Image Understand Notes

Depu Meng

Sept. 2018

1 Corner Detection

1.1 How to define a corner?

A corner is a region where two edges meet. Corners have the property that a movement of any direction will result in a significant change.

1.2 Metric to measure a corner

We use (weighted) summed square difference – WSSD to measure how much a region is likely to be a corner. WSSD with shift $\boldsymbol{u}$ is defined as follows:

$$E_{WSSD}(\boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)[I(\boldsymbol{x}_i + \boldsymbol{u}) - I(\boldsymbol{x}_i)]^2 \tag{1}$$

If we denote $\boldsymbol{u} = (u, v)$, $w(\boldsymbol{x}_i)$ is a window function with window area W, then we can rewrite equation 1 to

$$E_{WSSD}(u, v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(\boldsymbol{x}_i)]^2 \tag{2}$$

We call equation 2 *auto-correlation function.*

If we assume the shift $\boldsymbol{u}$ is small, then we can have the following approximation

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \tag{3}$$

Denote $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$ and rewrite the equation 3 in matrix form

$$I(x+u, y+v) \approx I(x,y) + \begin{bmatrix} I_x \; I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{4}$$

So the SSD function can be rewrite to

$$\begin{aligned} E(u,v) &\approx \sum_{(x,y)\in W} \left[\begin{bmatrix} I_x \; I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \right]^2 && (5) \\ &= \sum_{(x,y)\in W} \begin{bmatrix} u \; v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} && (6) \\ &= \begin{bmatrix} u \; v \end{bmatrix} A \begin{bmatrix} u \\ v \end{bmatrix} && (7) \\ & && (8) \end{aligned}$$

Where

$$A = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}$$

We apply eigenvalue decomposition to A, then we can get eigenvalues λ_1, λ_2 and eigenvectors $\boldsymbol{x}_1, \boldsymbol{x}_2$ respectively. Assume that $\lambda_1 > \lambda_2$, then we have the following conclusion:

Theorem 1. *When* $\cos\langle \boldsymbol{u}, \boldsymbol{x}_1\rangle = 1$*, the* $E(\boldsymbol{u})$ *reaches maxima* λ_1*;*
When $\cos\langle \boldsymbol{u}, \boldsymbol{x}_2\rangle = 1$*, the* $E(\boldsymbol{u})$ *reaches minima* λ_2*;*

Proof.

$$E(u,v) = \begin{bmatrix} u \ v \end{bmatrix} A \begin{bmatrix} u \\ v \end{bmatrix}$$

where $A = Q\Lambda Q^{-1}$, $Q = [\boldsymbol{x}_1, \boldsymbol{x}_2]$, $\Lambda = diag(\lambda_1, \lambda_2)$, denote $\boldsymbol{u} = \begin{bmatrix} u \\ v \end{bmatrix}$, then we have

$$\begin{aligned} E(u,v) &= \boldsymbol{u}^T Q\Lambda Q^{-1}\boldsymbol{u} \\ &= \boldsymbol{u}^T \begin{bmatrix}\boldsymbol{x}_1, \boldsymbol{x}_2\end{bmatrix} diag(\lambda_1,\lambda_2) \begin{bmatrix}\boldsymbol{x}_1, \boldsymbol{x}_2\end{bmatrix}^{-1} \boldsymbol{u} \end{aligned}$$

Because A is a symmetrical matrix, we have $Q^{-1} = Q^T$, then

$$\begin{aligned} E(u,v) &= \boldsymbol{u}^T \begin{bmatrix}\boldsymbol{x}_1, \boldsymbol{x}_2\end{bmatrix} diag(\lambda_1,\lambda_2) \begin{bmatrix}\boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T\end{bmatrix} \boldsymbol{u} \\ &= \begin{bmatrix}\boldsymbol{u}^T \cdot \boldsymbol{x}_1, \boldsymbol{u}^T \cdot \boldsymbol{x}_2\end{bmatrix} diag(\lambda_1,\lambda_2) \begin{bmatrix}\boldsymbol{x}_1^T \cdot \boldsymbol{u} \\ \boldsymbol{x}_2^T \cdot \boldsymbol{u}\end{bmatrix} \\ &= \lambda_1(\boldsymbol{u}^T \cdot \boldsymbol{x}_1)^2 + \lambda_2(\boldsymbol{u}^T \cdot \boldsymbol{x}_2)^2 \end{aligned}$$

Because $\boldsymbol{x}_1 \perp \boldsymbol{x}_2$, the $E = const$ construct an ellipse with semi-major axis $\boldsymbol{x}_1$ and semi-minor axis $\boldsymbol{x}_2$, $a = \lambda_1$, $b = \lambda_2$. Then we can easily get that $\boldsymbol{x_1}$ is the maximum gradient direction, $\boldsymbol{x_2}$ is the minimum. □

Then we can conclude that

W is a *edge* region if $\lambda_1 \gg \lambda_2$,
W is a *flat* region if both λ_1 and λ_2 are small ,
W is a *corner* region if both λ_1 and λ_2 are large .

With the properties above, we know that whether a region is a corner is determined by the smaller eigenvalue λ_2. So we can use a simple threshold like $\lambda_2 \leqslant t$ to detect corners. In order to get the exact location of a corner, we choose the local maximum as the corner location.

