View point robust visual search technique

POLITECNICO DI TORINO

POLITECNICO DI TORINO

SCUOLA DI DOTTORATO

Dottorato in Telecomunicazioni – XXVIII ciclo

Tesi di Dottorato

View point robust visual search technique

Biao ZHAO

Tutore
prof. Enrico Magli

Aprile 2015

April 20, 2015

Summary

In this thesis, we have explored visual search techniques for images taken from different view points and have tried to enhance the matching capability under view point changes. We have proposed the Homography based back-projection as post-processing stage of Compact Descriptors for Visual Search (CDVS), the new MPEG standard; moreover, we have defined the affine adapted scale space based affine detection, which steers the Gaussian scale space to capture the features from affine transformed images; we have also developed the corresponding gradient based affine descriptor. Using these proposed techniques, the image retrieval robustness to affine transformations has been significantly improved.

The first chapter of this thesis introduces the background on visual search.

In the second chapter, we propose a homography based back-projection used as the post-processing stage of CDVS to improve the resilience to view point changes. The theory behind this proposal is that each perspective projection of the image of 2D object can be simulated as an affine transformation. Each pair of affine transformations are mathematically related by homography matrix. Given that matrix, the image can be back-projected to simulate the image of another view point. In this way, the real matched images can then be declared as matching because the perspective distortion has been reduced by the back-projection. An accurate homography estimation from the images of different view point requires at least 4 correspondences, which could be offered by the CDVS pipeline. In this way, the homography based back-projection can be used to scrutinize the images with not enough matched keypoints. If they contain some homography relations, the perspective distortion can then be reduced exploiting the few provided correspondences. In the experiment, this technique has been proved to be quite effective especially to the 2D object images.

The third chapter introduces the scale space, which is also the kernel to the feature detection for the scale invariant visual search techniques. Scale space, which is made by a series of Gaussian blurred images, represents the image structures at different level of details. The Gaussian smoothed images in the scale space result in feature detection being not invariant to affine transformations. That is the reason why scale invariant visual search techniques are sensitive to affine transformations. Thus, in this chapter, we propose an affine adapted scale space, which employs the affine steered Gaussian filters to smooth the images. This scale space is flexible to different affine transformations and it well represents the image structures from different view points. With the help of this structure, the features from different view points can be well captured.

In practice, the scale invariant visual search techniques have employed a pyramid structure to speed up the construction. By employing the affine Gaussian scale space principles, we also propose two structures to build the affine Gaussian scale space. The structure of affine Gaussian scale space is similar to the pyramid structure because of the similiar sampling and cascading

properties. Conversely, the affine Laplacian of Gaussian (LoG) structure is completely different. The Laplacian operator, under affine transformation, is hard to be affine deformed. Differently from a simple Laplacian operation on the scale space to build the general LoG construction, the affine LoG can only be obtained by affine LoG convolution and the cascade implementations on the affine scale space. Using our proposed structures, both the affine Gaussian scale space and affine LoG can be constructed.

We have also explored the affine scale space implementation in frequency domain. In the second chapter, we will also explore the spectrum of Gaussian image smoothing under the affine transformation, and propose two structures. General speaking, the implementation in frequency domain is more robust to affine transformations at the expense of a higher computational complexity.

It makes sense to adopt an affine descriptor for the affine invariant visual search. In the fourth chapter, we will propose an affine invariant feature descriptor based on affine gradient. Currently, the state of the art feature descriptors, including SIFT and Gradient location and orientation histogram (GLOH), are based on the histogram of image gradient around the detected features. If the image gradient is calculated as the difference of the adjacent pixels, it will not be affine invariant. Thus in that chapter, we first propose an affine gradient which will contribute the affine invariance to the descriptor. This affine gradient will be calculated directly by the derivative of the affine Gaussian blurred images. To simplify the processing, we will also create the corresponding affine Gaussian derivative filters for different detected scales to quickly generate the affine gradient. With this affine gradient, we can apply the same scheme of SIFT descriptor to generate the gradient histogram. By normalizing the histogram, the affine descriptor can then be formed. This affine descriptor is not only affine invariant but also rotation invariant, because the direction of the area to form the histogram is determined by the main direction of the gradient around the features. In practice, this affine descriptor is fully affine invariant and its performance for image matching is extremely good.

In the conclusions chapter, we draw some conclusions and describe some future work.

Acknowledgements

The research of visual search techniques robust to view point changes has been financially supported and technically aided by Telecom Italia. The candidate will especially thank Skjalg Lepsoy, Gianluca Francini and Balestri Massimo, who provided me very important information of visual search standard, the original code to begin with and very informative discussions. I will also very thanks to my surpervisor, prof. Enrico Magli, who made the whole research plan, arranged periodical discussion and provide very detailed aid. Without them, I would never had this chance to obtain the Phd degree.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The proliferation of digital cameras equipped in the smartphone and consumer-level cameras is producing an explosion of media data. Computer or computing device based intelligent technologies can be used to facilitate the acquisition, analysis, understanding and retrieving of multimedia documents, including digital images, videos, and text in an integrated way through programming. Typical sub-topics in this area include content analysis, management, rendering and retrieval. Our research focus is on the content-based visual search techniques.

A visual search system is generally based on a computer system for browsing, searching and retrieving visual data, including images and videos, from a large database. Most traditional and common methods of visual search utilize some method of adding metadata such as captioning, keywords, or descriptions to the media so that retrieval can be performed over the annotation words. Manual annotation is time-consuming, laborious and expensive; to relieve the conflict between low efficient media annotation and rapid increasing of media data, a large effort has been made to automatically annotate the visual data by analyzing its content, leading to the content-based visual search techniques or content-based retrieval [1]. Since the video is a collection of images following the elapsing of time, a video retrieval system is also based on techniques of image retrieval. A good image retrieval system have great facilitate for the development of a video retrieval system.

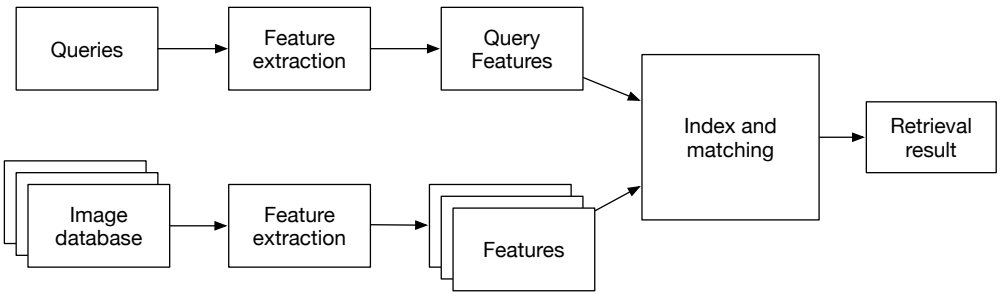


Figure 1.1. A typical content-based image retrieval system.

Content-based image retrieval (CBIR), also known as query by image content (QBIC) is the issue of media retrieval on digital images, that is, the problem of searching for digital images in large databases. CBIR is opposed to traditional concept-based approaches, which annotate the image by

manually manipulated tag of words [2].

Content-based means that retrieval of the image is based on the contents of the image rather than the metadata such as keywords, or tags associated with the image. Content of the image can refer to colors, shapes, textures, edges, or any other inherent character of the image. CBIR is more desirable than the metadata based retrieval system whose accuracy depends on the annotation quality and completeness. A manual annotation of images by tagging keywords or metadata in a large database can be time consuming and more apt to confuse with the desired keywords. In CBIR, the images in the database will be automatically tagged according to the content of the image, such as the colors, the shapes, the textures or some other associated informations contended in the image. Such information will be extracted as features stored in the database, and usually organized by multi-dimensional feature vectors. When the multimodal queries are input to the retrieval system, the same type of features must be extracted as the image features in database. Finally, the similarity or distance between queries features and features in the database will be compared with each other, and the most likely one will be selected as the retrieved image [3].

Figure 1.1 presents a typical pipeline of CBIR system. It works by comparing the similarity between the features of query image and the features of database stored image. Both these features of the images are automatically tagged by extracting the features according the content of the images, including the colors, the shapes or some other specific image qualities. The features of the database stored images are also indexed and organized to speed up the retrieval.

Theoretically, any inherent characteristic of the image content, including colors, shapes, textures and edges can be used as feature for image retrieval. But a valid and robust image retrieval system should be based on the features that are constant and robust enough to the fluctuation of the scene. Ideally, the retrieval result should not be affected by the illumination, which may cause the change of colors and textures, the focus of scene, which may render the shapes not complete, and also the scale and the view-point. For the object recognition, the background of the images of the same object may also be different, making the texture and the statistics of the complete image not reliable for an image retrieval system. Thus, local feature characterizing the local image content are a better option to distinguish the matched part and non-matched part of a pair of images and to determine whether two images really match or not.

Image feature can be any repeatable and detectable quantity characterized by certain types of local information, like colors, textures or edges to identify the location and neighboring pixels around. Considering the complexity of the detection and description processes, image features are generally defined around corners and associated neighboring pixels, including edge, corner and blob.

Edges are points where the image brightness changes sharply or, more formally, has discontinuities. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed edges. In general, an edge can be of almost arbitrary shape, and may include junctions. In practice, edges are usually defined as sets of points in the image which have a strong gradient magnitude [4].

A corner can be defined as the intersection of two edges or a point for which there are two dominant and different edge directions in a local points. A corner based feature should have a well-defined position and can be robustly detected. Equivalently, the determination for a corner to be a feature relies on its ability to be detected in multiple similar images, under conditions of different lighting, translation, rotation and other transforms [5].

A blob is a region of a digital image which differs in properties, such as brightness or color, compared to the surrounding areas; all the points in a blob can be considered to be similar to each other. Given some property of interest expressed as a function of position on the digital image, there are two main classes of blob detectors: (i) differential methods, which are based on derivatives of the function with respect to position, and (ii) methods based on local extrema, which are based on finding the local maxima and minima of the function. With the more recent terminology used in the field, these detectors can also be referred to as interest point operators, or alternatively interest region operators [6].

The blob can be used to signal the presence of informative image features based on local image statistics. One main reason to base the interest points on blob is to provide complementary information, which can not be obtained from edges or corners. These features could signal the presence of objects or parts of objects in the image domain.

There are several different feature detectors respectively based on these image characteristics, such as edge-based region (EBR) [7] and scale-invariant shape features (SISF) [8] based on edge detection; Harris [9], smallest univalue segment assimilating nucleus (SUSAN) [10] and features from accelerated segment test (FAST) [10] based on corner detection; and Laplacian of Gaussian (LoG) [11], Difference of Gaussian (DoG) [12], Determinant of Hessian (DoH) [13], and Maximally stable extremal region (MSER) [14] based on blob detection.

As has been discussed above, one advantage of the blob based detector is that any region with properties, such as brightness, focus or colors, that differs from the surrounding area, can be detected as a blob. Differently from the edges and corners, this simple requirement of the blob can easily be satisfied, implying that, more blobs can potentially be extracted than edges and corners. What is more, the internal part of the blob can be deemed as a homogeneous region. Initially, the blob is accounted as a region, which has the advantage to signal the object region and presence robustly to different scales. The state of the art feature detectors are all based on the blob detections. Stable, robust, applicable, and also amenable to multiple scale detection, all these properties render the blob based detection the most widespread and accepted detector for image retrieval.

The most commonly applied blob detection to extract features is LoG, which detects the local extrema on Laplacian of Gaussian blurred images. Usually, LoG results in strong positive responses for dark blobs and strong negative responses for bright blobs. Thus, given a discrete two-dimensional input image, a three-dimensional discrete scale-space volume can be computed and a point is regarded as a bright (dark) blob if it is the local maximum or minimum around its neighborhood. A specialty of this operator at a single scale is that the operator response is strongly related with the size of the blob structures in the image domain and the size of the Gaussian kernel used for pre-smoothing [15]. In order to automatically capture blobs of different size in the image domain, where the size is not known in advance, a multi-scale approach is therefore necessary, which inspires the creation of scale space.

A collection of these pre-smoothed images by Gaussian kernels of different size form a scale space. In general, scale space theory is a framework for multi-scale signal representation for handling image structures at different scales, by employing a one-parameter family of smoothed images. This framework provides a scale-invariant representation, which is necessary for dealing with the size variations that may occur in image data, because real-world objects may be of different sizes and the distance between the object and the camera may vary depending on the circumstances. The motivation for generating a scale-space representation of a given data set originates from the basic

observation that real-world objects are composed of different structures at different scales. This implies that real-world objects, in contrast to idealized mathematical entities, may appear in different ways depending on the scale of observation [6].

A highly useful property of scale space is that the blob detection can be made invariant to scales, by performing automatic scale selection based on normalized derivatives. A simple approach to this scale invariant blob detection is based on the Laplacian of the scale space, which can be created by LoG operator. Alternatively, scale space provide the structure for the image multiple scale representation, and Laplacian is used for automatic blob detection. This automatic scale invariant blob detection can also be established based on DoG and DoH. A well designed image retrieval algorithm based on DoG has been proposed by Lowe, named Scale Invariant Feature Transform (SIFT). Our research is also based on this algorithm.

1.1 Scale Invariant Feature Transform

Scale Invariant Feature Transform (SIFT) is a content-based feature extracting and matching algorithm for image matching and recognition developed by Lowe [12]. This algorithm has been used for a large number of applications in computer vision. The SIFT algorithm is invariant to translations, rotations, scaling transformations, illumination and partially robust to perspective transformations. Additionally, SIFT can identify objects even among clutter and under partial occlusion, because the image matching identification is based on local feature matching and not the whole image. In the real word, it has been proven to be very efficient and accurate in practice for image retrieval and object recognition. Generally, SIFT algorithm can be divided into two parts, including the SIFT feature extraction and the SIFT feature descriptor, which is used to identify the similarity of the features from different but related images [16].

The SIFT features are local interest points detected using the theory of scale space, and are invariant to image scale and rotation. They are also robust to changes in illumination, noise, and minor changes in viewpoint. In addition to these properties, they are highly distinctive, relatively easy to extract and allow for correct object identification with low probability of mismatch. They are relatively easy to match against a large database of local features but however the high complexity can be an issue in some applications.

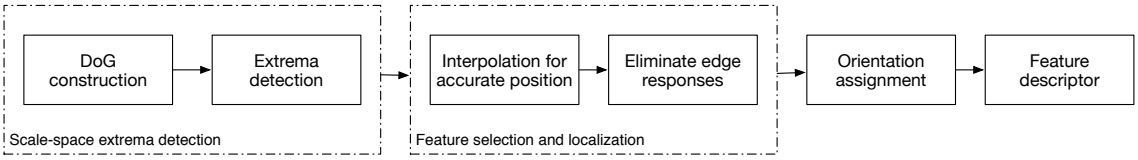


Figure 1.2. The pipeline of SIFT algorithm.

The SIFT descriptors comprise a method for detecting interest points from a grey-level image where statistics of local gradient of image intensities are accumulated to give a summarized description of the local image structures in a local neighborhood around each interest point, with the intention that this descriptor should be used for matching corresponding interest points between

different images.

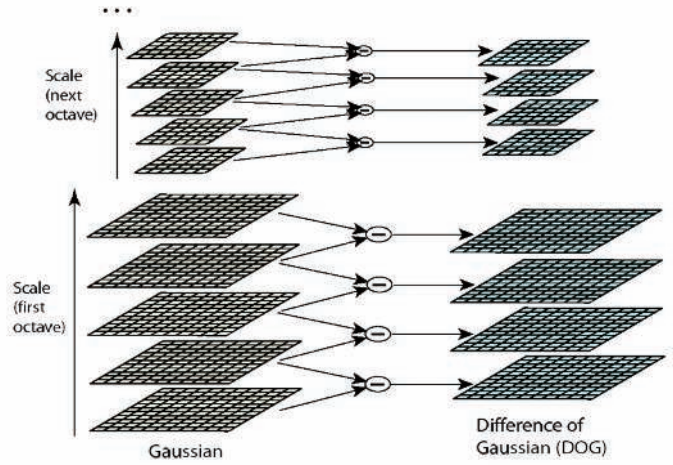


Figure 1.3. The structure of DoG pyramid.

Generally, SIFT algorithm can be divided into the following steps:

1. Scale-space extrema detection.
2. Features accurate localization.
3. Orientation assignment based on image gradient.
4. Descriptor based on gradient histogram.

Each steps also contain few small procedures. The whole pipeline of SIFT is demonstrate in the Figure 1.2.

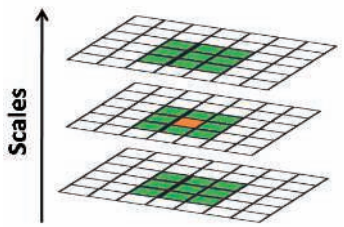


Figure 1.4. The extrema selection from DoG.

The first step of SIFT is to construct the DoG, where the feature are located and extracted. SIFT is also based on the blob detection in the framework of scale space. Differently from the LoG as the derivative form of the scale space, features in SIFT are detected by DoG, the approximation to the scale space derivative. The definition of DoG is given by

$$\begin{aligned} D(x,y,\sigma) =& (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \\ =& L(x,y,k\sigma) - L(x,y,\sigma), \end{aligned} \tag{1.1}$$

where $G(x, y, \sigma)$ is the Gaussian filter with the kernel σ, and $L(x, y, \sigma)$ is the corresponding Gaussian blurred image.

Since

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2\Delta^2 G \tag{1.2}$$

Then the difference of Gaussian can be used to approximate the Laplacian of Gaussian. In practice the construction of the DoG is accomplished by a Gaussian pyramid. A Gaussian pyramid is constructed from the input image by repeated smoothing and subsampling, and the DoG pyramid is computed from the differences between the adjacent levels in the Gaussian pyramid. Adjacent Gaussian blurred images are created by repeatedly convolving with Gaussian filters. These images form an octave. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process is repeated. Then, interest points are obtained from the points at which the DoG values assume extrema with respect to both the spatial coordinates in the image domain and the scale level in the pyramid [17]. The construction of the DoG pyramid and the detection of the extrema are demonstrated in the Figures 1.3 and 1.4.

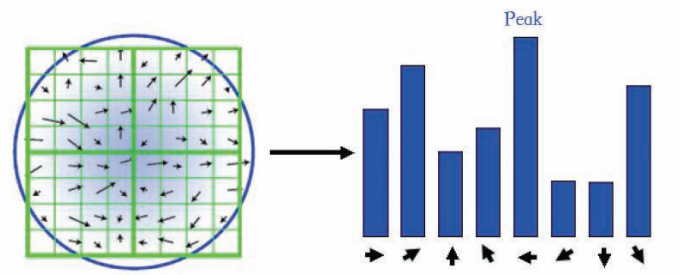


Figure 1.5. Orientation assignment in SIFT.

The next step in the algorithm is to perform a detailed fit to the nearby data for accurate location, scale, and ratio of principal curvatures. For each candidate feature, interpolation of nearby data is used to accurately determine its position. The initial approach was to just locate each feature at the location and scale of the candidate feature. A more efficient approach calculates the interpolated location of the extremum, which substantially improves matching and stability by the quadratic Taylor expansion of the DoG scale space function [12]. Features with low contrast and strong responses to edges will also be discarded.

In the next step, each feature will be assigned with one or more orientations based on local image gradient. This is the key step in achieving invariance to rotation gradient magnitude and orientation are defined as.

$$\begin{aligned} m(x, y) &= \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}, \\ \theta &= \arctan\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right). \end{aligned} \tag{1.3}$$

The magnitude and direction calculations for the gradient are done for every pixel in a neighboring region around the feature in the Gaussian-blurred image. To find the dominant orientation, peaks are detected in this orientation histogram divided into 36 bins, with each bin covering 10

degrees. Each sample in the neighboring window added to a histogram bin is weighted by its gradient magnitude and by a Gaussian-weighted circular window. The peak in this histogram is the dominant orientation [12].

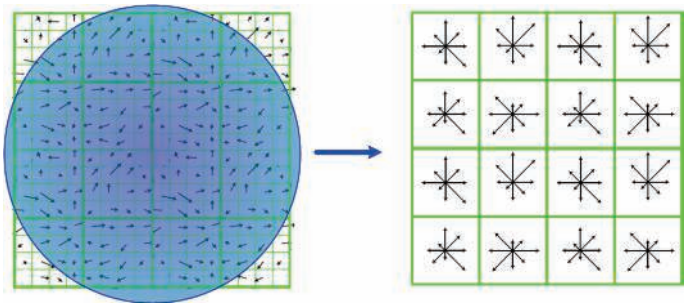


Figure 1.6. Calculating of SIFT descriptor.

Given these scale and orientation, a rectangular grid is laid out in the image domain, centered at the detected feature, with its orientation determined by the main peak(s) in the histogram and with the spacing proportional to the detection scale of the interest point. This grid will be further divided into 4×4 grid to form another gradient histogram. To give stronger weights to gradient orientations near the features, the entries in the histogram are also weighted by a Gaussian window function centered at the interest point and with its size proportional to the detection scale. Taken together, the local histograms computed at all the 4×4 grid points and with 8 quantized directions lead to an image descriptor with $4 \times 4 \times 8 = 128$ dimensions for each detected feature. This resulting image descriptor is referred to as the SIFT descriptor.

1.2 Compact Descriptors for Visual Search

Compact Descriptors for Visual Search (CDVS) is the standard, proposed by MPEG, in order to enable an interoperable, efficient and cross-platform solution for Internet-scale visual search applications and services. The forthcoming CDVS standard is particularly important because it will ensure interoperability of visual search applications and databases, enabling high level of performance of implementations conforming to the standard, simplifying design of descriptor extraction and matching for visual search applications. It will also enable low complexity, low memory hardware support for descriptor extraction and matching in mobile devices and significantly reduce load on wireless networks carrying visual search-related information. All this will stimulate the creation of an ecosystem benefiting consumers, manufacturers, content and service providers alike [18].

Although SIFT is a very successful algorithm for image matching and retrieval, it is far from an Internet based real application, which is confronted with the task of handling of millions of images or videos within a very short time. This large scale application of visual search requires the standard to provide fast retrieval. From this point of view, SIFT is not good enough for the widespread Internet application.

In particular, two procedures for descriptor comparison are implemented in CDVS, defining two fundamental tasks for real visual search systems: pairwise matching and retrieval. The former regards automated verification of whether two images depict the same objects or scene; in this case,

descriptors extracted from a query image are matched against the descriptors of a reference image, in order to determine whether they match or not. The latter regards the search and discovery of images contained within a large collection that depict the same objects or scenes as those depicted by a query image; this requires the database images to be processed for the creation of a database which may be searched using the descriptors extracted from the query [19]. The architecture of the pairwise matching is similar to the pipeline of a typical image retrieval system. Figure 1.7 depicts the architecture of a retrieval procedure.

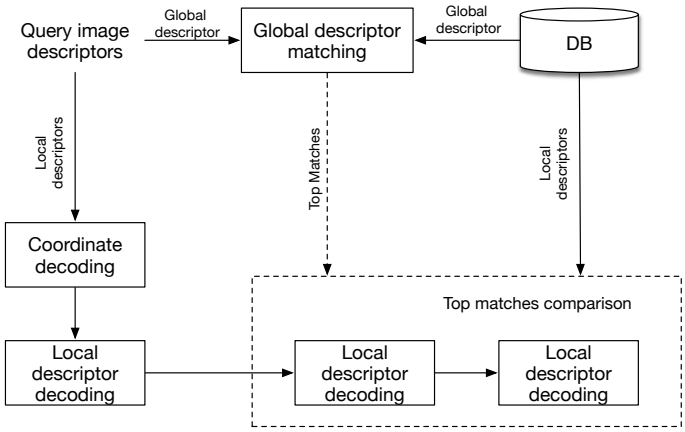


Figure 1.7. CDVS retrieval.

Generally speaking, the CDVS standard applies a lot of up-to-date techniques to improve the efficiency and accuracy of visual search, including [19]:

1. **Interest point detection:** Identification of interest points based on the LoG scale space and the subsequent identification of extrema by means of polynomial approximations. This feature detection is called A Low-degree Polynomial (ALP). It replies on LoG and polynomials to approximate the continue scale space function.

2. **Feature selection:** Selection of a limited number of interest points on the basis of their characteristics, in order to maximize a measure of expected quality for subsequent matching.

3. **Local descriptor computation:** It is the same as SIFT descriptor computation.

4. **Local descriptor compression:** Transform and scalar quantisation-based compression of the selected local descriptors.

5. **Coordinate coding:** Compression of the coordinates of the selected interest points.

6. **Global descriptor aggregation:** Aggregation of local descriptors, to form a single global descriptor employed for a first gross but quick retrieval from millions of images in the dataset and to be further refined from these selected candidates for the final image matching.

Our research is based on the framework of ALP. Similar to SIFT, ALP is also a scale space based feature detector, which approximates the continue function of LoG scale space by means of polynomials from few known LoG filtered images, which are used to find extrema in the scale space and to refine the spatial position of the detected points [20].

The construction of the ALP begins with four Gaussian blurred images with their parameter σ forthcoming an exponentially increasing sequence. The image shall be processed in a scale space representation obtained by Gaussian blur with different scale factors σ. These few Gaussian blurred image form the first octave.

For each pixel (x, y) in the image, a polynomial approximation to the scale-space function shall be formed as

$$p(x, y, \sigma) = a_3(x, y)\sigma^3 + a_2(x, y)\sigma^2 + a_1(x, y)\sigma + a_0(x, y) \tag{1.4}$$

The coefficients $a_3 \sim a_0$ shall be obtained by computing weighted sums of the images in the octave. By some complex coefficient transformation, the extrema can be obtained by the same theory as SIFT.

After the key-point detection, the points will be further computed by the same gradient based technique as the SIFT descriptor, to form the local descriptor. With the combination of local descriptor, it will further form the global descriptor for fast retrieval of the target image from the dataset with millions of candidates. What is more, the local descriptors will also be compressed and transformed to lighten the computation and bandwidth memory consumption. The standard also includes some other relevant procedures to speed up the whole process and to provide accurate retrieval results.

1.3 The motivation of this research

Robustness to different view-point is also a important criterion to evaluate the performance of a visual search technique. Compared to some other visual search techniques, SIFT is partially robust to affine transformation, but not enough for matching images taken from very different view-points.

The performance evaluation by Mikolajczyk and Schmid [21] also presents a performance comparison of different visual search techniques in terms of their robustness to view point changes. Figure 1.8 shows that 50 degrees view-point change can be considered as a small view-point change compared to the typical perspective difference. What is more, even the best performance of SIFT has just a little above 50% correct matching ratio, which is rather low referring to typical image matching precision. The correct matching precision refers to the ratio of how many matching images correctly matched. Even if SIFT has the best performance under view point change, 50% is far below the application requirement.

The evaluation above reveals a very important issue, which has not been properly addressed, namely the robustness of visual search algorithm to view-point change. In practice, all the current content based visual search techniques are unable to provide invariance to view point changes. In

reality, a scene may be acquired from several different view points, resulting in a large number of images of the same content but quite different perspectives. Without invariance or robustness to affine transformations, the application on the content-based visual search will be seriously limited. In advance, some real application, such as 3D object reconstruction, object recognition and architecture retrieval rely on accurate visual search of multiplexed planar object. Without a valid and accurate fully affine invariant image retrieval system, such application can not be carried out.

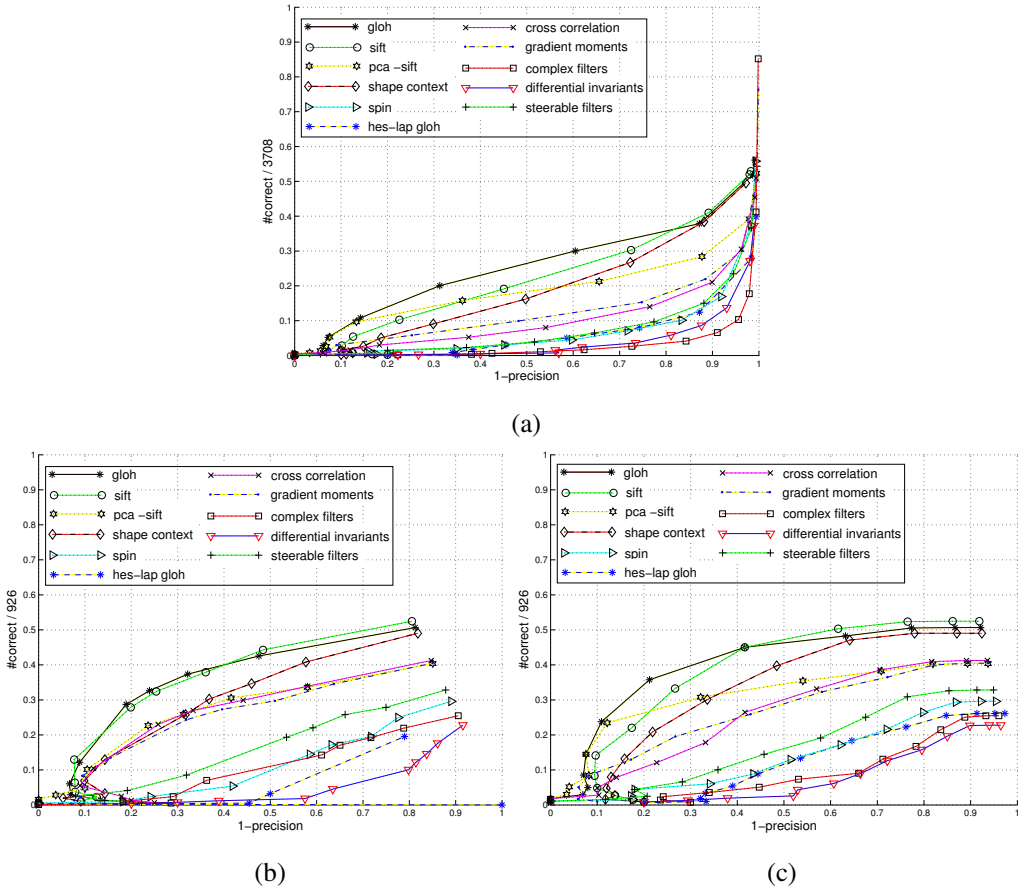


Figure 1.8. Some visual search techniques performance on affine robustness with the viewpoint change around 50 degrees. (a) Threshold based matching. (b) Nearest neighbor matching. (c) Nearest neighbor distance ratio matching [21].

Essentially, SIFT and most recent image retrieval systems are not designed to provide a full affine invariance, and in practice they are quite sensitive to affine transformations. This sensitivity to the affine transformation is not just due to the detector but also to the descriptor, since these two procedures are mainly based on the framework of scale space, which is constructed using isotropic Gaussian blurred images. To the isotropic filters, the image is considered to be transformed without directional preference, which is not the case for the image affine transformation. Directional preference is a natural characteristic for the image affine transformation; the direction is indicated by its eigenvectors and its level is indicated by its eigenvalues. However, the Gaussian filter cannot be steered to detect this directional preference. This makes the SIFT and some other scale space

based feature detection and description techniques unable to fully adapt to affine transformations unless they adopt an affine transformed scale space.

In addition to SIFT, there are also some other image retrieval algorithms, specially designed for affine invariance or comparably robust to affine transformations, such as Maximally Stable Extremal Regions (MSER) [14], Harris-Affine [22], Hessian-Affine and also Affine-SIFT (ASIFT) [23].

Figure 1.9. Feature matching comparision [23]. In the figure, SIFT (shown), Harris-Affine, Hessian-Affine and MSER(shown) find 2, 3, 1 and 42 correct matches.

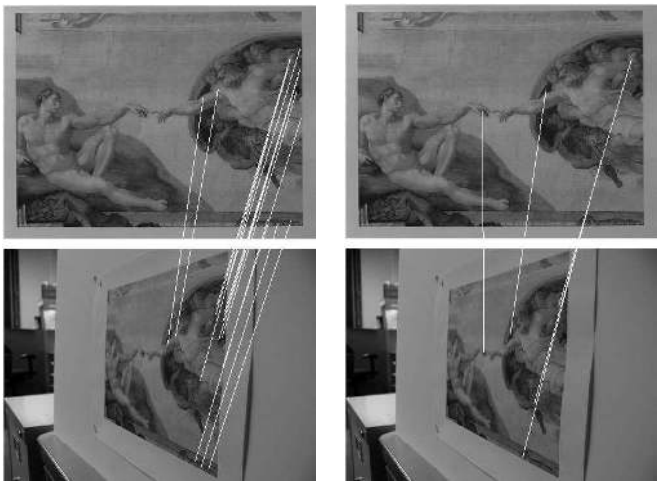


Figure 1.10. Another feature matching comparision [23]. Correspondences between the painting images taken from short distance with 75°. SIFT (shown), Harris-Affine, Hessian-Affine, and MSER (shown) find respectively 15, 3, 1, and 5 correct matches.

Harris-Affine and Hessian-Affine first detect key points in the scale-space using the approach

proposed by Lindeberg [15]. Then affine normalization is realized by an iterative procedure that estimates the parameters of elliptical regions and normalizes them to circular ones: at each iteration the parameters of the elliptical regions are estimated by minimizing the difference between the eigenvalues of the second order moment matrix of the selected region; the elliptical region is normalized to a circular one. The advance of Harris-Affine and Hessian-Affine techniques is that they are also based on the framework of scale space, which allows them to be scale invariant. But the feature detection is still performed on the isotropic scale space, and the descriptor has not be fully optimized for the affine transformation. Both these shortages make the result not so encouraging. But the theory behind inspires us that the feature detection and description based on a modified scale space is possible.

MSER tries to be affine invariant by an affine normalization of the most robust image level sets and level lines. It normalizes all of the parameters in the affine transform. It iteratively compares the stability of each region. Once maximum stable regions are obtained, an affine normalization is performed before they can be compared. Affine normalization up to a rotation is achieved by diagonalizing each MSER's second order moment matrix. MSER is not based on the framework of scale space, thus it is not fully scale invariant [14].

ASIFT is an extension of SIFT to enhance its affine invariance. It simulates three parameters: the scale, the camera longitude angle and the latitude angle (which is equivalent to the tilt) and normalizes the other three (translation and rotation) [23]. In other words, ASIFT exhaustively matches the image with all the possible simulated affine transformed versions. It inherits all the advantages of SIFT but also compensates its affine transform sensitivity by simulating all the possible image perspectives. It has a very good result for the affine invariance at the expense of a high computational complexity to match with the extra simulated images. In addition, the algorithm itself has not solved the affine invariance property in the framework of scale space, but by some extra simulations.

