**Features and Image Matching**

**I overview:**

**Features matching** or generally **image matching**, a part of many computer vision applications such as **image** registration, camera calibration and object **recognition**, is the task of establishing correspondences between two **images** of the same scene/object.

**II.Features matching:**

Features matching or generally image matching, a part of many computer vision applications such as image registration, camera calibration and object recogni- tion, is the task of establishing correspondences between two images of the same scene/object. A common approach to image matching consists of detecting a set of interest points each associated with image descriptors from image data. Once the features and their descriptors have been extracted from two or more images, the next step is to establish some preliminary feature matches between these images as illustrated in Fig. [14](#_bookmark13).

Without losing the generality, the problem of image matching can be formulated as follows, suppose that *p* is a point detected by a detector in an image associated with a descriptor

*Φ(p)* = {*φk(P)* | *k* = 1*,* 2*,..., K*} (29)

where, for all *K*, the feature vector provided by the *k*-th descriptor is

*φk(p)* = *(f k , f k ,..., f k*

1*p*

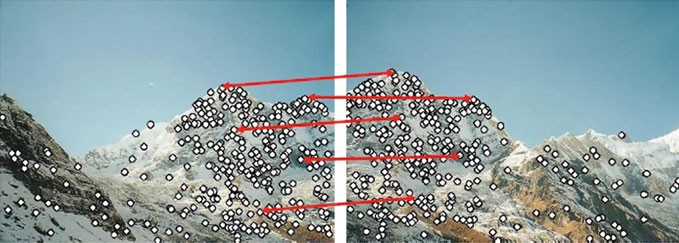
2*p*

*nk p*

*)* (30)

The aim is to find the best correspondence *q* in another image from the set of *N* interest points *Q q*1*, q*2*,..., qN* by comparing the feature vector *φk(p)* with those of the points in the set *Q*. To this end, a distance measure between the two interest points descriptors *φk(p)* and *φk(q)* can be defined as

= { }



**Fig. 1** Matching image regions based on their local feature descriptors [[79](#_bookmark88)] Ⓧc

2011 Springer

*dk(p, q)* = |*φk(p)* − *φk(q)*| (31)

Based on the distance *dk*, the points of *Q* are sorted in ascending order indepen- dently for each descriptor creating the sets

*Ψ (p, Q)* = {*ψk(p, Q)* | *k* = 1*,* 2*,..., k*} (32)

Such that,

*ψk(p, Q)* = .*(ψ* 1*,ψ*2*,...,ψ N )* ∈ *Q* | *dk(p,ψi )* ≤ *dk(p,ψj ),* ∀*i > j*Σ (33)

*k*

*k*

*k*

*k*

*k*

A match between the pair of interest points *(p, q)* is accepted only if (i) *p* is the best match for *q* in relation to all the other points in the first image and (ii) *q* is the best match for *p* in relation to all the other points in the second image. In this context, it is very important to devise an efficient algorithm to perform this matching process as quickly as possible. The nearest-neighbor matching in the feature space of the image descriptors in Euclidean norm can be used for matching vector- based features. However, in practice, the optimal nearest neighbor algorithm and its parameters depend on the data set characteristics. Furthermore, to suppress matching candidates for which the correspondence may be regarded as ambiguous, the ratio between the distances to the nearest and the next nearest image descriptor is required to be less than some threshold. As a special case, for matching high dimensional features, two algorithms have been found to be the most efficient: the randomized k-d forest and the fast library for approximate nearest neighbors (FLANN) [[78](#_bookmark87)].

On the other hand, these algorithms are not suitable for binary features (e.g., FREAK or BRISK). Binary features are compared using the Hamming distance calculated via performing a bitwise XOR operation followed by a bit count on the result. This involves only bit manipulation operations that can be performed quickly. The typical solution in the case of matching large datasets is to replace the linear search with an approximate matching algorithm that can offer speedups of several orders of magnitude over the linear search. This is, at the cost that some of the nearest neighbors returned are approximate neighbors, but usually close in distance to the exact neighbors. For performing matching of binary features, other methods can be employed such as [[80](#_bookmark89)–[82](#_bookmark90)].

