ISSN: 2834-7706

Research Article

Journal of Mathematical Techniques and Computational Mathematics

Proximity Modulated Thresholding for Hessian Matrix Feature Detection

Greg Passmore*

PassmoreLab, Austin, Texas, USA

*Corresponding Author
Greg Passmore, PassmoreLab, Austin, Texas, USA.

Submitted: 2025, Jan 18; **Accepted:** 2025, Feb 26; **Published:** 2025, Mar 03

Citation: Passmore, G. (2025). Proximity Modulated Thresholding for Hessian Matrix Feature Detection. *J Math Techniques Comput Math, 4*(1), 01-06.

Abstract

Our lab processes large volumes of multispectral drone data. We use feature identification for image alignment, object recognition, and scene reconstruction. Traditional methods using the Hessian matrix detect features like corners or blobs. This method is commonly used for color images, and the issues with changes in lighting and exposure are well known. However, for our multispectral data, it proved especially problematic. Fixed thresholds worsened these issues, causing inefficiencies and inaccuracies in feature matching and image alignment.

This paper presents a dynamic thresholding approach that adjusts the feature detection threshold based on the disparity between the current and pre-defined feature count. It starts with an initial detection phase using a standard threshold to establish a baseline. The threshold is then adjusted incrementally until the feature count converges towards the target. This iterative refinement improves responsiveness and efficiency by considering the proximity between the current and desired counts.

Experimental results demonstrate that adaptive thresholding reduces computational costs on the order of one magnitude and can increase the granularity in feature detection processes, making it useful for complex image processing tasks. This approach is particularly beneficial in environments with significant variability in multispectral environments, or where image quality and lighting conditions present challenges.

1. Significance

Feature identification is important for tasks such as image alignment, object recognition, and scene reconstruction. One common approach to feature identification involves using the Hessian matrix to detect areas in an image that contain distinct features, such as corners or blobs. However, the effectiveness of feature detection can vary significantly based on image characteristics like expo-sure, lighting conditions, and spectral bands. This variation often results in different numbers of detected features across images, which complicates alignment and other comparative tasks.

2. Simple Hessian Matrix

Definitions

For an image, *I*, the intensity at pixel is denoted by *I* (*x, y*). The Hessian matrix (*x, y*) at a pixel in terms of image intensity is defined as:

$$H(x,y) = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix}$$

3. Second-Order Derivatives

The second-order derivatives required for the Hessian matrix are approximated using finite differ-ences:

Second Partial Derivative in $x(\frac{\partial^2 I}{\partial x^2})$:

$$\frac{\partial^2 I}{\partial x^2} \approx I(x+1,y) - 2I(x,y) + I(x-1,y)$$

Second Partial Derivative in $y(\frac{\partial^2 I}{\partial y^2})$:

$$\frac{\partial^2 I}{\partial y^2} \approx I(x,y+1) - 2I(x,y) + I(x,y-1)$$

Mixed Partial Derivative:

$(\frac{\partial^2 I}{\partial x \partial y})$ and $(\frac{\partial^2 I}{\partial x \partial y})$, which are equal by Schwarz's theorem by the symmetry of the second derivatives:

$$\frac{\partial^2 I}{\partial x \partial y} \approx \frac{1}{4}(I(x+1,y+1) - I(x-1,y+1) - I(x+1,y-1) + I(x-1,y-1))$$

4. Hessian Matrix Determinant

The determinant of the Hessian matrix H at pixel (x, y) is given by:

$$\det(H(x,y)) = \left(\frac{\partial^2 I}{\partial x^2}\right)\left(\frac{\partial^2 I}{\partial y^2}\right) - \left(\frac{\partial^2 I}{\partial x \partial y}\right)^2$$

5. Interest Point Detection

Interest points are then detected by applying a threshold θ to $\det(H(x, y))$:

If $\det(H(x, y)) > \theta$, then (x, y) is an interest point

This mathematical formulation helps in understanding the underlying computations and can be directly translated into algorithms for processing images to detect features or interest points based on changes in local curvature represented by the Hessian matrix.