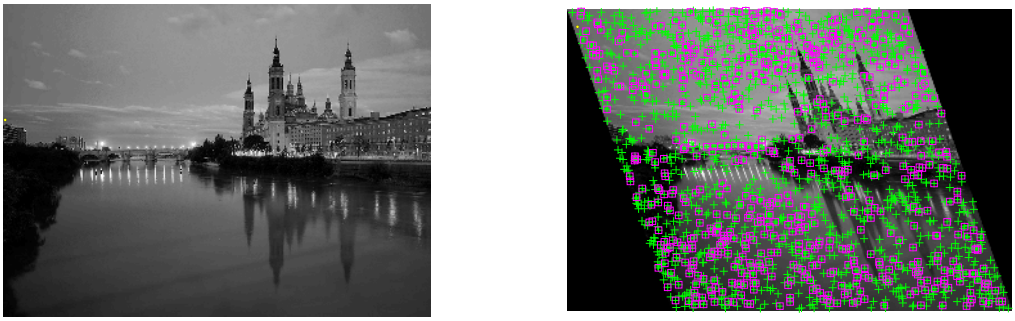
θ	SIFT	HarAff	HesAff	MSER	ASIFT
50°	267	131	144	150	1692
60°	20	29	39	117	1012
70°	1	2	2	69	754
80°	0	0	0	17	349

Table 1.1. Number of correct matches of ASIFT, SIFT, Harris-Affine (HarAff), Hessian-Affine (HesAff), and MSER for viewpoint angles between 50° and 80° on a Magazine cover [23].

Overall speaking, MSER has a better affine robustness performance than SIFT in most cases. But the lack of scale invariance for MSER is a shortcoming. Accordingly, MSER is not a good option for the visual search of different view points because accompanied with the view point changes, the distances between the cameras will also vary. Figures 1.9 and 1.10 present the performance of image matching respectively by SIFT and MSER. The view point change in the figure 1.9 is comparably large, resulting in a better performance for MSER. In the figure 1.10, there is a scale change accompanied with the change of view points. SIFT has a better performance than MSER. Generally speaking, both SIFT and MSER are not qualified for the image matching of different view point. One is quite sensitive to the view point changes and the other is not scale invariant. Comparably, Harris-Affine, Hessian-Affine are even less competitive than these two algorithms. Table 1.1

presents the matching performance of these algorithms under different view point changes. ASIFT has the best performance among all these algorithms under all the view point changes. The performance of SIFT under a small view point changes is quite remarkable. Whereas, it drops quickly following the increasing of the angles. To MSER, it is comparably robust to the view point changes, but its fundamental matching capability is less competitive than SIFT.

Except for ASIFT, the performance of all the other visual search algorithms cannot satisfy the requirements for the view point robustness. ASIFT is based on the image simulations of different view point, resulting in a high redundancy of computation when dealing with the images of different perspectives. An ideal visual search algorithm shall be based on SIFT to inherit its scale, rotation and illumination invariance, but will also be able to adapt to affine transformations. To design a visual search algorithm extended from SIFT, it is better to clarify the reason of SIFT sensitivity to affine transformation. A complete SIFT algorithm can be divided into detection procedure and description procedure, and a reasonable way to track the sensitivity to affine transformation shall be to distinguish on which part the problem lies.



(a) (b)

Figure 1.11. Affine robustness evaluation on detection and description

θ	Total extraction	Matching features	Correct detection	Missing detection
50°	3572	0	1090	0
60°	4318	0	615	0
70°	4065	0	237	0
80°	2991	0	74	0

Table 1.2. Affine robustness evaluation of SIFT on detection and description with viewpoint angles between 50° and 80°.

To clarify the source of affine transform sensitivity, we have design an affine evaluation on both detection and description, depicted in the Figure 1.3. In the Figure, the green crosses depict the total extracted features in the affine transformed image. The magenta crosses stand for the detected features from the original image while projected to the affine transformed image. If the green cross

matches with the magenta cross, it means the feature has not been affected by the affine transformation. If we can declare a match with the corresponding feature from the original image, it can also be matched with the feature calculated from the image with a similar perspective difference. These matched features will be colored with red triangle. The projected features, which have not been matched with any affine transformed image feature but happen to be at the same geometric coordinates and same scale value, will also be declared as a correct detection and depicted by a magenta square icon. An image in the framework of scale space can be fully represented by both its geometric coordinates and scaling value. If two features of the same content appear at the same coordinates and same scaling value, they are the same feature to be matched. If these two expected to be matched features do not match in reality, it is the descriptor that fails for a correct identification, and not the detection. In this way, we can track from which part the affine sensitivity of SIFT is coming. Table 1.2 displays the result of this evaluation under different levels of affine transformations. From the table, we can notice that none of the extracted features matched, revealing SIFT is rather sensitive to affine transformations. At 50˚, there are 3572 total extractions and among them 1090 are correct detection, which is about one third of them. But at 80˚, there are only 74 left correct detections. This number drops quickly with an increase level of affine transformation. Overall speaking, the mismatch of SIFT due to affine transformations occurred on both the detection and description procedures.

To design a view point robust visual search technique based on SIFT, the detection and description are both to be adapted to the affine transformation. In chapter 2, we describe a post-processing stage integrated in CDVS to improve its retrieval accuracy under affine transformations. Chapter 3 introduces an affine adapted SIFT based feature detection and the theory behind. In chapter 4 introduce an affine adapted SIFT based descriptor. Combining both these chapters, a new affine robust visual search technique will be presented.

1.4 Contribution of this work

To handle the sensitivity to view point changes of the up-to-date visual search techniques, we mainly focus on two different scenarios of view point invariance.

1. CDVS-compatible scheme, which aims to improve its robustness of image retrieval under different view points.

2. A more general scheme based on an affine invariant scale space, including the scale space based feature detector and feature descriptor.

The CDVS-compatible scheme is based on the theory that each pair of affine transformations are mathematically related by homography matrix. Given that matrix, the image can be modified according to the matrix to simulate its appearance from another view point. In this way, the image under different perspective projection can be declared as matching because the distortion by different projection can then be reduced by the coordinates modification. An accurate homography estimation from the images of different view point requires at least 4 correspondences, which can be obtained by the CDVS pipeline as will be described in chap. 2.

As we have discussed above, scale space is the foundation of the up-to-date scale invariant visual search techniques. It is a structure to represent an image of different level of details by blurring with a parameterized family of Gaussian filters. The scale space provides a scale-invariant representation, which is necessary for dealing with the size variations that may occur in image data. However, the scale space representation makes the image unsuited be linear transformed to compensate for a different viewpoint without a modification on the Gaussian filters. A linear transformed image convolved with a non-linear transformed filter will undermine the linear relationship, which existed between two perspective projections and can be captured by our brain. Thus, we propose an affine scale space to retain the linear relations of the scale invariant representation on the affine transformed images. In the framework of affine scale space, both the scale and affine invariant feature detection and descriptor has been re-designed to adapt to the modifications necessary to perform re-alignment of images taken from different viewpoints.

1.4.1 CDVS-compatible scheme

The idea for CDVS-compatible scheme is to detect and simulate the image under a different perspective projection by affine transforming the image coordinates. By this simulation, no perspective distortions between the query image and the reference image will exist to affect the matching process of CDVS, thus this scheme can be applied to the images of any view points by reducing the view points difference with the stored image. The applicability of this scheme is based on an accurate detection of the perspective difference between the query image and stored image, which can be mathematically described by homography matrix. A homography relation between the images of different view points is based on the mathematical model, that an image from a certain view point is obtained by a perspective projection of the target image. Thus, the images from different view points can be originated from the same image by different perspective projections. The relationship of different perspective projections can then be described by a homography matrix, which relates the coordinates of the same object points on different images. Given that matrix, one of the images can be re-projected to simulate the image from another view point. In this way, images can be matched by reducing the perspective distortion between them. Thus, an improvement to the robustness to view point changes is now obtained via an accurate estimation of the homography matrix between the images.

Since the homography relation of two perspective projections is homogenous to all the pixels on the images, the homography matrix can be estimated by a few correspondences. These few correspondences which depict the same points on the object, provide the coordinates relationship. By analysis these relationships, a homography matrix can then be estimated. These correspondences can be provided by CDVS matching scheme without a perspective re-projection.

In practice, we designed a homography based post-processing stage integrated in the general CDVS pipeline to improve its resilience to view point changes. This post-processing stage will not trigger unless the number of matched features are not enough to declare image matching. This post-processing stage will act as a double check to find whether the fail of match is due to the view point difference or not. We have also prepared the algorithm to the images of multiple planes by recursively applying RANSAC to rule out the features not belong to a certain homography relations.

By testing the images of different categories, we have show this post-processing stage can be used to improve the matching precision.

1.4.2 Affine scale space

The purpose of the affine scale space is to create a more general approach to the affine invariant image scale representation by steering the Gaussian filters to the specific affine transformations. It has the objective to retain a linear relationship of the scale space by steering the filters following the transformation of the image. With this linear relationship, this representation can be kept as a homogeneous transformations to all the scales. Thus a homogeneously transformed scale space can easily be modified by the same adaptation to cope with the view point changes especially when the transformation is given.

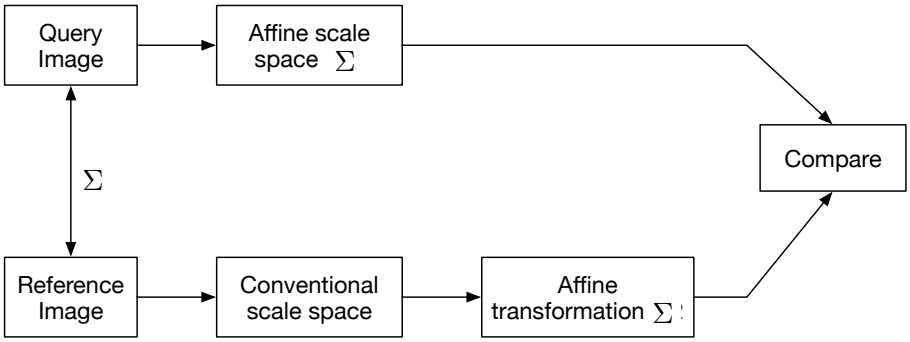


Figure 1.12. The affine robustness of the affine scale space implementation will be evaluated in this way. Σ is the supposed affine transformation.

This affine scale space aims to establish a more general approach to the scale and affine invariant image retrieval, including affine feature detection and affine feature descriptor. We should first understand how the detection and descriptor change in an image taken from another view point. As we have discussed above, an image from another view point can be described by an affine transformation, whose coordinates are linearly related with the original image. This linear relationship can easily be captured and calculated by our brain but cannot be automatically captured by the conventional scale space, which the scale invariant feature detection and feature descriptor are based on. Even the image itself has an approximately linear relationship with the original image, the convolution with a parametrized family of Gaussian filters will spoils this relationship. Thus, the scale space of the image from another view point cannot be related with the original one by a simple linear relationship and the feature detection and feature descriptor can be quite different. To address this issue, we can either steer the image to simulate a similiar view point, or steer the filters to establish a scale space that is still linear related with the original one. Our post-processing stage in chap. 2 and ASIFT choose the first method to cope with this. It is not economical especially in the image retrieval system, dealing with millions of images. We will also introduce the affine scale scale as a more general approach to capture and describe the features in an affine invariant way. Since the idea of scale space has been applied for both the feature detection and feature descriptor, the affine scale space can also be applied to both these areas.

In practice, scale invariant visual search techniques have employed a pyramid structure to speed up the scale space construction. This structure is based on the scale space principles, including sampling and cascading. By iteratively sampling and convolving with the smoothed image, the complete scale space can be constructed using Gaussian filters and the LoG can be created by a

Laplacian operation on the scale space. For the affine scale space, we will also propose two structures to build the affine Gaussian scale space and affine LoG. The structure of affine Gaussian scale space can also be created by the pyramid structure because sampling and cascading are also properties of affine Gaussian operators. However, to cope with the affine LoG, we needed to develop specific structures. The Laplacian operator, under affine transformation, is hard to be affine deformed because of the constraint on its size. Differently from the Laplacian operation to obtain the general LoG, the affine LoG can only be calculated by affine LoG filter integrating the convolution and cascade implementations in the construction of affine scale space. By our proposed structures, the affine Gaussian scale space and affine LoG can be simultaneously constructed. We have also proposed two scale space implementations in frequency domain by some special properties of affine alignment to obtain a more accurate affine robust scale space. The affine scale space is a forward model, allowing to predict what will happen to an image under a different view point.

The performance of affine scale space will be evaluated according to the figure 1.12. Supposing a known affine transformation Σ, we can construct both the affine scale space and the conventional scale space and by comparing the forward transformed general scale space, we can obtain the evaluation of the affine robustness for the implementation. Generally speaking, the precision of affine scale space implementation in spatial domain ranges from 36dB to 42 dB for SNR and in frequency domain ranges from 39dB to 18dB for SNR. The change of precision is mainly due to the change of scale. In spatial domain, the larger the scale the more accurate the scale space implementation is, whereas in frequency domain, the tread is the opposite.

We have also applied the affine scale scale to the feature detection. It has been proved to have a better performance than the general scale space especially when the difference of view point is large. General speaking, it can guarantee that at least 30% of features will be retained the same without being affected by the affine transformation, whereas for the conventional scale space, only 1% of the features will be detected when the view point change is larger than 70°.

By adopting the same idea, we will also propose an affine invariant descriptor, which borrows the idea of steerable filters to create the affine gradient as the histogram element. Unlike the conventional image gradient method, our proposed descriptor employs the Gaussian derivative filters to create the gradient of each scale space. Combined with the Gaussian filter, this gradient operator can easily be affine adapted, similar to the method we have applied to the affine scale space. With this affine gradient, we can apply the same scheme of SIFT to generate the histogram. By normalizing the histogram and selecting the area over which to bin the gradient, an affine invariant but also standard compatible affine descriptor can be formed. We have tested this descriptor by comparing one feature with 200 randomly selected points from the image to test whether the target feature will be matched or not following the tilting of the image. The result shows that, the performance of matching precision for the general feature descriptor drops quickly, whereas the affine descriptor is quite robust to the affine transformation. Over 85° view point difference, the affine descriptor is 35% more precise than the conventional descriptor. In practice, this affine descriptor is fully affine invariant and its performance for image matching is extremely good.

Chapter 2

Homography model and Back-projection

As introduced in the first chapter, CDVS is an MPEG proposed standard that will enable efficient and interoperable design of visual search applications by applying feature detection and description. Such techniques are invariant to rotation and scaling, but are not very robust towards viewpoint changes. In this chapter, we address this problem and propose a post-processing of the CDVS pipeline that employs image back-projection to compensate for perspective distortion via a re-ranking stage. The proposed technique is based on the Homography derived from the correspondence extracted from pairs of matching keypoints.

To our best knowledge, this is the first time that the homography transform is employed to improve robustness of local descriptors to viewpoint changes. Extensive tests on the CDVS database show that the proposed technique can improve the matching precision up to 3%.

2.1 Homography model

In a content based image retrieval system, two images of the same scene are to be matched. If the viewpoints between these images are different, *i.e.*, there is perspective distortion between them, then a correct matching might not be possible. To reduce the perspective distortion, we propose to estimate the Homography between the two images [28]. With projective cameras, any two images of the same planar surface in space are related by a Homography [29]. Homography can be used to estimate the projective position and projective plane. Once the Homography is known, back-projection can be used to reduce the perspective distortion and improve the image matching accuracy, as we show in Sec. 3.

Figure 2.1 depicts a Homography model. Image 1 and Image 2 are images of a 2D planar object. One point on Image 1 correspond to another point on Image 2 when they both reflect the same point on the object. Images of 2D planar objects are obtained via projective reflection. Thus images with different viewpoints are from a different reflection. These reflections are projectively related in geometry. This relationship can be estimated after knowing corresponding pairs of points because it is a homogenous relationship among all the points on the planes.

The mathematical definition of a Homography is given below:

$$P_a = \begin{bmatrix} x_a \\ y_a \\ 1 \end{bmatrix}, P_b = \begin{bmatrix} w' x_b \\ w' y_b \\ w' \end{bmatrix} \tag{2.1}$$

Then: $P_b = \mathbf{H}_{ab} P_a$ where $\mathbf{H}_{ba} = \mathbf{H}_{ab}^{-1}$. P_a and P_b are the corresponding points on different 2D planes. Notice that points laying on $\mathbf{R}^2$ are normally represented as a pair $(x, y)^T$. However in projective geometry intersection points of lines or planes are more relevant. For a homogenous representation, a third coordinate is added as a scale variable [29]. Therefore, an arbitrary homogeneous vector representative of a point is of the form $P = (x_1, x_2, x_3)^T$, representing the point $(x_1/x_3, x_2/x_3)^T$ in $\mathbf{R}^2$. The points at infinity can be represented with $x_3 = 0$. $\mathbf{H}_{ab}$ is the Homography matrix, representing the projection of point P_a to P_b. $\mathbf{H}_{ba}$ is the corresponding inverse transformation.

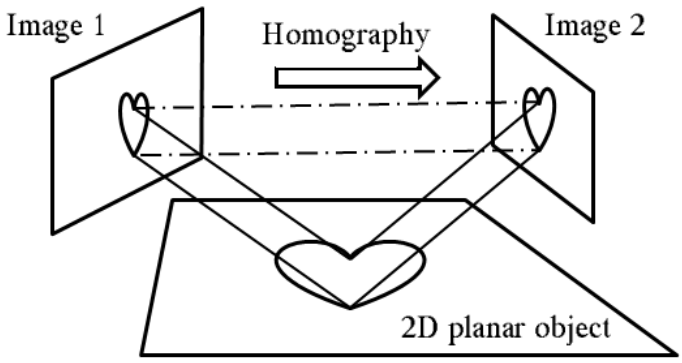


Figure 2.1. Homography model.

To model the perspective distortion, we need to estimate the Homography matrix. Direct Linear Transformation (DLT) is one of algorithms to determine $\mathbf{H}_{ab}$, given a set of 2D to 2D point correspondences $P_a \leftrightarrow P_b$ [29]. P_a and P_b are the corresponding points on different planes.

Letting pairs of correspondences be related by $P_b = \mathbf{H} P_a$ then,

$$\mathbf{H} P_a = \begin{pmatrix} \mathbf{h}^{1T} P_a \\ \mathbf{h}^{2T} P_a \\ \mathbf{h}^{3T} P_a \end{pmatrix}, with\ \mathbf{H} = \begin{pmatrix} \mathbf{h}^{1T} \\ \mathbf{h}^{2T} \\ \mathbf{h}^{3T} \end{pmatrix}, \tag{2.2}$$

Writing $P_b = (x_b, y_b, z_b)^T$, the cross product may then be given as

$$P_b \times \mathbf{H} P_a = \begin{pmatrix} y_b \mathbf{h}^{3T} P_a - z_b \mathbf{h}^{2T} P_a \\ z_b \mathbf{h}^{1T} P_a - x_b \mathbf{h}^{3T} P_a \\ x_b \mathbf{h}^{2T} P_a - y_b \mathbf{h}^{1T} P_a \end{pmatrix} \tag{2.3}$$

Since $\mathbf{h}^{jT} P = P^T \mathbf{h}^j$ for $j = 1, \ldots, 3$, it can be written in the form as,

$$\begin{bmatrix} \mathbf{0}^T & -z_b P_a^T & y_b P_a^T \\ z_b P_a^T & \mathbf{0}^T & -x_b P_a^T \\ -y_b P_a^T & x_b P_a^T & \mathbf{0}^T \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}. \tag{2.4}$$

These equations have the form $A_i \mathbf{h} = 0$, where A_i is a 3×9 matrix, and $\mathbf{h}$ is a 9-vector made up by the entries of matrix $\mathbf{H}$,

$$\mathbf{h} = \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix}, \mathbf{H} = \begin{pmatrix} \mathbf{h}^{1T} \\ \mathbf{h}^{2T} \\ \mathbf{h}^{3T} \end{pmatrix}, \tag{2.5}$$

If $\mathbf{H}$ is written in this way,

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}. \tag{2.6}$$

Then, $\mathbf{h} = \left(h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9\right)^T$. Thus, the solution of the equation $A_i \mathbf{h} = 0$ contains the entries for the Homography matrix $\mathbf{H}$.

The equations $A_i \mathbf{h} = 0$ is an equation linear in the unknown $\mathbf{h}$. The element of A_i are from the coordinates of known correspondences. Although, three vectors are, contained in the equations (2.4), only two of them are linearly independent. Thus each point correspondence provides two equations in the entries of $\mathbf{H}$. The third equation can easily be replaced by the other two and the solution to the combination of these three equations can be used to set up the Homography matrix $\mathbf{H}$. Thus the set of equations becomes,

$$\begin{bmatrix} \mathbf{0}^T & -z_b P_a^T & y_b P_a^T \\ z_b P_a^T & \mathbf{0}^T & -x_b P_a^T \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}. \tag{2.7}$$

The expression can also be simplified as, $A_i \mathbf{h} = \mathbf{0}$, where A_i is now a 2×9 matrix.

Since $\mathbf{h}$ is a 9 element vectors defined by to the entries of $\mathbf{H}$, the rank of $\mathbf{h}$ is the minim number of equations that are necessary to determine a unique solution of the linear equations $A_i \mathbf{h} = \mathbf{0}$. All the elements are mutually independent except h_9 presenting the scale of the transformation, which is determined by the rest of the elements. Thus, the rank of $\mathbf{h}$ is 8. If A_i is a collection of 8 independent equations as the constraints to $\mathbf{h}$, a unique solution can be directly obtained. In practice, each pair of correspondences give rise to two independent equations to the entries of $\mathbf{H}$. To satisfy the minimum number of equations, it is necessary to specify four point correspondences in order to constrain $\mathbf{H}$ fully. By detecting a set of four such matched correspondences, we obtain a set of equations $A\mathbf{h} = \mathbf{0}$, where A is the matrix of equation coefficients built from the matrix rows A_i contributed from each correspondence, and $\mathbf{h}$ is the vector of unknown entries of H. After all, the solution we seek should be non-zero, since the obvious $\mathbf{h} = \mathbf{0}$ is meaningless to us. Give 4 pairs of correspondence, a full rank A can be used to seek the uniquely determined entries of $\mathbf{H}$.

If exactly four pairs are given, then a unique solution for the matrix $\mathbf{H}$ can be obtained. However, since matching pairs are not known exactly, because of the non ideality of the keypoint detector, if more than four correspondences are given then these correspondences may not be fully compatible

with any projective transformation, and one will be faced with the task of determining the best transformation given the data.

If more than four point correspondences are given, then the set of equations $A\mathbf{h} = 0$ is over-determined. If the position of the keypoints are exact, we can seek an optimized approximation to the over-determined system $A\mathbf{h} = 0$ apart from the all-zero solution. However, we cannot be sure that all the available correspondence pairs are reliable, so we must identify and remove outliers before estimate the Homography.

To this end, we employ RANSAC [29], which is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers [30]. The idea is very simple: two of the points are selected randomly; these points define a line. The confidence score for this line is calculated as the number of points that lie within a maximum distance. This random selection is repeated a number of times and the line with highest confidence score is deemed the robust fit. The points within the threshold distance are the inliers. The aim of this stage is two-fold: first, to obtain an improved estimate of the Homography by using all the inliers available in the given correspondence pairs (rather than only the four points of the sample); second, during the following back-projection stage, to obtain more matches from the correspondence set because a more accurate Homography is available. An improved estimate of the Homography is then computed from the inliers.

After ruling out the outliers, the rest of the correspondence can be used as the constraint to the Homography matrix $\mathbf{H}$. If more than four of these correspondences are given, then the set of equations derived from (2.7) is over-determined. If the positions of all the given points are exact and the matrix A will still have rank 8, we have a one dimensional null-space, and there is still an exact solution for $\mathbf{h}$. Additionally, the equations related to the set of correspondences are cross-related with each other, until 8 unrelated equations are identified. In that case, the 8 independent equations shall firstly be selected among all available equations by the given correspondence and an exact solution can be seek by expectation sought.This will not be the case if the measurement of image coordinates is inexact (generally effected by noise) – there will not be an exact solution to the overdetermined system $A\mathbf{h} = 0$ except the zero solution. In that case, instead of demanding an exact solution, one attempts to find an approximate solution, namely a vector $\mathbf{h}$ that minimizes a standardized cost function. Generally, the norm of the target function will be used as the constraint to $\mathbf{h}$. Given that there is no exact solution to $A\mathbf{h} = 0$, it seems natural to attempt to minimize the norm $\|A\mathbf{h}\|$ instead, subject to $\|\mathbf{h}\| = 1$. The solution to minimize the norm is by the eigenvector of $A^T A$ with least eigenvalue. Equivalently, the solution is the unit singular vector corresponding to the smallest singular value of A. Specifically, if $A = UDV^T$ with D diagonal with positive diagonal entries, put into descending order down the diagonal, then $\mathbf{h}$ is the last column of V.

Another more sophisticated use of the Homography based back-projection is the Homography detection of multiple 2D planar objects. As discussed above, Homography is the matrix to calibrate the 2D plane projective difference. Theoretically, a Homography relation only exists between the same planes of different perspectives. Equivalently, if multiple planar objects are contained in the images with different view-point, Homography relations should be associated to each pair of planes rather than a universal Homography for the whole image, and the correspondence from different geometrical parts of image may be used to derive different Homography relations. In real applications, the number of planes and the complexity of the contents are not previously given, and must be estimated by the algorithm itself, which enormously increase the implementation complexity. In

practice, we collect all matched pairs indifferently and categorize the pairs by recursively applying RANSAC to remove the pairs different from the majority in the same plane. From the removed pairs, we identify another potential plane by using RANSAC again until all the planes have been identified or the number of pairs from that specific plane is not enough for a Homography estimation. In this way, the Homography based back-projection can also be extended to the images with multiple planes.

2.2 Proposed homography-based retrieval stage

CDVS is the standard under development in MPEG that will provide a highly efficient and interoperable pipeline for visual search; Figure 2.2 displays the descriptor extraction of CDVS [24]. It includes keypoint detection, feature selection, local descriptor computation, local descriptor compression and coordinate coding. Keypoint detection and descriptor computation are the fundamental operations of visual search. The purpose of feature selection is to preserve the most significant keypoints for a low memory consumption. Local descriptor compression and coordinate coding both aim to decrease the memory consumption and transmission bandwidth. The global descriptor is set aimed to quickly retrieve the target image from millions of candidates by applying some statistics and machine learning techniques [24].

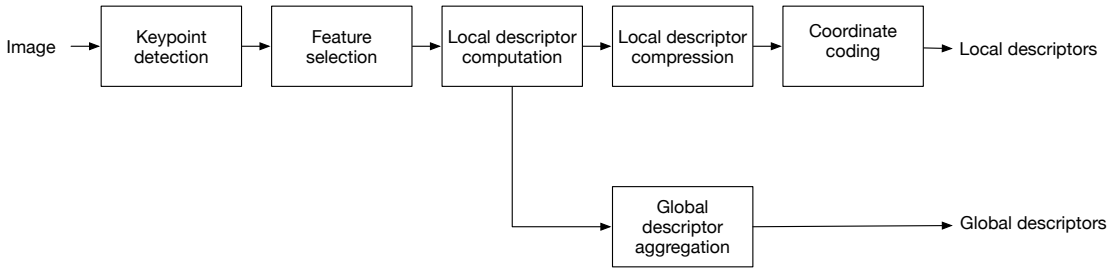


Figure 2.2. CDVS descriptor extraction.

In the CDVS standard, whether two images will be declared as matched or not depends on their matching score. Each pair of matched features will be assigned a score and the total image matching score is obtained by summing up all the scores of the matched features on that image. However, since CDVS is not viewpoint robust by construction, it may wrongly declare matching or non-matching images because of perspective distortion. In this paper we argue that perspective distortion can be reduced by homography estimation and back-projection. Back-projection consists in inverting the perspective transformation. The first step towards back-projection is to estimate the homography that defines the inverse transformation.

As has been said, a homography can be derived from at least 4 pairs of corresponding points. However, the image matching process will typically provide more than 4 matching pairs. In our proposed system, we used the DLT algorithm and RANSAC, as detailed in Sec. 2, to estimate the homography. Then, the perspective distortion can be reduced applying back-projection.

2.2.1 Perspective distortion reduction by back-projection

In particular, the proposed estimation and back-projection process operates as follows. Suppose two images I_a and I_b have an approximate homography relationship. The standard CDVS pipeline might declare a non-match between I_a and I_b because of the perspective distortion. Once the homography $\mathbf{H}_{ab}$ is estimated by DLT, the image I'_a is obtained as the set of points from the images satisfies $I'_a = \mathbf{H}_{ab} I_a$. In other words, we now have a new pair of images, I'_a and I_b, where the perspective distortion has been removed or at least strongly attenuated from image I_a. It is therefore reasonable to assume that, while CDVS might wrongly declare I_a and I_b as a non-match, it can be likely to correctly declare I'_a and I_b as a match without the perspective distortion. Thus I'_a and I_b are set as the new pair to be checked as matching or non-matching by CDVS [31].

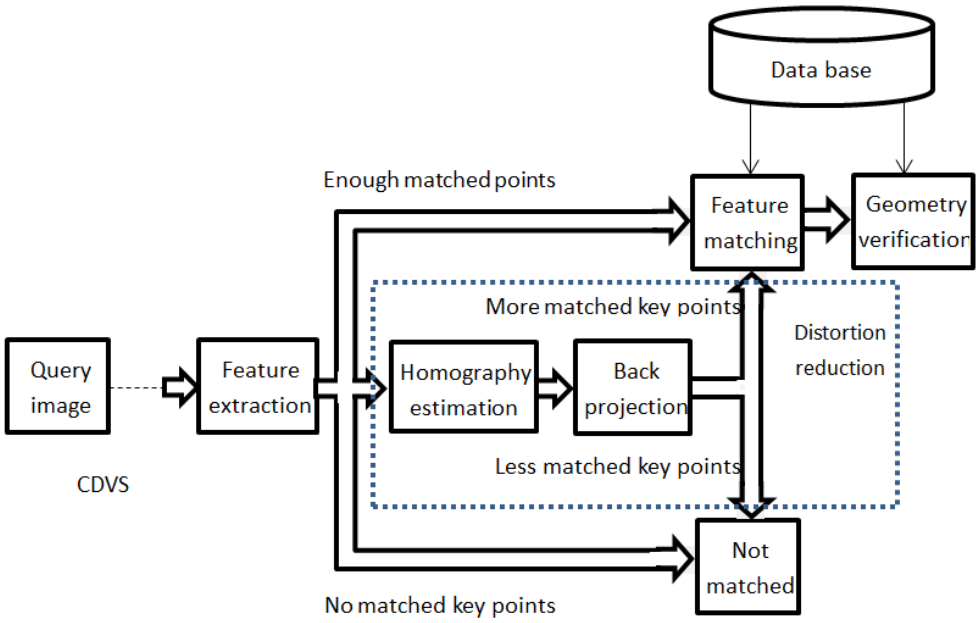


Figure 2.3. Integrated back-projection CDVS.

More in detail, the pipeline of our proposed method is displayed in Figure 2.3. From the pipeline, we can see that our proposed stage is integrated into the standardized CDVS visual search system. This guarantees to exploit CDVS's high efficiency and accuracy. The area inside the dotted rectangle is our proposed stage. It includes homography estimation, back-projection and re-matching of an image pair after compensating for perspective distortion.

In particular, the re-matching process is triggered only if the matching score does not exceed the threshold. That is, if CDVS believes the images are matched, we trust this as it is likely that the images had small perspective distortion. Instead, if CDVS decided that the image pair does not match, we perform back-projection and re-matching to see if a transformation can be found, which will estimate and correct the perspective distortion leading to a positive match. In particular, the re-matching stage checks whether more than 4 matched corresponding pairs are available. If this is

the case, a homography matrix is estimated and back-projection is used to remove the perspective distortion between the images. Thus, the image pair without perspective distortion will have more matched features. However, it is not guaranteed that matched features are truly matched or the positions of the matched features are exact. The back-projection based on non accurately estimated homography cannot help to decrease the perspective distortion. In the pipeline, to make sure that the perspective distortion of back-projected image is not worse than the initial one, the score after the re-matching stage is compared with the initial score. If the score has not been improved, the initial matching score and the related matching decision will be preserved [31].

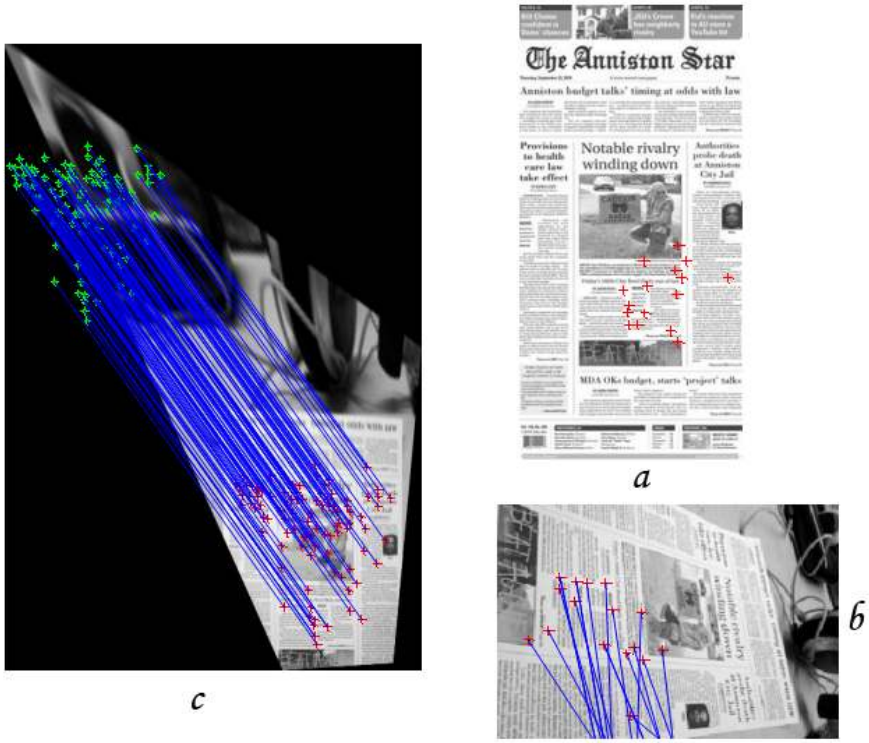
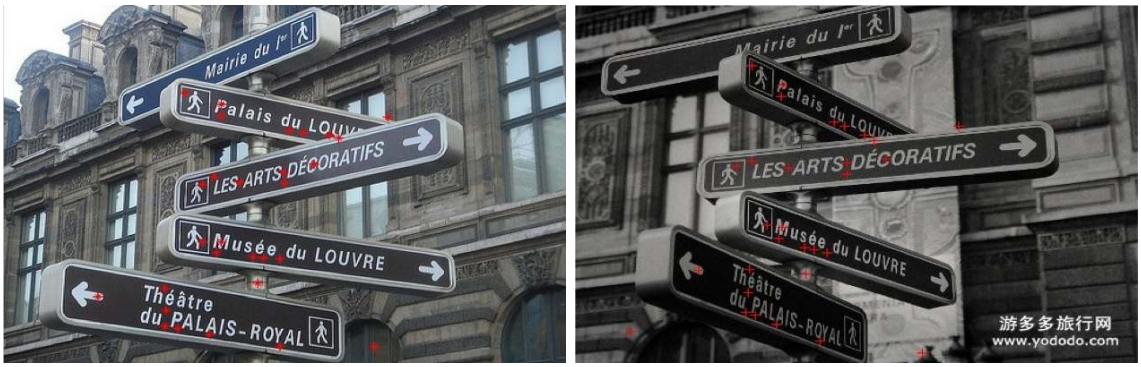


Figure 2.4. Improvement on the numbers of features.

