**Image Enhancement**

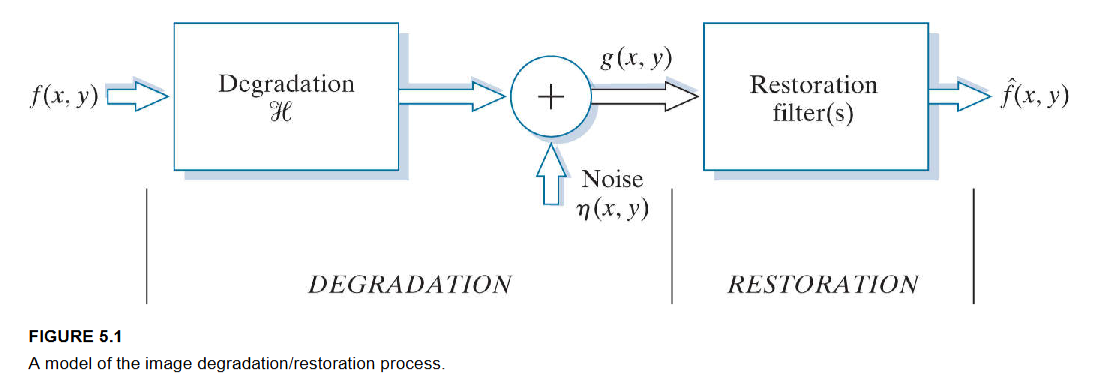
* **What it means:** It’s the process of improving the visual appearance of an image.
* **Subjective Process:**
  + "Subjective" means the improvements depend on personal taste or perception.
  + Enhancements like adjusting brightness, contrast, or color saturation are done to make the image look better to the viewer, but what looks “better” can vary from person to person.

**Image Restoration**

* **What it means:** It’s the process of recovering an image that has been degraded (e.g., by noise, blur, or other distortions) to get it closer to its original state.
* **Objective Process:**
  + "Objective" means the process is based on known, measurable factors and mathematical models.
  + Restoration uses a priori knowledge about how the image was degraded and applies a defined inverse process, aiming to restore the true, original image regardless of personal preferences.

In summary, image enhancement is about making an image look nicer based on personal or artistic criteria (subjective), whereas image restoration is about accurately reversing degradation using fixed, measurable models (objective).

**A Model of the Image Degradation/Restoration Process**



1. **Original Image**
   * Denote it as f(x,y). This is the “true” image before anything bad happens to it.
2. **Degradation Operator**
   * Represented by H (often a blur or distortion).
   * If H is linear and does not change with position, we can model it as a convolution with some function h(x,y).
3. **Additive Noise**
   * Denoted by n(x,y). This represents random noise added on top of the blurred image (like sensor noise or grain).
4. **Degraded Image**
   * Denoted by g(x,y).
   * It can be written in simple text form as:
   * g(x,y) = f(x,y) \* h(x,y) + n(x,y), Here, \* means convolution.
5. **Restoration Filter(s)**
   * These are algorithms or methods designed to “undo” or reduce the effects of H (the blur) and n(x,y) (the noise).
   * The output is an estimate of the original image, which we call f\_hat(x,y).
6. **Goal of Restoration**
   * We want f\_hat(x,y) to be as close as possible to the true original f(x,y).
   * The more we know about how the image was blurred (h(x,y)) and the characteristics of the noise (n(x,y)), the better our restoration can be.

**Frequency Domain Version (Optional Note)**

* Sometimes, it’s easier to work with the Fourier transforms of these images and functions. In the frequency domain, the degradation model becomes:
* G(u,v) = F(u,v) \* H(u,v) + N(u,v)

where G, F, H, and N are the Fourier transforms of g, f, h, and n, respectively.

**Summary**

* **Degradation** = Blurring or distortion (via H) + Noise (n).
* **Restoration** = Attempt to reverse that degradation to recover the original image f(x,y).
* **Key Insight**: Restoration is typically more “objective” because it tries to mathematically invert known degradations.