6. Scale Invariance

It is possible to extend the simple Hessian Matrix to become scale invariant. Let's take a look at the extended version.

The scale invariant Hessian Matrix, H can be written as:

$$H = \begin{bmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial^2 D}{\partial x \partial \sigma} \\ \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial^2 D}{\partial y^2} & \frac{\partial^2 D}{\partial y \partial \sigma} \\ \frac{\partial^2 D}{\partial x \partial \sigma} & \frac{\partial^2 D}{\partial y \partial \sigma} & \frac{\partial^2 D}{\partial \sigma^2} \end{bmatrix}$$

To eliminate the keypoints at the edges, the eigenvalues λ_1, λ_2, λ_3 of H are computed. A point is discarded if:

$\min(|\lambda_1|, |\lambda_2|, |\lambda_3|) \times r < \max(|\lambda_1|, |\lambda_2|, |\lambda_3|)$

Where r is a constant, used to decide the ratio between the smallest and largest eigenvalues.

7. Orientation Assignment

An orientation is assigned to each keypoint to achieve invariance to image rotation. The local image gradient directions are sampled around each keypoint, and a histogram is formed from the gradient orientations.

Given a keypoint located at (x,y) in image I, and we let $L(x,y,\boldsymbol{\sigma})$ with an orientation of $\theta(x,y)$ be the scale space representation (i.e., the image convolved with a Gaussian kernel of standard deviation $\boldsymbol{\sigma}$) of I.

The gradient magnitude M(x,y) and the orientation $\theta(x,y)$ at each image point in a neighboring region around the keypoint can be calculated as follows:

$$M(x,y) = \sqrt{\left(\frac{\partial L}{\partial x}\right)^2 + \left(\frac{\partial L}{\partial y}\right)^2}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{\frac{\partial L}{\partial y}}{\frac{\partial L}{\partial x}}\right)$$

A histogram is then formed of the gradient orientations. Each sample in the neighborhood contributes to the histogram with a weight equal to its gradient magnitude. The histogram could be formed over 36 bins covering the 360-degree range of orientations. The highest peak in the histogram is detected, and any other local peak that is within 80% of the highest peak is also considered to specify an orientation. This leads to the assignment of one or more orientations θ_{keypoint} to each keypoint.

8. Keypoint Descriptor

Finally, local image gradients are measured at the selected scale in the region around each keypoint. These are then transformed into a descriptor that allows for significant levels of local shape distortion and change in illumination.

While the descriptor is mainly computed through algorithmic steps, the underlying mathematics involves capturing local image gradients in a region around each keypoint. These gradients are then transformed into a more stable form that serves as the descriptor. Here is an outline of the mathematical foundations of this step:

As above, let L(x,y,σ) be the scale space representation of the image at the keypoint's scale, and consider a $d \times d$ region around the keypoint. Typically, $d = 16$. This region around a keypoint serves as the local area where gradient magnitudes and orientations are computed for feature descriptor creation. The choice of d determines the extent of the local neighborhood and is critical for capturing sufficient detail while also allowing for generalization. In standard implementations of SIFT, $d = 16$, which means a 16 × 16 square region is examined around each keypoint.

The d × d region is further divided into smaller regions, often 4 × 4, and for each of these sub-regions, an orientation histogram is generated. The orientation histogram is typically constructed with 8 bins, capturing the frequency of occurrence of gradient orientations in the 4×4 sub-region. Each pixel's gradient within these sub-regions contributes to these histograms.