To understand the re-matching process, note that, as Figure 2.4 displays, after reducing the perspective distortion, there will normally be an increase of the number of matched features. If the increased score exceeds the threshold after the back-projection, then the previous non-match will be turned into a correct match. In Figure 2.4, *a* and *b* are the initial images. Due to the perspective distortion, the number of the matched features is around 15. *c* is the image after reducing the perspective distortion from *b*. The matched features between *a* and *c* are around 125. It is a great increase of the numbers of matched features, which can lead to a correct match, while the two initial images would not have been matched.

2.2.2 Implementation on images of multiple planar objects

A more sophisticated integrated CDVS pipeline based on Homography and back-projection can be set up to handle the view-point changes to multiple 2D planar objects. Up to the descriptions above, a Homography relation exists between any 2 different perspectives of the planes. An image containing more than one 2D plane (either multiple 2D objects or a very complicated multiple planar object or both) may rise ambiguities for the Homography estimation since the Homography estimation is based on all the correspondences whose perspective constraints are homogeneous. But the correspondences from different plans will correspond to different Homography relations, thus a unique Homography relation cannot be fixed by indifferently collecting the correspondences from the images with different perspectives. A reasonable approach to identify each corresponding Homography relation is to categorize in advance the correspondences of each plane and then identify the related Homography matrix. That is not easy to implement because the perspective changes can be vary a lot. However, a Homography matrix can be fully constrained by 4 correspondences. Equivalently, every set of more than 4 correspondences responding to the same plane can be grouped and added to as the identification of the source of the correspondence. In practice, the correspondences which do not refer to the estimated Homography matrix by the main group of points will be rejected as outliers by RANSAC. In this way, the Homography referring to the main set of points can be fixed. Iteratively, from the rejected points, we can then rule out the outliers and fix the sub-main plane composed by the remaining points until all Homographies have been estimated or the correspondences from the planes are not enough for an Homography estimation.



(a) (b)

Figure 2.5. Image with multiple planes

Figure 2.5 presents a typical image matching containing multiple planes at different perspectives. For this kind of image matching, a homography based back-projection cannot completely reduce the perspective distortion between them. The perspective distortion exists between any two of the planes in the images, but a back-projection can only be used to reduce one of them.

Figure 2.6 demonstrates the process of each step to identify the correspondence from different planes by an iterative RANSAC-Homography method.

Figure 2.7 presents the pipeline of the Homography estimation for the images with multiple 2D planes. Generally, after an appropriate Homography estimation of each 2D plane, the image will be back-projected respectively to each Homography. Then, the matching pairs from that specific

(a) (b) (c)

Figure 2.6. The correspondence on each plane by iterative RANSAC

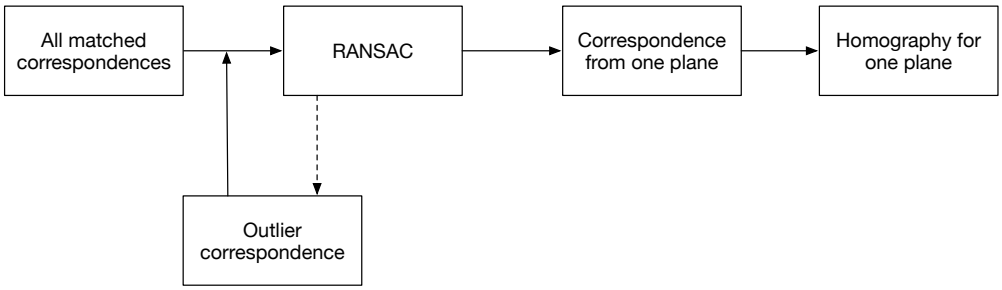


Figure 2.7. Improved pipeline for Homography estimation for multiple planes.

plane will be expected to increase by reducing the perspective distortion between them.

2.3 Experiments and results

Our proposed method has been integrated into the CDVS test model. Experiments are conducted by employing the MPEG dataset used for the evaluation of CDVS. In the dataset, there are 5 image categories. Additionally, Category 1 has 3 sub-categories. These dataset are defined as follows.

- Mixed text and graphics
- Mixed text and graphics at VGA resolution
- Mixed text and graphics at VGA resolution with heavy JPEG compression
- Paintings
- Video frames
- Buildings and landmarks
- Common objects

Totally, there are 33590 images in the dataset.

The experiment for evaluating the performance of the proposed scheme is designed as follows. In each category, original CDVS and our integrated back-projection CDVS are tested calculating both matching precision and non-matching precision. Our experiment has been run on all categories. As expected, the proposed back-projection method is more efficient in the categories of objects where the planar assumption is reasonable, although no performance decrease is observed in the other categories, leading to an overall improvement.

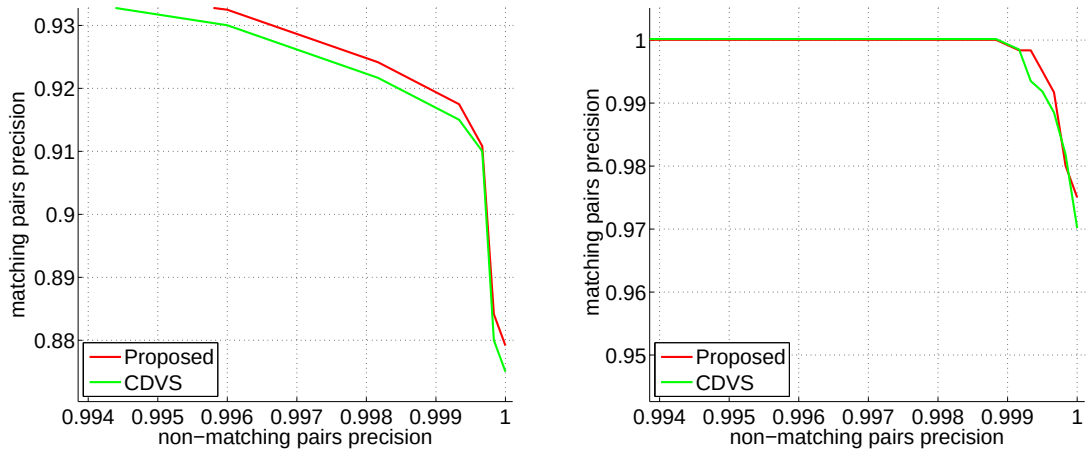


Figure 2.8. Proposed method precision evaluation, (a) prints; (b) dvds.

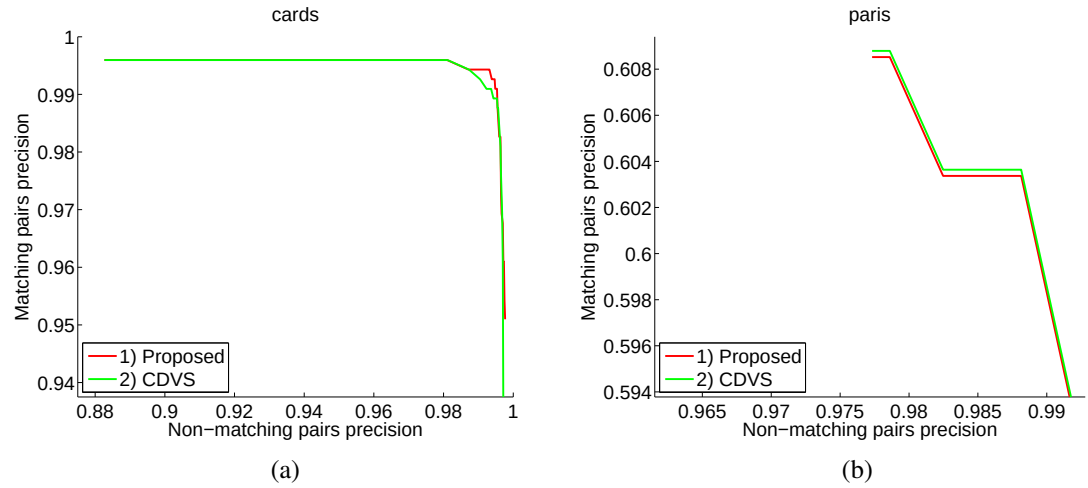


Figure 2.9. Proposed method precision evaluation, (a) card; (b) paris.

Books	**MP**	**NMP**	**CDs**	**MP**	**NMP**	**Dvds**	**MP**	**NMP**
CDVS	0.980	1.000	CDVS	0.900	1.000	CDVS	0.970	1.000
Proposed	0.990	1.000	Proposed	0.920	1.000	Proposed	0.975	1.000
Improvement	0.010	0.000	Improvement	0.020	0.000	Improvement	0.005	0.000
Books-vga			**CDs-vga**			**Dvds-vga**		
CDVS	0.980	1.000	CDVS	0.921	1.000	CDVS	0.921	1.000
Proposed	0.985	1.000	Proposed	0.938	1.000	Proposed	0.938	1.000
Improvement	0.005	0.000	Improvement	0.017	0.000	Improvement	0.017	0.000
Books-vga-jpeg			**CDs-vga-jpeg**			**Dvds-vga-jpeg**		
CDVS	0.983	1.000	CDVS	0.901	1.000	CDVS	0.976	1.000
Proposed	0.987	1.000	Proposed	0.918	1.000	Proposed	0.985	1.000
Improvement	0.004	0.000	Improvement	0.017	0.000	Improvement	0.009	0.000
Cards			**Print**			**Video**		
CDVS	0.960	0.997	CDVS	0.872	1.000	CDVS	0.858	0.999
Proposed	0.965	0.997	Proposed	0.880	1.000	Proposed	0.838	0.999
Improvement	0.005	0.000	Improvement	0.008	0.000	Improvement	0.020	0.000
Cards-vga			**Print-vga**			**Stanford**		
CDVS	0.935	0.997	CDVS	0.848	1.000	CDVS	0.555	1.000
Proposed	0.952	0.997	Proposed	0.882	1.000	Proposed	0.561	1.000
Improvement	0.017	0.000	Improvement	0.034	0.000	Improvement	0.001	0.000

Table 2.1. Performance of the proposed pipeline on the categories of partial CDVS dataset.

Figure 2.8 shows some result on the matching precision, in particular (a) displays the result in *print* and (b) displays the result in *dvds*. The red line represents the precision of the proposed back-projection method and the green one represents the precision of original CDVS, and it can be seen that the proposed algorithm consistently outperforms CDVS. Figure 2.9 presents the matching precision on the images of *cards* and *paris architectures*. Apparently, the improvement on different categories of images is different.

The result are further analyzed in Table 1, including planar objects at original resolution, at VGA resolution and at VGA resolution with heavy JPEG compression, buildings, landmarks and video frames. In the table, MP is short for matching pairs precision and NMP is short for non-matching pairs precision. Generally speaking, our proposed back-projection method can improve the matching precision more than the non-mathing precision. However, the improvement varies among categories and resolutions. As expected, the improvement on the 2D planar objects is more obvious compared with buildings and landmarks. But even on buildings and landmarks, our method can still somewhat improve the matching precision. The average improvement on the buildings and landmarks is about 0.3% and the average improvement on the 2D planar objects is 2.9%. JPEG compression will not affect the improvement but resolution indeed has an effect. Typically, a image of higher resolution will generate more matched pairs of keypoints, but these matches are not generally more correct than in a lower resolution image. More incorrect matched keypoints do not contribute to a correct homography estimation, hence a higher image resolution generally did not

provide better results.

Cards-vga-jpeg			**Print-vga-jpeg**			**Turin**		
CDVS	0.962	0.999	CDVS	0.852	1.000	CDVS	0.662	1.000
Proposed	0.967	0.999	Proposed	0.872	1.000	Proposed	0.663	1.000
Improvement	0.005	0.000	Improvement	0.020	0.000	Improvement	0.001	0.000
Paintings			**Objects**			**Zubud**		
CDVS	0.972	1.000	CDVS	0.938	1.000	CDVS	0.879	1.000
Proposed	0.979	1.000	Proposed	0.940	1.000	Proposed	0.879	1.000
Improvement	0.005	0.000	Improvement	0.002	0.000	Improvement	0.000	0.000
Etri			**Huawei**			**Paris**		
CDVS	0.984	1.000	CDVS	0.688	1.000	CDVS	0.604	1.000
Proposed	0.985	1.000	Proposed	0.688	1.000	Proposed	0.605	1.000
Improvement	0.001	0.000	Improvement	0.000	0.000	Improvement	0.001	0.000

Table 2.2. Performance of the proposed pipeline on the categories of partial CDVS dataset.

2.4 Conclusion

In this chapter, we proposed a new method based on the CDVS pipeline, attempting to improve the matching precision of images pairs taken at different viewpoints, which is known to be a difficult case for SIFT descriptors. The method employs Homography, and is fully integrated into the CDVS standard, its complexity is low and it can improve the matching precision, especially on images of 2D planar objects. In particular, performance improvement is up to 3% on those image categories that satisfy the planar model, such as print and CDs.

Chapter 3

Affine scale space

Scale space is the fundamental theory of multi-scale signal representations, handling image structures at different scales by representing an image as a one-parameter family of smoothed images. The scale-space representation, parametrized by the size of the smoothing kernel, is used for suppressing fine scale structures [32].

The scales space representation is also the basis of scale invariant image retrieval. In the real world, objects may be of different sizes and the distance between the object and the camera is variable [15]. For a feature detections algorithm, it becomes quite necessary to be invariant to scales.

A side effect of the feature detection algorithm based on the scale space is that it is not robust to many geometric transformations. In practice, visual search appears quite sensitive to a change of view-point, which is analogous to a geometric linear transformation. Therefore, we will introduce an affine scale space representation which is specifically adapted for linear transformation. In that structure, not only the image, but also the smoothing kernel will be converted according to the transformation. Moreover, we will also introduce a practical technique for its implementation. Based on that implementation, the classical scale space will be interpreted as a special case of affine scale space.

3.1 Introduction to scale space

The scale space framework provides a basic representation tool for feature detection, allowing to extract features from multiple scales. The local derivative operator detects local edges and blobs at all the scales. Those selected blobs, with maximum response over all scales, will be selected as feature candidates [15].

Isotropic scale space is the fundamental representation tool for recent feature detection algorithms. It is easily applicable since many attractive properties derive from scale-space axioms. Based on these properties, some special construction methods have been designed to speed up the processing.

For a given image $I(x, y)$, its scale-space representation is given by a family of images $L(x, y, \sigma)$

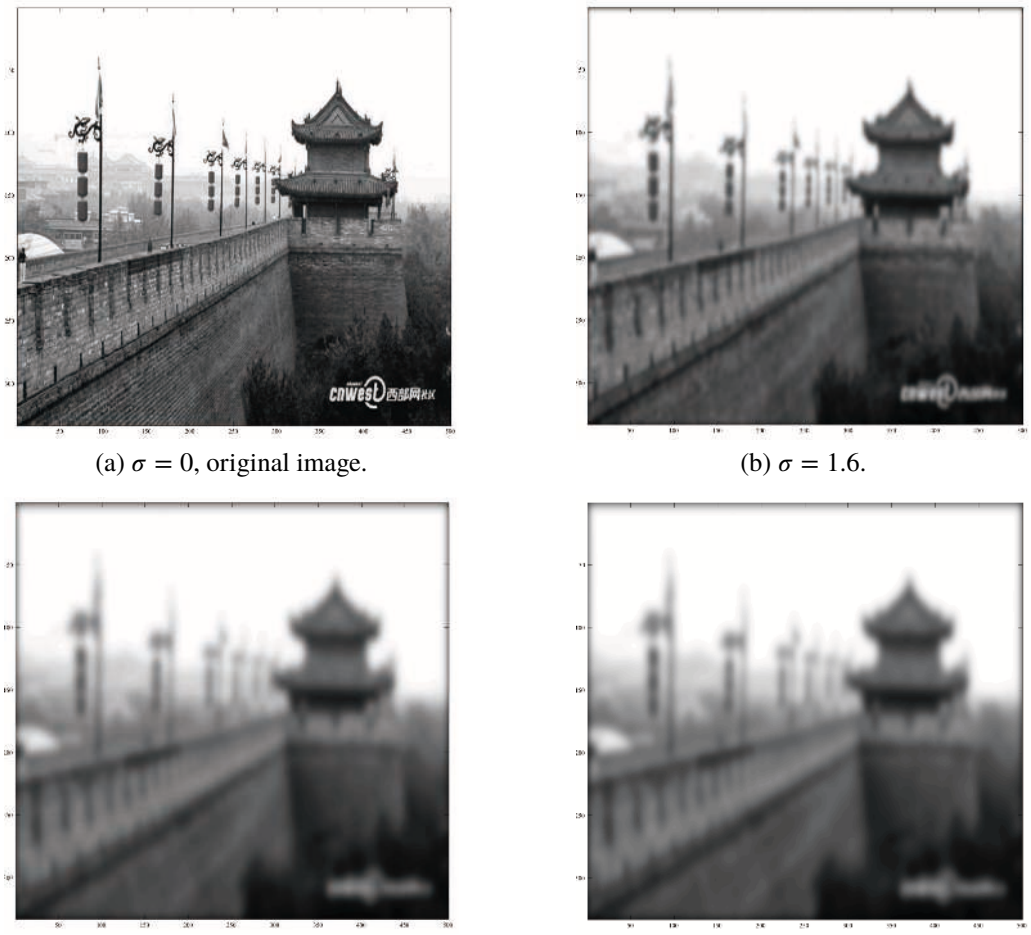
smoothed by a two dimensional Gaussian kernel, whose parameter is defined according to the kernel size:

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{3.1}$$

such that

$$L(x, y; \sigma) = g(x, y; \sigma) * I(x, y) \tag{3.2}$$

where the semicolon in the argument of L implies that the convolution is performed only over the variables x, y, while the scale parameter σ after the semicolon just indicates which scale level is being defined [33].



(a) $\sigma = 0$, original image. (b) $\sigma = 1.6$.

(c) $\sigma = 1.6 \times 2$. (d) $\sigma = 1.6 \times 2\sqrt{2}$.

Figure 3.1. A typical scale space, including the images smoothed with different size of Gaussian filter.

The scale parameter σ is the standard deviation of the Gaussian filter. When $\sigma = 0$ the filter g becomes an impulse function such that $L(x, y; 0) = I(x, y)$, that is, the scale-space representation at level 0 is the image itself. Figure 3.1 represents a typical scale space. We could notice that as the

increasing of σ, the corresponding image is smoothed with a larger Gaussian kernel and hence it contains fewer details.

The motivation for generating a scale-space representation of a image originates from the basic observation that real-world objects are composed of different structures at different scales. This implies that real-world objects, in contrast to idealized mathematical entities such as points or lines, may appear in different ways depending on the scale of observation. For example, the concept of a "tree" is appropriate at the scale of meters, while concepts such as leaves and molecules are more appropriate at finer scales. For a computer vision system analysing an unknown scene, there is no way to know a priori what scales are appropriate for describing the interesting structures in the image data. Hence, the only reasonable approach is to consider descriptions at multiple scales in order to be able to capture the unknown scale variations that may occur. Taken to the limit, a scale-space representation considers representations at all scales [15].

Gaussian filter is the most common choice for the scale space. Essentially, the smoothing filter should not introduce new spurious structures at coarse scales and not correspond to simplifications of structures at finer scales. This special requirement can only be satisfied by the filters with a smooth response from low pass band to high pass band and introducing no spurious structures at any scales. Based on that theory, Gaussian filter becomes a very good choice among all the low pass filters [34].

3.1.1 Feature detector

The scale space representation contains interesting image structures at all the scales. In order to capture the structure at the corresponding scale, an appropriate metric should be found. The local extrema in the scale space are a good choice for this metric. Indeed, derivative operations in the scale space could be applied for automatic scale selection to obtain the local maxima and minima citelindeberg1998feature.

At any scale in scale space, we can apply local derivative operators to the scale-space representation as

$$L_{x^m y^n}(x, y, \sigma) = (\partial_{x^m y^n} L)(x, y, \sigma). \tag{3.3}$$

Where $\partial_{x^m y^n}$ stands for the 'm'-th and 'n'-th derivative of the function respectively on the variable 'x' and 'y'.

Due to the commutative property between the derivative operator and the Gaussian smoothing operator, such scale-space derivatives can equivalently be computed by convolving the original image with Gaussian derivative operators.

$$L_{x^m y^n}(x, y, \sigma) = \partial_{x^m y^n} g(x, y, \sigma) * I(x, y). \tag{3.4}$$

In this way, the derivative of the scale space can be simplified as a convolution of the image with the corresponding derivative of Gaussian filter [13].

Following the idea of expressing visual operation in terms of differential invariants, we can obtain the local maximum and minimum applying the Laplacian operator on each smoothed image in the scale space structure.

The Laplacian operator is defined as,

$$\nabla^2 L = L_{xx} + L_{yy} \tag{3.5}$$

The approach of the Laplacian of Gaussian operator can also be expressed by means of differential between two Gaussian scale images.

$$\nabla^2_{norm} L(x, y, t) \approx \frac{t}{\Delta t}(G(x, y, t + \Delta t) - G(x, y, t - \Delta t)), t = \sigma^2 \tag{3.6}$$

This approach has been used in SIFT for feature detection [17].

3.1.2 Implementation of isotropic scale space

The implementation of a scale space is inspired by exploring the scale-space axioms and using the properties which can be used to speed up the processing.

The linear scale space representation $L(x, y, \sigma) = g(x, y, \sigma) * I(x, y)$ of signal $I(x, y)$ obtained by smoothing with the Gaussian kernel $g(x, y, \sigma)$ has a number of useful properties [11].

- Linearity
  $g(x, y; \sigma) * (aI_1 + bI_2) = ag(x, y; \sigma) * I_1 + bg(x, y; \sigma) * I_2$
  where I_1 and I_2 are signals while a and b are constants

- Shift invariance
  $g(x, y, \sigma) * I(x - \Delta x, y - \Delta y) = g(x - \Delta x, y - \Delta y, \sigma) * I(x - \Delta x, y - \Delta y)$

- Semi-group structure
  $g(x, y, \sigma_1^2) * g(x, y, \sigma_2^2) = g(x, y, \sigma_1^2 + \sigma_2^2)$
  with the associated cascade smoothing property
  $L(x, y, \sigma_1^2 + \sigma_2^2) = g(x, y, \sigma_1^2) * L(x, y, \sigma_2^2)$
  Since $L(x, y, \sigma_1^2 + \sigma_2^2) = g(x, y, \sigma_1^2 + \sigma_2^2) * I(x, y)$
  $L(x, y, \sigma_1^2 + \sigma_2^2) = g(x, y, \sigma_1^2) * g(x, y, \sigma_2^2) * I(x, y)$
  $L(x, y, \sigma_1^2 + \sigma_2^2) = g(x, y, \sigma_1^2) * L(x, y, \sigma_2^2)$

- Normalization
  $\int_{x=-\infty}^{\infty} \int_{y=-\infty}^{\infty} g(x, y, \sigma) dx dy = 1$

A practical structure for the scale space implementation is a pyramid, which allows to obtain a computationally efficient approximation to scale space. There are mainly two types of structure: Gaussian pyramid, an approach to the Gaussian scale space and Laplacian pyramid, an approach to the derivative of Gaussian scale space. For the pyramids, both the cascaded implementation of scaled Gaussian smooth but also sub-sampling would be applied to be efficiently construct the scale space.

Figure 3.2 illustrates a typical pyramid structure. Different from the normal scale space, the pyramid is constructed by sub-sampling the smoothed image to reduce the exponentially increased Gaussian kernel. In this way, the processing can be accelerated by reducing convolution size.

A common procedure to construct the Gaussian pyramid is as follows: the original image is convolved with a low-pass filter and sub-sampled by a factor of two; the filter-sub sample procedure is repeatedly operated to generate the smoothed image of each scale space. The traditional pyramid keeps the construction by recursively sub-sampling and smoothing with the same Gaussian filter. The new pyramid applies several Gaussian filters to separately generate the first few smoothed

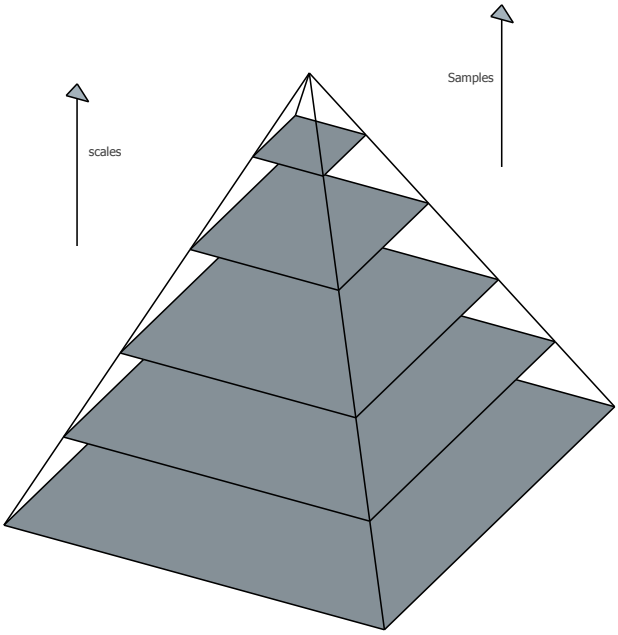


Figure 3.2. Pyramid structure.

images, than one of the image will be sub-sampled as the initial image for the next operation, next few images will be smoothed by the same filters. This smooth-sub sampling method will be recursively operated to until the whole scale space has been created.

- Semi-group structure
  $g(x, y, \sigma_1^2) * g(x, y, \sigma_2^2) = g(x, y, \sigma_1^2 + \sigma_2^2)$
  with the associated cascade smoothing property
  $L(x, y, \sigma_2^2) = g(x, y, \sigma_2^2 - \sigma_1^2) * L(x, y, \sigma_1^2)$

- Sub-sampling
  $L(2x,2y, \sigma^2) = I(2x,2y) * g(2x,2y, \sigma^2) = I(2x,2y) * g(x, y, \frac{\sigma^2}{2})$
  A sub-sampled blurred image is equal to a sub-sampled image blurred with a half-scale Gaussian filter

Figure 3.3 presents the structure of Gaussian pyramid. A smoothed image can be sub-sampled without losing information because Gaussian filter is a low pass filter, and sub-sampling only removes the high frequency spectrum. An image with its whole frequency spectrum cannot be sub-sampled without creating any aliasing. So the sub-sampling and filtering are well cooperated in the procedure of constructing the scale space.

Another type of pyramid, which is more practical and accurate, introduces a new structure called octave for the scale space construction. The octave is a structure containing several neighboring smoothed images of the same size. Within each octave, every smoothed image is obtained by convolving the previous smoothed image with an increasing sized Gaussian kernel. The initial

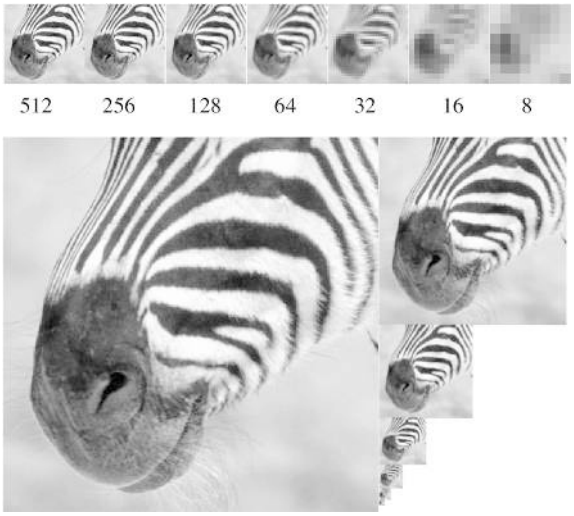


Figure 3.3. Pyramid images

smoothed image of each octave is the sub-sampling of the third image of the previous octave. Given the sub-sampled image, the other smoothed images of the octave can be generated by a series of longer Gaussian kernels. Based on this recursive operation, an efficient and accurate implementation can be obtained.

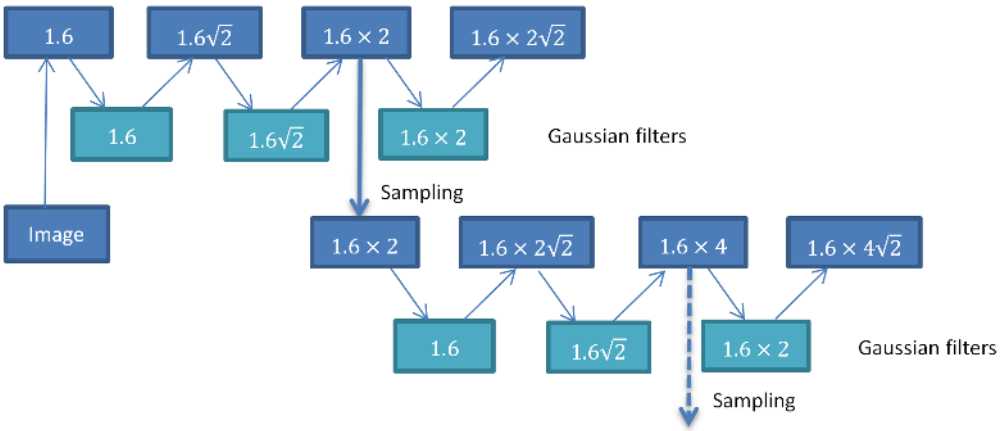


Figure 3.4. ALP pyramid structure.

Figure 3.4 presents the pyramid structure that has been used in ALP [24], an integral part of CDVS standard, we can notice that only three Gaussian filters have been used to construct the whole Gaussian pyramid. The third image of each octave will be sub-sampled as the initial image for the next octave. The next octave will also be generated by the same given filters. Finally, the whole pyramid will be constructed via this smoothing-sub sampling recursive operation.

3.2 Affine scale space

Scale-space theory provides a well-founded framework for modeling image structures at multiple scales, and the output from the scale-space representation can be used as input to a large variety of visual modules. On the other hand, most works on multi-scale representations focus on the definition of isotropic scale space, characterized by equal accessibility in all the directions [32]. But that kind of scale space is not compatible with non-isotropic image structures generated by a perspective affine transformation. A simple solution to this issue is an adaptation of isotropic scale space to account for geometry linear transformation. This new adaptation will be termed as "affine scale space".

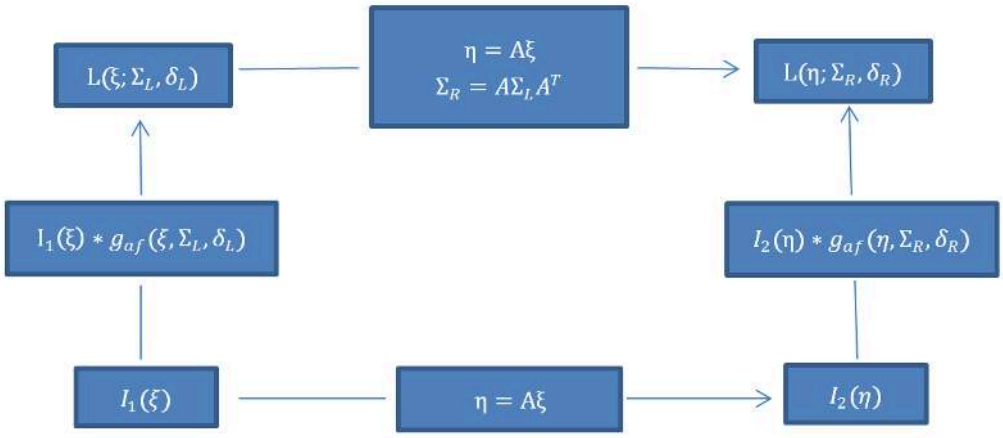


Figure 3.5. The equivalent relations of affine scale space

3.2.1 Affine Gaussian scale space

Let us first analyses the mathematical theory behind this new scale space. Assume that two images I_1 and I_2 are related by an affine transformation

$$I_1(\xi) = I_2(\eta) \text{ ,where } \eta = A\xi. \tag{3.7}$$

Both ξ and η are two dimensional vectors. The isotropic Gaussian scale space is defined by

$$L_1(x, y, \sigma) = g(x, y, \sigma) * I_1(x, y),\ L_2(x, y, \sigma) = g(x, y, \sigma) * I_2(x, y) \tag{3.8}$$

It is clear that

$$\begin{aligned} g(\xi; \sigma) * I_1(\xi) \neq & g(\eta; \sigma) * I_2(\eta), \\ g(\xi; \sigma) \neq & g(\eta; \sigma). \end{aligned} \tag{3.9}$$

Definitely, a linear Gaussian scale space cannot be described into the equation using an affine deformation. This inequality also implies that an isotropic scale space cannot capture the structure of an affine deformed image. A reasonable approach to handle this deformed structure is to generate

the corresponding scale space also by a deformed Gaussian kernel. Let us that suppose the new Gaussian kernel is denoted as g_{af}. So we have

$$\begin{aligned} g(\xi;\sigma) * I_1(\xi) &= g_{af}(\eta;\sigma) * I_2(\eta) \\ g(\xi;\sigma) &= g_{af}(\eta;\sigma) \end{aligned} \tag{3.10}$$

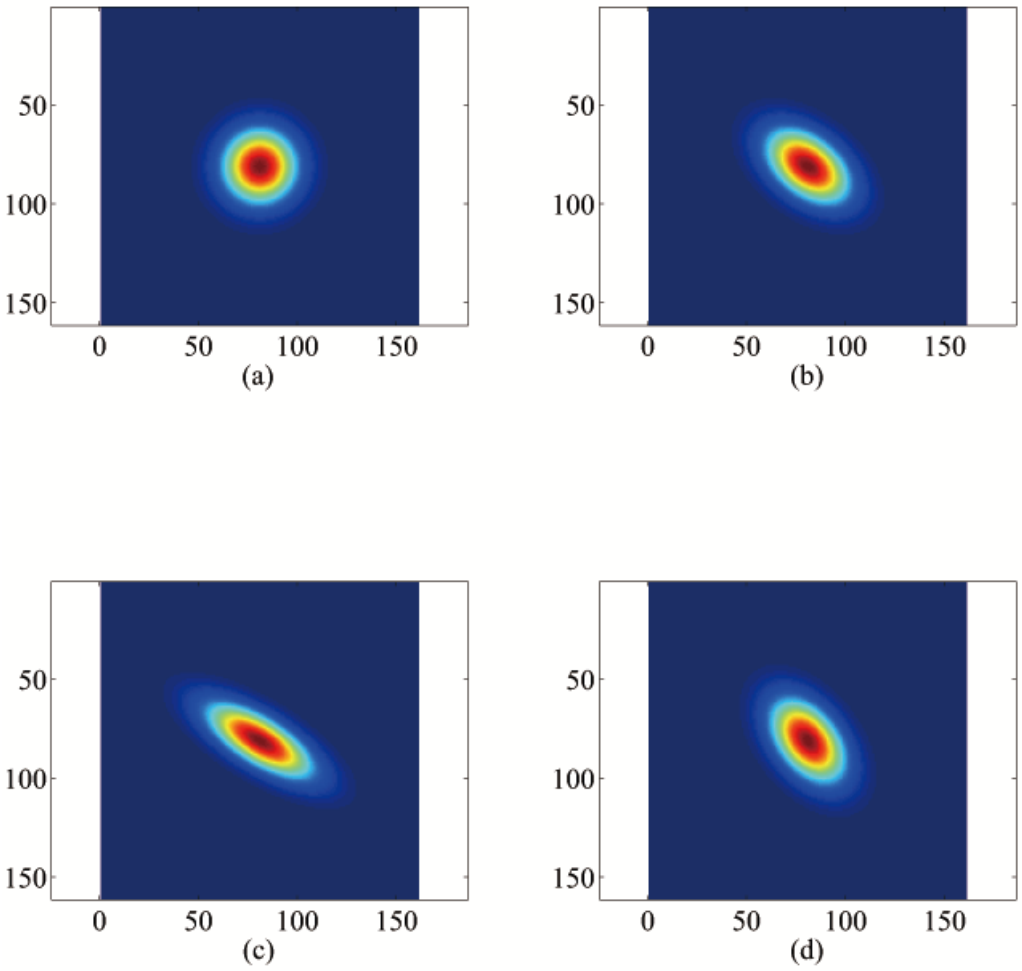


Figure 3.6. Examples of affine Gaussian kernel.