**Noise Model in Images**

When we capture or transmit an image, it often becomes degraded due to blurring and added noise. This degradation can be modeled as:

**g(x, y) = f(x, y) ∗ h(x, y) + n(x, y)**

Let’s break it down:

* **f(x, y)**: This is the original, clean image you want to capture.
* **h(x, y)**: This function represents the blurring or distortion that occurs. For example, if your camera shakes, the image becomes blurred. In our model, h(x, y) is assumed to be linear and does not change with position.
* **∗ (Convolution)**: This is a mathematical operation that “spreads” or “smears” the original image f(x, y) according to the pattern defined by h(x, y). You can think of it like stamping an ink pattern over the original image.
* **n(x, y)**: This term represents noise – the random unwanted variations (like grain, sensor noise, or interference) that are added to the image.
* **g(x, y)**: This is the final observed image after it has been blurred and had noise added.

**Spatial and Frequency Properties of Noise**

Noise can be looked at in two different “domains”:

**Spatial Domain**

* **White Noise**: This noise is random; every pixel’s noise is independent, with no predictable pattern.
* **Colored Noise**: Here, there may be some correlation between pixels (i.e., neighboring pixels may have similar noise values).

**Frequency Domain**

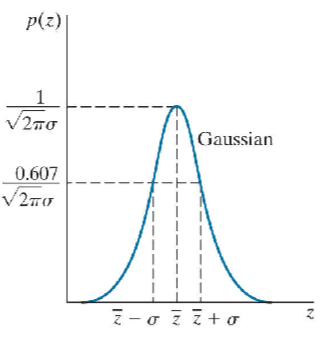
* When you transform the image using a Fourier transform, you see its frequency components:
  + **White Noise** appears as a flat (uniform) spectrum, meaning all frequencies are equally represented.
  + **Periodic Noise** (which happens when the noise has a repeating pattern) appears as spikes at certain frequencies.

In many image processing tasks, we assume noise is **uncorrelated** (each pixel’s noise is independent), which is simpler to work with.

**Common Noise Probability Density Functions (PDFs)**

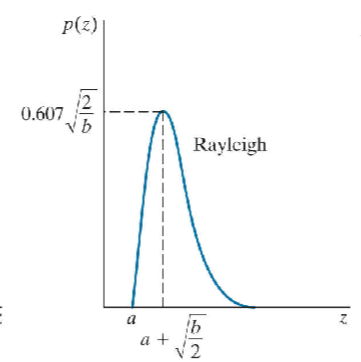
A probability density function (PDF) describes the likelihood of each noise value (z) occurring. Here are some common noise models:

**3.1 Gaussian (Normal) Noise**

****

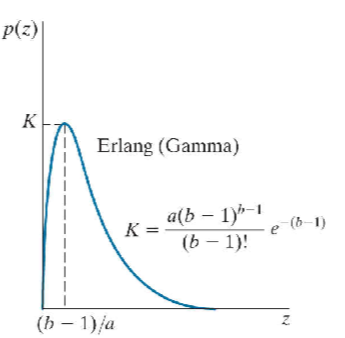
* **PDF:**  
    p(z) = (1 / (√(2π) · σ)) · exp[ – (z – m)² / (2σ²) ]
* **Parameters:**  
    m = mean  
    σ² = variance
* **Explanation:**  
  Gaussian noise is very common because it results from many small, independent sources of error. Its histogram is bell-shaped.

**3.2 Rayleigh Noise**

****

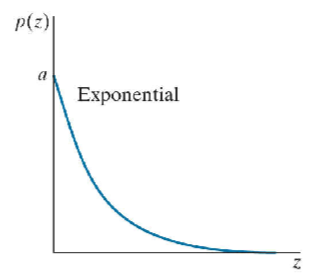
* **PDF (for z ≥ 0):**  
    p(z) = (z / b²) · exp( – z² / (2b²) )
* **Parameter:**  
    b > 0 (scale parameter)
* **Mean:**  
    m = b √(π/2)
* **Variance:**  
    σ² = ((4 – π) / 2) · b²
* **Explanation:**  
  Rayleigh noise has a skewed (asymmetric) shape. It is useful in applications like radar or range imaging.

