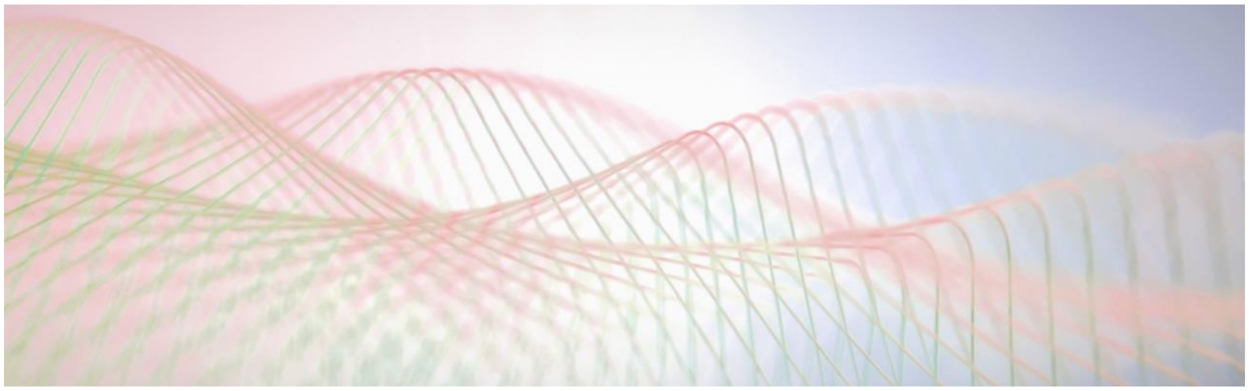


Detailed Note on the Haar Wavelet-based Blur Detection Algorithm

The Haar wavelet-based blur detection algorithm analyzes the edges in an image using multi-level Haar wavelet transforms to determine the presence and extent of blur. It detects sharp transitions (edges) at different scales and classifies the type of structure present, such as Dirac structures (sharp transitions), step-like transitions, or roof-like structures (indicative of blur).

1. Image Conversion to Grayscale

The algorithm begins by converting the input image to a grayscale format. This reduces the computational complexity since only the intensity values are needed for the wavelet transform.

Let the grayscale image be represented as Y with dimensions M×N.

2. Discrete Wavelet Transform (DWT)

The core of the algorithm involves applying the Discrete Wavelet Transform (DWT) to the image at multiple levels to decompose it into different components that highlight horizontal, vertical, and diagonal edges.

The first-level wavelet transform on the grayscale image Y produces four components:

- LL1: Approximation (low-frequency component)
- LH1: Horizontal details (high-frequency in the vertical direction)
- HL1: Vertical details (high-frequency in the horizontal direction)
- HH1: Diagonal details (high-frequency in both directions)

Mathematically:

$$Y(DWT) \rightarrow LL1, LH1, HL1, HH1$$

This process is repeated on the approximation component (LL) for further levels to extract edge information at multiple scales:

$$LL1(DWT) \rightarrow LL2, LH2, HL2, HH2$$

$$LL2(DWT) \rightarrow LL3, LH3, HL3, HH3$$

3. Edge Map Construction

At each level, edge maps are constructed by calculating the energy of the edges using the following equation:

$$Ei = \sqrt{(LHi2 + HLi2 + HHi2)}, i = 1,2,3$$

where:

- Ei is the edge map at level i.
- LHi , HLi , and Hi are the horizontal, vertical, and diagonal detail coefficients, respectively.

This formula represents the combined magnitude of the edges detected in the horizontal, vertical, and diagonal directions.

4. Sliding Window Analysis

A sliding window technique is used to analyze the edge maps. Maximum edge values are extracted from each window at each level to identify the strongest edges in that region:

$$E\max 1 = \max\left(E1_{(window)}\right), E\max 2 = \max\left(E2_{(window)}\right), E\max 3 = \max\left(E3_{(window)}\right)$$

This helps in identifying regions with significant edge activity.

5. Edge Classification Rules

Edges are classified based on the relationship between the maximum edge values across different scales using the following rules:

- **Rule 2 (Dirac/Astep Structures):** An edge is classified as a Dirac or step-like structure if the maximum edge values follow this pattern:

$$E\max 1 > E\max 2 > E\max 3$$

This pattern suggests a sharp edge, indicating a non-blurred region.

- **Rule 3 (Roof/Gstep Structures):** An edge is classified as a roof or Gstep structure if the maximum edge values increase as we move to finer scales:

$$E\max 1 < E\max 2 < E\max 3$$

This pattern indicates a more gradual transition, often seen in blurred regions.

- **Rule 4 (Roof Structures):** An edge is classified as a roof structure if the maximum edge value at the middle scale is greater than the other two:

$$E\max 2 > E\max 1\ and E\max 2 > E\max 3$$

This suggests a peak-like structure, which can also be indicative of blur.

6. Blur Confidence Calculation

The algorithm calculates a blur confidence value for each edge point using these rules:

$$\mathrm{BlurC[i]} = \begin{cases} 1 & \text{if RGstructure[i] or RSstructure[i] and Emax1[i]} < \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

7. Blur Extent Calculation

Finally, the overall blur extent of the image is calculated using the ratio of the number of blurred edge points to the total number of classified edge points:

$$\mathrm{Per} = \frac{\sum \mathrm{DAstructure}}{\sum \mathrm{EdgePoint}}$$

$$\mathrm{BlurExtent} = \begin{cases} 100 & \text{if } \sum(\mathrm{RGstructure} + \mathrm{RSstructure}) = 0 \\ \frac{\sum \mathrm{BlurC}}{\sum(\mathrm{RGstructure}+\mathrm{RSstructure})} & \text{otherwise} \end{cases}$$

Explanation of Parameters

- **Per:** Represents the percentage of Dirac and Astep structures detected, indicating sharp edges or transitions.
- **BlurExtent:** Measures how many edges follow the gradual transition or roof-like structures typical of blur.