The Gaussian filter can be expressed as

$$g(\xi;\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{\xi^T * \xi}{2\sigma^2}}. \tag{3.11}$$

Since $\eta = A\xi$, then we have

$$\begin{aligned} \xi &= A^{-1}\eta \\ g(\xi;\sigma) &= \frac{1}{2\pi\sigma^2} e^{-\frac{(A^{-1}\eta)^T * (A^{-1}\eta)}{2\sigma^2}} \\ g(\eta;\sigma)_{af} &= \frac{1}{2\pi\sigma^2} e^{-\frac{\eta^T (AA^T)^{-1}\eta}{2\sigma^2}} \end{aligned} \tag{3.12}$$

With the definition of covariance matrices $\Sigma_s = A\sigma^2 A^T$ we obtain

$$g(\eta;\Sigma_s)_{af} = \frac{1}{2\pi\sqrt{det\Sigma_s}} e^{-\frac{\eta^T \Sigma_s^{-1}\eta}{2}}. \tag{3.13}$$

If $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, the linear scale space becomes a special case of affine scale space.

This is the affine Gaussian kernel extended from the linear Gaussian expression. Employing this kind of filters, we can obtain the corresponding affine Gaussian scale space. Regarding the application in image processing and computer vision, this means that image structure subject to affine transformations can be perfectly captured by the new affine scale space.

In practice, there are two different ways of computing scale space representations under affine alignment, namely by deforming the filter shapes or by deforming the image before Gaussian filtering. Figure 3.5 demonstrates the equivalent relations of the smoothed images of different perspectives, either by affine deforming the smoothed images or by convolving the affine deformed images with affine adapted Gaussian kernels.

Theoretically, these two approaches are mathematically equivalent. But in a practice, however, former based approach can be expected to be more accurate because it can be performed in the continuous-space domain.

Figure 3.6 presents some affine Gaussian kernel with different covariance matrix. The corresponding matrix is $\begin{bmatrix} 144 & 0 \\ 0 & 144 \end{bmatrix}$ for (a); $\begin{bmatrix} 180 & 72 \\ 72 & 144 \end{bmatrix}$ for (b); $\begin{bmatrix} 288 & 144 \\ 144 & 144 \end{bmatrix}$ for (c) and $\begin{bmatrix} 144 & 72 \\ 72 & 180 \end{bmatrix}$ for (d).

This representation satisfies all the scale space properties except those related to rotational symmetry.

3.2.2 Affine Laplacian of Gaussian

As we have discussed before, the local feature representation can be automatically extracted by the operation of Laplacian operator by means of differential derivation on a set of points at multiple scales. Following this idea, the derivative of the affine Gaussian scale space can also be obtained by adapting the corresponding Laplacian operator for the affine alignment.

In practical applications, Laplacian operator is usually simplified as a 3×3 Laplacian filter in spatial domain.

$$Lap = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{3.14}$$

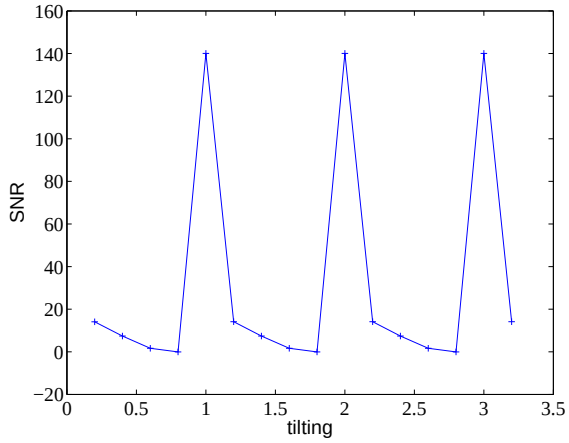


Figure 3.7. The performance of affine deformation on the Laplacian operator.

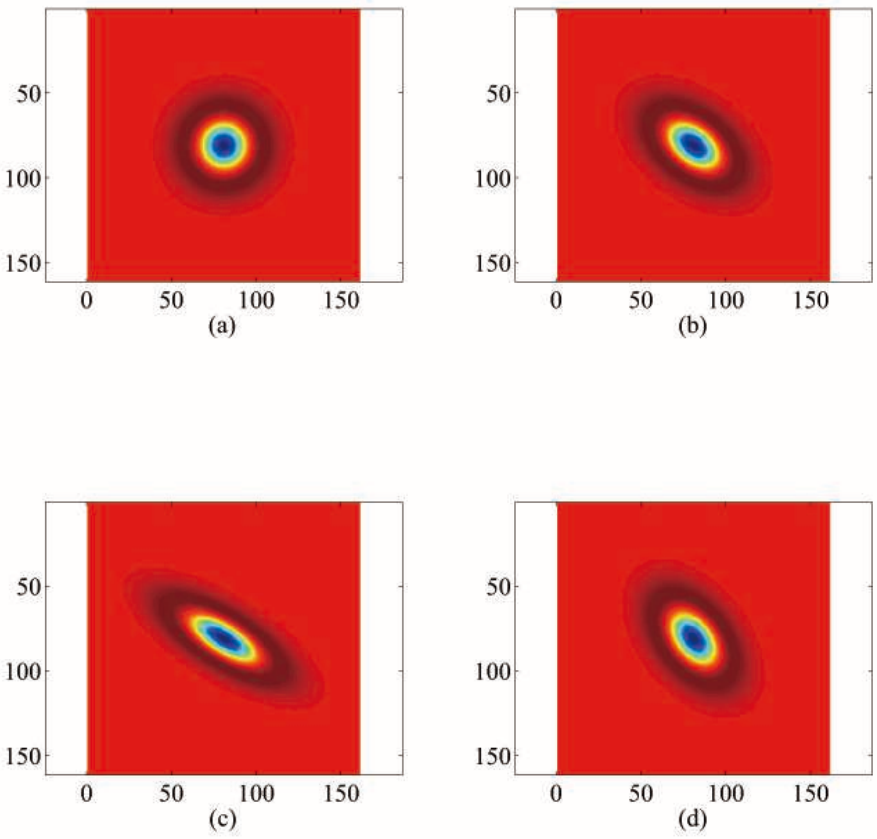


Figure 3.8. Examples of affine Laplacian of Gaussian kernel.

In ALP [24], the LoG scale space is obtained by the Laplacian operator convolved with the corresponding Gaussian scale space. As we have discussed before, the Gaussian kernel has been

adapted for the affine deformation.

A simple and straightforward idea about the affine adaptation on the LOG scale space is to transform the Laplacian operator for the corresponding affine alignment. Different from the Gaussian kernel which is originally a continuous function, the simplified Laplacian approach is based on a 3×3 matrix, which is too small to be manipulated for an affine deformation. Figure 3.7 presents the accuracy of the affine deformation, by Signal to Noise Ratio (SNR), comparing affine adaptation implementation either on Laplacian operator or on the Laplacian images. X axis reports a level of deformation and Y axis reports the SNR, which is used as the accuracy metric. In this thesis, all the following tests will employ SNR, which is calculated according to the scheme presented in the Figure 1.12. In practice, except some special cases, a lot of noise will be brought in for the interpolation on a 3×3 matrix.

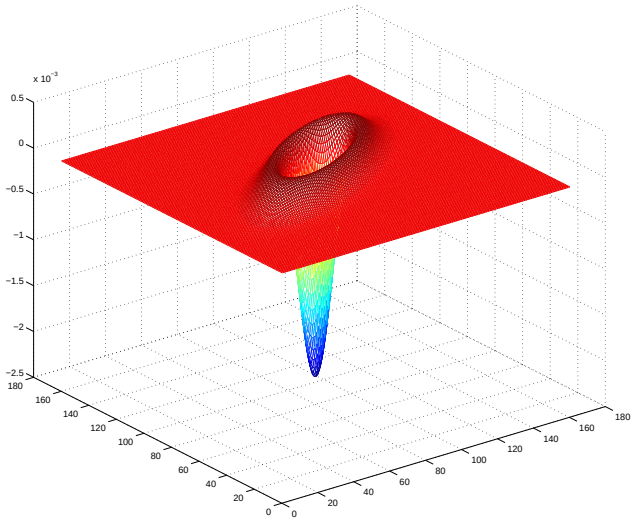


Figure 3.9. 3D figure of affine Laplacian of Gaussian kernel

Another way to approach the derivative of the Gaussian scale space is the Difference of Gaussian (DoG) which is based on the subtraction of the neighboring smoothed images. It has been applied in SIFT as the approach to the derivative of Gaussian scale space. The expression of DoG is:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma). \tag{3.15}$$

DoG can be linearly deformed to approach the derivative of affine scale space by the subtraction of the neighbouring affine-Gaussian smoothed images. But the selection of features on the DoG scale space also relies on their spatial neighbours, not only their scale neighbours. It is hard to gather the spatial neighbours since all pixel relations have been deformed by the affine transformation. In addition, the affine deformed DoG needs to be re-normalized. For this reason, we will mainly focus on the LoG scale space and its affine deformation.

The Laplacian operator in two dimensions is given by

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \tag{3.16}$$

The expression of LoG can be derived:

$$\begin{aligned} L_\Delta(x, y, \sigma) =& \nabla^2 g(x, y, \sigma) \\ =& \frac{\partial^2 g(x, y, \sigma)}{\partial x^2} + \frac{\partial^2 g(x, y, \sigma)}{\partial y^2} \\ =& -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2 + y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}} \end{aligned} \tag{3.17}$$

Another equivalent expression is given in the vector form,

$$L_\Delta(\xi; \sigma) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{\xi^T\xi}{2\sigma^2}\right) e^{-\frac{\xi^T\xi}{2\sigma^2}}, \tag{3.18}$$

$$\xi = \begin{pmatrix} x \\ y \end{pmatrix}.$$

The affine LoG filter can also be derived like affine Gaussian filter. Suppose an image deformation

$$I_1(\xi) = I_2(\eta), \text{ where } \eta = A\xi. \tag{3.19}$$

Since $\eta = A\xi$, then

$$\xi = A^{-1}\eta \tag{3.20}$$

$$\begin{aligned} L_\Delta(\eta; \sigma) =& -\frac{1}{\pi\sigma^4}\left(1 - \frac{(A^{-1}\eta)^T(A^{-1}\eta)}{2\sigma^2}\right) e^{-\frac{(A^{-1}\eta)^T(A^{-1}\eta)}{2\sigma^2}} \\ =& -\frac{1}{\pi\sigma^4}\left(1 - \frac{\eta^T(AA^T)^{-1}\eta}{2\sigma^2}\right) e^{-\frac{\eta^T(AA^T)^{-1}\eta}{2\sigma^2}} \end{aligned} \tag{3.21}$$

with the definition of covariance matrices $\Sigma_s = A\sigma^2 A^T$ we have

$$L_\Delta(\eta; \sigma) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{\eta^T\Sigma_s^{-1}\eta}{2}\right) e^{-\frac{\eta^T\Sigma_s^{-1}\eta}{2}}. \tag{3.22}$$

This is the affine Laplacian of Gaussian kernel. It has a similar expression as the affine Gaussian kernel. So the implementation is also similar. Comparing Figure 3.6 with Figure 3.8, we can find the shape of these two affine alignments are quite similar. Figure 3.8 displays some LoG kernels with different covariance matrix. The corresponding matrix is $\begin{bmatrix} 144 & 0 \\ 0 & 144 \end{bmatrix}$ for (a); $\begin{bmatrix} 180 & 72 \\ 72 & 144 \end{bmatrix}$ for (b); $\begin{bmatrix} 288 & 144 \\ 144 & 144 \end{bmatrix}$ for (c) and $\begin{bmatrix} 144 & 72 \\ 72 & 180 \end{bmatrix}$ for (d).

3.2.3 Accuracy of affine scale space

The assessment of affine Gaussian and affine Laplacian of Gaussian scale space is essential and necessary for practical application. Only an accurate implementation of the affine adapted scale space can guarantee the precision of the feature detection for images with different perspectives.

For this assessment, two small patches describing the same scene but with different perspective projection will be used to measure the accuracy of the scale spaces, one generated by the affine deformed kernel and one by isotropic kernel. The result is very important for the applicability assessment of affine scale space. The result will also help to evaluate for which deformations and scales, the affine scale space will perform better than the isotropic one for the image matching.

In this assessment, rotation will also be used to evaluate the noise level in the affine scale spaces, since the amount of noise brought in by rotation in isotropic scale space is acceptable. Here, the noise refers to the small errors due to the imprecise pixel value estimation because of the transformed non-integer coordinates. If the noise quantity brought in by rotation and affine deformation are of the same level, the affine scale space can be considered equally robust to both.

In the affine scale space assessment, the noise level will be separately evaluated for different deformation and scales level. In our test, the noise level will be defined by SNR, defined as the ratio of signal power to the noise power, normally expressed in decibels. It can be expressed as:

$$SNR_{dB} = 10 \log_{10} \left[\left(\frac{A_{signal}}{A_{noise}} \right)^2 \right] \tag{3.23}$$

A_{signal} and A_{noise} respectively denote the magnitude of signal and noise.

In the figure 3.10, (a) illustrates the performance on affine Gaussian scale space: the curve reports SNR versus scales and deformation. The SNR of the Gaussian scale space ranges from 30dB to 42dB, which is normally acceptable for the image quality. There is a coherent growth trend of SNR over the scales but little fluctuation over different deformation. We will show later that these SNR levels are suitable for image matching. In the figure 3.10, (b) displays the curves at different scales with the affine deformation $A = \begin{bmatrix} 1 & 0 \\ 0.25 & 0 \end{bmatrix}$. There is a uniform growth of SNR as scale level increase. Initially, SNR is significantly increased at small scales. Then, the precision gradually stays constant when the scale is larger than 5. In the figure, (c) illustrates two curves of SNR for different affine deformations. The affine Gaussian kernel results in a better performance at the scale 6.4 than 1.6. Overall speaking, SNR is not correlated with a specific affine deformation, as it is mainly due to interpolation.

To explain the SNR versus scale, we note that interpolation is inevitable for an affine image transformation, and this will introduce noise. A large scale of the Gaussian filter translates to a narrow band low pass filter and will reduce interpolation noise. This can explain why the larger the scale is, the less noise will appear on the corresponding smoothed images. Conversely, interpolation noise will be less if the deformation can be largely constructed from known points. One special case of interpolation is represented by the integer levels deformation,e.g. $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. In this case, since the deformation can be fully constructed within the range of given points and no interpolation is required, less or no noise will be expected after the transformation. In figure 3.10, (d) provides the SNR versus scale curve with the affine deformation $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. As expected, the deformation

can be completely constructed by the given set of points and no interpolation is applied. The SNR appears quite large compared with the image deformation of the same level. It ranges from 100dB to 125dB, while the SNR of a similar deformation can only range from 30dB to 40dB. This very special case helps to explain the main source of the noise.

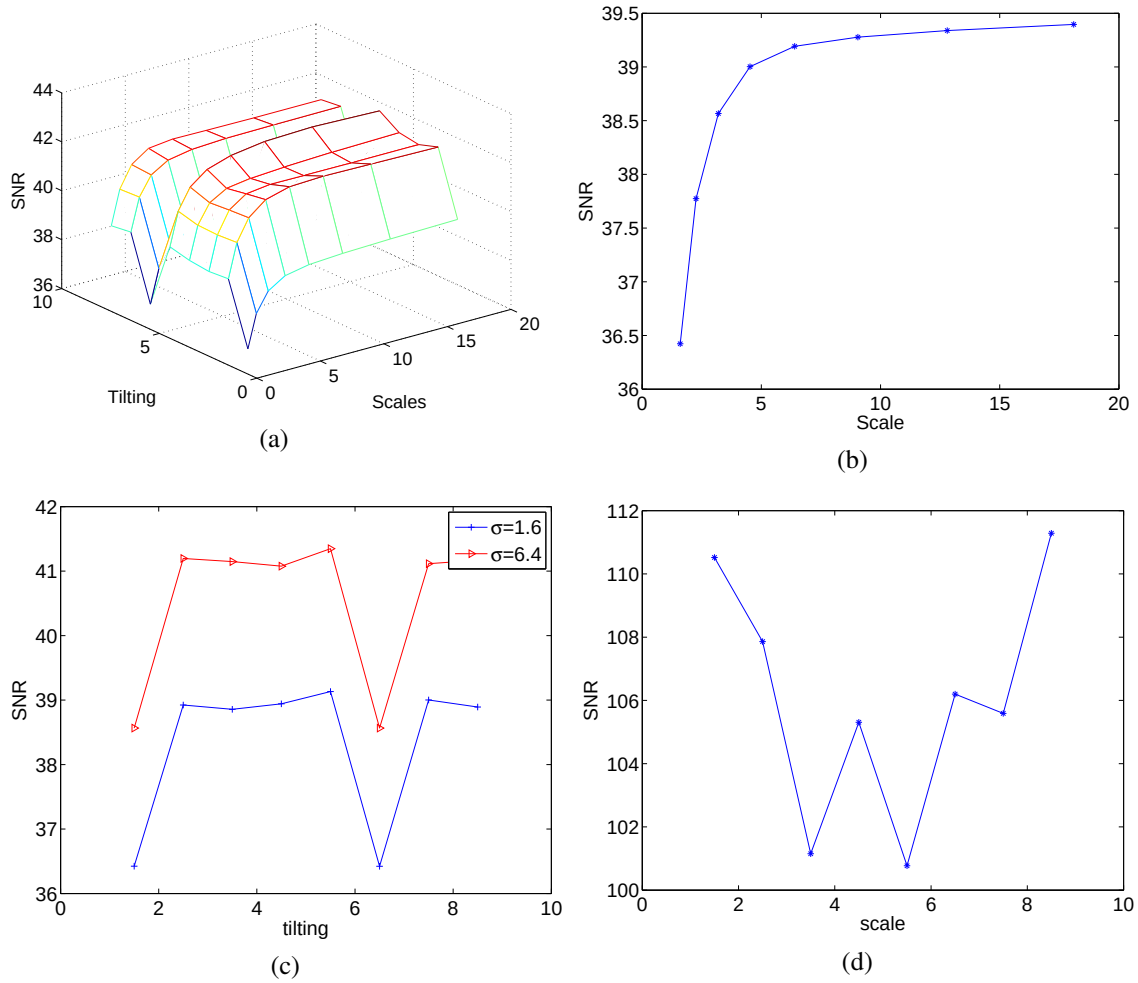


Figure 3.10. Performance of affine scale space. (a) SNR versus Tilting and Scales. (b) SNR versus Scales. (c) SNR versus deformation. (d) A special case with an integer deformation.

Affine LoG has a similiar performance to the affine transformation. In the figure 3.11, (a) illustrates the performance on affine LoG: the curve reports SNR versus scales and deformation. The SNR of the affine LoG ranges from 25dB to 40dB, which is a bit smaller than the SNR of affine scale space. (b) displays the curves at different scales with the affine deformation $A = \begin{bmatrix} 1 & 0 \\ 0.25 & 0 \end{bmatrix}$. There is a uniform growth of SNR as scale level increase. Initially, SNR is significantly increased at small scales. Then, the precision gradually stays constant when the scale is larger than 5. In the figure, (c) illustrates two curves of SNR for different affine deformations. The affine Gaussian kernel results in a better performance at the scale 6.4 than 1.6. The affection of SNR is mainly due to

image interpolation. (d) also illustrates a special case, which provides the SNR versus scale curve with the affine deformation $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. The deformation can be completely constructed by the given set of points and no interpolation is needed. It ranges from 62dB to 73dB. This special case helps to find the origin of the noise for the affine LoG.

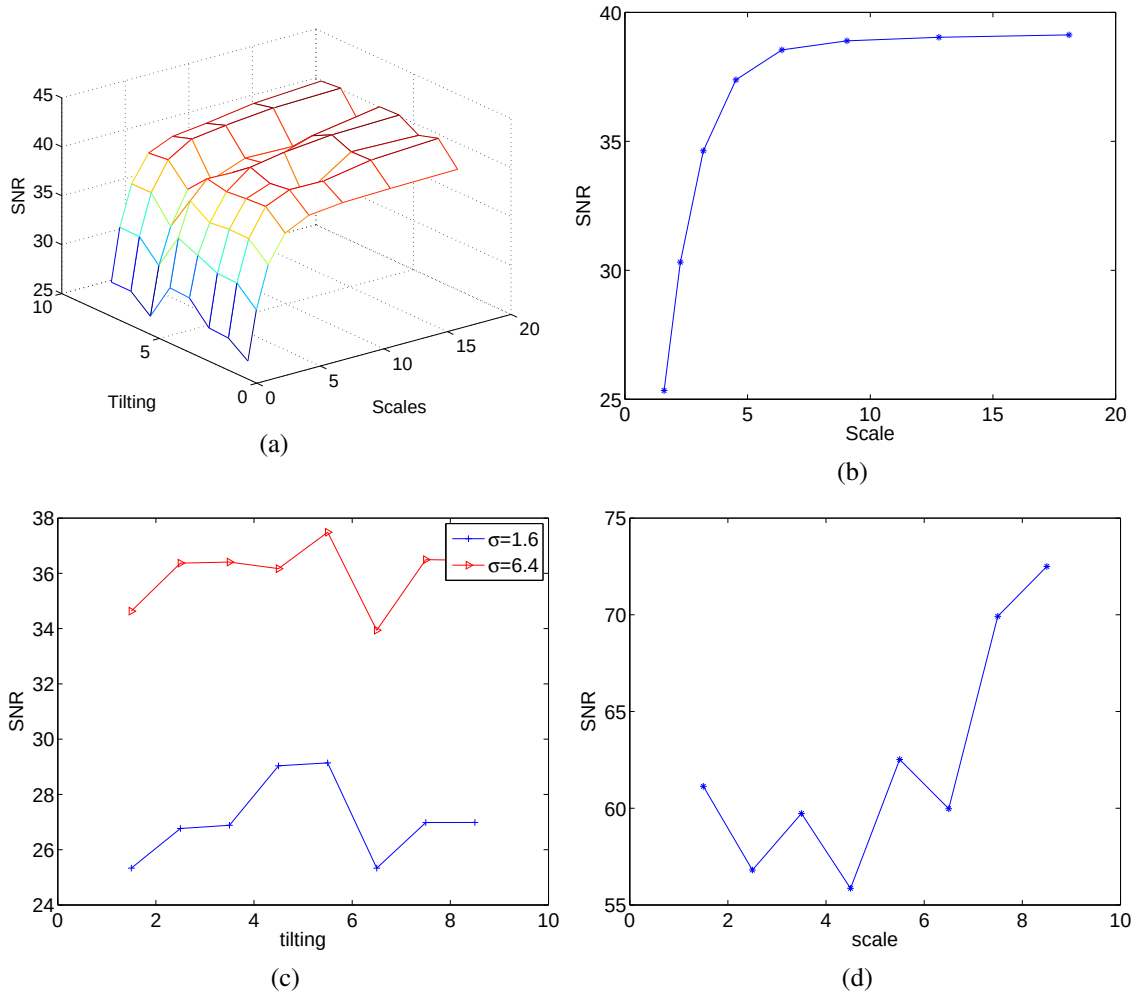


Figure 3.11. Performance on Laplacian of Gaussian. (a) SNR versus scales and deformation. (b) SNR versus scales. (c) SNR versus deformations. (d) SNR versus scales with a integer deformation.

Given the noise level of an affine Gaussian scale space and affine LoG, a further analysis is needed to confirm the amount of noise acceptable for image retrieval. A reasonable approach for this analysis is to compare the noise level of an affine deformation with that of a pure rotation. As we known, image retrieval on isotropic scale space is rotation invariant. Noise will also be introduced by interpolation applied in the rotation. For this reason, if the noise level in rotation is acceptable, so is the noise level in affine deformation.

As in the case of the experiments in affine deformation, a special case involving no interpolation

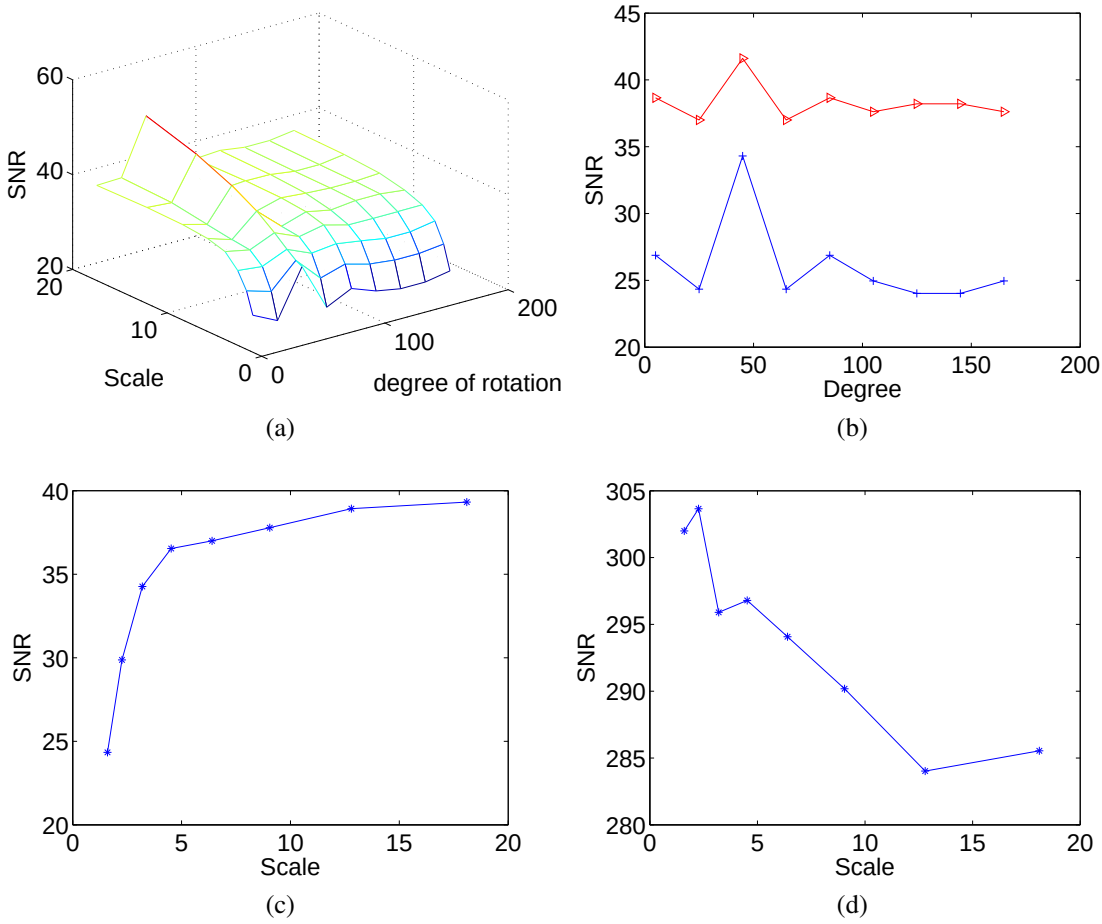


Figure 3.12. Performance on rotation invariance. (a) SNR versus scales and rotation. (b) SNR versus rotation. (c) SNR versus scales with 85° of rotation. (d) SNR versus scales with 90° of roation

.

will also be shown afterwards. In this way, we can also identify the main noise source for the image rotation.

Figure 3.12 shows the performance of the scale space for rotation. The SNR of the affine Gaussian deformation and isotropic rotation is quite similar and in most cases it also ranges from 30dB to 40dB. A uniform increase of SNR with the scale can also be found in Figure (c). Figure (d) reveals that the main noise source for rotation in isotropic scale spaces is also interpolation. Since the noise level from rotation on the isotropic scale space does not affect feature extraction, the noise level on the affine deformation is also acceptable for feature extraction from an affine projected image.

3.3 Structure to construct the affine scale space

As we have discussed before, a pyramid structure can be used to efficiently implement the scale space. It is necessary to create such a computationally efficient structure to build the scale space especially for large images, since the construction of the scale space is computationally complex and time consuming. If a similar structure can be designed for handling both affine Gaussian and LoG scale space, it will be very appealing in practice. Therefore we first investigate whether required properties for the pyramid construction are satisfied by affine Gaussian and LoG.

Basically, two properties are important for the pyramid structure: semi-group and sub-sampling. We need to recall these two properties.

$$\begin{aligned} g(x,y,\sigma_1^2) * g(x,y,\sigma_2^2) =& g(x,y,\sigma_1^2+\sigma_2^2) \\ \text{with } \xi =& \begin{pmatrix} x \\ y \end{pmatrix} \\ g(\xi,\sigma_1^2) * g(\xi,\sigma_2^2) =& g(\xi,\sigma_1^2+\sigma_2^2) \\ \eta = A\xi \quad \text{so} \quad & \xi = A^{-1}\eta \\ g(A^{-1}\eta,\sigma_1^2) * g(A^{-1}\eta,\sigma_2^2) =& g(A^{-1}\eta,\sigma_1^2+\sigma_2^2) \end{aligned} \tag{3.24}$$

So, an affine Gaussian scale space also satisfies the semi-group property.

It can also be seen that,

$$L_\Delta(x,y,\sigma_1^2) = \frac{\partial^2 g(x,y,\sigma_1^2)}{\partial x^2} + \frac{\partial^2 g(x,y,\sigma_1^2)}{\partial y^2} \tag{3.25}$$

$$\begin{aligned} L_\Delta(x,y,\sigma_1^2) * g(x,y,\sigma_2^2) =& \frac{\partial^2 g(x,y,\sigma_1^2)}{\partial x^2} * g(x,y,\sigma_2^2) + \frac{\partial^2 g(x,y,\sigma_1^2)}{\partial y^2} * g(x,y,\sigma_2^2) \\ L_\Delta(x,y,\sigma_1^2) * g(x,y,\sigma_2^2) =& \frac{\partial^2 g(x,y,\sigma_1^2+\sigma_2^2)}{\partial x^2} + \frac{\partial^2 g(x,y,\sigma_1^2+\sigma_2^2)}{\partial y^2} \end{aligned} \tag{3.26}$$

$$L_\Delta(x,y,\sigma_1^2) * g(x,y,\sigma_2^2) = L_\Delta(x,y,\sigma_1^2+\sigma_2^2) \tag{3.27}$$

This cascade implementation of LoG is a bit different as large scale LoG operation cannot be replaced as a convolution of two small scale LoG operations, but instead as a convolution of one Gaussian operation and one LoG operation. The corresponding scale is the summation of the Gaussian and LoG results. This property enables the construction of LoG generated by cascaded Gaussian smoothing and LoG operation.

In the same way

$$L_\Delta(A^{-1}\eta,\sigma_1^2) * g(A^{-1}\eta,\sigma_2^2) = L_\Delta(A^{-1}\eta,\sigma_1^2+\sigma_2^2) \tag{3.28}$$

and hence affine LoG has the same property.

This cascade implementation of LoG has not drawn much attention before. In a practical application, the LoG scale space can be easily approximated by DoG or a Laplacian operation on the

corresponding Gaussian scale space. Generally, it is more efficient to implement the LoG by Laplacian operation than by cascade implementation, but the Laplacian operation is unable to guarantee the precision of the affine adaptation. In comparison, the affine adapted LoG is easier to be achieved by a cascade of affine adaptable LoG filters. Based on the LoG filters, the cascade implementation can be more efficient and robust to construct the LoG.

Another important property for the Gaussian pyramid is sub-sampling.

