Table of Contents

Feature Detection and Matching 1
- Feature Detectors 2
- Scale Invariance 3
- Rotational Invariance 3
- Feature Descriptors 4
  - The SIFT Algorithm 4
  - Scale Invariant Feature Detection 5
  - Feature Matching and Indexing 5
  - Cluster Identification 5
  - Hypothesis Verification 5
- SIFT Algorithm 5
- Feature Matching 7
- Feature Tracking 8
- Edge Detection 8
  - Edge Descriptors 9
  - Model of the Ideal Step Edge 9
  - Canny's Edge Detection Algorithm 9
  - Non-Maximum Suppression 9
  - Other Edge Detectors 10
  - Corner Points (or Blobs) 11
  - Laplacian of Gaussian 11

Feature Detection and Matching

Feature detection and matching are essential components of computer vision applications. Feature points usually form a set of sparse corresponding locations in different images. They can be used to align images, perform video stabilization, object recognition, and more.

There are two main approaches to finding feature points and their correspondences. The first uses a local search, such as correlation or least-squares. The second approach consists of independently detecting features in all images under consideration and then perform a match based on local appearances. The first approach is more suited to video imagery, where disparities are expected to be small while the second approach is capable of handling larger disparities. The detection and matching sequence consists of four sequential phases:

1. **Feature Detection**: Images are searched for locations likely to correctly match in other images
2. **Feature Description**: Regions where detected key points are found are converted into compact and invariant descriptors
3. **Feature Matching**: Search for likely matching candidates in other images

using descriptors

4. **Feature Tracking**: Search in small image regions around key points to evaluate their motion, from frame to frame

Feature Detectors

The simplest matching criterion is the sum of squared differences between two image regions. It is formulated as:

$$\mathrm{SSD}(\vec{u}) = \sum_{i \in Q} \omega(\vec{x}_i) \left[I_1(\vec{x}_i + \vec{u}) - I_0(\vec{x}_i) \right]^2$$

where Q is an image region, $\vec{u}$ is a displacement vector, ω is a (possibly Gaussian) weighted window, and I_0 and I_1 are the two images being compared. This metric is stable in the sense that it has a minimum in regions where there is enough variation in both the x and y directions. Over an edge, the minimum is represented by a line, and matching becomes uncertain. This is yet another manifestation of the aperture problem, this time in the context of correlating two surfaces.

This is exemplified by using auto-correlation:

$$\mathrm{S}(\vec{u}) = \sum_{i \in Q} \left[I_0(\vec{x}_i + \vec{u}) - I_0(\vec{x}_i) \right]^2$$

over say, an edge. The auto-correlation surface will not have a minimum. A Taylor series expansion of $I_0(\vec{x}_i + \vec{u}) \approx I_0(\vec{x}_i) + \nabla I_0(\vec{x}_i) \cdot \vec{u}$ approximates the auto-correlation surface:

$$\begin{aligned} S(\vec{u}) &= \sum_{i \in Q} \left[I_0(\vec{x}_i + \vec{u}) - I_0(\vec{x}_i) \right]^2 \\ &\approx \sum_{i \in Q} \left[I_0(\vec{x}_i) + \nabla I_0(\vec{x}_i) \cdot \vec{u} - I_0(\vec{x}_i) \right]^2 \\ &= \sum_{i \in Q} \left[\nabla I_0(\vec{x}_i) \cdot \vec{u} \right]^2 \\ &= \vec{u}^T A \vec{u} \end{aligned}$$

where $\nabla I_0(\vec{x}_i)$ is the image gradient at $\vec{x}_i$. The auto-correlation matrix A is written as:

$$A = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

The inverse of this matrix provides a lower bound on the uncertainty of in the location of a matching image region. An eigenvalue analysis of A demonstrates

this, as it has two eigenvectors $\vec{e}_1$ and $\vec{e}_2$. Each of these vectors also has an eigenvalue associated with it. In this case, λ_1 and λ_2 , where $\lambda_1 > \lambda_2$. Since the larger uncertainty depends on the smallest of the two eigenvalues, it is then correct to find maxima in the smallest eigenvalue to locate good features to track.

The uncertainty in the direction of the eigenvector $\vec{e}_2$ associated with the smallest eigenvalue λ_2 is proportional to:

$$\frac{1}{\sqrt{\lambda_2}}$$

The uncertainty in the direction of eigenvector $\vec{e}_1$ associated with the largest eigenvalue λ_1 is proportional to

$$\frac{1}{\sqrt{\lambda_1}}$$

Other, similar measures have been proposed. For instance

$$\frac{\det(A)}{\text{trace}(A)} = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

which turns out to be rotationally invariant and well behaved at edges.