Generally, the performance of matching methods based on interest points depends on both the properties of the underlying interest points and the choice of associated image descriptors [[83](#_bookmark91)]. Thus, detectors and descriptors appropriate for images con- tents shall be used in applications. For instance, if an image contains bacteria cells, the blob detector should be used rather than the corner detector. But, if the image is an aerial view of a city, the corner detector is suitable to find man-made structures. Fur- thermore, selecting a detector and a descriptor that addresses the image degradation is very important. For example, if there is no scale change present, a corner detector that does not handle scale is highly desirable; while, if image contains a higher level of distortion, such as scale and rotation, the more computationally intensive SURF

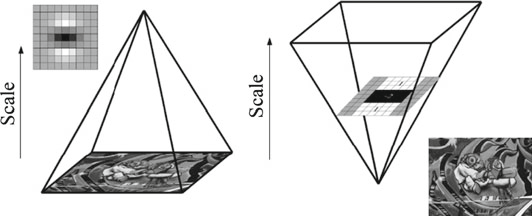
feature detector and descriptor is a adequate choice in that case. For greater accuracy, it is recommended to use several detectors and descriptors at the same time. In the area of feature matching, it must be noticed that the binary descriptors (e.g., FREAK or BRISK) are generally faster and typically used for finding point correspondences between images, but they are less accurate than vector-based descriptors [[74](#_bookmark83)]. Statis- tically robust methods like RANSAC can be used to filter outliers in matched feature sets while estimating the geometric transformation or fundamental matrix, which is useful in feature matching for image registration and object recognition applications

**Definitions and Principles:**

**Scale and Affine Invariance**

Actually, finding correspondences based on comparing regions of fixed shape like rectangles or circles are not reliable in the presence of some geometric and photo- metric deformations as they affect the regions’ shapes. Also, objects in digital images appear in different ways depending on the scale of observation. Consequently, scale changes are of important implications when analyzing image contents. Different techniques have been proposed to address the problem of detecting and extracting invariant image features under these conditions. Some are designed to handle scale changes, while others go further to handle affine transformations. In order to address the scale changes, these techniques assume that the change in scale is same in all directions (i.e., uniform) and they search for stable features across all possible scales using a continuous kernel function of scale known as scale space. Where, the image size is varied and a filter (e.g., Gaussian filter) is repeatedly applied to smooth sub- sequent layers, or by leaving the original image unchanged and varies only the filter size as shown in Fig. [2](#_bookmark1). More details about feature detection with scale changes can be found in [[36](#_bookmark45)], where a framework is presented for generating hypotheses about interesting scale levels in image data by searching for scale-space extrema in the scale normalized Laplacian of Gaussian (LoG).

On the other hand, in the case of an affine transformation the scaling can be dif- ferent in each direction. The nonuniform scaling has an influence on the localization,



**Fig. 2** Constructing the scale space structure

the scale and the shape of the local structure. Therefore, the scale invariant detec- tors fail in the case of significant affine transformations. Hence, detectors designed to detect the image features under uniform scaling need to be extended to be affine invariant detectors that can detect the affine shape of the local image structures. Thus, these affine invariant detectors can be seen as a generalization of the scale invariant detector.

Generally, affine transformations are constructed using sequences of translations, scales, flips, rotations, and shears. An affine transformation (affinity) is any linear mapping that preserves collinearity and ratios of distances. In this sense, affine indi- cates a special class of projective transformations that do not move any object from the affine space R3 to the plane at infinity or conversely. Briefly, the affine transfor-

mation of R*n* is a map *f* : R*n* → R*n* of the form

*f (Y)* = A*Y* + B (1)

for all *Y* R*n*, where A is a linear transformation of R*n*. In some special cases, if det*(*A*)>* 0, the transformation is called orientation-preserving, while, if det*(*A*)<* 0, it is orientation-reversing. It is well known that a function is invariant under a certain family of transformations if its value does not change when a transformation from this family is applied to its argument. The second moment matrix provides the theory for estimating affine shape of the local image features. Examples of affine invariant detectors are Harris-affine, Hessian-affine, and maximally stable extremal regions (MSER). It must be borne in mind that the detected features by these detectors are transformed from circles to ellipses.