Let's look at representing the 16×16 region and the smaller 4×4 regions:

For the 16×16 region:

• Let d represent the side length of the 16×16 square region.
• The 16×16 region can be represented as:

$$\sum_{i=1}^{d}\sum_{j=1}^{d}$$

For the 4×4 sub-regions within the 16×16 region:

• Let d d represent the side length of the 4×4 square sub-regions.
• Each 4×4 sub-region can be represented as:

$$\sum_{k=1}^{d_d}\sum_{l=1}^{d_d}$$

Combining these notations, we can represent the entire process as:

$$L(x,y,\sigma) = \sum_{i=1}^{d}\sum_{j=1}^{d}\left(\sum_{k=1}^{d_d}\sum_{l=1}^{d_d} \text{PixelGradient } (x+i, y+j)\right)$$

Where:

• $L(x, y, \sigma)$ represents the scale space representation of the image at the keypoint's scale.
• (x, y) denote the keypoint coordinates.
• σ represents the scale.
• d is the side length of the 16×16 square region.
• d_d is the side length of the 4×4 square sub-regions.
• PixelGradient$(x + i, y + j)$ represents the gradient of the pixel at coordinates $(x + i, y + j)$.

9. Differences

The difference between the two representations of the Hessian matrices lies primarily in their dimensions and the function variables they are derived from, and the proximity modulation works for either the simple or extended method. Let's discuss these differences and their implications:

9.1. Function Variables:

• The first Hessian matrix $H(x, y)$ is derived from a function I that depends wholly on spatial variables x and x. This is typical for basic image processing tasks where the image intensity I is a function of spatial coordinates. This matrix is primarily used for tasks such as feature detection in images, where the interest is in spatial shifts.
• The second Hessian matrix uses a function D that depends on spatial variables x , y , and a scale variable σ. This type of formulation is often used in scale-invariant feature detection methods, where the goal is to identify features across different scales. The inclusion of σ allows D to represent a function like the scale space representation of an image, commonly used in algorithms like Scale-Invariant Feature Transforms.

9.2. Matrix Dimensions:

• The first matrix is a 2×2 matrix, which is suitable for two-dimensional spatial analysis. This matrix evaluates the curvature of the function I concerning the spatial dimensions alone.
• The second matrix is a 3×3 matrix, reflecting the addition of the scale dimension σ . This larger matrix can evaluate changes not only across spatial dimensions, but also how these changes vary with scale. This is crucial for detecting features that are consistent across different sizes or zoom levels of images.

9.3. Applications:

• The 2×2 Hessian matrix $H(x, y)$ is adequate for image processing tasks that require edge detection, corner detection, or other feature detections where only spatial considerations are necessary.
• The 3×3 Hessian matrix is used in more complex scenarios where scale invariance is important, such as in the detection of features that need to be recognized at different scales, distances or differing optical settings. This is essential for computer vision applications involving multiscale analysis, like object recognition across different cameras or optics.

9.4. Interpretation and Complexity:

• The complexity in interpreting the 2×2 matrix is lower since it deals only with spatial derivatives, which are easier to compute and visualize.
• The 3×3 matrix introduces additional complexity due to the inclusion of derivatives regarding σ, indicating how the detected features change as the image is progressively blurred (a common method for simulating changes in scale).

The issue becomes determining if we need scale invariance. When the images I_1, I_2, I_3... are intrinsically at the same scale (independent of resolution differences), we can skip this more involved process. However, even minor shifts in scale can frustrate the later process of determining associated points for image alignment.

10. Density of Features

When identifying associated features across different images, achieving a consistent number of features for comparison is very helpful. However, due to variations in density, exposure, lighting, or detail, different images often yield differing counts of features when processed with a uniform threshold. To address this, we can dynamically adjust the threshold as necessary. This is also helpful because using a static rate of adjustment can lead to slow convergence. To improve efficiency, the rate at which the threshold changes should be proportional to the discrepancy between the current number of detected features and the desired count. The dynamic adjustment accelerates convergence by adapting more aggressively when the difference is larger, and more slowly as it narrows.

11. Convergence Mechanism Using the Hessian Matrix

The convergence mechanism involves adaptively adjusting a threshold to achieve a desired number of detected interest points based on the Hessian determinant. The key to understanding the convergence mechanism lies in the iterative adjustment process:

Initial Conditions: Start with an initial threshold for the determinant of the Hessian matrix.