Suppose the image can be sub-sampled given a scale space:

$$L(x, y, \sigma^2) = I(x, y) * g(x, y, \sigma^2) = I(x, y) * \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.29}$$

by sub-sampling, one obtains

$$\begin{aligned} L(2x,2y, \sigma^2) =& 4I(2x,2y) * g(2x,2y, \sigma^2) \\ =& 4I(2x,2y) * \frac{1}{2\pi\sigma^2} e^{-\frac{4x^2+4y^2}{2\sigma^2}} \\ =& 4I(2x,2y) * \frac{\frac{1}{4}}{2\pi\frac{\sigma^2}{4}} e^{-\frac{x^2+y^2}{2\frac{\sigma^2}{4}}} \\ =& I(2x,2y) * g(x, y, \left(\frac{\sigma}{2}\right)^2) \end{aligned} \tag{3.30}$$

or

$$\begin{aligned} L(2x,2y, (2\sigma)^2) =& I(2x,2y) * g(x, y, \sigma^2) \\ L(4x,4y, (4\sigma)^2) =& I(4x,4y) * g(x, y, \sigma^2) \end{aligned} \tag{3.31}$$

$$\vdots$$

In this way, the whole scale space can be generated by one filter through sub-sampling. If sub-sampling and cascade implementation are combined, we have:

$$\begin{aligned} & I(2x,2y) * g(x, y, \sigma_1^2) * g(x, y, \sigma_2^2) \\ =& I(2x,2y) * g(x, y, \sigma_1^2 + \sigma_2^2) \\ =& L(2x,2y,2(\sigma_1^2 + \sigma_2^2)) \end{aligned} \tag{3.32}$$

If the smoothed image of the scale space is sub-sampled we have

$$\begin{aligned} & L(2x,2y, \sigma_1^2) * g(x, y, \sigma_2^2) \\ =& I(2x,2y) * g(x, y, \left(\frac{\sigma_1}{2}\right)^2) * g(x, y, \sigma_2^2) \\ =& L(2x,2y, \sigma_1^2 + 2\sigma_2^2) \end{aligned} \tag{3.33}$$

The sub-sampling can also be applied during the cascade implementation. Letting $\xi = \begin{pmatrix} x \\ y \end{pmatrix}$

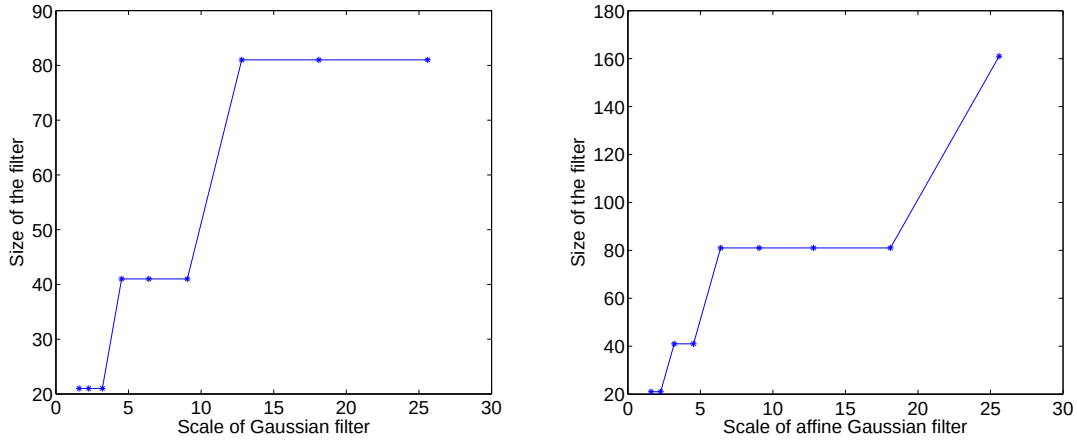
we have,

$$\begin{aligned} L(\xi, \sigma^2) =& I(\xi) * g(\xi, \sigma^2) \\ L(2\xi, (2\sigma)^2) =& I(2\xi) * g(\xi, \sigma^2) \\ \eta = A\xi \quad \text{so} \quad \xi =& A^{-1}\eta \\ L_1(A^{-1}2\eta, (2\sigma)^2) =& I_1(A^{-1}2\eta) * g(A^{-1}\eta, \sigma^2) \\ L_2(2\eta, (2\sigma)^2) =& I_2(2\eta) * g_{af}(\eta, \sigma^2) \end{aligned} \tag{3.34}$$

This shows that sub-sampling is also a property of the affine Gaussian scale space.

All the calculations above helps to prepare the mathematical foundation to design an affine Gaussian pyramid that is computational efficient. Due to the large number of candidate images for feature detections, a fast implementation is very desirable. On the other hand, the filter size increases exponentially with the scale, and hence the computations for convolution also grow exponentially, which is not practical in a real time application. That is why a pyramid structure becomes practical and applicable to approximate the scale space.

Even though the mathematical theory behind this implementation has been described above, a further analysis about its performance is necessary, in order to assess the effect of interpolation noise and computational efficiency.



(a) (b)

Figure 3.13. The effect of the filter size. (a) The size of Gaussian filters of different scale. (b) The size of affine Gaussian filters of different scale.

According to Figure 3.13, the size of the Gaussian filter keeps growing with the scale. The convolution complexity is related to both image size and filter size. Suppose the image size is $M \times M$ and the filter size is $N \times N$. The complexity for the convolution is $O(M^2N^2)$, which is very large.

On the contrary, if the scale space is built completely based on sub-sampling, at the next scale, the filter will keep the same size but the image size will be halved. Therefore, the computation complexity will not increase but drop.

3.3.1 Cascade implementation on affine scale space

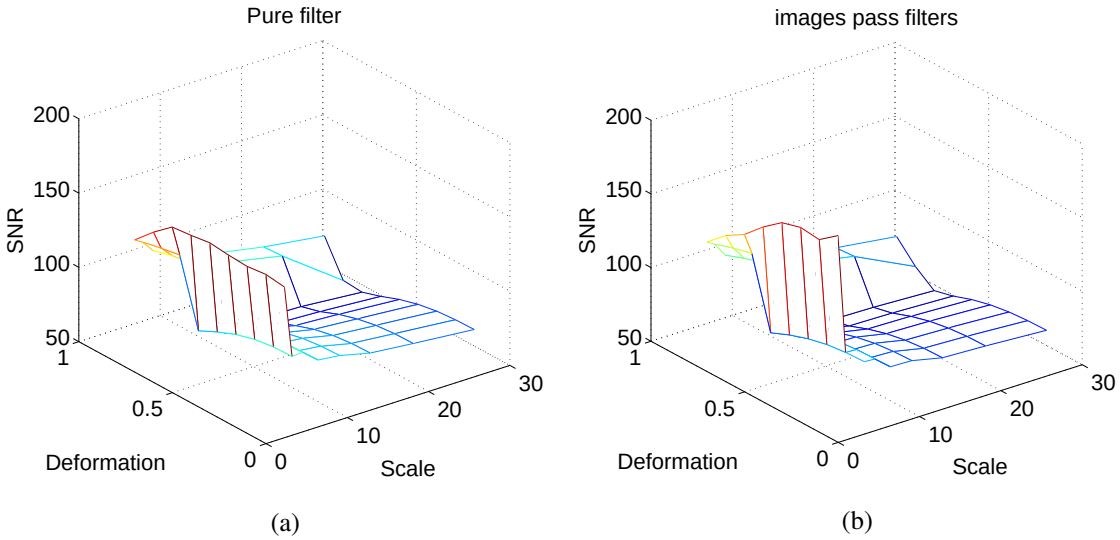


Figure 3.14. The performance of cascade filters. (a) SNR of cascade filters compared to ordinary filters. (b) Cascade filters compared to images by ordinary filters.

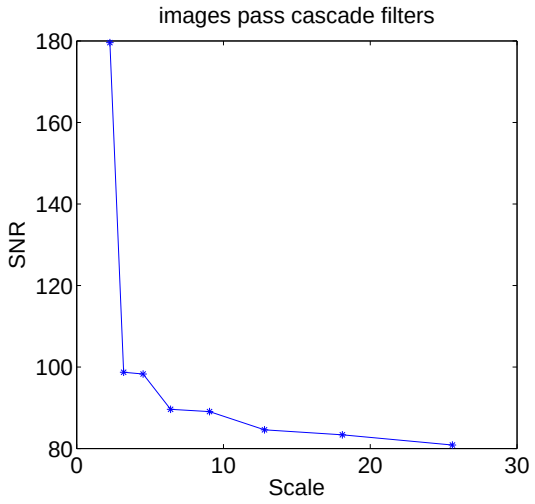


Figure 3.15. A special case of affine Gaussian scale space without interpolation.

Figure 3.14 presents the performance of affine cascade implementation by SNR, which compares the general and cascade implementation based affine scale space, ranging from 70dB to 160dB for the filter and 65dB to 120dB for the smoothed images. Overall speaking, the result implies that SNR is large enough to make the cascade implementation practical. Additionally, the difference among affine alignment is not obvious and the corresponding precision is also comparable with the non-deformed scale space. In the experiment, the scale space contains 8 scales, based

on 8 steps of cascade implementation.

In Figure 3.15, the SNR ranges from 98dB to 78dB, decreasing after each stage of cascade implementation.

Even though the decrease of SNR after each stages not constant, on average each stage only causes a drop around 2 ∼ 3dB. Considering the very good accuracy of affine Gaussian scale space at high scales, the result is good enough for the affine deformed cascade implementation.

As we have proved, a LoG operation can be separated as a convolution of a smaller scaled LoG operation and Gaussian operation. The Gaussian operation can also be approximated by several cascade Gaussian operations. A LoG operation can also be approximated as the convolution of one LoG operation and several cascade Gaussian operations, which is depicted in the figure 3.16.

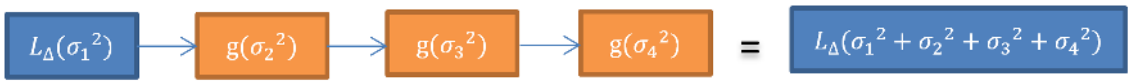


Figure 3.16. Cascade structure of Laplacian of Gaussian filter.

This combined cascade implementation of Gaussian and LoG provide us a simple way to approximate the LoG operation. The performance of this cascade combination is similar to the cascade Gaussian implementation (shown in Figure 3.17).

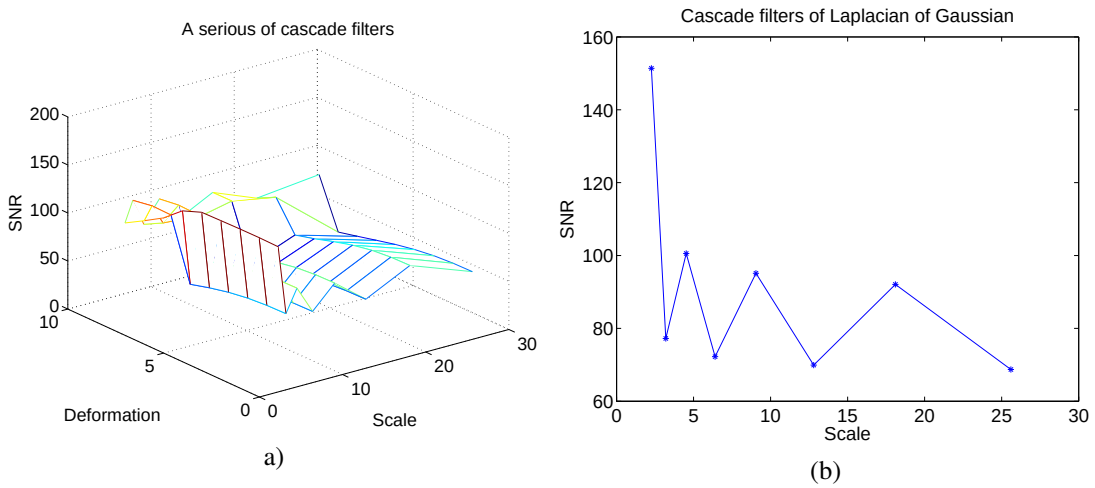


Figure 3.17. Cascade filters of affine Laplacian of Gaussian. (a) A cascade filters approximation to Laplacian of Gaussian of multiple deformations. (b) Cascade filters of affine Laplacian of Gaussian.

Apart from a LoG filter, the rest of this structure is exactly the same as a cascade Gaussian implementation. So a reasonable method to further simplify the process of constructing the scale space is to make the kernel of LoG filter as small as possible. We have another test to estimate the minimum parameter we need to leave to the Laplacian part. As the Figures 3.18 shows, a reasonable scale parameter to the Laplacian part should be larger than 1. When the parameter of scale on Laplacian part is large enough, it can guarantee the interpolation noise is small enough to not

affect the cascade performance.

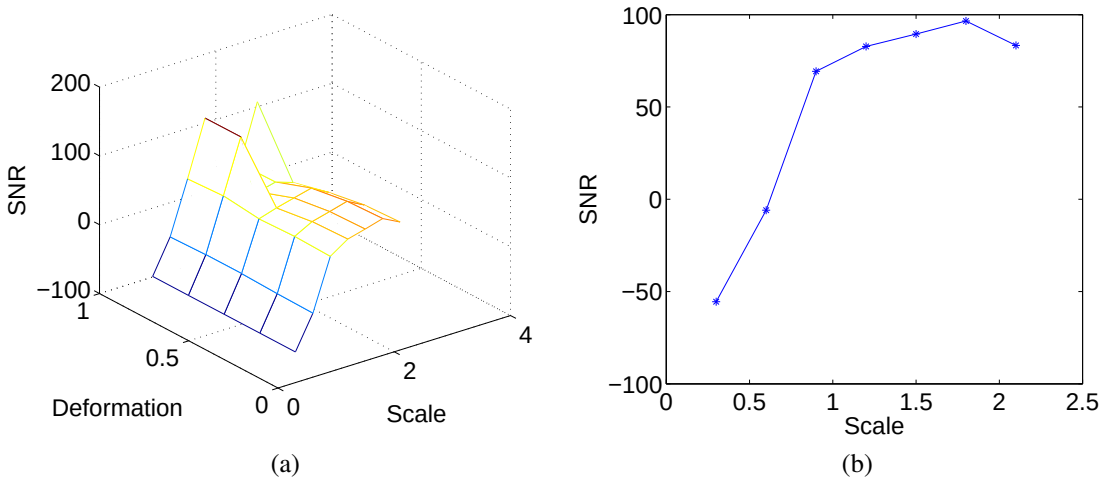


Figure 3.18. The performance of the affine cascade LoG. (a) Multiple deformation with different scales left to the Laplacian of Gaussian filter. (b) The performance with different Laplacian of Gaussian scales to the first filter.

Just as we have discussed, the performance of the cascade implementation, operated on LoG scale space, is good enough for the real application. An affine LoG structure is based on the corresponding Gaussian cascade implementation. A good method is to take the LoG as the first operation and the rest of the structure will be generated by the corresponding Gaussian operations.

The construction of the LoG scale space based on semi-group property will be performed as it presented in the figure 3.19.

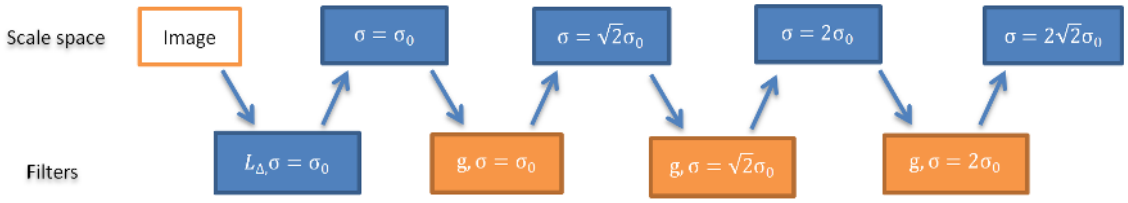


Figure 3.19. Structure to build the Laplacian of Gaussian scale space on cascade filters.

In this way, the structure to implement LoG is the same as the structure to implement Gaussian scale space. Instead of having all the spaces generated by Gaussian or affine Gaussian filters, the first LoG space will be generated by LoG filter and the rest of space will be obtained via Gaussian filtering from the first one.

Up to now, we have developed a simple but efficient structure to construct both the affine Gaussian and affine LoG scale space based on cascade implementation. The structures of the affine Gaussian and affine LoG are quite similar, only differing in the first filter. This structure will be quite simple but efficient for the implementation, especially when the total scale space is not large.

As we have discussed, following the growth of the scale, the size of the corresponding Gaussian filters increase exponentially and the computational complexity is proportional to the square of the size. The cascade method effectively squeezes the size of affine Gaussian and affine LoG filters which are used to generate the corresponding scale spaces. Based on this method, the computational complexity is acceptable for a real time application.

3.3.2 Sub-sampling on affine scale space

Another important property for efficient construction of the scale space, especially for the implementation in the pyramid structure, is sub-sampling. It successfully reduces the image size without much information losts due to the low pass Gaussian filter. But LoG filter is a band-pass not low pass filters. The image generated by LoG filter cannot be sub-sampled and cannot be approximated directly by a pyramid. For an isotropic scale space, LoG is simply generated by Laplacian operating on the same parametrized Gaussian scale space. There is no need to explore the LoG pyramid. But the affine LoG cannot be generated directly from an affine Gaussian scale space. Affine Gaussian scale space and affine LoG must be created separately with different structures. At the same time the images in LoG cannot be directly sub-sampled. On the other hand, the cascade implementation for the affine LoG provides us a possible solution. The images of Gaussian part are still amenable to sub-sampling. Based on the sub-sampled smoothed images, the computation of both Gaussian part and LoG part can be simplified.

The performance of the sub-sampling on the affine Gaussian scale space is evaluated in Figures 3.20 ∼ 3.21.

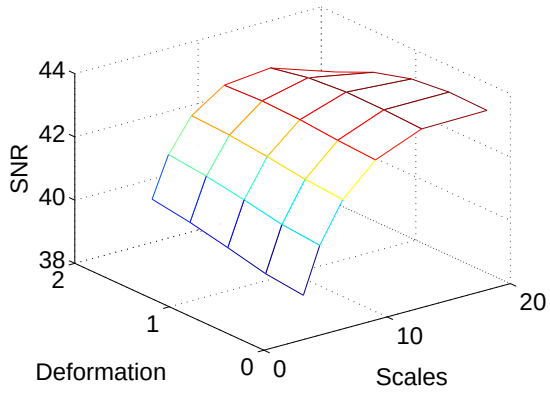


Figure 3.20. Effect of sub-sampling on affine Gaussian scale spaces of multiple deformation and scales.

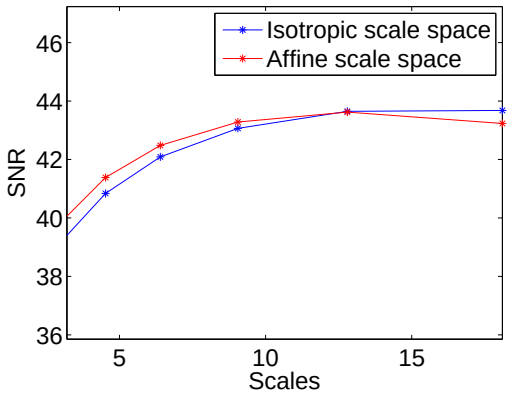


Figure 3.21. Comparison of sub-sampling performance on isotropic scale space and affine Gaussian scale space.

According to the sub-sampling property of a Gaussian kernel, sub-sampling a Gaussian blurred image is equivalent to blurring a sub-sampled image by a Gaussian kernel, with the scale divided by the sampling rate. Figure 3.20 contains the SNR obtained comparing this equivalence at different scales and affine alignment. It performs similarly for sub-sampling at different affine deformed scale spaces. At some scales, the performance of affine Gaussian is even better than an isotropic scale

spaces. Figure 3.21 compares the sub-sampling on isotropic scale space and on a typical affine scale space. The difference between these two curves is rather small.

Sub-sampling on affine scale space is quite reliable comparing its performance on both isotropic and affine scale space. But an automatic extraction of features relies on the derivative of the scale space rather than the scale space itself. In practice, it would be more efficient and reliable if sub-sampling could be used to generate LoG. While the images on LoG are band-pass, the fundamental requirement of sub-sampling has been undermined. Considering the relations between Gaussian and LoG operations, it is possible to obtain the LoG by employing sub-sampling only on the Gaussian part.

According to the structure we have designed in Figure 3.19, there will be no low-pass images to be sub-sampled during the generation of LoG. Since the Gaussian scale space and LoG are generated separately for the affine alignment, the low-pass image cannot be directly obtained in the LoG. To apply the filters recursively, we need to design the structure containing the corresponding Gaussian images. There are two possible ways to generate such images. We will display both these structures and assess the feasibility and accuracy.

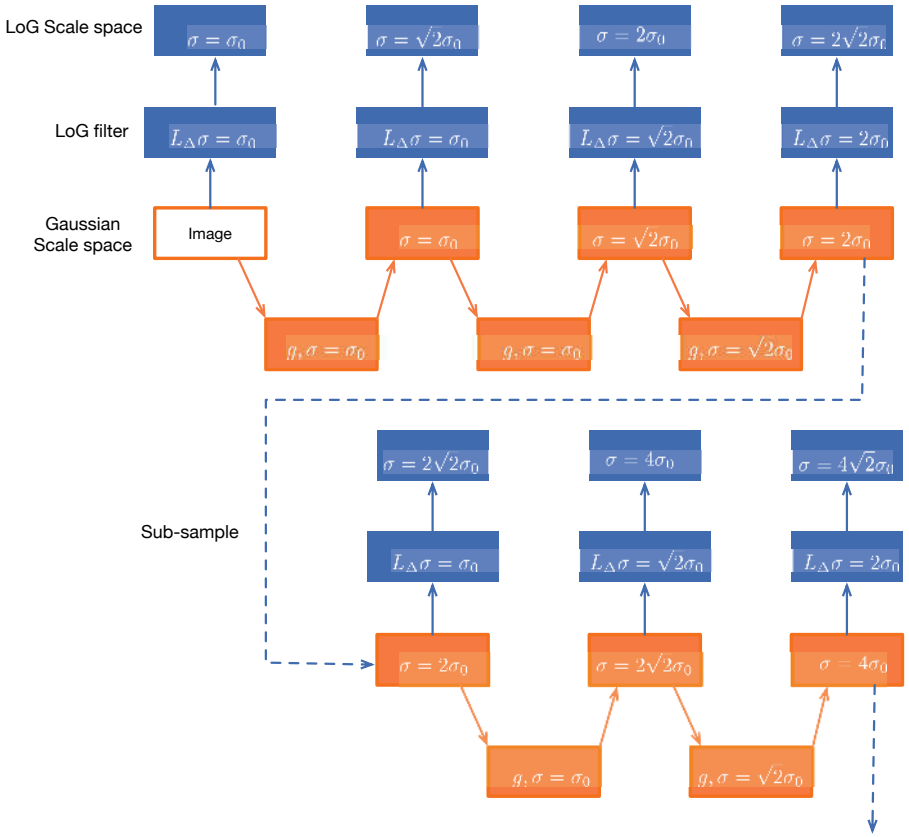


Figure 3.22. A designed affine Laplacian of Gaussian pyramid.

As is presented in Figure 3.22, this is an LoG pyramid we propose for affine alignment. This pyramid is completely based on the Gaussian pyramid, adjusted by a final LoG operation. It applies

just two Gaussian filters and three LoG filters for recursive construction. A significant advantage of this structure is that it can construct both the Gaussian and LoG scale space at the same time.

In this pyramid, four LoG scale spaces lay in the first octave, and three in each of the others. This special structure in the first octave makes it a bit complex for a parallel computation. But the rest of the structure is exactly the same, easy to be concurrently processed. Meanwhile, Gaussian pyramid contains three images in each octave. The size of the LoG images becomes smaller and smaller following the sub-sampling. Like the isotropic Gaussian pyramid, there is also a constraint of the minimum size of image to be sub-sampled. The scale space should be large enough to represent all the structures in the image.

In the pyramid, σ_0 is the initial scale and the rest of scales are $\sqrt{2}$ times the initial one. So normally, the initial scales should be small enough to precisely cover the whole space, even if a high precision also means a high computational complexity. In practice, we will set $\sigma_0 = 1.6$, keeping the balance between the precision and efficiency. This pyramid is especially designed for the affine LoG construction, but it is also suitable for an isotropic scale space. Actually, isotropic scale space is a special case in the framework of Gaussian affine alignment.

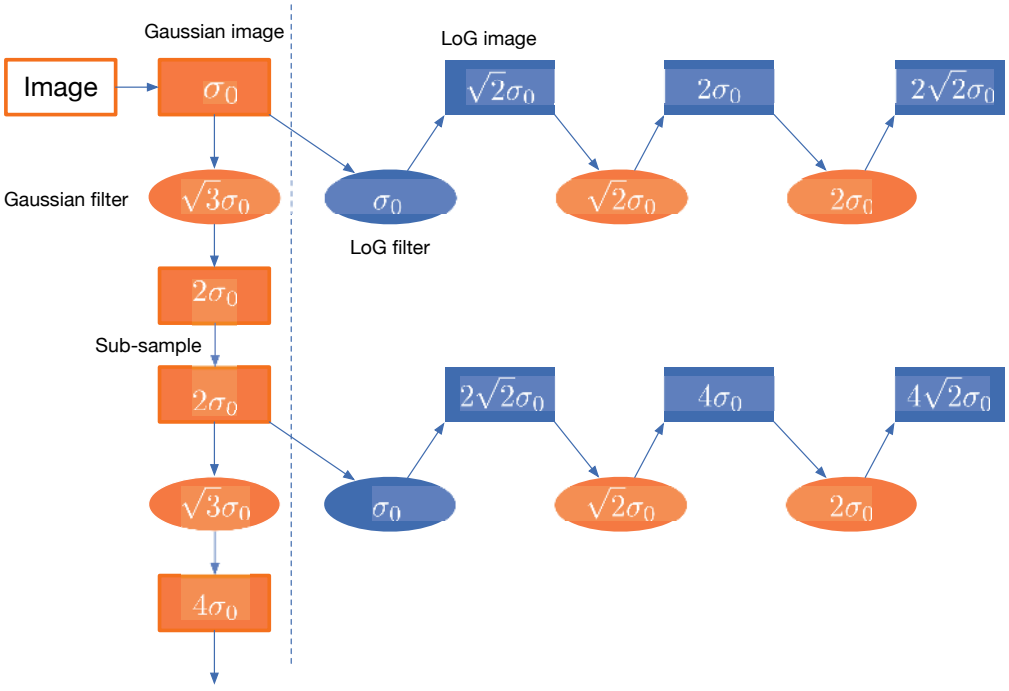


Figure 3.23. Another structure for affine Laplacian of Gaussian pyramid.

Another pyramid structure we designed for the affine LoG scale space is shown in Figure 3.23. It is also based on the scale space construction of affine Gaussian and affine LoG. This structure can be simply separated into two parts, Gaussian part and LoG part. The separation is illustrated by a blue dotted line in Figure 3.23. In this structure, the initial image of each octave is generated by Gaussian smoothing and image sub-sampling. There is just one Gaussian filter recursively applied in this part and its scale is $\sqrt{3}\sigma_0$. Three other filters are used to generate the other images in each octave,

including two affine Gaussian filters and one LoG filer. The LoG filter will be firstly convolved with the initial image, guaranteeing the rest of the images are all LoG filtered. Overall speaking, this structure of affine LoG is simpler and easier to be constructed, comparing with the pyramid presented in Figure 3.22. It contains the structure we proposed in Figure 3.19, making the structure in each octave simple.

It should also be noted that we cannot obtain a Gaussian scale space in this pyramid. So if the application also requires a Gaussian scale space, like Gaussian gradient descriptor, the pyramid in Figure 3.23 may be more appropriate. For a normal application, 7 steps are needed to construct each octave, including image smoothing and sub-sampling in the pyramid presented in Figure 3.22 and only 5 steps are required for the same task in Figure 3.22. So if the application requires only an automatic extraction of features from the derivative of scale space, we recommended the pyramid in Figure 3.22 for space construction.

Up to now, we have explored the implementation of affine Gaussian scale space and LoG scale space in spatial domain. The structure to implement the affine Gaussian scale space is exactly the same as the structure of isotropic Gaussian scale space since the properties of semi-group and sub-sampling are both satisfied by affine Gaussian. With these two properties, we can build a pyramid structure approximating isotropic Gaussian scale space. But the structure of affine LoG pyramid is quite different. In SIFT and ALP, LoG or DoG can be easily obtained by a simple subtraction between the neighbouring images in each octave or a convolution with the Laplacian operator. The affine LoG cannot be simply split as a Laplacian operator and Gaussian scale space, which can be easily constructed by the Gaussian pyramid. To make the whole scale space robust to the affine alignment, the affine LoG can only be achieved through the convolution with LoG filters, which has also been adjusted for affine deformation. Meanwhile, we have designed two pyramids for efficient and accurate construction of affine LoG. These two special structures can be used to extract features from images under variable affine deformations.

On the other hand, all the structures are implemented in spatial domain. Since most of the work of scale space construction is due to image convolution, there are several advantages in doing this in frequency domain, hence we will also explore the possibility of the affine scale space implementation in frequency domain.

3.4 Implementation of affine scale space in frequency domain

A particular feature of Gaussian filter is that its expression in spatial domain and frequency domain is quite similar. Comparing with the implementation in spatial domain, there are several advantages to process the image in frequency domain. Firstly, multiplication is quite fast and accurate compared to convolution, especially when dealing with a large number of different sized filters. Secondly, multiple scaled Gaussian filters are easily created in frequency domain.

Before the actual implementation, we will recall the corresponding relations of the Gaussian filters in spatial domain and frequency domain.

The expression of a Gaussian filter in spatial domain is,

$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.35}$$

The corresponding expression in frequency domain is,

$$G(\mu,\omega) = e^{-\frac{\mu^2+\omega^2}{2\tau^2}}\ , \tau = \frac{F_s}{2\pi\sigma}, \tag{3.36}$$

where F_s is the sampling rate for the digital implementation of Gaussian filter in frequency domain and here it equal to the size of filter. The expressions in both spatial domain and frequency domain are quite similar. Thus we can apply the same structure to build the scale space in frequency domain.

Denote the Fourier transform of Gaussian kernel as:

$$g(x,y) \leftrightarrow G(\mu,\omega) \tag{3.37}$$

The affine deformation is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = M \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.38}$$

Then,

$$\begin{aligned} \mu x + \omega y &= \begin{bmatrix} \mu & \omega \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\ &= \begin{bmatrix} \mu & \omega \end{bmatrix} \begin{bmatrix} a & b \\ d & e \end{bmatrix}^{-1} \begin{bmatrix} x' \\ y' \end{bmatrix} \\ &= \frac{1}{det(M)} \begin{bmatrix} e\mu - d\omega & -b\mu + a\omega \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} \end{aligned} \tag{3.39}$$

Hence,

$$G(\mu,\omega) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(ax+by, dx+ey)\cdot e^{2j\pi(\mu x+\omega y)}dxdy \tag{3.40}$$

Denote,

$$\begin{bmatrix} \mu' \\ \omega' \end{bmatrix} = \frac{1}{det(M)} \begin{bmatrix} e\mu - d\omega \\ -b\mu + a\omega \end{bmatrix} = (M^T)^{-1} \begin{bmatrix} \mu \\ \omega \end{bmatrix} \tag{3.41}$$

$$\begin{aligned} G(\mu,\omega) &= \frac{1}{|det(M)|}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')\cdot e^{j(2\pi/det(M))(\mu' x'+\omega' y')}dx'dy' \\ &= \frac{1}{|det(M)|}G(\mu',\omega') \end{aligned} \tag{3.42}$$

In this way, we can obtain the Fourier pairs:

$$g(x',y') \leftrightarrow \frac{1}{det(M)}G(\mu',\omega') \tag{3.43}$$

It can also be written that

$$\begin{aligned} g(\eta) &\leftrightarrow \frac{1}{det(M)} e^{-\frac{\Psi^T M M^T \Psi}{2\tau^2}} \\ \Psi &= \begin{bmatrix} \mu \\ \omega \end{bmatrix},\ \tau = \frac{F_s}{2\pi\sigma} \end{aligned} \tag{3.44}$$

The Fourier transformation of a Gaussian kernel can be affine adapted according to the equations above. Accordingly, affine scale space can also be constructed in frequency domain by the Gaussian filter in frequency domain. The procedures for the implementation in frequency domain are not complex. The main procedures are listed below.

- Calculate the minimum zero pad to be added to the image and define the minimum size of the filter.
- Create the corresponding frequency filters.
- Multiply the spectrum of image with the corresponding filter and obtain the spectrum of the blurred image.
- Inversely transform the images to spatial domain.
- Remove the blank part of image.

According to the scaling property of Fourier transformation, a smaller scale of Gaussian kernel in spatial domain reflects to a larger Gaussian kernel when implemented in frequency domain. The size of Gaussian kernel is related to the capability to reduce the interpolation noise. So the scale space processed in frequency domain will result in a high accuracy at small scale and comparable less accuracy at large scale. Considering that most features will be extracted at small scales, the implementation in frequency domain can be used to improve the overall extraction accuracy.

On the other hand, the scale of the Gaussian kernel spectrum is related to the filter size according to equation 3.4 and the spectrum of image size can be adjusted by adding different amount of zero pad. In this way, the accuracy of each scale space is under control. In addition, the computational complexity will be reduced since multiplication is easier to be implemented compared to convolution, especially when there is just one image but a lot of filters.

Figure 3.24 presents the accuracy of affine Gaussian kernel implemented in frequency domain. We can compare these figures with Figure 3.10. One significant difference is that the implementation in frequency domain provides a better performance at small scales rather than at large scales, which is in contrast to the result in spatial domain.

The Laplacian of Gaussian kernel spectrum can easily be derived thanks to the differentiation property of Fourier transform i.e. If $f(x)$ is a differentiable function with Fourier transform $F(\omega)$, then the Fourier transform of its derivative is given by $j\omega F(\omega)$. This can be used to transform differential equations into algebraic equations. LoG is the partial derivative of a Gaussian. It can also be derived by translating into algebraic equations thanks to the differential property of Fourier transformation:

$$L(\eta) \leftrightarrow -\frac{4\pi^2(\Psi^T M M^T \Psi)}{det(M)(F_s^2)} e^{-\frac{\Psi^T M M^T \Psi}{2\tau^2}} \tag{3.45}$$

With the spectrum expression of LoG filter, the LoG scale space can also be created in the frequency domain.

