varying parameters in the image retrieval process was studied.

Overall, this new method performed significantly worse than the existing SIFT descriptors. While other studies showed promising results of the pattern spectra they did not live up to this expectation in this application. The descriptors are less diverse, indicated by the low number of codebooks that yielded the best ANMRR on the UC Merced and ZuBuD datasets.

Many configurations for MorphSIFT were left out due to scope limitations. First of all, only two pattern spectra attributes were used whereas many other attributes can be used. The used attributes are scale and rotation-invariant, while the method should be able to incorporate attributes which are not rotation-invariant due to the extracted orientations by SIFT. Additionally, the averaged pattern spectra show a lower diversity than SIFT descriptors, such that PCA could potentially be used to reduce the descriptor size. The next point of interest is the difference in computation time. MorphSIFT appeared to be faster overall, but more rigorous research should be done to make conclusions. Finally, the image retrieval method could be improved by using the VLAD indexing scheme as described by Bosilj et al. [3].

References

[1] Suraya Abu Bakar, Muhammad Suzuri Hitam, and Wan Nural Jawahir Hj Wan Yussof. “Content-Based Image Retrieval using SIFT for binary and greyscale images”. In: *Signal and Image Processing Applications (ICSIPA), 2013 IEEE International Conference on.* IEEE. 2013, pp. 83–88.

[2] Petra Bosilj et al. “Local 2D pattern spectra as connected region descriptors”. In: *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing.* Springer. 2015, pp. 182–193.

[3] Petra Bosilj et al. “Retrieval of remote sensing images with pattern spectra descriptors”. In: *ISPRS International Journal of Geo-Information* 5.12 (2016), p. 228.

[4] Ming-Kuei Hu. “Visual pattern recognition by moment invariants”. In: *IRE transactions on information theory* 8.2 (1962), pp. 179–187.

[5] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[6] David G Lowe. “Object recognition from local scale-invariant features”. In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on.* Vol. 2. Ieee. 1999, pp. 1150–1157.

[7] Bangalore S Manjunath et al. “Color and texture descriptors”. In: *IEEE Transactions on circuits and systems for video technology* 11.6 (2001), pp. 703–715.

[8] S.A. Nene, S.K. Nayar, and H. Murase. *Columbia object image library (COIL-100).* Tech. rep. CUCS-006-96. Columbia University, Department of Computer Science, 1996. URL: `http://www.cs.columbia.edu/CAVE/%20research/softlib/coil-100.html`.

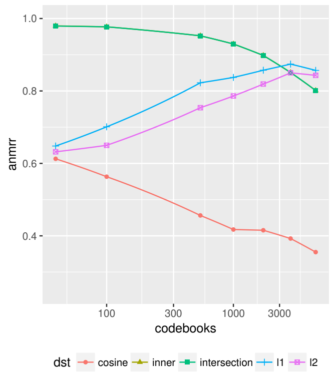
[9] H. Shao, Tomas Svoboda, and Luc Van Gool. “ZuBuD ± Zurich Building Database for Image Based Recognition”. In: (2003).

[10] Florence Tushabe and Michael HF Wilkinson. “Content-based image retrieval using combined 2D attribute pattern spectra”. In: *Workshop of the Cross-Language Evaluation Forum for European Languages.* Springer. 2007, pp. 554–561.

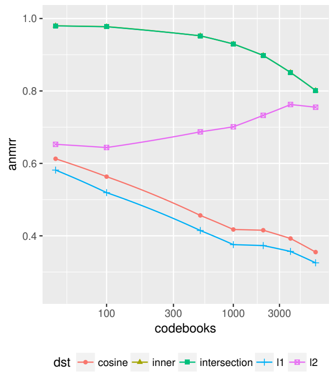
[11] Yi Yang and Shawn Newsam. “Bag-of-visual-words and spatial extensions for land-use classification”. In: *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems.* ACM. 2010, pp. 270–279.

[12] Yi Yang and Shawn Newsam. “Geographic image retrieval using local invariant features”. In: *IEEE Transactions on Geoscience and Remote Sensing* 51.2 (2013), pp. 818–832.