1.3 Harris Detector

In Harris detector, the measure of corner response is

$$R = \det(A) - k(\mathrm{tr}(A))^2 \tag{9}$$

where k is a empirical constant and $\det(A)$ and $\mathrm{tr}(A)$ have the following properties

$$\det(A) = \lambda_1\lambda_2$$
$$\mathrm{tr}(A) = \lambda_1 + \lambda_2$$

So when λ_2 is large, R will be large too, and from the definition we can see that the computation cost of R is smaller than directly calculate λ_2.

1.4 Adaptive Non-Maximal Suppression

The method that take local maxima as corner point can lead to an uneven distribution of corner points. For example, some regions might be rich of textures so these regions might have lots of local maxima. In order to balance the corner point distribution, we adopt *Adaptive Non-Maximal Suppression* (ANMS), that only consider points that are (a) local maxima (b) larger than its neighborhood by at least 10% in an adaptive radius as corner points.

2 Feature Detection

2.1 Blob Detection Filters

Laplacian of Gaussian The Laplacian of Gaussian (LoG) filter is defined as

$$\bigtriangledown^2 G = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2} \tag{10}$$

where G is a Gaussian kernel

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{11}$$

so combine equation 10 and equation 11, we can get

$$\bigtriangledown^2 G(x, y, \sigma) = \Big(\frac{x^2+y^2}{\sigma^4} - \frac{2}{\sigma^2}\Big) G(x, y, \sigma) \tag{12}$$

Remark. The original idea of LoG is to calculate the second derivative of the image blurred by a Gaussian kernel, i.e.,

$$LoG(f) = \bigtriangledown^2 (g * f) = \mathcal{L} * (g * f) = (\mathcal{L} * g) * f = \bigtriangledown^2 g * f \tag{13}$$

where $\mathcal{L}$ is the Laplacian kernel.

Difference of Gaussian The Difference of Gaussian (DoG) filter is defined as

$$DoG(\frac{\sigma_1}{\sigma_2}, \sigma_2) = G(x, y, \sigma_1) - G(x, y, \sigma_2) = \frac{1}{2\pi}\Big(\frac{1}{\sigma_1^2}e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{\sigma_2^2}e^{-\frac{x^2+y^2}{2\sigma_2^2}}\Big) \quad (14)$$

DoG is a good approximation of LoG, because

$$DoG(k, \sigma) = G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \bigtriangledown^2 G(x, y, \sigma) \quad (15)$$

2.2 Scale Invariance

Definition A feature detector is scale invariant means that the detector can detect features at a variety of scale and match in all possible levels of resolution in a pyramid. That means that the detector should find the local maxima of a feature in x, y, σ domain, i.e.,

$$(x_0, y_0) = \arg\max_{(x_0,y_0)}\{\max_{\sigma} f(x, y, \sigma)\} \quad (16)$$

A naive approach is to detect the features using a series of DoG detectors with different σ. However, in this way, the computation is not efficient. In order to detect features with different scales efficiently, *Scale Invariant Feature Transform* (SIFT) is developed.

Frequency sampling in scale In order to find the local maxima in scale domain, we need sampling in scale. Suggest we adpot a sampling $\sigma, 2\sigma, 4\sigma, ...$, we will have features of $\sigma, 2\sigma, 4\sigma, ...$ scale.

An important equivalence A convolution with Gaussian kernel $G(2\sigma)$ performs on an image f is equivalent with a convolution with Gaussian kernel $G(\sigma)$ performs on an image h, where h is the image that f is downsampled by the scale of 2.

SIFT Algorithm Scale sampling in SIFT is $\sigma, 2^{\frac{1}{s}}\sigma, 2^{\frac{2}{s}}\sigma, ..., 2^1\sigma, 2^{1+\frac{1}{s}}\sigma, ...$. From the equivalence mentioned above, we can divide this sampling into

$$\begin{aligned}
&\sigma, 2^{\frac{1}{s}}\sigma, 2^{\frac{2}{s}}\sigma, ..., 2^{\frac{s-1}{s}}\sigma \\
&2\sigma, 2^{1+\frac{1}{s}}\sigma, 2^{1+\frac{2}{s}}\sigma, ..., 2^{1+\frac{s-1}{s}}\sigma \\
&... \\
&2^n\sigma, 2^{n+\frac{1}{s}}\sigma, 2^{n+\frac{2}{s}}\sigma, ..., 2^{n+\frac{s-1}{s}}\sigma
\end{aligned}$$

Instead of directly perform DoG with the above scale on the original image, we perform downsample with scale of 2 on the original image to generate a series of new images, we call them *octaves*. We only need to perform DoG with scale $\sigma, 2^{\frac{1}{s}}\sigma, 2^{\frac{2}{s}}\sigma, ..., 2^{\frac{s-1}{s}}\sigma$ on all these octaves to achieve all the scale sampling. With this modification, the computation cost is reduced.

To further reduce computation cost, we find that the Gaussian convolution can be reused. Suggest that we perform $G(\sigma)$ on the original image f and get a new image denoted as h_σ, then

$$DoG(2^{\frac{1}{s}}, \sigma) = G(x, y, 2^{\frac{1}{s}}\sigma) - G(x, y, \sigma)$$

Then we can calculate h_{σ_i} first and cache them, so each h_{σ_i} can be used twice.

After DoGs are calculated, we get the local maximas in each scale. In order to get the local maximas in scale domain, we compare the local maximas in each map with the same location in its neighborhood images in scale domain. Then we get the features from different scales, i.e., *Scale Invariant Feature Detection*.

3 Feature Description

After features are detected, a natural idea is to match the same feature points across the images, so description of features as well as distance between descriptions is required.