Scale Invariance

Some images have very little high-frequency content at their original scale, or resolution. It it useful then to extract features at different scales. This can be accomplished by extracting features at various levels from an image pyramid. For most applications, the scale of the object to detect is unknown. Thus, features that are stable in both location and scale are excellent key points.

DoG (Difference of Gaussians) operator can be used to decompose an image into its frequency bands, each representing a different scale (or resolution). Extrema in the response of the LoG (Laplacian of a Gaussian) outputs across multiple scales indicate reliable features.

Rotational Invariance

Matching and object recognition algorithms need to be robust to rotations at least within the image plane. However, explicitly rotationally invariant operators map different-looking image regions to the same descriptor, creating problems. Alternatively, one can estimate a dominant orientation at each detected key point. Once the local orientation and scale of a key point is known, a scaled and

oriented image region can be used to form a feature descriptor. Probably the simplest orientation estimate is the average gradient within a region around a key point.

Scale and rotation invariance are sometimes insufficient for some applications which may require full affine invariance. Usually, the frame of reference used for the affine transformation is that composed of the eigenvectors of the Hessian matrix, computed at the location of the key point.

An affine transformation is written as:

$$\vec{y} = A\vec{x} + \vec{b}$$

and, using homogeneous coordinates, is concisely expressed as:

$$\begin{bmatrix} \vec{y} \\ 1 \end{bmatrix} = \begin{bmatrix} & A & & \vec{b} \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

Feature Descriptors

Once feature detection has been performed, we must be able to match them in other images under consideration. There are many feature descriptors in the literature. However, one that stands out for its success is the Scale-Invariant Feature Transform (SIFT).

We can think of SIFT as an algorithm to detect and describe local features in images. The main publication on SIFT is by D. Lowe, in the 1999 Proceedings of ICCV, pages 1150-1157.

For any object in an image, interesting points on the object can be extracted to provide a feature description of the object. These descriptions are obtained by the use of training images, in which the object in question is present or not. It is desirable to have some form of robustness with these descriptors, as they should be scale-invariant, noise-resilient, and illumination-invariant.

The SIFT Algorithm

- Key points of objects are first detected from a set of training images and stored into a database
- An object is recognized in a new image by comparing each feature from the new image to the database and by finding candidate matching features, using Euclidean distance
- Subsets of key points that agree on the object in terms of its location, scale, and orientation are identified

- Determining these clusters of consistent matches is performed with the generalized Hough transform
- The probability that a particular subset of features indicates the presence of the object is then estimated

Scale Invariant Feature Detection

- Transform an image into a large collection of feature vectors, invariant to effects mentioned earlier
- Key image locations (where features are found) are determined by localizing image areas where the DoG operator yields an extremum
- The DoG operator is applied in scale space onto a series of smoothed and re-sampled images inside a Gaussian pyramid
- Low contrast areas and edge points are discarded
- Dominant orientations are assigned to key points
- SIFT descriptors are obtained by considering regions around key locations

Feature Matching and Indexing

- Storing SIFT keys from the training stage
- Identifying keys from new image using k-d trees and best-bin first

Cluster Identification

- Uses a Hough transform to cluster reliable hypotheses and search for a particular model pose
- Each SIFT key point specifies a location, scale, and orientation

Hypothesis Verification

Each cluster is subject to verification performed by computing the least-squares solution of the affine model relating the model to the image

SIFT Algorithm

1. **Scale-Space Extrema Detection**:
    - Key points are detected at this stage
    - Images are convolved with Gaussian filters at different scales

- A DoG operator is applied at all scales
- Key points are those that have DoG extrema occurring at different scales

2. **Key Point Localization**:
   - Principal curvature of candidate key points allows to filter out those with low contrast or poor localization (along straight edges, for instance)
3. **Interpolation**:
   - Each candidate key point is interpolated for accurate localization
   - The extremum is located with a quadratic Taylor series expansion of the DoG scale space function $D(\vec{x})$ with candidate key point at the origin:

$$D(\vec{x}) = D + \frac{\partial D^T}{\partial x}\vec{x} + \frac{1}{2}\vec{x}^T\frac{\partial^2 D}{\partial \vec{x}^2}\vec{x}\Bigg|_{\vec{x}}$$

4. **Edge Responses**:
   - Computing the eigenvalues of the Hessian matrix yields the principal curvature:

$$H = \begin{pmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{pmatrix}$$

   The ratio of eigenvalues is used to discriminate edge points.