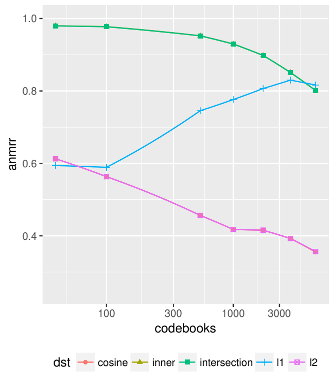
A Results of SIFT and MorphSIFT for varying codebooks



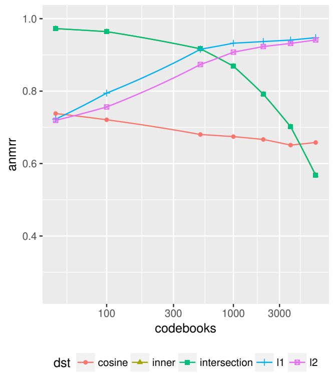
(a)



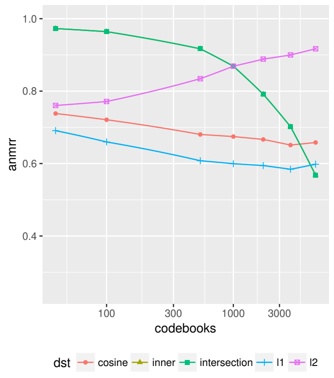
(b)



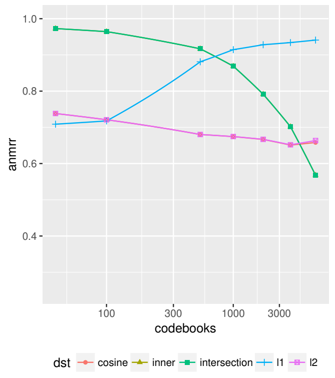
(c)



(d)

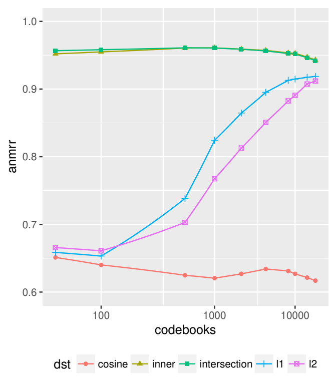


(e)

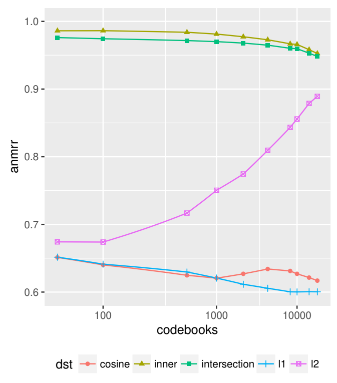


(f)

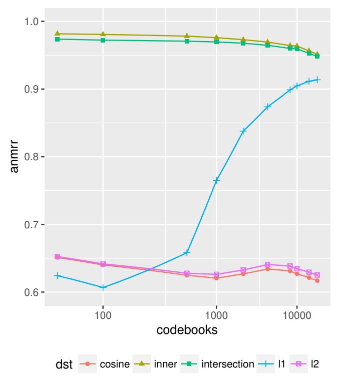
Figure 8: ANMRR on the COIL-100 dataset for SIFT (a, b, c) and MorphSIFT (d, e, f) under various distance and codebook sizes using (a, d) no normalization, (b, e) L1 normalization and (c, f) L2 normalization.