Detection and Counting: For each pixel (x, y) in the image, compute the determinant of the Hessian matrix. Count the number of pixels for which $\det(H(x, y)) > \theta$ exceeds the threshold, indicating potential interest points.

Adaptive Adjustment: Compare the current count of interest points to the desired count. Based on the difference, adjust the threshold:

• If the number of detected points is greater than N, increase the threshold to make it harder for points to qualify as interest points:

$$\theta = \theta + \Delta\theta \times \left(\frac{\text{count} - N}{N}\right)$$

• If count is less than N , decrease the threshold to allow more points to qualify:

$$\theta = \theta - \Delta\theta \times \left(\frac{N - \text{count}}{N}\right)$$

Here, $\Delta\theta$ represents a scaling factor for the threshold adjustment, which may be refined each iteration.

Convergence Criterion: The process is repeated until the number of detected points is close to N, or the changes in θ become negligibly small, indicating convergence.

Abstraction: The iterative adjustment can be mathematically expressed as:

$$\theta_{\text{new}} = \theta_{\text{old}} \pm \Delta\theta \times \left|\frac{\text{count} - N}{N}\right|$$

where ± depends on whether count is greater than or less than N.

This method, then, dynamically fine-tunes the sensitivity of interest point detection to achieve a target number of features, balancing between sensitivity and specificity based on the image content and the degree of feature detection needed. This approach is particularly useful in applications where consistent feature detection across varying image conditions is required.

12. Extending Convergence

The proximity modulated threshold holds potential across a wide array of applications, particularly in scenarios where the goal of an optimal fitness set or numeric convergence point is essential. One example application lies in artificial intelligence (AI), where the adjustment of hyperparameters, such as the learning rate or the activation function (e.g., sigmoid), plays a pivotal role in model optimization.

For now, however, let's consider a scenario involving the enhancement of the Hessian matrix through the application of Gaussian functions. The Gaussian function reduces noise. One way to determine its effectiveness is to measure repeatability of the final feature set after Hessian matrix calculation. The Hessian matrix characterizes the curvature of the image function's surface, and our proximity modulation controls the convergence behavior via iterative optimization. We can attack the noisy or erratic data which introduces instability in the Hessian matrix estimation with the Gaussian operator, minimizing suboptimal convergence or divergence.

By incorporating proximity modulated thresholding techniques inside the Gaussian function controlled by eventual feature repeatability derived by the Hessian matrix computation, it becomes possible to reduce the impact of image noise and increase final result reliability. Said another way, the proximity modulated threshold method can also dynamically adjust the Gaussian weights based on the observed repeatability leading towards a zero divergence from frame to frame. This adaptive thresholding mechanism, then, can effectively filter out noise to focus on more relevant image information, thereby improving the accuracy and reliability of the Hessian matrix calculation by way of improvement of the Gaussian function.

13. Gaussian Noise Easing

Here, we take a look at repeatedly factoring to converge on an ideal Gaussian process.

The building of the kernel creation can be expressed as:

$$\text{kernel}[y][x] = \frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{\sum_{y'=-\text{ radius}}^{\text{radius}} \sum_{x'=-\text{ radius}}^{\text{radius}} e^{-\frac{x'^2+y'^2}{2\sigma^2}}}$$

In this equation:

• kernel$[y][x]$ represents the weight assigned to the pixel at position (x, y) within the Gaussian kernel.
• σ denotes the standard deviation of the Gaussian distribution.
• radius = $\frac{\text{size}}{2}$ represents the radius of the kernel.
• x and y iterate over the spatial coordinates within the kernel, ranging from −radius to radius.
• The numerator calculates the Gaussian weight for the current pixel position (x, y).
• The denominator represents the normalization term, which is the sum of Gaussian weights over all pixel positions within the kernel.
• The sum is computed over y' and, representing all possible pixel positions within the kernel.

14. Application in X and Y

Next, we apply the Gaussian filer to X and Y separately.