5. **Orientation Assignment**:
   - Each key point is given one or more orientations based on local image gradient directions
   - The Gaussian smoothed image $L(\vec{x},\sigma)$ at the key-point's scale σ is used and all computations are performed in a scale-invariant manner
   - For an image $L(\vec{x},\sigma)$ at scale σ , the gradient magnitude and orientation are computed as pixel differences:

$$m(\vec{x}) = \sqrt{[L(x+1,y)-L(x-1,y)]^2+[L(x,y+1)-L(x,y-1)]^2}$$

$$\theta(\vec{x}) = \tan^{-1}\left[\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right]$$

- These calculations are performed for every pixel in the region around the key points in $L(\vec{x},\sigma)$. Following this step, an orientation histogram is formed. Peak orientations within this histogram correspond to dominant orientations and are assigned to the key point.

Feature Matching

We assume that the feature descriptors have been designed so that Euclidean (vector magnitude) distances in feature space can be directly used to rank potential matches. Given an Euclidean distance metric, the simplest matching strategy is to set a threshold and eliminate all matches not within this threshold value. Too low a threshold results in many false positives, or incorrect matches being considered as correct. On the other hand, too high a threshold results in false negatives, or correct matches being missed. If we use the following definitions:

- TP: True Positives (number of correct matches)
- FN: False Negatives (number of matches not correctly detected)
- FP: False Positives (number of matches that are incorrect)
- TN: True Negatives (number of non-matches correctly rejected)

we can evaluate the performance of a matching algorithm for a particular threshold:

- TPR: True Positive Rate: $TPR = \frac{TP}{TP + FN}$
- FPR: False Positive Rate: $FPR = \frac{FN}{FP + TN}$
- PPV: Positive Predictive Value: $PPV = \frac{TP}{TP + FP}$
- ACC: Accuracy: $ACC = \frac{TP + TN}{TP + FN + FP + TN}$

Any matching strategy at any threshold can be evaluated with the TPR and FPR numbers. The perfect situation is depicted when the TPR is 1.0 and the FPR is 0. As the threshold is varied, we obtain what is called a Receiver Operating Characteristic (ROC) curve.

Setting the matching threshold can be difficult as it depends partly on the imagery being processed. Ideally, its value should adjust depending on the different regions of the feature space.

Searching for matches can be computationally expensive. For instance, the simplest way would be to compare each feature with every other feature in each pair of potential matches. This is quadratic in the number of extracted features and not practical for most applications, including real-time ones.

One strategy to perform the matching efficiently is to devise an indexing structure that allows for a fast search for features that are near the feature we are trying to match. Hash tables and multi-dimensional trees are commonly used for that purpose. These data structures operate in the feature space, not in the image space directly.

Once a set of core (most reliable) matches have been determined, they can be used to densify the matching set. Less reliable matches that are consonant with the core matches can then be accepted. A technique such as RANSAC can be used for this purpose.

Feature Tracking

In sequences of images (such as video) it often makes sense to track the displacement of individual features from image to image. Good features to track are often found in the same way as good features to match. For matching techniques to work well, it is often the case that frame-to-frame displacements be small. If this assumption cannot be made, then it is natural to turn to scale space and perform the initial tracking at low resolution in order to propagate displacement estimates at finer scales progressively. This may be accomplished with the use of a Gaussian or Laplacian image pyramid.

One of the latest developments in feature tracking is the use of learning algorithms to build special-purpose recognizers (or classifiers) to rapidly search for features in entire images. Much larger displacements can be handled in this way. However, training on sets of images must be performed ahead of time. Coupling these classifiers with structure-from-motion techniques do increase the stability of 3D solutions.

Edge Detection

Edge points are pixels at or around which the image values undergo sharp variations. There are three steps to edge detection. First, it is required that we suppress noise in some way, then detect and enhance edges, and finally localize them accurately.

To perform noise suppression, we may convolve the image with a 2D Gaussian kernel, for instance. Edge enhancement may involve the design of filters that

respond to edges. Localization involves deciding which local maxima in the filter's response field are edges. This involves thinning and thresholding on local maxima. A very well known edge detection algorithm is Canny's edge detector.