Differently from construction of affine LoG scale space, which specifies affine LoG filter as its kernel, the affine LoG spectrum can be completely separated into a Gaussian kernel and Laplacian

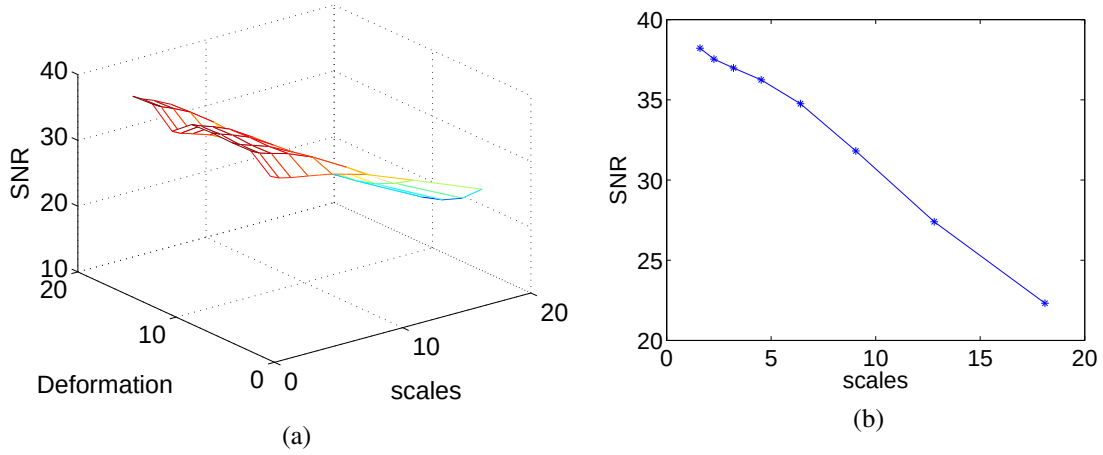


Figure 3.24. The performance of affine Gaussian scale space implemented in frequency domain. (a) SNR versus deformation and scales. (b) SNR versus scales.

operator. This makes it possible to construct the affine LoG scale space based on an affine Gaussian scale space by a simple operation. Here is the mathematical derivation.

Supposing the Fourier transform pair as:

$$L_{Lap}(x,y) \leftrightarrow L_{Lap}(\mu,\omega) \tag{3.46}$$

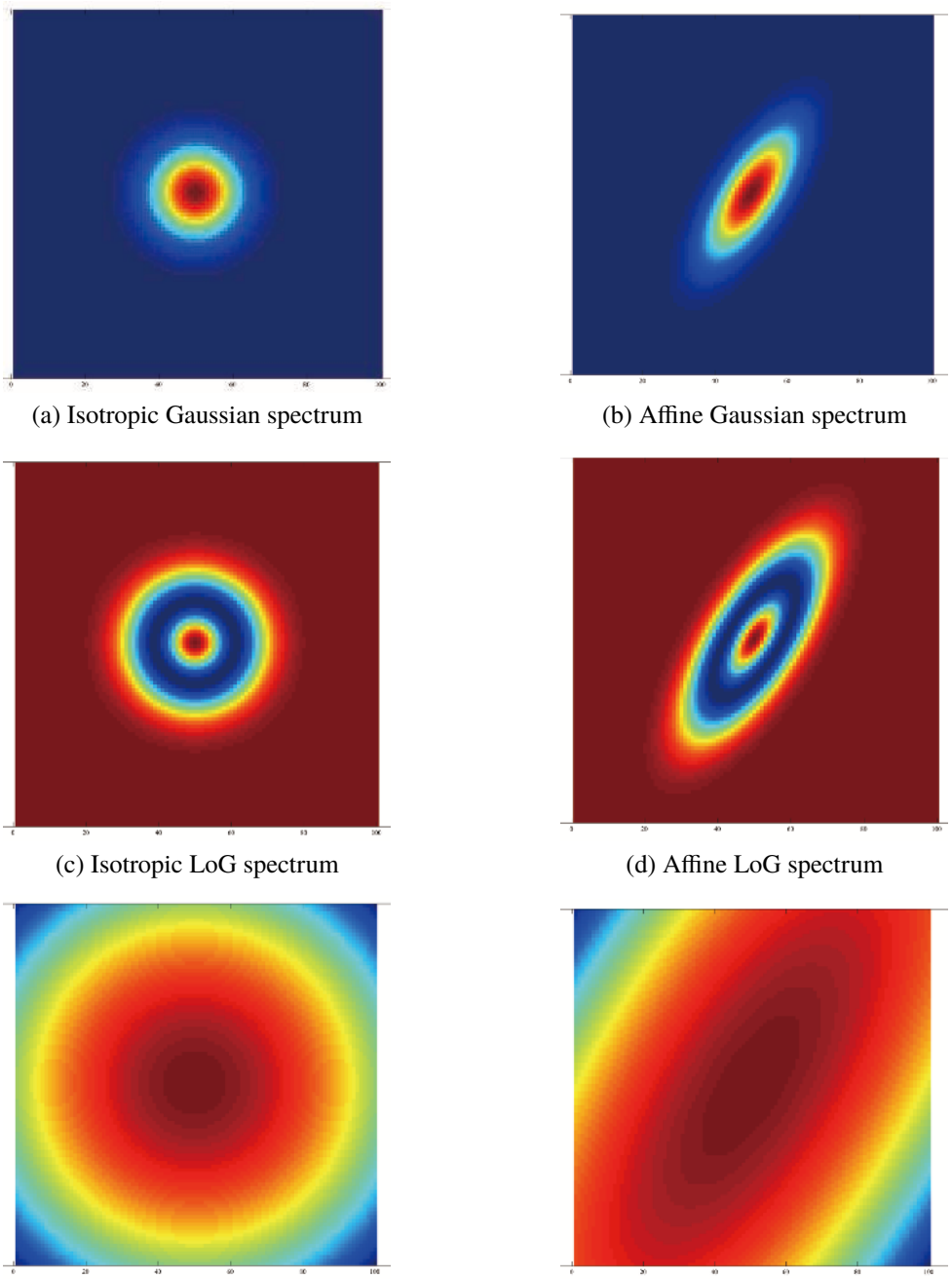
then,

$$\begin{aligned} L_{Lap}(\mu,\omega,\sigma) =& L_{\Delta}(\mu,\omega,\sigma)I(\mu,\omega) \\ =& I(\mu,\omega)\cdot\left(-\frac{4\pi^2(\Psi^T MM^T\Psi)}{det(M)(F_s^2)}e^{-\frac{\Psi^T MM^T\Psi}{2\tau^2}}\right) \\ =& I(\mu,\omega)\cdot\frac{1}{det(M)}e^{-\frac{\Psi^T MM^T\Psi}{2\tau^2}}\cdot\left(-\frac{4\pi^2(\Psi^T MM^T\Psi)}{(F_s^2)}\right) \\ =& I(\mu,\omega)\cdot G(\mu,\omega,\sigma)\cdot\left(-\frac{4\pi^2(\Psi^T MM^T\Psi)}{(F_s^2)}\right) \\ =& L(\mu,\omega,\sigma)\cdot\left(-\frac{4\pi^2(\Psi^T MM^T\Psi)}{(F_s^2)}\right) \end{aligned} \tag{3.47}$$

So the affine LoG scale space can be obtained from the corresponding Gaussian scale space by processing with an affine Laplacian operator. To make the procedure simpler, $-\frac{4\pi^2(\Psi^T MM^T\Psi)}{(F_s^2)}$ can be used as a Laplacian operator by multiplying with the corresponding Gaussian scale space.

In the Figure 3.4, (a) and (b) present the spectrum of Gaussian kernel; (c) and (d) present the spectrum of LoG filter; (e) and (f) present the frequency Laplacian operator.

With the help of this Laplacian operator, we do not need to specify a structure to construct the LoG scale space. We just need to focus on the construction of affine Gaussian scale space. The corresponding LoG scale space can easily be obtained by a simple Laplacian operation on the corresponding Gaussian scale space.



(a) Isotropic Gaussian spectrum

(b) Affine Gaussian spectrum

(c) Isotropic LoG spectrum

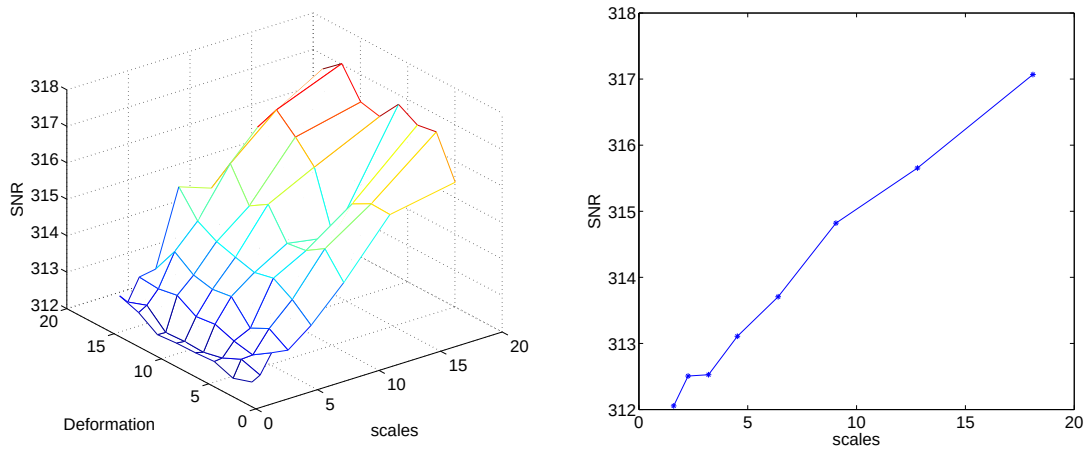
(d) Affine LoG spectrum

(e) Isotropic Laplacian operator

(f) Affine Laplacian operator

Figure 3.25. Operation spectrum.

Another important property which can be used to speed up the processing in frequency domain



(a) (b)

Figure 3.26. Precision of Laplacian operator. (a) SNR versus deformation and scales. (b) SNR versus scales.

is semi-group property. Given the property in spatial domain, which is:

$$g(\xi, \sigma_1^2) * g(\xi, \sigma_2^2) = g(\xi, \sigma_1^2 + \sigma_2^2) \tag{3.48}$$

The corresponing spectrum is given by,

$$g(\xi, \sigma) \leftrightarrow G(\Psi, \sigma) \tag{3.49}$$

We can Fourier transform both sides of equation 3.48. According to convolution theorem, we have

$$G(\Psi, \sigma_1^2)G(\Psi, \sigma_2^2) = G(\Psi, \sigma_1^2 + \sigma_2^2) \tag{3.50}$$

The equation above illustrates not only the semi-group property but also the simplicity of the frequency cascade implementation. This property can be applied for a new structure to speed up the frequency scale space construction.

Figure 3.27 presents the SNR of the affine Gaussian kernel cascade implemented in frequency domain. In that Figure, (c) and (d) illustrate the performance on affine deformed images. Compared with the graph (b), there is no obvious precision decrease after each step of cascade implementation. On the contrary, the cascade implementation is more efficient in frequency domain and it can even be used to reduce the interpolation noise in each step.

The cascade implementation in spatial domain is linked with convolution. In frequency domain

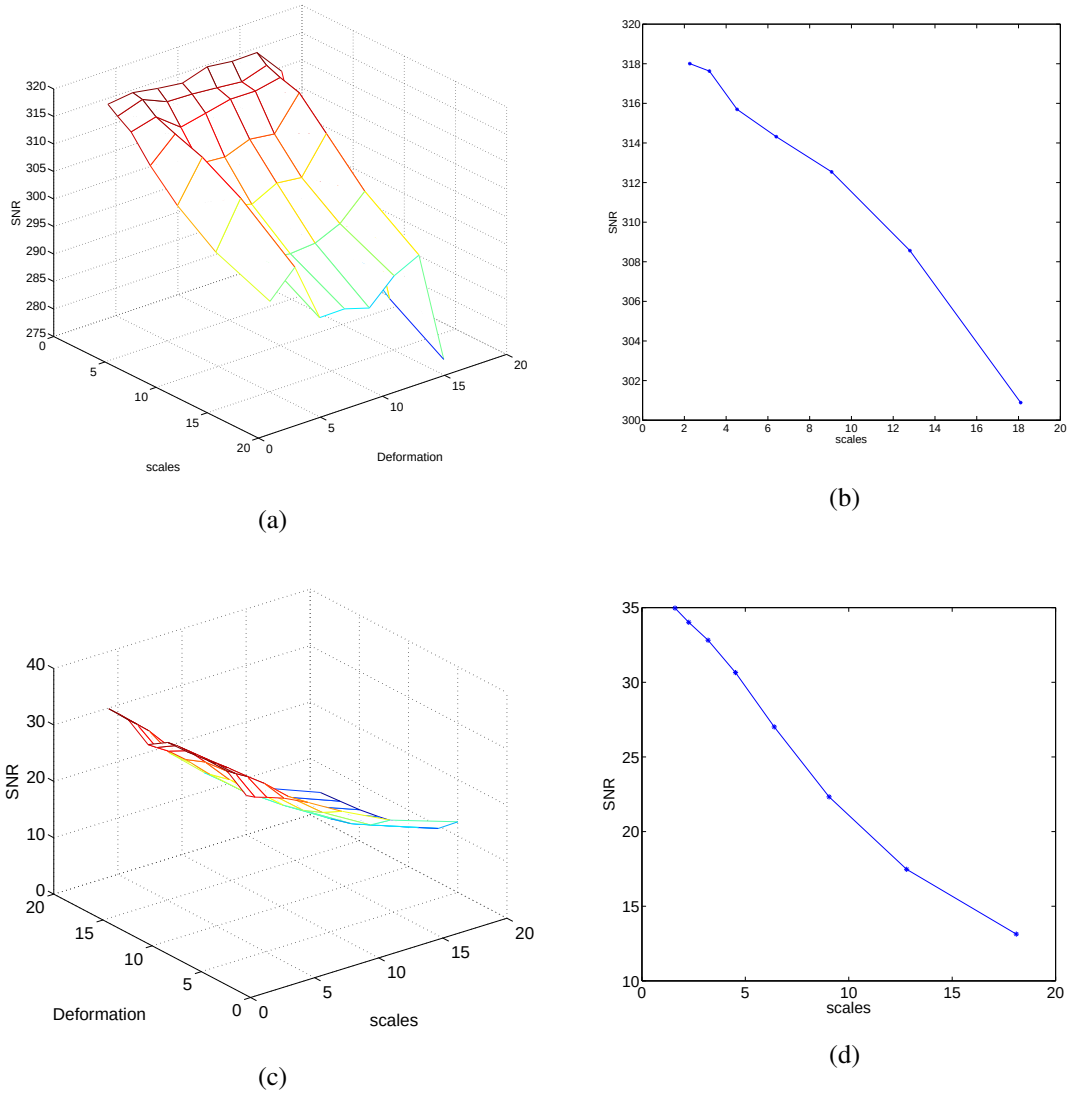


Figure 3.27. The performance of affine Gaussian scale space cascade implemented in frequency domain. (a) SNR versus deformation and scales. (b) SNR versus scales. (c) SNR versus deformation and scales for real image (d) SNR versus scales for real image

it is linked with multiplication, which can contribute to another new structure of scale space.

$$\begin{aligned} G(\Psi,\sigma_0^2)G(\Psi,\sigma_0^2) &= G(\Psi,\sigma_0^2+\sigma_0^2) \\ G(\Psi,\sigma_0^2)^2 &= G(\Psi,\sqrt{2}\sigma_0^2) \\ &\vdots \\ (G(\Psi,\sigma_0^2)\underbrace{\cdots}_{n})^2 &= G(\Psi,(\sqrt{2})^n\sigma_0^2) \end{aligned} \tag{3.51}$$

The properties above inspire us to design a new structure to build the scale space in frequency domain.

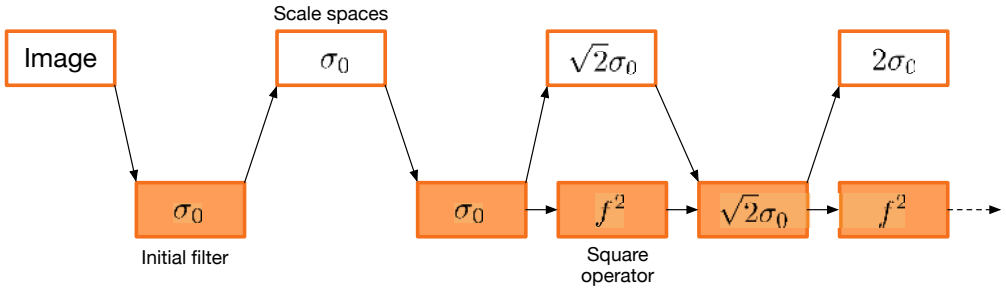


Figure 3.28. A new affine Gaussian structure in frequency domain.

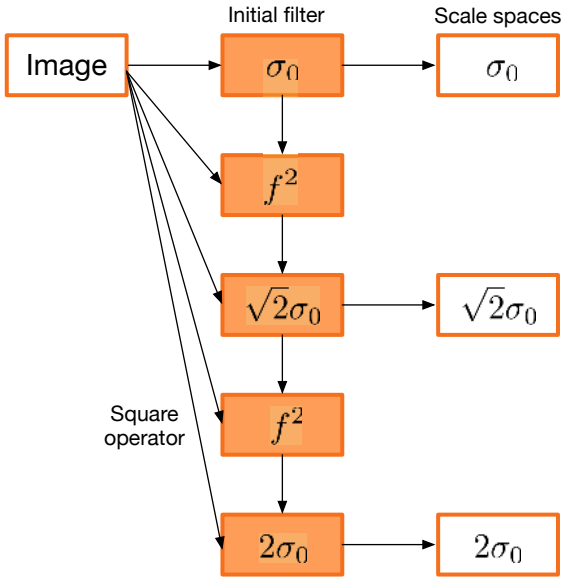


Figure 3.29. Implementation of Gaussian structure in frequency domain.

According to the equation 3.51, the cascade implementation in frequency domain is accomplished based on multiplication not convolution. Hence, cascade multiplication can be simplified as a square operation especially to the $\sqrt{2}$ scale based scale space. In Figure 3.28 the square operation will be used to generate the Gaussian filters in frequency domain by squaring the values of the previous filter. The property of Gaussian filter that its Fourier transform has zero imaginary part makes the square operation simpler to be applied. Based on the filters of the previous scale and square operation, the filter for the next cascade implementation can easily be generated by a square operator.

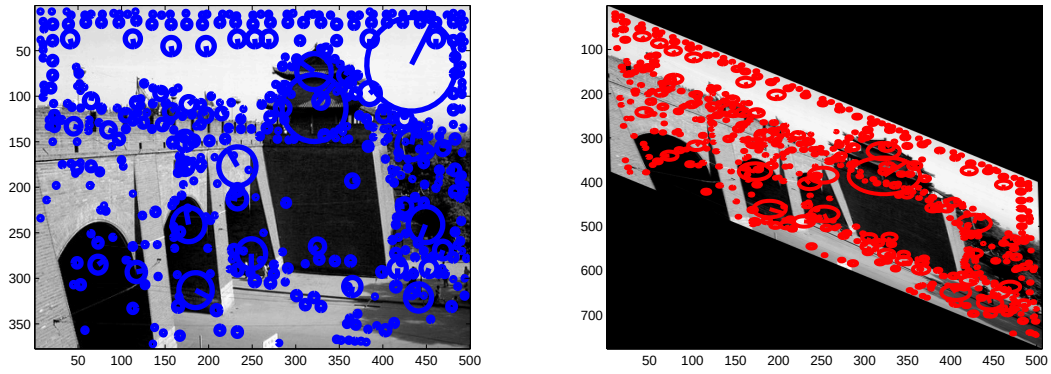
In this structure, the scale space is cascade implemented by a recursively squared affine Gaussian filter. Each successive scaled image is the multiplication of the current smoothed image with

the square of current filter. So each time, the next smoothed image is generated by the square of current filter multiplied by the current smoothed image. In this way, the scale space can be constructed simply by a multiplication operation, a square operation and an initial filter.

This is another structure not based on the cascade implementation. The parameter of current filter is also the square of the filter of the previous scale. By applying this structure, the scale space can also be constructed efficiently without a cascade implementation.

3.5 Result analysis

The performance of the affine scale space implementation will be evaluated by applying the feature detection to the images under different view points. A typical feature detection on the images of different view point is illustrated in figure 3.30. In the figure, (a) shows the feature detection on the original image and (b) shows the feature detection on an affine transformed image. Since an image under different viewpoint can be simulated by an affine transformation, we will employ the affine transformation on the image to evaluate its detection performance under different view point, as depicted in (b). In the figure (a), a general scale space will be established for the feature detection and in the figure (b), an affine scale space will be employed to detect the features from the affine transformed image.

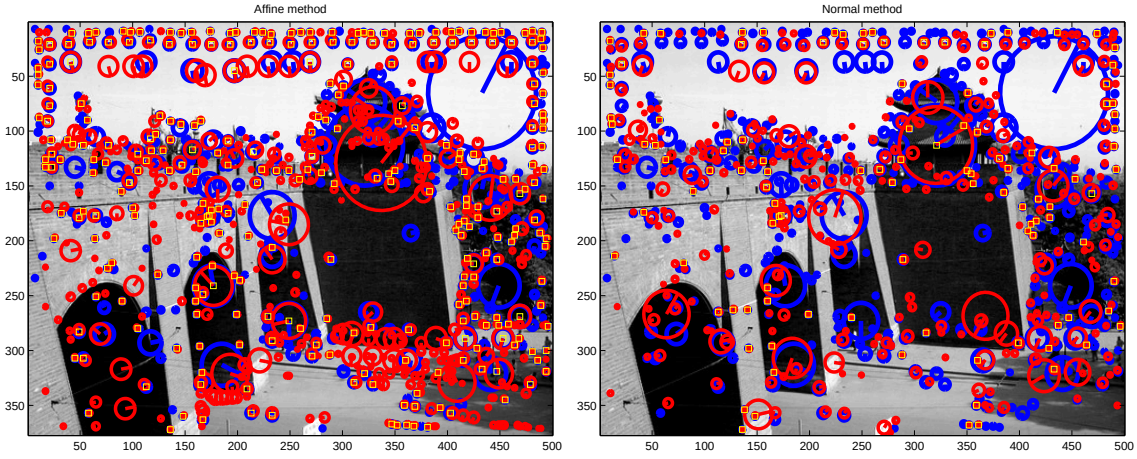


(a) (b)

Figure 3.30. Detection performance on the affine transformed images (a) Feature detection on original image. (b) Feature detection on the affine transformed image.

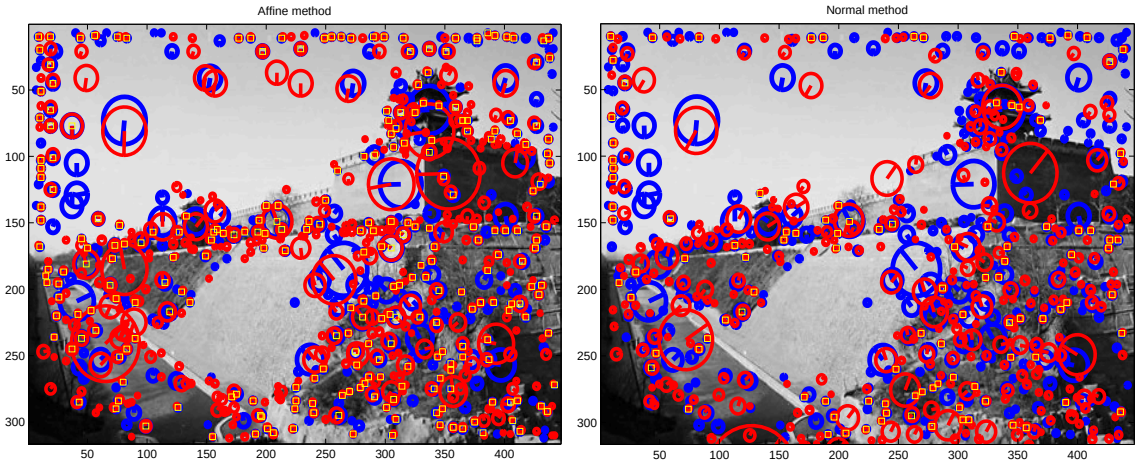
As we have discussed, the new detected features shall be retained almost the same by employing the specified affine scale space. To compare the detected features from the different perspectives of the image, the features detected in the affine transformed image will be back-projected to the original one. In the figures 3.31, 3.32 and 3.33, the blue circles depict the features detected from the original image, red ones are the back-projected features detected from the affine transformed image. The radius of a circle represent the corresponding scale and the yellow box depicts the features that have been detected by both the affine scale space and general scale space. For these figures in 3.31, 3.32 and 3.33, the affine transformation is $\begin{bmatrix} 1.0000 & 0.0050 \\ 0.6000 & 1.0000 \end{bmatrix}$. In the figure 3.31,

the correct detection by affine scale space is 0.6416 but for the general scale space, it is 0.2425.



(a) (b)

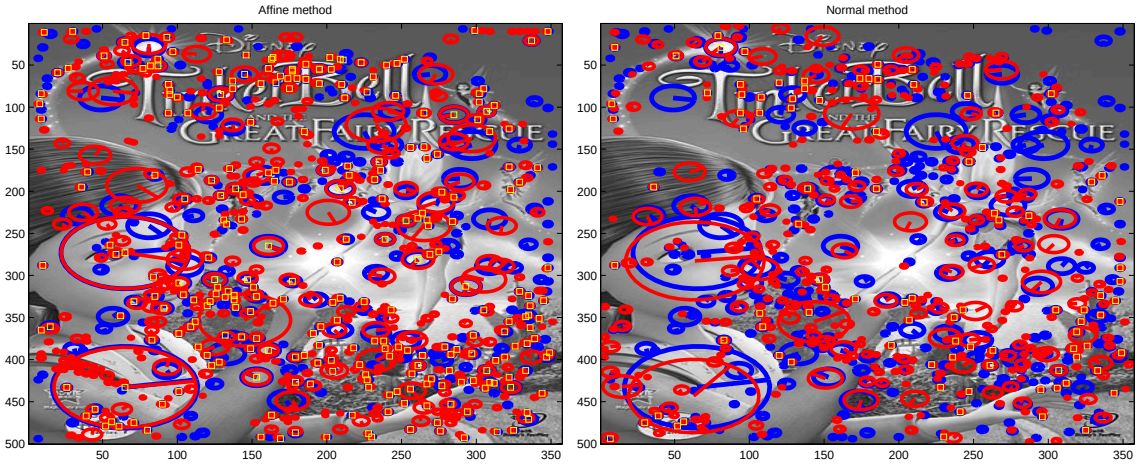
Figure 3.31. Comparison of detection performance by employing affine scale space (a) Feature detection by employing affine scale space. (b) Feature detection by general scale space. The blue circles depict the features detected from the original image, red ones are the back-projected features detected from the affine transformed image. The radius of the circle represents its scale and the yellow box depict the correctly detected features. The correct detection by affine scale space is 0.6416 but for the conventional scale space, it is 0.2425.



(a) (b)

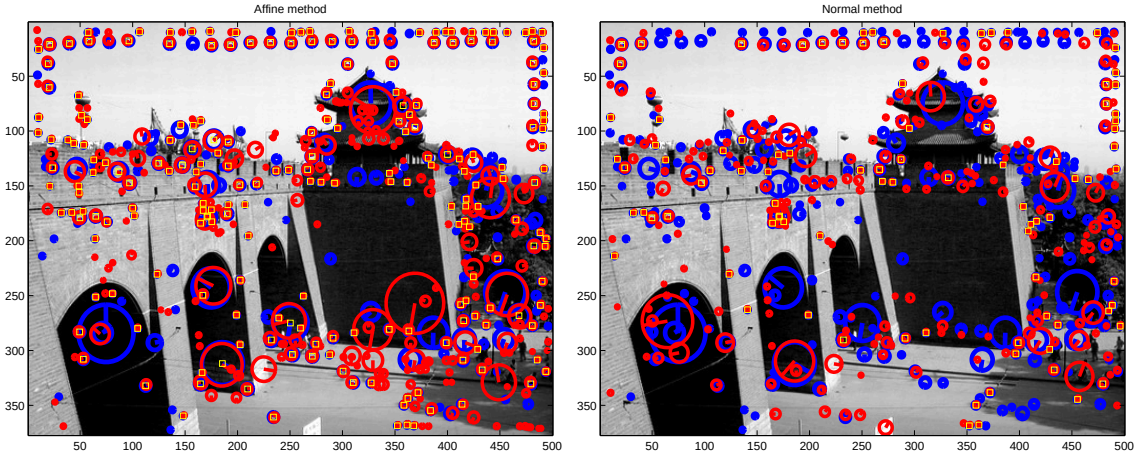
Figure 3.32. Comparison of detection performance by employing affine scale space (a) Feature detection by employing affine scale space. (b) Feature detection by general scale space. The correct detection by affine scale space is 0.6541 but for the conventional scale space, it is 0.2707.

The implementation of affine scale space in figures 3.31, 3.32 and 3.33 employs the structure depicted in figure 3.22. Figure 3.34 shows another detection comparison between affine scale space and general scale space implemented in frequency domain.



(a) (b)

Figure 3.33. Comparison of detection performance by employing affine scale space (a) Feature detection by employing affine scale space. (b) Feature detection by general scale space. The correct detection by affine scale space is 0.6522 but for the conventional scale space it is 0.2503.



(a) (b)

Figure 3.34. Comparison of detection performance by employing affine scale space implemented in frequency domain. (a) Feature detection by employing affine scale space implemented in frequency domain. (b) Feature detection by general scale space implemented in frequency domain. The correct detection by affine scale space implemented in frequency domain is 0.6945 but for the conventional scale space in frequency domain, it is 0.2958.

The scale space of these two figures are constructed from the structure depicted in figure 3.28 and the rate of the accuracy is 0.6945 by affine scale space and 0.2958 by general scale space. Figures 3.31, 3.32 and 3.33 demonstrate the comparison of detection of affine scale space on different images, including two architectures images and one cartons. General speaking, there is no much difference for the detection accuracy due to the image content.

Figure 3.34 shows the detection comparison by the affine scale space implemented in frequency domain. Apparently, it extracted less features but still maintains a high level of accuracy.

Tilting	Citywall		Castle		Carton	
	affine	general	affine	general	affine	general
0.6	0.6416	0.2425	0.6541	0.2707	0.6133	0.2178
0.7	0.6202	0.2060	0.6015	0.1855	0.5711	0.1667
0.8	0.5923	0.1159	0.5840	0.1654	0.5244	0.1400
0.9	0.5494	0.1009	0.5689	0.1203	0.4659	0.1133
1.0	0.5773	0.0944	0.5689	0.1103	0.4756	0.1044
1.1	0.5515	0.0579	0.5213	0.0526	0.4867	0.0667
1.2	0.5343	0.0429	0.5414	0.0526	0.4578	0.0489

Table 3.1. The ratio of correct detection from the affine transformed image respectively by affine scale space and general scale space. The affine scale space in this table is constructed by the structure depicted in the figure 3.22.

Tilting	Citywall		Castle		Carton	
	affine	general	affine	general	affine	general
0.6	0.6416	0.2425	0.6541	0.2707	0.6133	0.2178
0.7	0.6202	0.2060	0.6016	0.1855	0.5711	0.1666
0.8	0.5923	0.1159	0.5841	0.1654	0.5224	0.1400
0.9	0.5494	0.1009	0.5689	0.1203	0.4659	0.1133
1.0	0.5773	0.0944	0.5689	0.1103	0.4756	0.1044
1.1	0.5516	0.0575	0.5213	0.0526	0.4867	0.0667
1.2	0.5343	0.0429	0.5414	0.0526	0.4579	0.0489

Table 3.2. The ratio of correct detection from the affine transformed image respectively by affine scale space and general scale space. The affine scale space in this table is constructed by the structure depicted in the figure 3.23.

By comparing table 3.1 and 3.2, we can notice that the performance on the affine transformed image by the structure of figure 3.22 and of figure 3.23 is quite similar. The performance upon different images is also quite similar. The main difference for the detection performance is between the implementation in spatial domain and frequency domain. As depicted in the figure 3.35, the implementation in frequency domain generally has a more accurate feature detection on the affine transformed images, at the expense of a high computational complexity. In our experiment, it takes 2.63 seconds to extract the features by employing the implementation in spatial domain, whereas it takes 77.62 seconds by employing the implementation in frequency domain. The experiments are implemented on the computer with the CPU Inter Xeon(R) 5130 @ 2.00 GHz, RAM 4 GB and

operating system 64 bit windows 7.

Tilting	Citywall		Castle		Carton	
	affine	general	affine	general	affine	general
0.6	0.6945	0.2958	0.7128	0.3191	0.7375	0.2406
0.7	0.7235	0.2122	0.6915	0.2092	0.6875	0.1906
0.8	0.7042	0.1125	0.6277	0.1206	0.6937	0.1437
0.9	0.6302	0.0836	0.6277	0.0887	0.6406	0.0969
1.0	0.5852	0.0675	0.5709	0.0603	0.5687	0.0750
1.1	0.5370	0.0514	0.5390	0.0567	0.5281	0.0594
1.2	0.5595	0.0514	0.5213	0.0319	0.5062	0.0437

Table 3.3. The ratio of correct detection from the affine transformed image respectively by affine scale space implemented in frequency domain and general scale space implemented in frequency domain. The affine scale space in this table is constructed by the structure depicted in the figure 3.29.

By these experiments, we have shown that the affine scale space can be successfully employed to improve the feature detection for the images under different view point, especially when the tilting angle is large. The affine scale space implementation in frequency domain has even better performance regarding the invariance to affine transformation. But its computational complexity is also large. In practice, we can choose the appropriate affine scale space implementation for the specific requirement.

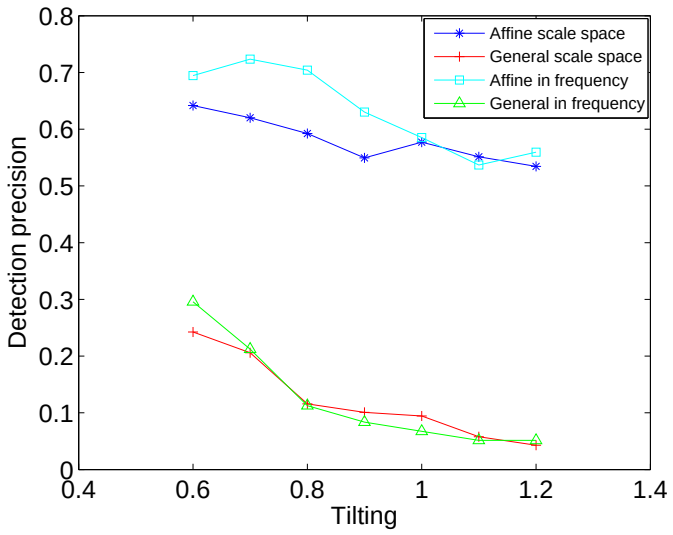


Figure 3.35. Comparison of feature detection by employing affine scale space respectively implemented in spatial domain and frequency domain.

Chapter 4

Affine descriptor

As we have discussed in the last chapter, the affine invariant scale space has been proved successful for affine image alignment. In this chapter we will propose an affine adapted feature descriptor, which is extended from an existing descriptor, and is based on the framework of Gaussian affine adaptation and thus compatible with the scale space that we have proposed in the last chapter.