Summary of the Algorithm's Steps

1. **Convert Image to Grayscale:** Reduces the image to intensity values for processing.
2. **Wavelet Decomposition:** Extracts edge information at multiple scales using the Haar wavelet transform.
3. **Edge Map Calculation:** Computes the magnitude of the edges at each level.
4. **Sliding Window Analysis:** Finds the maximum edge values in each region.
5. **Edge Classification:** Applies rules to classify edges into sharp or blurred structures.
6. **Blur Confidence Calculation:** Determines if the edges are more likely to be blurred.
7. **Blur Extent Calculation:** Provides the overall percentage of blur in the image.

Key Concepts and Equations

- **Discrete Wavelet Transform (DWT):** Captures edge information at multiple resolutions.
- **Edge Magnitude:** $Ei = \sqrt{(LHi2 + HLi2 + HHi2)}$, indicating the strength of edges.
- **Classification Rules:** Distinguish between sharp edges (Dirac) and blurred edges (roof-like) based on patterns in edge magnitudes.

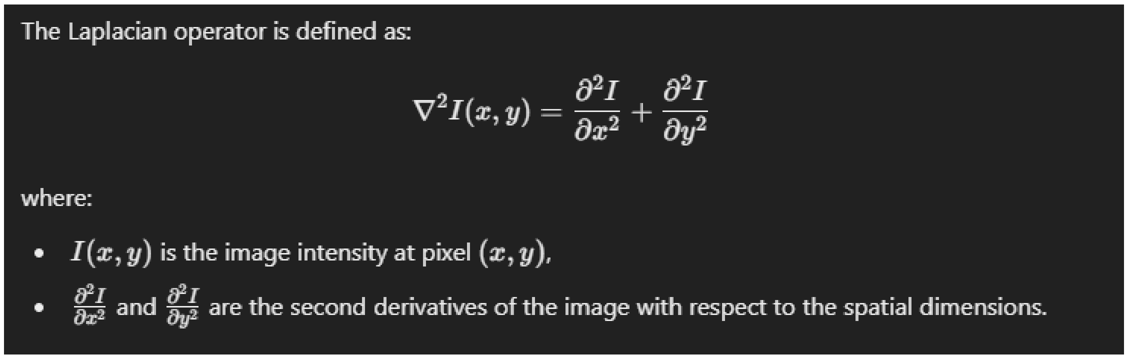
This method is effective in distinguishing blurred regions from sharp regions by analyzing changes in edge characteristics across multiple scales, making it a powerful tool for detecting image blur.

Laplacian Focus Measure for Image Sharpness Assessment

Introduction

In image processing, assessing the sharpness of an image is crucial for various applications, such as image enhancement, feature extraction, and object recognition. One of the commonly used methods to quantify image sharpness is the **Laplacian focus measure**. This method evaluates the second derivative of the image intensity, providing insight into the image's focus and clarity.

Mathematical Foundation



The Laplacian operator is defined as:

$$\nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

where:

- $I(x,y)$ is the image intensity at pixel (x,y),
- $\frac{\partial^2 I}{\partial x^2}$ and $\frac{\partial^2 I}{\partial y^2}$ are the second derivatives of the image with respect to the spatial dimensions.

The Laplacian operator detects regions of rapid intensity change, which are often associated with edges and fine details. Thus, an image with a high focus (sharpness) will yield a higher Laplacian variance.

Variance of the Laplacian

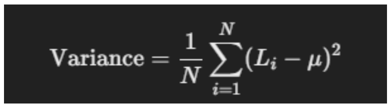
The **variance of the Laplacian** is computed as follows:

1. **Calculate the Laplacian:** The Laplacian of an image can be computed using convolution with a kernel, typically represented as:

$$L = \nabla^2 IL$$

In OpenCV, this is often done using the cv2.Laplacian function.

2. **Compute the Variance:** The variance of the Laplacian is defined as:



$$\text{Variance} = \frac{1}{N}\sum_{i=1}^{N}(L_i - \mu)^2$$

A higher variance indicates a sharper image, while a lower variance suggests blurriness.

potential problems and considerations regarding the Laplacian focus measure algorithm and its implementation:

1. Sensitivity to Noise

- **Issue:** The Laplacian operator is sensitive to noise in the image. Even small amounts of noise can lead to high values in the Laplacian, resulting in misleading focus measures.
- **Solution:** Preprocessing steps, such as applying a Gaussian blur before computing the Laplacian, can help reduce the impact of noise on the focus measure.

2. Thresholding for Blur Detection

- **Issue:** Determining an appropriate threshold for the variance of the Laplacian to classify an image as blurred or sharp can be challenging. A universal threshold may not work for all types of images due to variations in content and lighting.
- **Solution:** Consider adaptive thresholding techniques or use machine learning methods to dynamically adjust thresholds based on the image content.

3. Color Image Handling

- **Issue:** The algorithm typically operates on grayscale images, which may lose some color information relevant to sharpness.
- **Solution:** Alternatives can include computing the focus measure on individual color channels or using a combination of channels for a more holistic evaluation.

One way to solve some of the issues here to use higher kernel size than regular 3x3 kernel. 3x3 is most likely to be affected by the noise. So better use kernel_size = 11

The following github repo contains representative codes using these algorithms:

https://github.com/Salam068/Blurryness-detection