Edge Descriptors

An edge descriptor would contain attributes such as a) the normal direction to the edge (and conversely, its direction), b) the edge center, and c) the edge strength. The edge center describes the location of the edge, while the edge strength is a measure of local image contrast at the edge location.

Model of the Ideal Step Edge

Mathematically, a step edge (in 1D here) may be described by a discontinuous function:

$$G(x) = \begin{cases} 0 & x<0 \\ A & x\geq 0 \end{cases}$$

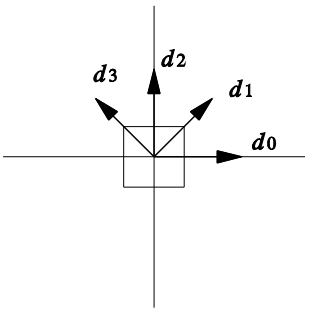
We generally assume that the edge enhancement filter is linear, and that the image noise is additive, white-Gaussian. In designing algorithms for noise detection, we need to minimize the probability of false positives. In addition, locating the edge must be bias-free.

Canny's Edge Detection Algorithm

The algorithm's input is an image I . Let G be a Gaussian with zero mean and standard deviation σ . Then $J=I*G$ apply a convolution of the original image with the Gaussian. And, for each pixel $J(i,j)$:

- Compute the gradient $\nabla J = (J_x, J_y)^T$
- Estimate the edge strength $e_s(i,j) = \|\nabla J\|_2$
- Estimate edge normal direction $e_o(i,j) = \tan^{-1}\left(\frac{J_y}{J_x}\right)$

Non-Maximum Suppression



Consider the two outputs from Canny's detector, e_s and e_o , and the four basic directions (d_0, d_1, d_2, d_3) from an image location (i,j) .
We need to find d_k that approximates best $e_o(i,j)$, the

edge orientation. If $e_s(i,j)$ is smaller than one of its two neighbors along d_k, then $I_n(i,j)=0$, else $I_n(i,j)=e_s(i,j)$ (See figure on the right).

To further enhance edge detection, we can use hysteresis thresholding. The input to this enhancement is I_n, the edge map that has been non-maximum suppressed, $e_o(i,j)$, the edge orientation, and two thresholds, τ_l and τ_h, such that $\tau_l<\tau_h$. Then, for all points in I_n, and scanning it in a fixed order:

- Locate the next unvisited edge point $I_n(i,j)$ such that $I_n(i,j)>\tau_h$
- Starting at that point, follow the chain of connected local maxima in both directions perpendicular to the edge normal, as long as $I_n(i,j)>\tau_l$
- Mark all visited points, and save a list of the locations of all points in the found contour

The output of this method is a set of lists, describing the positions of connected contours.

Other Edge Detectors

Roberts edge detector can be applied by following these steps:

- Filter the image with the following kernels: $\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ and $\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$ leading to two images I_1 and I_2
- Compute the magnitude of the gradient $G(i,j)=\sqrt{I_1^2(i,j)+I_2^2(i,j)}$
- edges are located with $G(i,j)>\tau$

With Sobel's detectors, only the kernels change. Hence, apply Roberts method

with the following kernels:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 2 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & -1 \end{bmatrix}$$

These methods locate edges with pixel precision only. When more precise edge locations are required, we resort to interpolation techniques for achieving sub-pixel accuracy.

Corner Points (or Blobs)

An image corner is an area where there is significant variation in both the x and y directions. Finding such image points involve the following steps:

- Form the matrix $\begin{bmatrix} \sum_Q I_x^2 & \sum_Q I_x I_y \\ \sum_Q I_y I_x & \sum_Q I_y^2 \end{bmatrix}$ over a region Q . This matrix is symmetric and can be diagonalized by a rotation yielding $\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$ where λ_1 and λ_2 are eigenvalues
- Eigenvectors are found as $C\vec{x}_i = \lambda_i \vec{x}_i$, where $\lambda_1 \geq \lambda_2$

Eigenvectors encode edge direction while eigenvalues encode edge strength.

Laplacian of Gaussian

The Laplacian of Gaussian is known as the LoG operator. In this method, an image $I(x, y)$ is convolved with the function

$$L(x, y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\pi^2}\right]\exp\left\{\frac{-x^2+y^2}{2\pi^2}\right\}$$

where σ controls the spatial extent of the function. The Laplacian of a Gaussian can also be approximated with a Difference of Gaussians (DoG) operator.