Currently, many different techniques for describing local image features have been developed. The simplest descriptor is a vector of image pixels. Cross-correlation can then be used to compute a similarity score between two descriptors. Zabih and Woodfill have developed an approach robust to illumination changes [35]. It relies on histograms of ordering and reciprocal relations between pixel intensities which are more robust than raw pixel intensities. The binary relations between intensities of several neighbouring pixels are encoded by binary strings and a distribution of all possible combinations is represented by histograms. This descriptor is suitable for texture representation but a large number of dimensions are required to build a reliable descriptor.

As we have discussed at the very beginning, Lowe has proposed a scale invariant feature named SIFT [12], which combines a scale invariant region detector and a descriptor based on the gradient distribution in the detected regions. This descriptor is represented by a 3D histogram of gradient locations and orientations, and the contribution to the location and orientation bins is weighted by the gradient magnitude. The quantization of gradient locations and orientations makes the descriptor robust to small geometric distortions and small errors in the key point location. The descriptor has been successfully used, for example, in the CDVS standard.

Gradient location and orientation histogram (GLOH) is a new descriptor, which extends SIFT by changing the location grid and employing principal component analysis (PCA) to reduce the dimensionality. It is designed to increase its robustness and distinctiveness. The SIFT descriptor is computed in a log-polar location grid with 3 bins in radial direction (the radius set to 6, 11 and 15) and 8 in angular direction, which results into 17 location bins, because the central bin is not divided in angular directions. The gradient orientations are then quantized into 16 bins. This gives a 272 bin histogram. The size of this descriptor is reduced by PCA and the 128 largest eigenvectors are used as the descriptor [21].

Shape context is similar to the SIFT descriptor [36] specialized by a 3D histogram of edge point locations and orientations. Edges are extracted by Canny detector. Location is quantized into 9 bins of a log-polar coordinate system with the radius set to 6, 11 and 15 and orientation quantized into 4 bins (horizontal, vertical and two diagonals), yielding a 36-dimensional descriptor. The histogram

is weighted by the corresponding gradient magnitude. This has shown to give better results than using the same weight for all edge points. Note that the original shape context was computed only for edge point locations but not for orientations.

PCA-SIFT [37] is the size-reduced SIFT descriptor by PCA. The descriptor is a vector of image gradients in the detected scale space and direction computed within the support region. The gradient region is sampled at 39×39 locations. Hence the vector dimension is 3042 reduced to 36 by PCA.

We can observe from the introduction that the state-of-the-art descriptors are all based on image gradient. They are also similar to each other regarding the matching strategies, mostly based on cross-correlation on histograms. Therefore, it is essential to explore the possibility to analytically characterize the image gradient deformation due to an affine alignment for an affine invariant feature descriptor. In this chapter, we proposed an affine invariant feature descriptor based on an affine gradient operation, which directly generates the image gradient by filtering the image with an affine transformed Gaussian derivative. By interpolation and re-alignment of the affine gradient, we can approach the image gradient compensating for the view point difference. The descriptor based on this gradient can largely compensate the distortion brought by affine transformation and thus be invariant to different view point. One purpose of the affine adaptation on the image gradient is because the state of the art feature descriptors including SIFT and GLOH which are also compatible with the scale invariant feature detection, are all based on the histogram of local gradient. The gradient for the feature descriptor is generated from the detected scale. The smoothed images from different scales are not robust to the affine transformation, neither does the gradient. Thus an affine adaptation on the image gradient of different view point is quite useful to compensate the perspective distortion.

4.1 Image gradient

An image gradient is an operator sensitive to directional change in the intensity or color in an image. The gradient of a two-variable function (here the image intensity function) at each image point is a 2D vector with components given by the derivatives in the horizontal and vertical directions. At each image point, the gradient vector points in the direction of largest possible intensity increase, and the magnitude of the gradient vector corresponds to the rate of change in that direction [38].

Since the intensity function of a digital image is only known at discrete points, derivatives of this function cannot be defined unless we assume that there is an underlying continuous intensity function which has been sampled at the image points. With some additional assumptions, the derivative of the continuous intensity function can be computed as a function on the sampled intensity function. Approximations of these derivative functions can be defined at varying degrees of accuracy.

The definition of gradient for an image f is given by [38] :

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right), \tag{4.1}$$

where $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ are the respectively calculated gradients in x direction y direction.

The gradient direction can be calculated by the formula,

$$\theta = \arctan 2\left(\frac{\partial f}{\partial y}, \frac{\partial f}{\partial x}\right). \tag{4.2}$$

Since the digital image can not be defined as a continuous function, the image gradient is normally approximated by the difference of two adjacent pixels. To prevent a half pixel shift, the image gradient is generally obtained by employing the gradient filter: $\begin{bmatrix}-1 & 0 & 1\end{bmatrix}$. Accordingly, the gradient formula in SIFT [12] is given by

$$\begin{aligned} m(x,y) =& \sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2}, \\ \theta =& \arctan\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right). \end{aligned} \tag{4.3}$$

Notice that in the formula, gradient is not generated from the original image but from the scale where the feature was extracted.

A more precise gradient approximation can be generated by Sobel operator [39] . It is another discrete differentiation operation, computing the gradient of the image intensity function. The expression of Sobel operator is given by,

$$\nabla f = \begin{bmatrix}-1 & 0 & +1\\ -2 & 0 & +2\\ -1 & 0 & +1\end{bmatrix} * f\hat{x} + \begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ +1 & +2 & +1\end{bmatrix} * f\hat{y} \tag{4.4}$$

The Sobel operator is still not “large” enough for an affine adaptation, but it conveys the idea that image gradient can be approximated by kernels of different length. A “long” operator enables more precise adaptation for an affine alignment.

Following this idea, we will seek a large gradient operator, which can be more precisely affine adapted.

4.2 Gaussian gradient operator

In the following, we will introduce a Gaussian gradient operator as an alternative for computing the descriptor in an affine invariant way. It should be noticed that the gradient used to build the descriptor is generated from the detected scale space. The approximation of the image gradient will be generated by computing weighted differences of adjacent pixels. The Gaussian filter is a continuous differentiable function, which can be used to build the derivative of the corresponding blurred image.

Denote G_x and G_y as the gradient filters along the x and y directions.

$$\begin{aligned} \nabla L \approxeq& L * G_x\hat{x} + L * G_y\hat{y} \\ =& (I * g) * G_x\hat{x} + (I * g) * G_y\hat{y} \\ =& I * (g * G_x)\hat{x} + I * (g * G_y)\hat{y} \end{aligned} \tag{4.5}$$

The gradient filters G_x and G_y approximate the derivative operator along the respective directions. Then,

$$G_x \approxeq \frac{\partial f}{\partial x} \quad , G_y \approxeq \frac{\partial f}{\partial y} \tag{4.6}$$

Thus,

$$\begin{aligned} \nabla L \approxeq & I * (g * G_x)\hat{x} + I * (g * G_y)\hat{y} \\ \approxeq & I * (\frac{\partial g}{\partial x})\hat{x} + I * (g * \frac{\partial g}{\partial y})\hat{y}, \end{aligned} \tag{4.7}$$

where I is the input image and the Gaussian smoothing gradient operator in x direction is given by:

$$\begin{aligned} \frac{\partial g}{\partial x} =& \partial \left[\frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \right] / \partial x \\ =& \frac{x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \end{aligned} \tag{4.8}$$

In the same way, the operator in y direction is:

$$\frac{\partial g}{\partial y} = \frac{y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.9}$$

A combination operator can be expressed as:

$$\begin{bmatrix} \partial g/\partial x \\ \partial g/\partial y \end{bmatrix} = \frac{1}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.10}$$

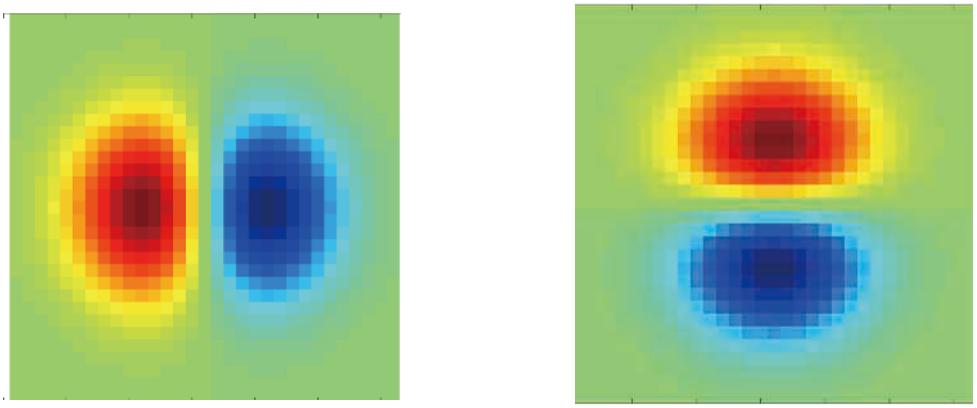
From the equations above, the Gaussian gradient operator is simple and easily achievable. The gradient used for the descriptor is calculated from the blurred image, at the scale where the feature was detected. The Gaussian gradient operation performs both image blurring and gradient generation at the same time simplifying the procedures for gradient generation and thereby decreasing the computational complexity.

As is shown in Figures 4.2 and 4.3, there is not so much difference between the gradient calculated by Gaussian gradient operator and the difference between adjacent pixels, showing that the gradient by Gaussian operator can also be employed for calculating a histogram of gradients.

However, the difference operator can hardly be adapted to an affine transformation. An affine transformation means a complex geometric shift that requires a long filter. Since the Gaussian gradient operation is accomplished by a long filter, it is more easily to be affine adapted in the same way we have done in the last chapter. In this way, the Gaussian gradient operator is suitable to provide a geometric affine adaptation on the filter itself.

4.3 Affine Gaussian gradient operator

A linear affine adapted Gaussian gradient operator can be derived based on the linear transformation on Gaussian gradient operator. The Gaussian gradient operator can be expressed in vector



(a) vertical direction (b) horizontal direction

Figure 4.1. Gaussian gradient operator.

(a) (b)

Figure 4.2. Gradient of vertical direction. (a) is generated by Gaussian gradient operation. (b) is generated by Gaussian blurring and adjacent pixels difference.

form. In mathematics, a linear transformation can be simplified as a product of the homography matrix and the coordinates vector as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = A \begin{bmatrix} x \\ y \end{bmatrix}. \tag{4.11}$$

Denote

$$\begin{aligned} \eta' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad \eta = \begin{bmatrix} x \\ y \end{bmatrix} \\ \eta' = A\eta \end{aligned} \tag{4.12}$$

(a) (b)

Figure 4.3. Gradient of horizontal direction. (a) is generated by Gaussian gradient operation. (b) is generated by Gaussian blurring and adjacent pixels difference.

x' and y' are the new coordinate system. Accordingly, the Gaussian gradient operator can be expressed in the new coordinate system as,

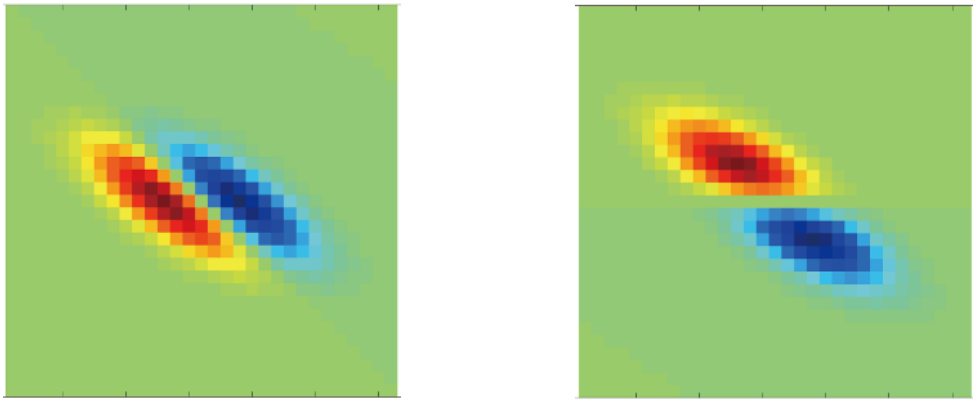
$$\begin{aligned}\begin{bmatrix}\partial f/\partial x \\ \partial f/\partial y\end{bmatrix} &= \frac{1}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \begin{bmatrix}x \\ y\end{bmatrix} \\ &= \frac{1}{2\pi\sigma^4} e^{-\frac{\eta^T\eta}{2\sigma^2}} \eta\end{aligned} \tag{4.13}$$

and hence, $\eta = A^{-1}\eta'$.

$$\begin{aligned}\begin{bmatrix}\partial f/\partial x' \\ \partial f/\partial y'\end{bmatrix} &= \frac{1}{2\pi\sigma^4} e^{\frac{(A^{-1}\eta')^T(A^{-1}\eta')}{2\sigma^2}} A^{-1}\eta' \\ &= A^{-1}\eta' \frac{1}{2\pi\sigma^4} e^{-\frac{\eta'^T(AA^T)^{-1}\eta'}{2\sigma^2}}\end{aligned} \tag{4.14}$$

The affine Gaussian gradient operator has been derived according to the formulas above, based on the same method used for the affine Gaussian and affine Laplacian of Gaussian kernel. These affine adaptation works in the same way, which is to retain a linear relation after blurring or gradient operations. The linear relation is essentially important for the image analysis especially when only geometric transformation occurred between a pair of images.

Figure 4.4 presents the affine Gaussian gradient operator with the affine transformation $\begin{bmatrix}1 & 1 \\ 0 & 1\end{bmatrix}$. Figure 4.5 is the gradient images generated by affine Gaussian gradient operator with the same affine transformation. An obvious directional preference can be found in the gradient images. A



(a) (b)

Figure 4.4. Affine Gaussian gradient operator.

(a) (b)

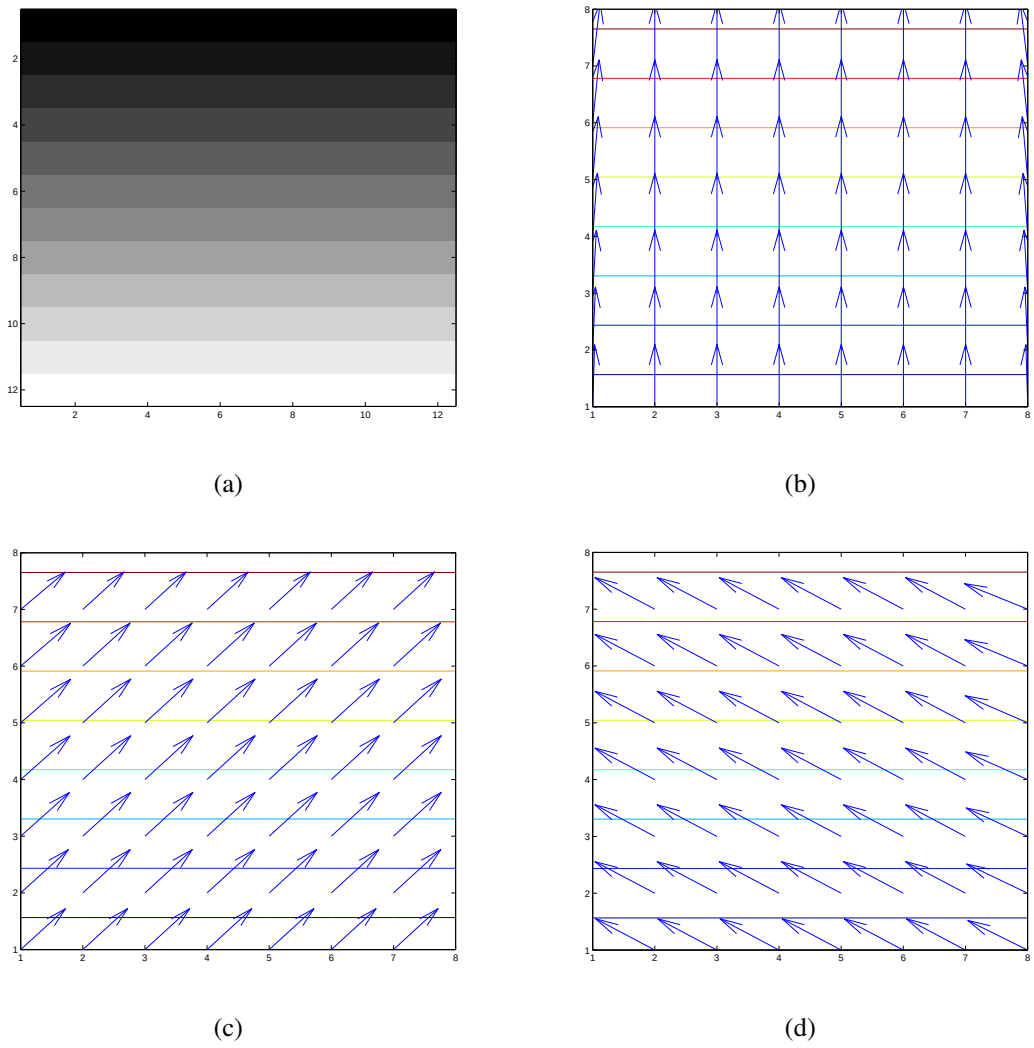
Figure 4.5. gradient by affine Gaussian gradient operator.

more intuitive illustration can be found from the gradient orientation and magnitude (OM) graphs. The orientation and magnitude of the Gaussian gradient is defined as,

$$\begin{aligned} m(x,y) &= \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2} \\ \theta(x,y) &= \arctan 2(\frac{\partial f}{\partial y}, \frac{\partial f}{\partial x}) \end{aligned} \tag{4.15}$$

where $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial x}$ are the gradients generated by affine Gaussian gradient operator.

It should be noticed that gradient graphs presented in Figure 4.6 are under different situations.



(a) (b)

(c) (d)

Figure 4.6. OM graph of image gradient under different transformation. (a) is the original image. (b) is OM graph of the gradient. (c) OM graph of the affine gradient under a skewing transformation. (d) OM graph of the affine gradient under a rotation transformation.

The gradient presented in (c) is obtained by a skewing transformation and (d) presents the gradient from a pure image rotation. Essentially, the affine Gaussian gradient operator is designed to cope with the affine alignment of the coordinates. Originally, the gradient is generated respectively from the horizontal and vertical directions. The original coordinates will be linear projected into the new coordinates. The gradients based on the original coordinates can be generated either by co-ordinates re-alignment or perspective back projection. The coordinates realignment is easier to be manipulated with no overall image transformation. Our proposed Gaussian gradient operator can be used as a coordinates re-alignment tool which specifies the derivative functions for the gradient generation from the detected scales.

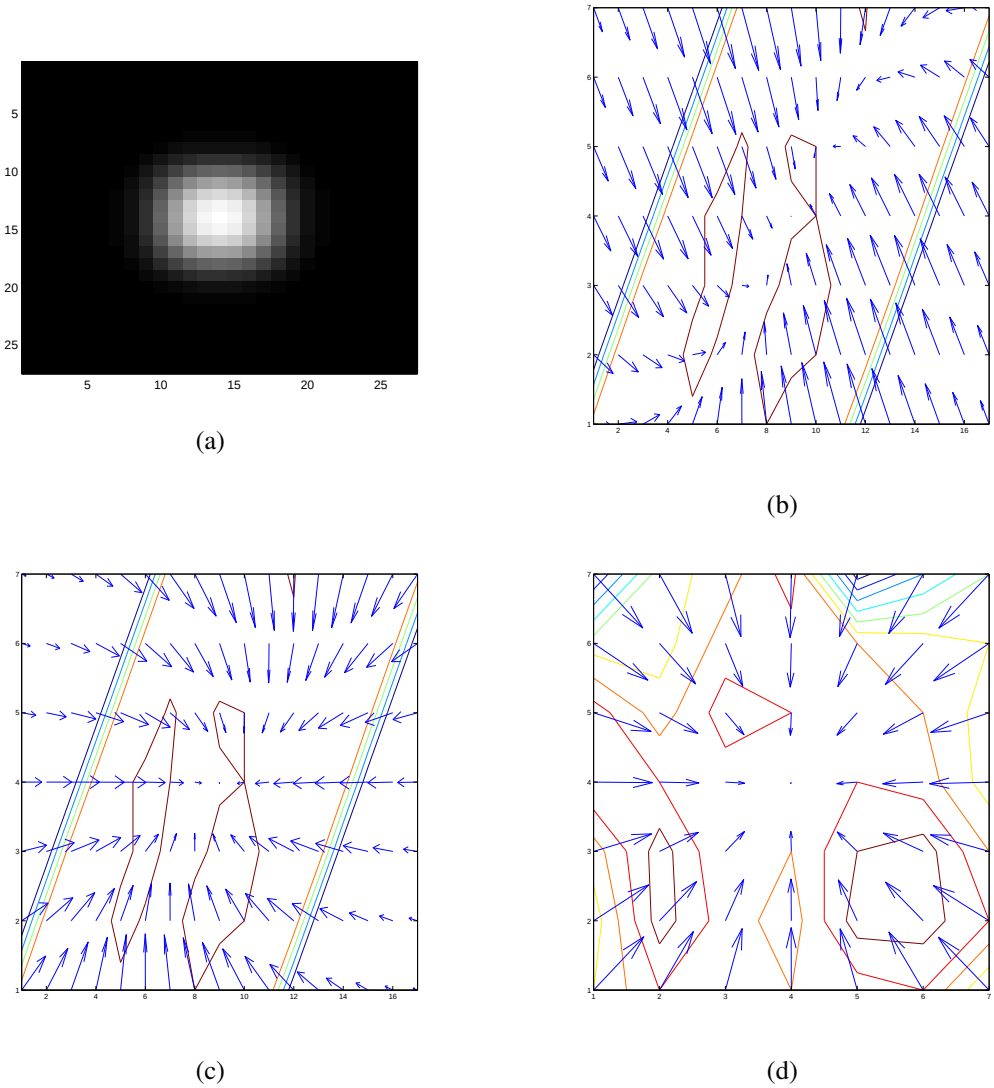


Figure 4.7. Affine gradient and normal gradient. (a) is the original image. (b) shows the gradient on the affine transformed image. (c) shows the affine gradient on the affine transformed image. (d) gradient on the original image.

Figure 4.7 presents the gradient from an affine deformed image respectively generated by a normal Gaussian gradient operator and an affine Gaussian gradient operator. The gradient by these two operators are different between each other once they are presented in different coordinates systems. However, the value of the gradient from the deformed image is the same as the gradient from the original image by a normal operator. They only differ in the corresponding gradient locations. This shows that, if the affine transformation is known, gradient can be compensated using the proposed operator. What is still missing is how to compensate the location distortion brought by the affine alignment. The objective is to collect the same set of gradient from the affine transformed image

to approach to the gradient around the original image.

4.4 Gradient re-location and interpolation

Even if the gradients generation have been affine adapted to retain the same value, it every pixel will also be moved to a new position according to the linear relation of different image views. The purpose of this step is to compensate the location distortion brought in by the affine adaptation and to re-locate each gradient for its original position to approach the descriptor compensating the view point difference.

Gradients from linear transformed images are related though the homography matrix between different views. The relations can be simplified by the expression (4.11). Since every gradient in the image from another view point is constrained by this relation, we can compensate the corresponding gradients according to this deformation. In this way, the gradient generated by affine Gaussian gradient operator can be transformed in so as to approximate the gradient of the original image. Since the gradient patch to generate the descriptor is not large, we can figure out the specific area to generate the gradient from the transformed image and further identify the relations for each gradient pair.

The procedures of the gradient interpolation after an affine Gaussian gradient operation are given by:

1. Specify the patch to generate the affine gradient on the skewed image according to the pre-defined patch on the original image. Denote H as the homography matrix of the affine transformation and $N_x \times N_y$ as the area to generate the gradient on the original image. Than the corresponding path on the affine transformed image is $\max(HN_x) \times \max(HN_y)$ as demonstrated in the Figure 4.8.

2. Fix the coordinate system for the affine patch and its relation with the pre-defined patch. For instance, we all take the right upper vertex as the zero point for both patch. Then they will be related with the Homography matrix H.

3. List all the positions on the pre-defined patch by its coordinates. In the Figure 4.8, we have listed them with the number 1,2,3, … . These points will also be given their coordinates.

4. Specify each of them with the corresponding positions on the affine patch by forward transformed the coordinates. If the coordinates of point A is $(x, y)^T$ than the point with the coordinate $H(x, y)^T$ is the corresponding point. In the Figure 4.8, the corresponded points on the affine transformed patch have also been labelled with 1,2,3, … .

5. Collect the gradient from the corresponded positions on the patch. Ideally, we could collect a set of points that approximate the gradients on the original image. We will also estimate the gradient values according to its neighborhoods if that position does not correspond to any given points.

We now provide more details about this procedure, assuming that a given image can be deemed as an affine deformed version of an original version stored in the dataset.

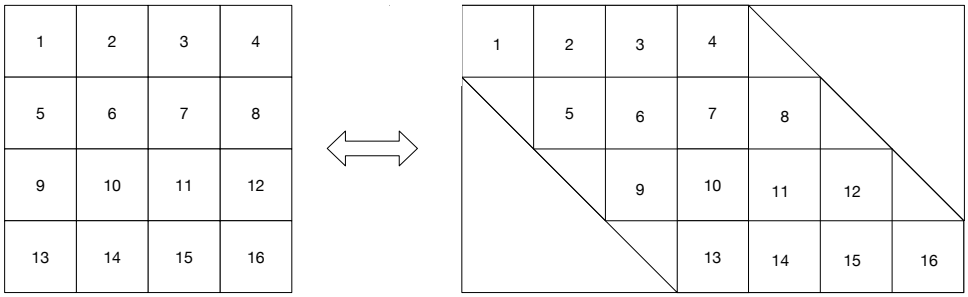


Figure 4.8. Gradient collection.

Our target affine descriptor is an extension of SIFT [12]. Instead of the gradient generation by difference of adjacent pixels, the gradient used in the affine descriptor will be created by affine Gaussian gradient operator, which adopts the affine coordinate system according to the affine deformation between the query and reference images. The area to calculate the gradient for the descriptor from the original image is determined by the detected location and scale of the feature. It covers a circular plan around the feature with a radius equal to 4.5 times the scale for assigning the feature orientation, and $15\sqrt{2}$ times the scale for calculating the descriptor. To simplify the computation, both areas can be enlarged to a tangential square containing the circular part.

Once the patch area from the original image in the dataset has been fixed, the patch area on the affine deformed image can also be calculated accordingly. The four vertexes of the area from the original image will be forward transformed according to the homography matrix to define the corresponding area in the query image. The new area on the query image is defined by the minimum rectangular tangent to quadrilateral formed by the projected vertexes. This area is used to calculate the gradients by the affine Gaussian gradient operator. Figure 4.8 presents a typical gradient patch, where only the numerated gradient will be collected to calculate the affine invariant descriptor.

Then the coordinate systems of these two areas are to be linked either by the area centre or one of the vertexes. These two system are equivalent, if the vertex transformation can be satisfied by both. In this way, each point in its own area can be uniquely determined by its coordinate system. If we forward transform all the locations from the original image domain to the affine deformed image domain, we can obtain the locations in the affine deformed image. In this way, each pair from the two images has been linked together. Thus, the gradients on the original image can be approximated by picking the corresponding gradient from the deformed version. However, it is not guaranteed that every point from the affine deformed patch will reflect to an available one from the original image.

On the other hand, forward transformation from the original patch to the affine one cannot guarantee to link points with integer coordinates since a digital image is not a continuous function and thus the non integer coordinate points cannot be found. But in a small scale, especially in the blurred image, the gradients are still correlated and given a set of known values, the target values can be estimated with interpolation. This gradient interpolation may introduce some noise, but it is better than the gradient based on the image perspective back-transformation which may introduce

more noise on both the pixel values and gradient values.

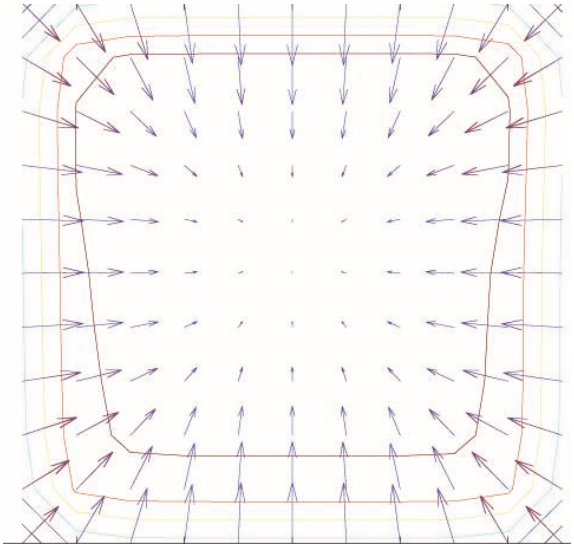


Figure 4.9. Comparison between the gradient patch from images of different perspective.

Figure 4.9 demonstrates the gradient calculated from images of different perspectives. In the figure, there are two types of gradient, the red one and blue one, respectively standing for a collection from the original image and affine deformed image. From the illustration above, there is not so much difference among these two types of gradient by a gradient collection and interpolation. The affine deformation upon the image is $\begin{bmatrix} 1 & 0.8 \\ 0 & 1 \end{bmatrix}$.

To make the gradient collection more accurate, we illustrate more outcomes of the gradient collection on images of different perspectives.

Figure 4.10 presents the accuracy of affine gradient operation. It is a typical result under the effect of interpolation noise. The curves are quite similar to those representing the performance of affine Gaussian scale space, implying some connections between each other. Similarly, the main limitation to accuracy is also from the interpolation noise which can be reduced to some extent by the Gaussian filter during gradient generation. The level of the noise reduction depends on the size of the Gaussian kernel, which also reflects the feature scale. This can be used to explain why with the growth of the scale, the signal to noise ratio becomes better, whereas the increase of skewing has no obvious influence.

It has been proved that a low pass filter is effective to reduce the interpolation noise in the previous chapters. To compensate the noise effect on the affine gradient, we would like to employ some low pass filters during the affine gradient generation. One typical filter is low pass Gaussian filter, which is also fundamental to create the scale space. What is more, the main target of affine gradient operation is to approximate the gradient from the referred images. Thus this additional Gaussian filter will also be affine transformed to compensate for the view point difference. This additional Gaussian filter is equivalent adding an extra value to the detected scales from which the gradient was created.With this added scale the gradient is still identifiable because this additional scale will be added to all the scale space. What is more, the gradient of different scale is still

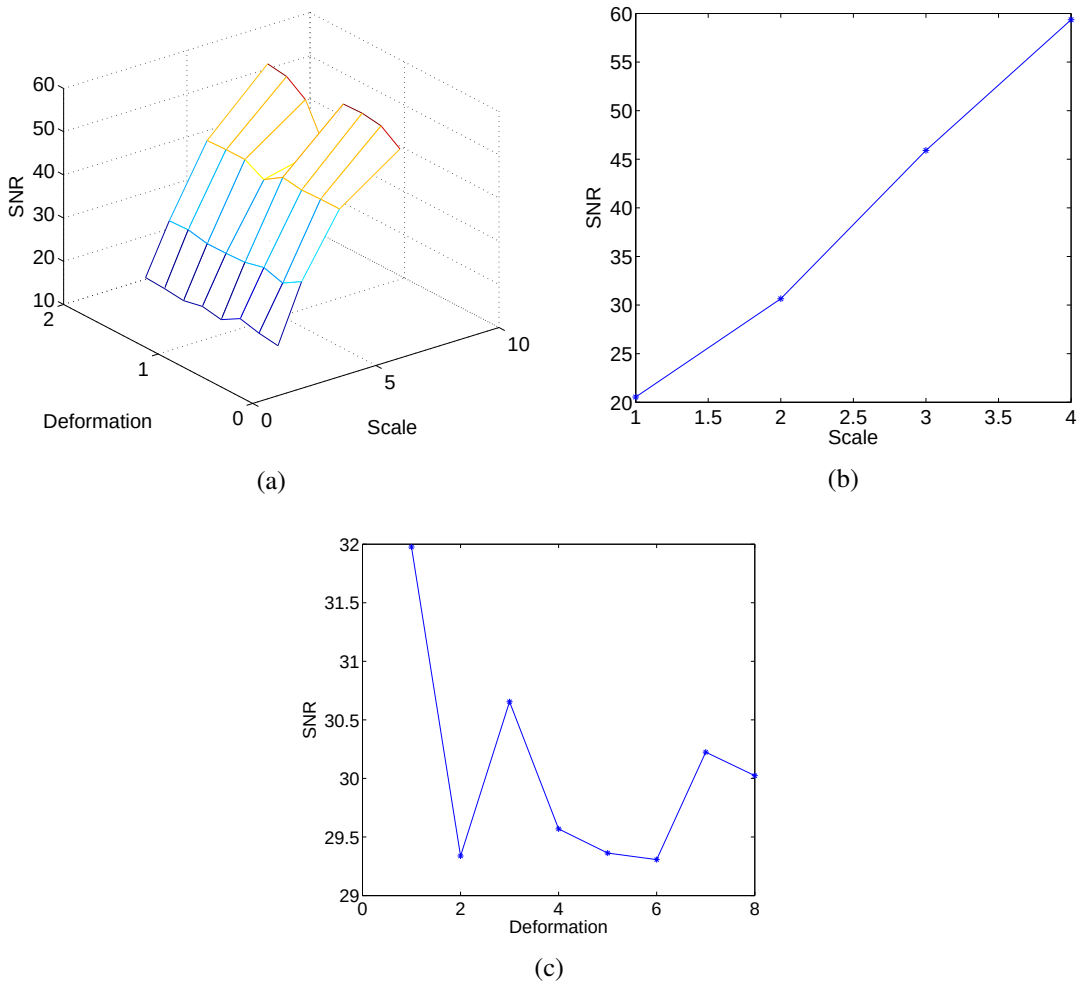


Figure 4.10. Affine gradient accuracy. (a) shows the accuracy under different scales and deformation. (b) shows accuracy under different scales. (c) shows accuracy under different deformation.

distinctive between each other. In this way, the interpolation noise can be further reduced.

Figure 4.11 presents the performance of the affine Gaussian gradient parametrized by an additional scale. The SNR becomes higher compared with the corresponding performance without the additional scale. The Figure shows that the precision of gradient can be effectively improved by the additional scale. This improved gradient can be used to build the affine descriptor.