∈

**Image Feature Detection:**

Feature detectors can be broadly classified into three categories: single-scale detec- tors, multi-scale detectors, and affine invariant detectors. In a nutshell, single scale means that there is only one representation for the features or the object contours

using detector’s internal parameters. The single-scale detectors are invariant to image transformations such as rotation, translation, changes in illuminations and addition of noise. However, they are incapable to deal with the scaling problem. Given two images of the same scene related by a scale change, we want to determine whether same interest points can be detected or not. Therefore, it is necessary to build multi- scale detectors capable of extracting distinctive features reliably under scale changes. Details of single-scale and multi-scale detectors as well as affine invariant detectors are discussed in the following sections.

**Harris Detection:**

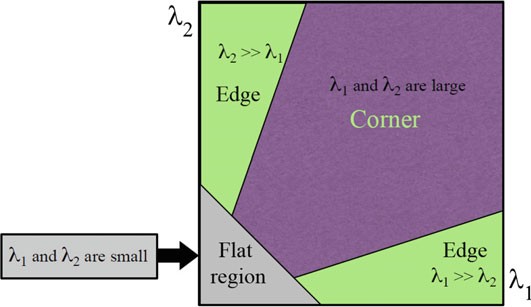
Harris and Stephens [[38](#_bookmark47)] have developed a combined corner and edge detector to address the limitations of Moravec’s detector. By obtaining the variation of the auto- correlation (i.e., intensity variation) over all different orientations, this results in a more desirable detector in terms of detection and repeatability rate. The resulting detector based on the auto-correlation matrix is the most widely used technique. The 2 2 symmetric auto-correlation matrix used for detecting image features and describing their local structures can be represented as

×

Σ ⎡ *I*2*(x, y) IxIy(x, y)* ⎤

*x*

*y*



**Fig. 4** Classification of image points based on the eigenvalues of the autocorrelation matrix *M*

where *Ix* and *Iy* are local image derivatives in the *x* and *y* directions respectively, and *w(u, v)* denotes a weighting window over the area *(u, v)*. If a circular window such as a Gaussian is used, then the response will be isotropic and the values will be weighted more heavily near the center. For finding interest points, the eigenvalues of the matrix *M* are computed for each pixel. If both eigenvalues are large, this indicates existence of the corner at that location. An illustrating diagram for classification of the detected points is shown in Fig. [4](#_bookmark3). Constructing the response map can be done by calculating the cornerness measure *C(x, y)* for each pixel *(x, y)* using

*C(x, y)* = det*(M)* − *K(trace(M))*2 (5)

where

det*(M)* = *λ*1 ∗ *λ*2*, and trace(M)* = *λ*1 + *λ*2 (6)

The *K* is an adjusting parameter and *λ*1*, λ*2 are the eigenvalues of the auto-correlation matrix. The exact computation of the eigenvalues is computationally expensive, since it requires the computation of a square root. Therefore, Harris suggested using this cornerness measure that combines the two eigenvalues in a single measure. The non- maximum suppression should be done to find local maxima and all non-zero points remaining in the cornerness map are the searched corners.