Given an input matrix image(x, y) of size $width \times height$, and assuming a Gaussian kernel of size $size \times size$ with standard deviation σ , the method calculates the filtered output matrix (x,y) for each pixel as follows:

$$\text{output}(x,y) = \sum_{i=-\text{radius}}^{\text{radius}} \sum_{j=-\text{radius}}^{\text{radius}} \text{image}(x+i, y+j) \cdot \text{kernel}[i + \text{radius}][j + \text{radius}]$$

Where:

• radius = $\frac{\text{size}}{2}$ represents the radius of the Gaussian kernel.
• kernel[i + radius][j + radius] denotes the value of the Gaussian kernel at position (i, j).
• The sums are taken over all possible offsets i and j , ranging from − radius to r adius , effectively covering the entire kernel region around pixel (x, y).

The Gaussian kernel weights are multiplied by the corresponding pixel values within the kernel neighborhood, and the resulting weighted sums are accumulated to obtain the filtered output for each pixel.

15. Crossproduct of X and Y

Next, we use the x and y results to produce the derivatives I_{xx} , I_{yy} , and I_{xy} . These derivatives are obtained by squaring and multiplying the elements of the input derivative matrices I_x and I_y. The Gaussian filtering process helps smooth and refine these derivatives, which is useful for our image processing tasks such as edge detection and feature extraction.

Here's the mathematical explanation of the process:

Given input derivative matrices $I_x(x, y)$ and $I(x,y)$y of size *width* × *height* , the method calculates the product matrices $I_{xx}(x, y)$, $I_{yy}(x, y)$, and $I_{xy}(x, y)$ as follows:

$$I_{xx}(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (I_x(x+i, y+j))^2 \cdot \text{kernel}[i][j]$$

$$I_{yy}(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (I_y(x+i, y+j))^2 \cdot \text{kernel}[i][j]$$

$$I_{xy}(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I_x(x+i, y+j) \cdot I_y(x+i, y+j) \cdot \text{kernel}[i][j]$$

Where kernel[i][j] represents the value of the Gaussian kernel at position (i, j).

In these equations:

• (x, y) represent the spatial coordinates of the output matrices I_{xx}, I_{yy}, and I_{xy}.
• i and j represent the spatial offsets within the Gaussian kernel.
• The sums are taken over all possible offsets i and j , effectively spanning the entire image domain.
• The Gaussian kernel is applied to each pixel in the input matrices, and the resulting weighted sums are accumulated to compute the output matrices I_{xx} , I_{yy}, and I_{xy}.

The returned matrix (in this case, I_{xy} represents the result of applying the Gaussian filter to the crossproduct derivative matrix.

To combine the Gaussian filtering process for both x and y directions, as well as the cross-product operation, into a single sigma notation equation, we can express the entire process as a convolution operation using the $_2$D Gaussian kernel. Let's denote the input matrix as *i* mage(x, y) and the Gaussian kernel as kernel(i, j).

The combined process can be represented as follows:

$$\text{output}(x,y) = \sum_{i=-\text{radius}}^{\text{radius}} \sum_{j=-\text{radius}}^{\text{radius}} \left(\left(\sum_{k=-\text{radius}}^{\text{radius}} \sum_{l=-\text{radius}}^{\text{radius}} \text{image}(x+k, y+l) \cdot \text{kernel}(k,l) \right)^2 \right) \cdot \text{kernel}(i,j)$$

Where:

• radius = $\frac{\text{size}}{2}$ represents the radius of the Gaussian kernel.
• kernel(i, j) denotes the value of the Gaussian kernel at position (i, j).
• The outer sums iterate over the pixel positions (x, y) in the output matrix.
• The inner sums represent the convolution operation for each pixel, where the input matrix image is convolved with the Gaussian kernel.
• Within the inner sums, the cross-product operation is performed by squaring the result of the convolution.

This single sigma notation equation encapsulates the entire process of applying Gaussian filtering and computing the products of derivatives I_{xx} , I_{yy} , and I_{xy} simultaneously for each pixel (x, y) in the input matrix.