**3.3 Erlang (Gamma) Noise**

****

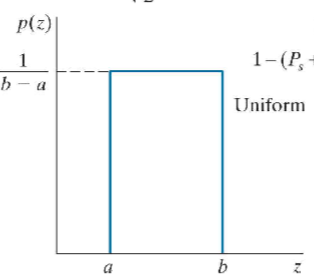
* **PDF (for z ≥ 0):**  
    p(z) = [ (bᵃ) / ((a – 1)!) ] · z^(a – 1) · exp( – b·z )
* **Parameters:**  
    a = a positive integer (shape parameter)  
    b = a positive real number (rate parameter)
* **Mean:**  
    m = a / b
* **Variance:**  
    σ² = a / b²
* **Explanation:**  
  This distribution (often called Gamma when using the gamma function) is skewed and is useful in modeling noise from sources like laser imaging.

**3.4 Exponential Noise**

****

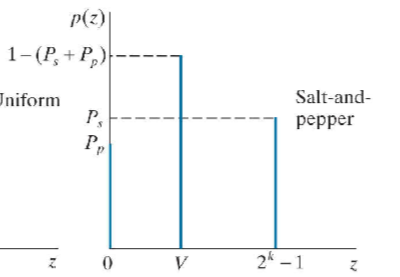
* **PDF (for z ≥ 0):**  
    p(z) = (1 / a) · exp( – z / a )
* **Parameter:**  
    a > 0
* **Mean:**  
    m = a
* **Variance:**  
    σ² = a²
* **Explanation:**  
  Exponential noise is a simpler case, and it is actually a special case of the Gamma distribution.

**3.5 Uniform Noise**

****

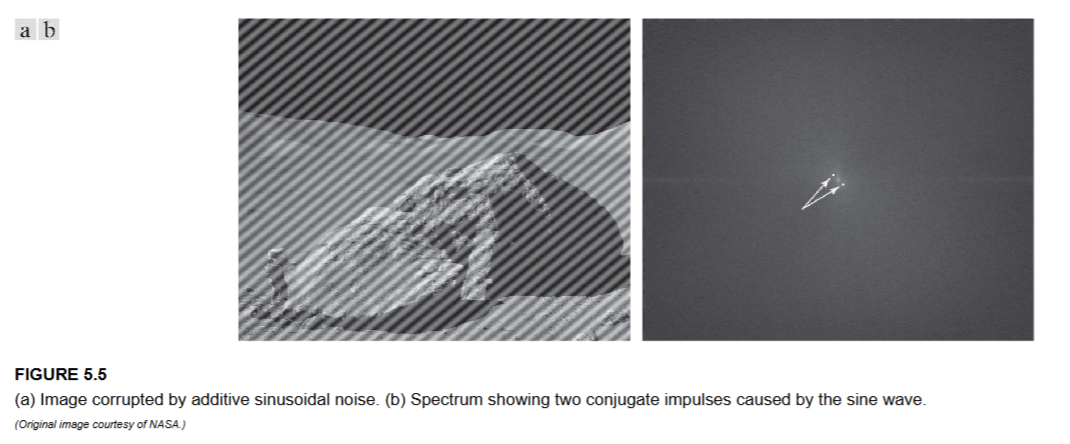
* **PDF (for a ≤ z ≤ b):**  
    p(z) = 1 / (b – a)
* **Parameters:**  
    a and b are the lower and upper limits (with a < b)
* **Mean:**  
    m = (a + b) / 2
* **Variance:**  
    σ² = (b – a)² / 12
* **Explanation:**  
  Every noise value between a and b is equally likely. This model is simple and is often used as a starting point in simulations.

**3.6 Salt-and-Pepper Noise (Impulse Noise)**

****

* **How it works:**  
  Instead of having a continuous range of noise values, salt-and-pepper noise replaces some pixels with fixed extreme values. Typically:
  + p(z = a) = Pₐ (e.g., a might be 0 for pepper)
  + p(z = b) = Pᵦ (e.g., b might be 255 for salt)
  + p(z = any other value) = 0 (for noise generation purposes)
* **Explanation:**  
  This noise appears as random black (pepper) and white (salt) dots in the image. The total noise density is given by P = Pₐ + Pᵦ, which tells you the fraction of pixels that are corrupted.