**Image Feature Descriptor**

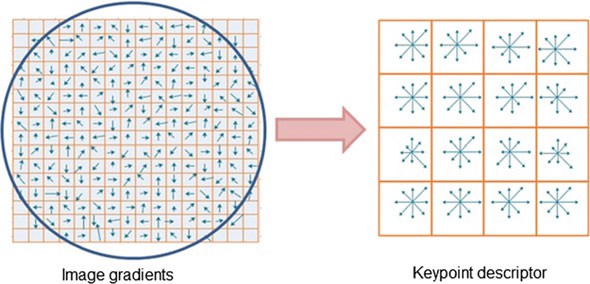
Scale Invariant Feature Transform (SIFT)

Lowe [[31](#_bookmark40)] presented the scale-invariant feature transform (SIFT) algorithm, where a number of interest points are detected in the image using the Difference-of-Gaussian (DOG) operator. The points are selected as local extrema of the DoG function. At each interest point, a feature vector is extracted. Over a number of scales and over a neighborhood around the point of interest, the local orientation of the image is estimated using the local image properties to provide invariance against rotation. Next, a descriptor is computed for each detected point, based on local image infor- mation at the characteristic scale. The SIFT descriptor builds a histogram of gradient orientations of sample points in a region around the keypoint, finds the highest orien- tation value and any other values that are within 80 % of the highest, and uses these orientations as the dominant orientation of the keypoint.

The description stage of the SIFT algorithm starts by sampling the image gradient magnitudes and orientations in a 16 16 region around each keypoint using its scale to select the level of Gaussian blur for the image. Then, a set of orientation histograms is created where each histogram contains samples from a 4 4 subregion of the original neighborhood region and having eight orientations bins in each. A

×

×



**Fig 4** A schematic representation of the SIFT descriptor for a 16 16 pixel patch and a 4 4 descriptor array

× ×

Gaussian weighting function with *σ* equal to half the region size is used to assign weight to the magnitude of each sample point and gives higher weights to gradients closer to the center of the region, which are less affected by positional changes. The descriptor is then formed from a vector containing the values of all the orientation histograms entries. Since there are 4 4 histograms each with 8 bins, the feature vector has 4 4 8 128 elements for each keypoint. Finally, the feature vector is normalized to unit length to gain invariance to affine changes in illumination. However, non-linear illumination changes can occur due to camera saturation or similar effects causing a large change in the magnitudes of some gradients. These changes can be reduced by thresholding the values in the feature vector to a maximum value of 0*.*2, and the vector is again normalized. Figure [4](#_bookmark7) illustrates the schematic representation of the SIFT algorithm; where the gradient orientations and magnitudes are computed at each pixel and then weighted by a Gaussian falloff (indicated by overlaid circle). A weighted gradient orientation histogram is then computed for each subregion.

× × =

×

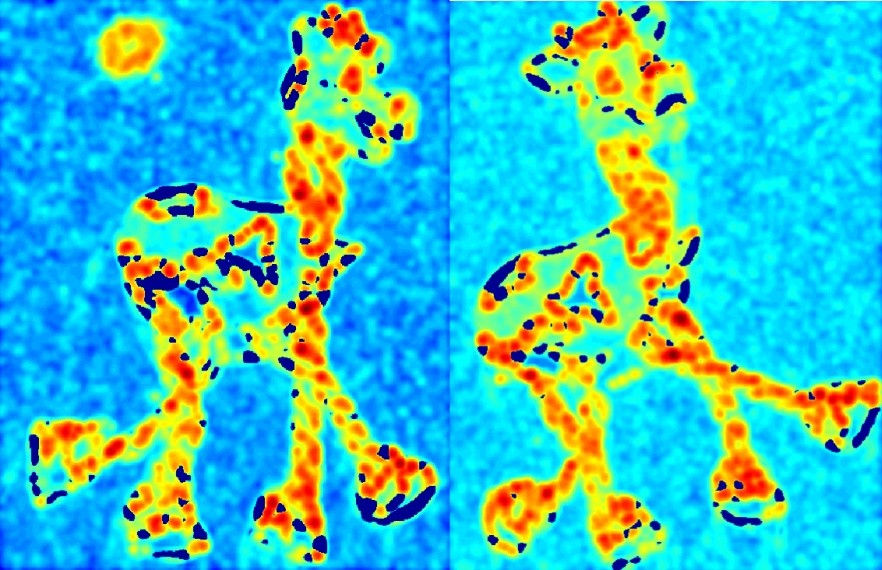
The standard SIFT descriptor representation is noteworthy in several respects: the representation is carefully designed to avoid problems due to boundary effects- smooth changes in location, orientation and scale do not cause radical changes in the feature vector; it is fairly compact, expressing the patch of pixels using a 128 element vector; while not explicitly invariant to affine transformations, and the representation is surprisingly resilient to deformations such as those caused by perspective effects. These characteristics are evidenced in excellent matching performance against com- peting algorithms under different scales, rotations and lighting. On the other hand, the construction of the standard SIFT feature vector is complicated and the choices behind its specific design are not clear resulting in the common problem of SIFT its high dimensionality, which affects the computational time for computing the descrip- tor (significantly slow). As an extension to SIFT, Ke and Sukthankar [[54](#_bookmark63)] proposed PCA-SIFT to reduce the high dimensionality of original SIFT descriptor using the standard Principal Components Analysis (PCA) technique to the normalized gradi-