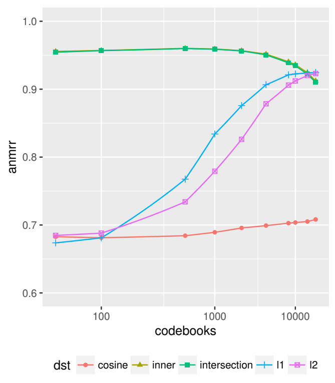
(a)



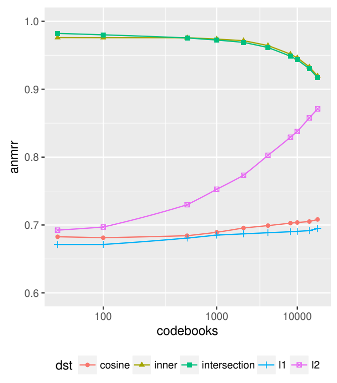
(b)



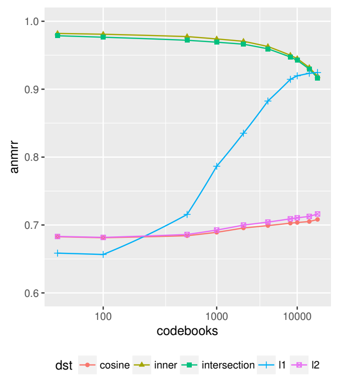
(c)



(d)

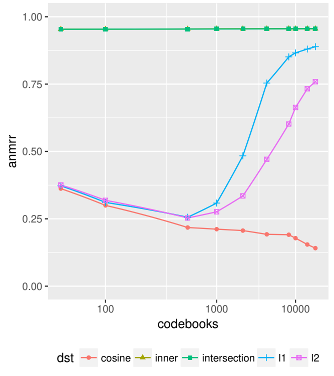


(e)

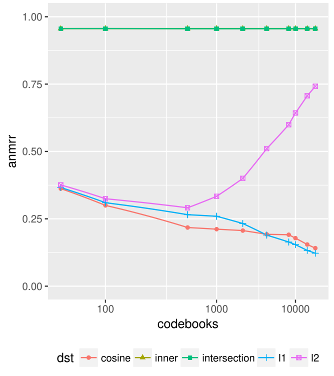


(f)

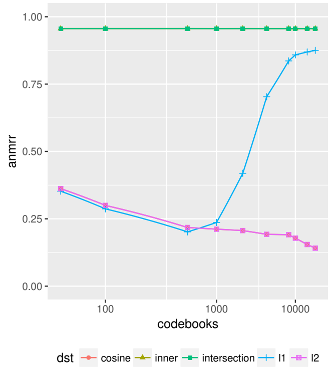
Figure 9: ANMRR on the UC Merced dataset for SIFT (a, b, c) and MorphSIFT (d, e, f) under various distance and codebook sizes using (a, d) no normalization, (b, e) L1 normalization and (c, f) L2 normalization.



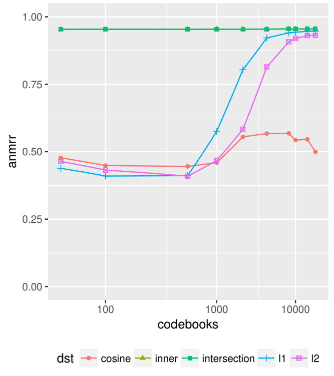
(a)



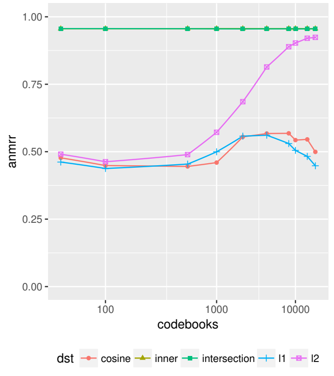
(b)



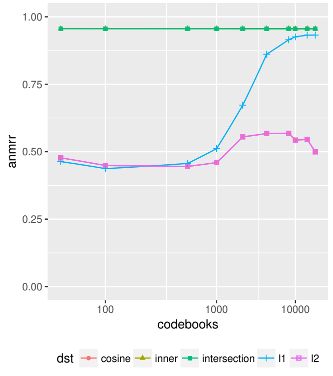
(c)



(d)



(e)



(f)

Figure 10: ANMRR on the ZuBuD dataset for SIFT (a, b, c) and MorphSIFT (d, e, f) under various distance and codebook sizes using (a, d) no normalization, (b, e) L1 normalization and (c, f) L2 normalization.