4.5 Orientation assignment

The affine Gaussian gradient re-aligned by gradient collection and interpolation can now be used to build the affine descriptor. Generally, before the descriptor, each feature should be assigned with an orientation, reflecting the main direction of the gradient around it. This orientation is the angle which compensates the direction difference upon the gradient area from images having

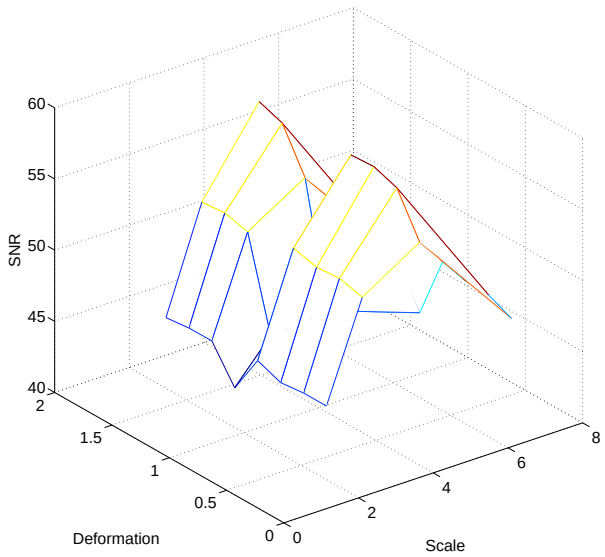


Figure 4.11. Affine gradient performance with an additional scale.

different rotations. This procedure helps to enhance the invariance to image rotations.

To calculate a SIFT descriptor, an orientation histogram with 36 bins will be formed, with each bin covering 10 degrees. Each sample in the neighboring window added to a histogram bin is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the key-point. The peaks in this histogram correspond to dominant orientations. Once the histogram is filled, the orientations corresponding to the highest peak and local peaks that are within 80% of the highest peaks are assigned to the key-point. In the case of multiple orientations being assigned, an additional key-point is created having the same location and scale as the original key-point for each additional orientation [12].

In our proposal, the orientation assignment is also based on the SIFT scheme. Since the neighbourhood to collect the gradient for the orientation assignment has been pre-defined, we just need to further specify the corresponding area on the affine deformed image. That area can be specified though an affine alignment on the plan. Once the area is fixed, that image area can be processed with the affine Gaussian gradient operation and the successive gradient collection and interpolation. The gradient obtained after these procedures can then be used as the approximation of the gradient from the original image. With this approximation of the gradient from the original image, the orientation assignment operation can be further processed based on the SIFT. At the very beginning, a histogram with 36 bins will be created with each bin covering 10 degrees. Each entry of the histogram will be weighted by its magnitude in order to stabilize the main orientation estimation. The highest peak of the histogram will be referred as the main orientation of the feature and the local peaks within 80% of the highest peak will be referred as the multiple orientations of that feature. By statistics, 15% of the features will be assigned with more than one orientations [12].

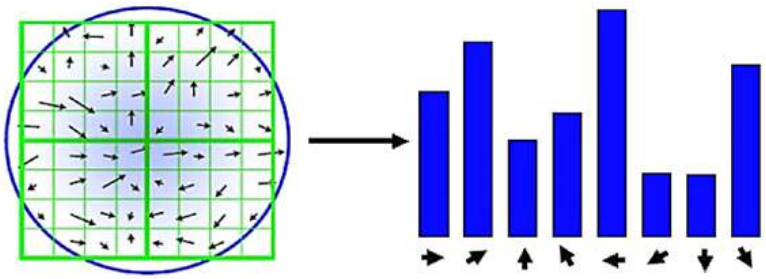


Figure 4.12. Orientation assignment by SIFT

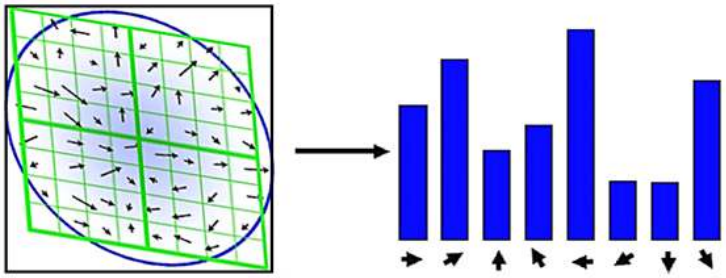


Figure 4.13. Orientation assignment by our proposal

Figure 4.12 illustrates the main procedures of orientation assignment operated by SIFT, including the creation of the orientation histogram with each bin weighted by the gradient magnitude. Figure 4.13 illustrates our proposed orientation assignment operation. The main difference between the typical SIFT assignment and our proposed assignment lies in the gradient re-alignment. Once the gradients are collected from the re-ordered positions, a histogram of gradient orientation can be established with the peak referred as the dominant orientation.

In the Figure 4.14, figure (a) presents the histogram created based on the conventional gradient. The peak of the histogram points to 185°. Figure (b) presents the collected gradient by affine Gaussian gradient operator from the affine deformed image. The peak also point to 185°. Figure (c) also presents the collected gradient by affine Gaussian gradient operator from an affine deformed image with a 30° rotation. The peak of this histogram corresponds to 155°, perfectly reflecting the 30° counter-clockwise rotation.

4.6 Affine descriptor

The area to compute the descriptor invariant to scale, rotation and partially to illumination can be fixed once the orientation has been assigned. For example, SIFT descriptor is computed by a rectangular grid laid out in the image domain, centered at the interest point, with its orientation determined by the orientation assignment and with the spacing proportional to the detection scale of the interest point. From experiments, Lowe [12] found that a 4×4 grid is often a good choice.

For each point on this grid, a histogram of local gradient directions at the scale of the feature

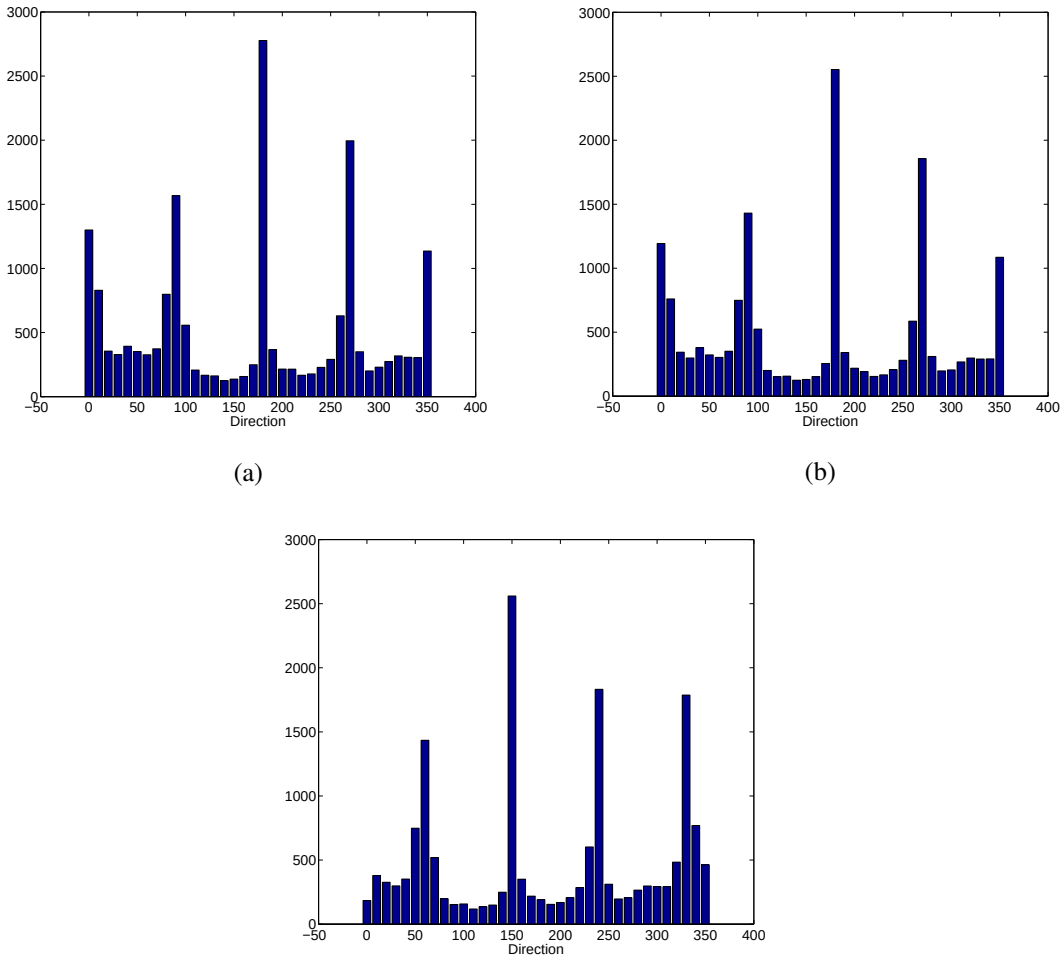


Figure 4.14. Affine gradient histogram. (a) shows gradient histogram from original image. (b) shows Affine gradient histogram from affine deformed image. (c) shows the affine gradient histogram from affine deformed image with 30° of rotation.

is computed over a pre-defined local neighborhood, with the gradient directions quantized into 8 discrete directions. During the accumulation of the histograms, the increments in the histogram bins are weighted by the gradient magnitude at each feature to give stronger weights to image points where the gradient estimates can be expected to be more reliable. To give stronger weights to gradient orientations near the interest point, the entries in the histogram are also weighed by a Gaussian window function centered at the feature and with its size proportional to the detection scale. Taken together, the local histograms computed at all the 4×4 grid points and with 8 quantized directions lead to an image descriptor with $4 \times 4 \times 8 = 128$ dimensions for each feature. This 128 dimensional vectors is referred to as the SIFT descriptor[12].

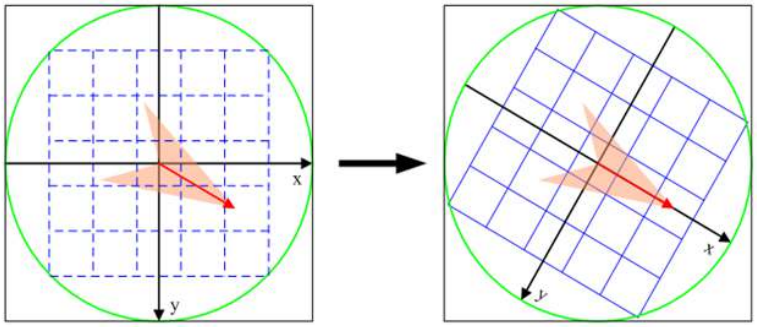


Figure 4.15. Square neighborhood for descriptor rotating to the orientation of the detected feature.

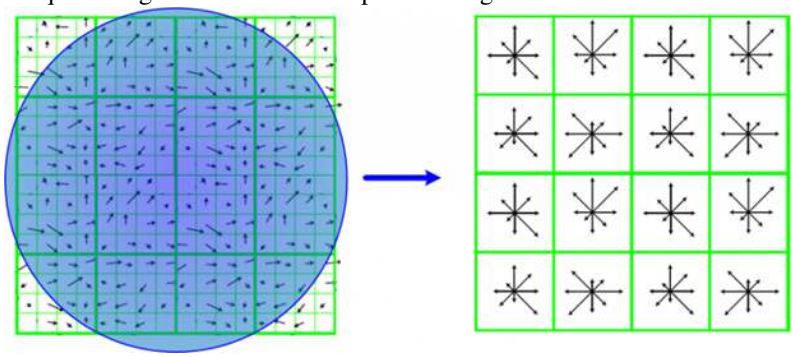


Figure 4.16. Descriptor created on the gradient from the rotated square neighbourhood.

Figures 4.15 and 4.16 illustrate how the SIFT descriptor is computed from few sampled gradients over a local adapted grid at the detected scale, with the orientation determined by the dominant peak in a gradient orientation histogram. Similar to the procedures of the orientation assignment based on the affine Gaussian gradient, the first step to build the affine invariant descriptor is to define the area to collect the gradient. Differently from the area pre-determined to estimate the main direction, the rectangular grid to compute the descriptor is not pre-defined but rotated by the assigned orientation. In the pipeline, the rectangular grid will be rotated to the feature's assigned orientation to remain the same direction after any rotation. This makes it more complicated to define the corresponding area on the affine deformed images, since rotation and affine transformation both get involved.

To simplify the definition of the area to collect the gradient from the non-deformed images, we will define a maximum area which covers all the image rotation cases. In this way, this maximum area represents the corresponding area to collect the affine Gaussian gradient for our proposed feature descriptor.

Figure 4.17 presents the area to computer the descriptor. In the figure, σ stands for the detected scale, d is the number of the sub-regions on each size and $m\sigma$ stands for the size of each sub-region ($d = 4, m = 3$ in SIFT).

The coordinate system within the maximum area will also be rotated to define the precise area

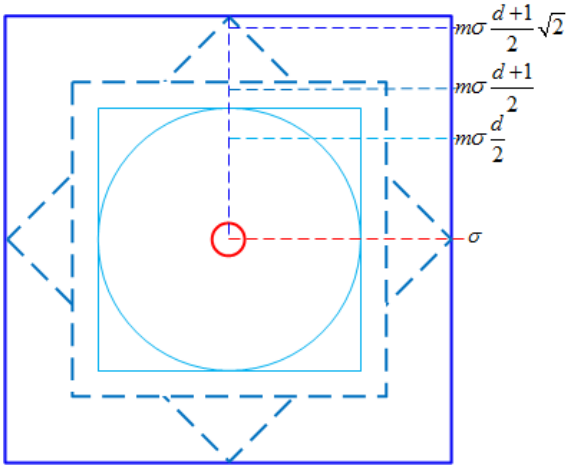


Figure 4.17. The area to compute the descriptor.

for the gradient collection. The relation of the coordinate system is given by

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{4.16}$$

The transformation of the coordinate system can also be seen as an affine transformation upon the image, because a linear transformation of the coordinates will be processed on each point in that coordinate system. Coordinates rotation is equal to image rotation and coordinates skewing is also equal to image skewing. Therefore, affine Gaussian gradient operation is suitable for the gradient adaptation with compensation of image linear transformation. Since the coordinates transformation for rotation is equal to an image rotation, and image rotation is a special case of affine transformation, then affine Gaussian gradient operation can also be used for the area rotation. In other words, the affine gradient generation and rotation transformation can be combined in the affine Gaussian gradient operation.

The new affine Gaussian gradient operation combined with the coordinate transformation can be specified by the new affine kernel,

$$A' = A \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{4.17}$$

Figure 4.18 presents the combination of affine gradient collection and rotation transformation. After this operation, the descriptor can be formed following the pipeline of SIFT descriptor. In this way, a feature descriptor based on the affine adapted gradient can be created given the affine deformation. The design of this feature descriptor aims to construct a descriptor compatible with affine scale space based feature detection.

Only when the affine feature detection works well, and is matched with an appropriate descriptor, the feature detection algorithms can become very resilient to viewpoint changes. On the other hand, the affine descriptor is designed in the framework of SIFT descriptor, thus it can be deemed as an extension of SIFT for view-point robustness, thus fully compatible with SIFT which has been widely used in many applications.

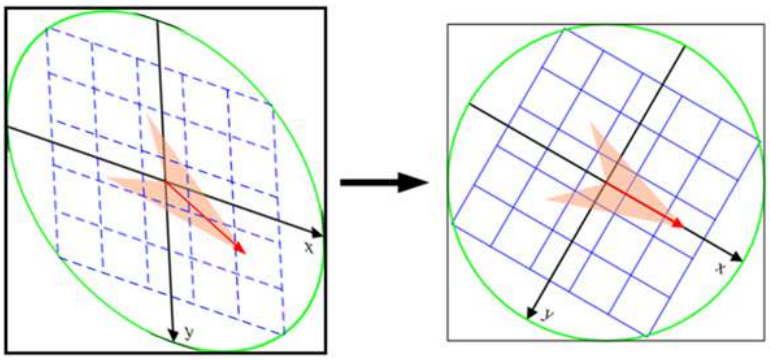
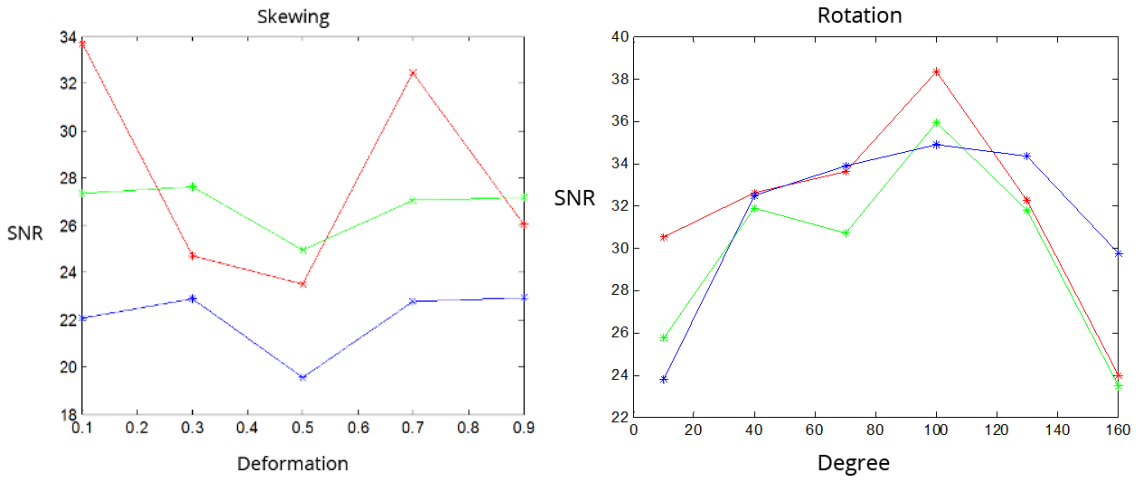


Figure 4.18. The combination of affine gradient and coordinates transformation.



(a) (b)

Figure 4.19. The accuracy of affine descriptor. Figure (a) presents the accuracy for different skewing. Figure (b) presents the accuracy for different rotations.

Figure 4.19 presents the affine descriptor accuracy to skewing and rotation respectively. It should be noticed that, when we refer to rotation, it is not pure rotation but an additional rotation contained in an affine transformation. Normally, an affine transformation can be divided as the product of pure rotation and pure skewing. The invariance to pure rotation depends on the orientation assignment which determines the direction of the rectangular area where the gradient is calculated. But the invariance to skewing can only be accomplished by affine gradient operation. The experiments set on rotation and skewing aim to evaluate the accuracy on the robustness of orientation assignment and affine gradient operation. In Figure (a), there are three curves, respectively representing the additional rotation of 30°, 60° and 45°. In Figure (b) there are also three curves, representing performance of skewing equal to 0.1, 0.2 and 0.5 in the vertical direction. Overall speaking, there is no distinction between each other and the signal to noise ratio ranges from 20 to 40 dB, which is an acceptable level to feature matching. A further evaluation of the matching

precision will be tested in the following.

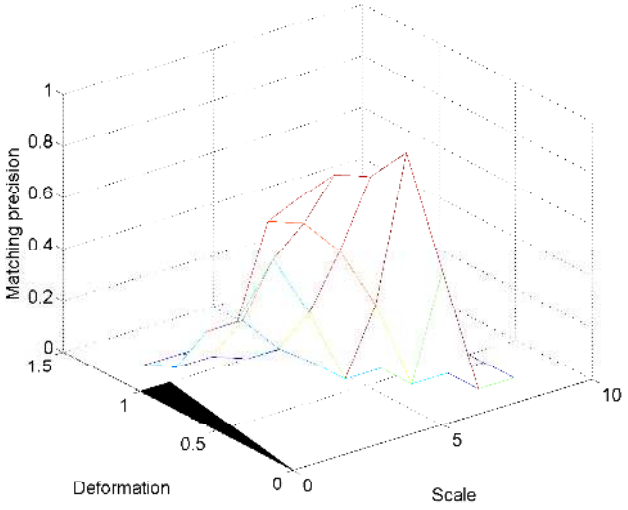


Figure 4.20. The matching precision of SIFT descriptor.

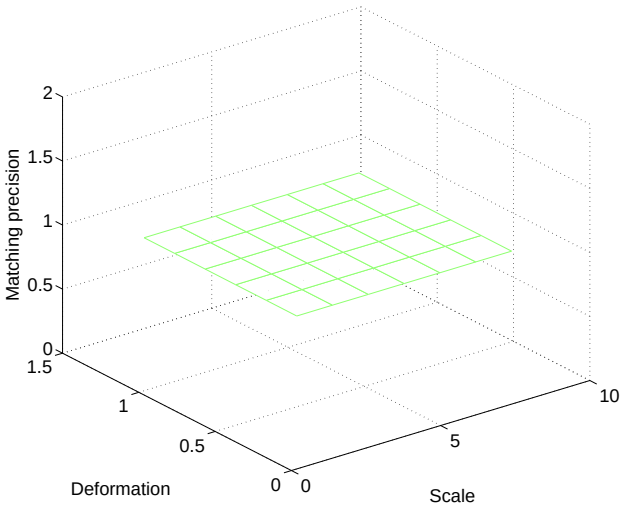


Figure 4.21. The matching precision of affine descriptor.

Figures 4.20 and 4.21 respectively present the matching precision of SIFT descriptor and the proposed affine descriptor. In the experiment, we have randomly selected 200 pairs of key points from the reference image and affine image to evaluate the descriptor's matching precision. Among all of these selected pairs, only 1 pair is a correct match. We have repeated the experiment at different scales and deformations to have an overall assessment of the affine descriptor. From the graphs

illustrated above, the SIFT descriptor only achieves a good performance at large scale and low affine deformations. The larger the scale and smaller the deformation undergone by the affine image, the higher the matching precision. On the contrary, the result of the affine descriptor is excellent since the only matching pair has always been successfully matched among the other 200 non-matching pairs at all scales and deformations. Compared with the SIFT descriptor, the matching precision of the affine descriptor is very encouraging. It points out that an affine adaptation on the gradient generation is a sound approach to cope with the effects of view-point variation.

Combining the affine detection and affine descriptor, this affine SIFT extension can be used to handle image search with images of difference perspectives. It can be used in a lot of real applications especially when the deformation is given.

Chapter 5

Conclusion

In this thesis, we have tried to explore the principles behind the sensitivity to view-point change of the current content based image retrieval algorithms and tried to enhance their robustness to view-point changes by a post processing stage using Homography based back-projection, and affine scale space based feature detection and description algorithms. The main purpose of this thesis is to truly reveal the reason behind the sensitivity of content-based visual search techniques and to improve their performance of retrieval precision under different affine transformations.

By the introduction of Homography based back projection, we have also introduced the mathematical model of affine transformation in the form of a 2 dimensional matrix, which is also the foundation of affine scale space. This mathematical description simplified the change of view point as an affine transformation, which makes it possible to simulate the images of different perspectives and assess the corresponding performance of detection and description. What is more, this mathematical model helps to explore the inner principle to the feature detection and description of the content based visual search techniques. The proposed Homography based back projection algorithm can be used to detect the affine transformation a posteriori. When integrated into the standard pipeline of CDVS it can relieve the computational complexity compared with the simulation of all the possible perspectives by ASIFT [23].

At the same time, this Homography based post processing stage can be used to improve the retrieval performance on the visual content of different perspectives, but it does not resolve the problem in a general way. Therefore, we have also explored the theory of affine scale space and designed the practical structure for the feature detection and description. The affine skewed Gaussian filter was first proposed by Tony Lindeberg [15] and applied by Krystian Mikolajczyk [22]. This skewed Gaussian was used as an adaptation to the affine transformations in affine Harris. But the result does not meet the requirement for the detection. Considering that a simple affine adaptation on the normal scale space does not provide enough improvement on the robustness to different perspectives, we have tried to established an affine scale space, completely based on skewed affine Gaussian blurred images which directly reflect the corresponding affine transformations. In a way, the affine skewing in the filter compensate the affine distortion from the image, thus the affine skewed scale space can well capture the normal features from the affine transformed images. In the thesis, we have proposed four implementation structures to construct the affine scale space, two structures in spatial domain and two in frequency domain. Since the Laplacian operator is not easy to be affine skewed in spatial domain, we have also proposed the corresponding affine LoG. The

computation complexity for the affine scale space is of the same level of the pyramid structure for normal scale space.

Once a feature has been detected, it requires a suitable descriptor for its identification in the visual content based retrieval system. For the same reason, the current image gradient based descriptor is also not robust to an affine transformation, since the descriptor is not calculated by the gradient from an affine skewed scale space. The calculated gradient can not well represent the image structures from a specific perspectives, which cannot be present by a normal scale space. In the third chapter, we have also proposed an affine gradient based affine descriptor. The affine gradient is obtained by the derivation of the detected affine scale space representing the structure of the affine transformed image. Combing this with some gradient skewing collection and interpolation, we form our affine descriptor, which inherits the main characteristics of the common SIFT descriptor. It owns the same form of SIFT descriptor, thus it is compatible with the CDVS [18] retrieval standard which has introduced a lot of other techniques for quick and accurate retrieval.

By combining the pipeline of both the affine detection and affine description, a fully affine robust visual search technique can be formed based on the theory of affine transformation and affine scale space. Overall speaking, scale space structure is the kernel for the scale invariant visual search. Combining this with affine adaption of the scale space can then be used to well represent the multi-scale image and also well capture the structures from different perspectives.

5.1 Future work

In this thesis, we have designed several different structures to build the affine scale space for a certain affine transformation and re-designed the affine scale space based feature detection and descriptor. The efforts we have made are successful to improve the robustness to view point changes for visual search techniques. But there are still several unsolved problems we need to confront with.

1. The affine scale space can represents the details of the image steering to a certain view point, but the view point difference should be previously known for the feature detection and descriptor. Except the simulated images, the view point difference in practice is hard to calibrate.

2. The homography matrix we are using to specify the view point difference not only contains the information of view point difference but also introduces the change of camera distance. By using the homography matrix to steering the scale space, it will also bring the change of camera distance, which will influence the scale ranges for the feature detection. Meanwhile, the camera distance will also affect the pre-defined area for the image gradient collection to form the feature descriptor.

3. The state of the art visual search techniques have a certain level of view point robustness. Within the range of their robustness, the implementation of affine scale space will not improve the image matching precision. On the other hand, the affine scale space is more complex and time consuming to implement. It should be figured out when the affine scale space will be used to detect and describe the features and when it will not be.

To the first issue, we will establish an integrated structure combining all the scale space from different perspectives in order to capture the image details not only from different scales but also

from different perspectives. In this way, the view point difference does not need to be previous known because the features from any potential view point will be detected. This integrated structure is demonstrated in the figure 5.1. In theory, a complete perspective projections of an image will cover the view point for a half sphere from a certain distance. Thus we also need to explore how sparsely to set up the affine scale space of different view points in order to economically capture the features of the whole visual sphere.

To the second issue, we will try to define a certain distance for the view point changes by specifying a freedom of homography matrix in order to prevent the scale changes due to the affine adaptation. By normalizing the homography matrix to define the camera distance, the scale range will not be affected by the affine adaptation on the scale space. Thus the feature detection and descriptor will be within the same range of scales and this will guaranteer the accuracy of detection and descriptor for different view point. In advance we will also try to establish some special structures to simplify the implementation, just like the structure to implement the affine scale space. The affine scale spaces from different perspectives are also mathematically related, we will explore their inner relations to simplify the construction process.

To the third issue, we need to do more experiments to evaluate the specific performance of the visual search techniques on different level of image skewing and tilting. With this performance evaluation, it can be specified when to implement the affine scale space and when to implement the conventional scale space to speed up the whole image retrieval system.

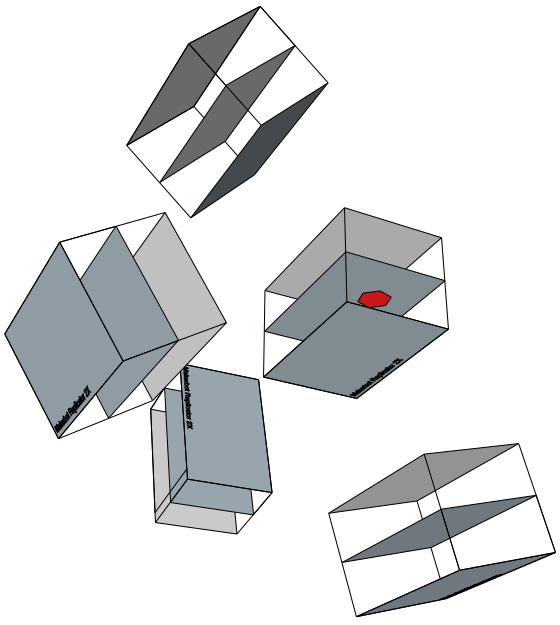


Figure 5.1. Scale spaces from different perspectives.

There are also a lot of real applications based on the view-point robust visual search techniques, including 3D objects reconstruction, navigation, auto driving etc. We will also keeps our mind on

the research of that areas.

Bibliography

[1] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Computing Surveys (CSUR)*, vol. 40, no. 2, pp. 5, 2008.

[2] Michael S Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain, "Content-based multimedia information retrieval: State of the art and challenges," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, vol. 2, no. 1, pp. 1–19, 2006.

[3] Venkat N Gudivada and Vijay V Raghavan, "Content based image retrieval systems," *Computer*, vol. 28, no. 9, pp. 18–22, 1995.

[4] Tony Lindeberg, "Edge detection and ridge detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 117–156, 1998.

[5] Wolfgang Förstner and Eberhard Gülch, "A fast operator for detection and precise location of distinct points, corners and centres of circular features," in *Proc. ISPRS intercommission conference on fast processing of photogrammetric data*, 1987, pp. 281–305.

[6] Tony Lindeberg, "Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention," *International Journal of Computer Vision*, vol. 11, no. 3, pp. 283–318, 1993.

[7] Krystian Mikolajczyk, Andrew Zisserman, Cordelia Schmid, et al., "Shape recognition with edge-based features," in *British Machine Vision Conference (BMVC'03)*, 2003, vol. 2, pp. 779–788.

[8] Frédéric Jurie and Cordelia Schmid, "Scale-invariant shape features for recognition of object categories," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. IEEE, 2004, vol. 2, pp. II–90.

[9] Lionel Gueguen and Martino Pesaresi, "Multi scale harris corner detector based on differential morphological decomposition," *Pattern Recognition Letters*, vol. 32, no. 14, pp. 1714–1719, 2011.

[10] Stephen M Smith and J Michael Brady, "Susan—a new approach to low level image processing," *International journal of computer vision*, vol. 23, no. 1, pp. 45–78, 1997.

[11] Tony Lindeberg, *Scale-space theory in computer vision*, Springer, 1993.

[12] David G Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[13] Tony Lindeberg, "Generalized axiomatic scale-space theory," *Advances in Imaging and Electron Physics*, vol. 178, pp. 1, 2013.

[14] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and vision computing*, vol. 22, no. 10, pp. 761–767, 2004.

[15] Tony Lindeberg, "Feature detection with automatic scale selection," *International journal of computer vision*, vol. 30, no. 2, pp. 79–116, 1998.

[16] T. Lindeberg, "Scale Invariant Feature Transform," 2012.

[17] David G Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Ieee, 1999, vol. 2, pp. 1150–1157.

[18] etc Danilo Pau, Giovanni Cordara, "White paper on compact descriptors for visual search," Tech. Rep., MPEG, 04 2013.

[19] Giovanni Cordara Stavros Paschalakis, Gianluca Francini, "Test model 12: Compact descriptors for visual search," testmodel ISO/IEC JTC1/SC29/WG11/N14961, MPEG, Strasbourg, France, 12 2014.

[20] Giovanni Cordara Stavros Paschalakis, Gianluca Francini, "Information technology — multimedia content description interface — part 13: Compact descriptors for visual search," ISO/IEC JTC 1/SC 29, 04 2014.

[21] Krystian Mikolajczyk and Cordelia Schmid, "A performance evaluation of local descriptors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 10, pp. 1615–1630, 2005.

[22] Krystian Mikolajczyk and Cordelia Schmid, "An affine invariant interest point detector," in *Computer Vision—ECCV 2002*, pp. 128–142. Springer, 2002.

[23] Guoshen Yu and Jean-Michel Morel, "Asift: an algorithm for fully affine invariant comparison," *Image Processing On Line*, vol. 2011, 2011.

[24] Ling-Yu Duan, Feng Gao, Jie Chen, Jie Lin, and Tiejun Huang, "Compact descriptors for mobile visual search and mpeg cdvs standardization," in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, May 2013, pp. 885–888.

[25] CuipingZhang RonKimmel, AlexanderM Bronstein, and Michael M Bronstein, "Affine invariant interesting descriptors," 2009.

[26] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool, "A comparison of affine region detectors," *International journal of computer vision*, vol. 65, no. 1-2, pp. 43–72, 2005.

[27] Adam Baumberg, "Reliable feature matching across widely separated views," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. IEEE, 2000, vol. 1, pp. 774–781.

[28] Paresh Kumar Jain and CV Jawahar, "Homography estimation from planar contours," in *3D Data Processing, Visualization, and Transmission, Third International Symposium on*. IEEE, 2006, pp. 877–884.

[29] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2003.

[30] Martin A. Fischler and Robert C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[31] Biao Zhao and E. Magli, "A homography-based cdvs pipeline for image matchingwith improved resilience to viewpoint changes," in *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*, Sept 2014, pp. 2215–2219.

[32] Alfons H Salden, Bart M ter Haar Romeny, and Max A Viergever, "Linear scale-space theory from physical principles," *Journal of Mathematical Imaging and Vision*, vol. 9, no. 2, pp. 103–139, 1998.

[33] Luc Florack, Robert Maas, and Wiro Niessen, "Pseudo-linear scale-space theory," *International Journal of Computer Vision*, vol. 31, no. 2-3, pp. 247–259, 1999.

[34] Tony Lindeberg, "Scale-space behaviour of local extrema and blobs," *Journal of Mathematical Imaging and Vision*, vol. 1, no. 1, pp. 65–99, 1992.

[35] Ramin Zabih and John Woodfill, "Non-parametric local transforms for computing visual correspondence," pp. 151–158. Springer, 1994.

[36] Serge Belongie, Jitendra Malik, and Jan Puzicha, "Shape matching and object recognition using shape contexts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 4, pp. 509–522, 2002.

[37] Yan Ke and Rahul Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. IEEE, 2004, vol. 2, pp. II–506.

[38] Rafael C Gonzalez, Richard E Woods, and Steven L Eddins, "Digital image processing using matlab," *Upper Saddle River, N. J: Pearson Prentice Hall*, 2004.

[39] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker, "Design of an image edge detection filter using the sobel operator," *Solid-State Circuits, IEEE Journal of*, vol. 23, no. 2, pp. 358–367, 1988.