ent image patches extracted around keypoints. It extracts a 41 41 patch at the given scale and computes its image gradients in the vertical and horizontal directions and creates a feature vector from concatenating the gradients in both directions. There- fore, the feature vector is of length 2 39 39 3042 dimensions. The gradient image vector is projected into a pre-computed feature space, resulting a feature vec- tor of length 36 elements. The vector is then normalized to unit magnitude to reduce the effects of illumination changes. Also, Morel and Yu [[55](#_bookmark64)] proved that the SIFT is fully invariant with respect to only four parameters namely zoom, rotation and translation out of the six parameters of the affine transform. Therefore, they intro- duced affine-SIFT (ASIFT), which simulates all image views obtainable by varying the camera axis orientation parameters, namely, the latitude and the longitude angles, left over by the SIFT descriptor.

×

× × =

Harris detection example:

**fHarris value (red high, blue low)**

**Threshold (fHarris > threshold val**

**III. Image Matching**

**Image Matching with OpenCV’s Template Matching**

In this article, we reviewed a somewhat out of the ordinary real-world data science problem which required a tailor-made solution. We dug into OpenCV’s Template Matching and understood how it can be used to fulfill the requirements of the problem at hand



**Blob detection technique:**

The results below show that Blob Detection methods require high contrast and particularly intense lighting to provide accurate results. The advantages of using Blob detection method is that it is suitable for cases which require lesser number of gestures because it is simpler to implement. It is also dynamic in nature. The main disadvantage of these blob detectors is that they are dependent on gray scale space. It also shows that blob detection is really sensitive to noise and leads to over segmentation in results. Hence, we move on to template matching, an improved version of blob detection methods.

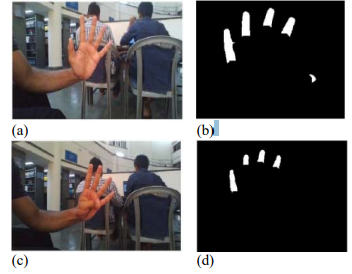


Fig 5 . (a) Hand showing 5 fingers, (b) Blob detection result of (a) where only 4 fingers have been identified, (c) Hand showing 4 fingers, (d) Blob detection result showing the 4 fingers been identified correctly

7 Conclusions

**IV:Conclusion:**

The objective of this chapter is to provide a straight-forward, brief introduction for new researchers to the image feature detection and extraction research ﬁeld. It introduces the basic notations and mathematical concepts for detecting and extract- ing image features, then describes the properties of perfect feature detectors. Vari- ous existing algorithms for detecting interest points are discussed brieﬂy. The most frequently used description algorithms such as SIFT,… are also discussed and their advantages/disadvantages are highlighted. Furthermore, it explains some approaches to feature matching. Finally, the chapter discusses the techniques used for evaluating the performance of these algorithm