16. Bringing It All Together

The goal is to modify the Hessian threshold, and the Gaussian weights to converge on a set of identified features that converge on both the correct number of features, as well as exhibiting high repeatability.

This recursive function updates the Hessian matrix, applies feature detection, evaluates the detected features, and adjusts parameters iteratively until convergence. Let F^n represent the set of detected features at iteration n , T_n represent the Hessian threshold, and $\text{kernel}^n_{\text{mod}}(i, j)$ represent the modified Gaussian kernel weights at iteration n. The iterative optimization process can be described as follows:

$$F^n = \text{Optimize}(F^{n-1}, T_{n-1}, \text{kernel}^{n-1}_{\text{mod}}(i,j))$$

Where Optimize is a function that encompasses the following steps:

16.1. Gaussian Filtering and Hessian Computation:

• Apply the Gaussian filter to the image gradients to compute the modified Hessian matrix elements $H_{\text{mod}}(x, y)$ using kernel $^{n-1}_{\text{mod}}(i, j)$.

• This step can be represented as:

$$H_{\text{mod}}(x,y) = \sum_{i=-\text{radius}}^{\text{radius}} \sum_{j=-\text{radius}}^{\text{radius}} \left(\left(\sum_{k=-\text{radius}}^{\text{radius}} \sum_{l=-\text{radius}}^{\text{radius}} I_{\text{mod}}(x+k, y+l) \cdot \text{kernel}_{\text{mod}}^{n-1}(k,l) \right)^2 \right) \cdot \text{kernel}_{\text{mod}}^{n-1}(i,j)$$

16.2. Feature Detection and Evaluation:

• Detect features F^n using the modified Hessian threshold T_{n-1} and $H_{\text{mod}}(x, y)$.

16.3. Parameter Adjustment:

• Adjust the Hessian threshold T_n and the Gaussian kernel weights $\text{kernel}^n_{\text{mod}}(i, j)$ based on the evaluation results.

16.4. Convergence Check:

• Check for convergence based on predefined criteria, such as changes in feature count, distribution, or repeatability metrics.

16.5. Recursive Call:

• Recursively call the function with updated parameters until convergence is achieved.

17. Further Investigation

The areas for further investigation are briefly discussed below.

In crack detection, it could improve discrimination between cracks and noise, potentially leading to more accurate results. Similarly, in image enhancement, there's potential for selectively amplifying features while reducing artifacts, though effectiveness may vary. Adaptive segmentation thresholds enabled by proximity modulated thresholding may offer more precise object delineation, contingent on image complexity. Furthermore, in image fusion, dynamic parameter adjustment could lead to more informative composite images, though actual improvements depend on image compatibility and fusion algorithms.

18. Conclusion

The integration of proximity modulated thresholding within Hessian matrix techniques exhibited fast and effective solutions in our multispectral aerial imagery. The adaptive convergence increased speed, and increased realiability by more closely matching correlated feature points between images. While the versatility of this approach extends to other Hessian matrix-based techniques such as texture analysis and pattern recognition, the extent of enhancement across different domains requires further investigation and refinement [1-4].

References

1. Zhao, Y., Zhao, Q., He, Y., & Lu, G. (2016, January). A crack extraction algorithm based on im-proved median filter and Hessian matrix. In *Seventh International Symposium on Precision Me-chanical Measurements* (Vol. 9903, pp. 775-780). SPIE.
2. Chen, X., Wu, Y., Zhu, C., & Liu, H. (2022). Research on Image Quality Enhancement Algorithm Using Hessian Matrix. *Journal of New Media, 4*(3), 117.
3. Ge, S., Shi, Z., Peng, G., & Zhu, Z. (2019). Two-steps coronary artery segmentation algorithm based on improved level set model in combination with weighted shape-prior constraints. *Journal of medical systems, 43*, 1-10.
4. Li, X., Wang, X., Cheng, X., Tan, H., & Li, X. (2022). Multi-focus image fusion based on hessian matrix decomposition and salient difference focus detection. *Entropy, 24*(11), 1527.