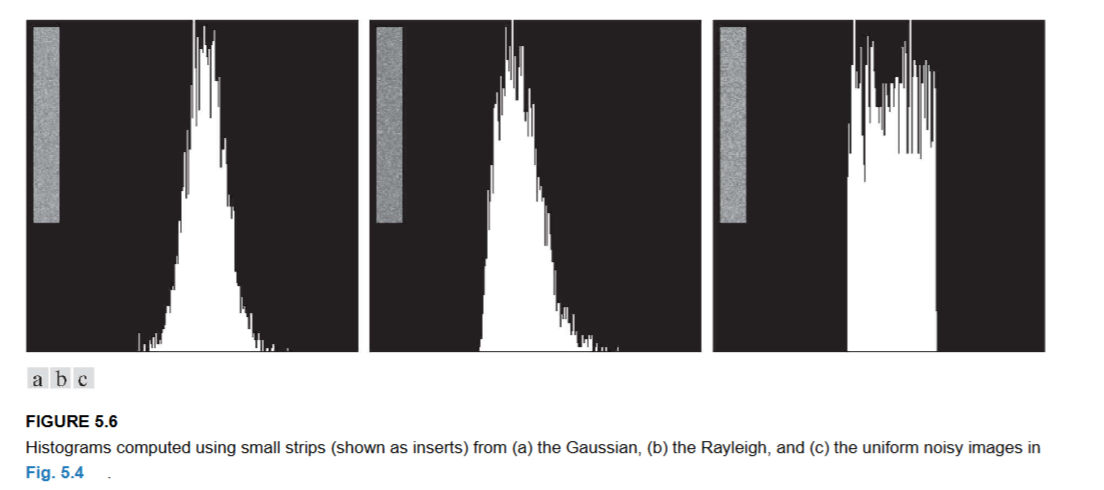
**4) Periodic Noise**

****

Periodic noise is different from random noise:

* **Origin:**  
  It typically comes from electrical interference or mechanical vibrations during image acquisition.
* **Spatial Domain:**  
  The noise shows up as repeating patterns or stripes.
* **Frequency Domain:**  
  When you analyze the image in the frequency domain, periodic noise creates distinct pairs of spikes (or impulses) at certain frequencies.
* **Removal:**  
  You can often reduce periodic noise by applying filters in the frequency domain to remove these spikes.

**5) Estimating Noise Parameters**

****

Knowing the noise parameters (like mean and variance) is essential to design effective denoising filters. Here’s how you can estimate them:

**Step 1: Visual and Frequency Analysis**

* **Periodic Noise:**  
  Look at the Fourier spectrum of the image. Spikes in specific locations indicate periodic noise.
* **Random Noise:**  
  Identify flat or uniform regions in the image where the true image is nearly constant. These regions help in isolating the noise.

**Step 2: Using Histograms**

* **Collect Data:**  
  Choose a small patch (subimage) from an area of nearly constant intensity.
* **Calculate the Normalized Histogram:**  
  Let p(zᵢ) be the probability (frequency) of intensity zᵢ, where zᵢ ranges over all possible intensity levels (for example, 0 to 255 for an 8-bit image).
* **Compute Mean (m) and Variance (σ²):**

  m = Σ (from i = 0 to L – 1) [ zᵢ · p(zᵢ) ]

  σ² = Σ (from i = 0 to L – 1) [ (zᵢ – m)² · p(zᵢ) ]

Here, L is the total number of intensity levels.

**Step 3: Matching to a Known PDF**

* **Compare Shapes:**  
  Compare the histogram shape from your sample to known PDFs (like Gaussian or Rayleigh).
  + If it looks like a bell curve, it’s likely Gaussian, and the estimated m and σ² fully describe it.
  + If it is skewed, you may need to solve for the parameters of a Rayleigh or Erlang distribution using the computed m and σ².

**Step 4: Special Case for Salt-and-Pepper Noise**

* **Direct Estimation:**  
  Count the fraction of pixels that have the extreme values (e.g., 0 or 255). This gives you Pₐ and Pᵦ directly, and their sum is the overall noise density.