Object Recognition & Restoration

Object Recognition in Computer Vision

Object Recognition is the task of identifying and classifying objects within an image or video. It answers two key questions:

- **What** objects are present?
- **Where** are they located?

It is a fundamental part of many computer vision applications like self-driving cars, facial recognition, security surveillance, medical image analysis, etc.

Key Steps in Object Recognition

1. **Input Image**
   - An image or video frame is given as input to the system.
2. **Preprocessing**
   - The image may be resized, normalized, or enhanced to improve model performance.
3. **Feature Extraction**
   - Important characteristics (like edges, textures, shapes) are extracted.
   - Earlier methods used techniques like SIFT, SURF, HOG manually.
   - Modern methods use **deep learning** (CNNs) to automatically learn features.
4. **Object Detection**
   - The system identifies *where* objects are in the image.
   - Output: bounding boxes around objects.
5. **Object Classification**
   - Each detected object is classified into one of the known categories (e.g., cat, dog, car).
6. **Post-processing**
   - Filter redundant detections (e.g., using Non-Maximum Suppression).

Common Techniques

1. Traditional Methods (Before Deep Learning)

- **Template Matching**: Compare image patches to stored templates.
- **Feature-based Methods**: Detect keypoints and match features using descriptors (like SIFT, SURF).

2. Deep Learning-based Methods

- **Convolutional Neural Networks (CNNs)** are the backbone.
- Models like:
  - **R-CNN, Fast R-CNN, Faster R-CNN** (Region-based object detection)
  - **YOLO (You Only Look Once)** (Real-time object detection)
  - **SSD (Single Shot MultiBox Detector)** (Fast and accurate detection)

Popular Object Recognition Models

Model	Key Feature	Use-case
R-CNN	Region proposals + CNN	Accurate but slow
YOLO	Single CNN for detection & classification	Real-time detection
SSD	Multiscale feature maps	Fast and reasonably accurate
Vision Transformers (ViTs)	Transformer architecture instead of CNNs	High accuracy in newer tasks

---

Applications of Object Recognition

- **Autonomous Vehicles**: Recognizing pedestrians, vehicles, traffic signs.
- **Healthcare**: Detecting tumors in medical images.
- **Security**: Face recognition in surveillance.
- **Retail**: Automated checkout systems.
- **Robotics**: Grasping and manipulating objects.

Challenges in Object Recognition

- **Occlusion**: Objects partially hidden.

- **Scale Variations**: Objects appearing larger or smaller.
- **Lighting Changes**: Different brightness, shadows.
- **Intra-class Variability**: Objects of same type looking different (e.g., different dog breeds).
- **Real-time Performance**: Recognizing objects fast enough for real-world use.

Quick Example:

Imagine a self-driving car camera feed:

- The system **detects** a bounding box around an object.
- It **classifies** the object inside the box as "pedestrian".
- Based on the detection, the car decides to slow down or stop.

Object Detection vs. Object Recognition in Computer Vision

□ Main Difference:

Aspect	Object Detection	Object Recognition
Goal	Find *where* objects are and what they are	Identify *what* is present
Output	Bounding box + Class label	Class label only
Example	"There is a cat at (x, y, width, height)"	"This image has a cat"
Use Case	Self-driving cars (need location)	Image search (need label)

□ What is Object Recognition?

- Recognizing or *classifying* the **entire image** or **detected object** into a category.
- It does **NOT** focus on *where* the object is.
- Focus: **"What is in the image?"**

□ Example:

- An image is given → The model says "Dog" or "Car".

□ What is Object Detection?

- **Finding** (locating) objects in the image **AND** classifying them.
- It gives **both**:
  - The **type** of object
  - The **location** (bounding box coordinates)

□ Example:

- An image of a street is given →
  Model detects:
  - **Car** at (x1, y1, w1, h1)
  - **Pedestrian** at (x2, y2, w2, h2)

□ Quick Analogy:

Think about looking at a crowded scene:

- **Recognition**: You say "I see people and cars."
- **Detection**: You point your finger and say, "There is a person here, a car there, another person over there."

□ Example Scenario:

Imagine a **self-driving car**:

- **Recognition**:
  - Car sees the image and knows "there is a traffic light."
- **Detection**:
  - Car *locates* the traffic light, *identifies its position*, and reacts (e.g., slow down).

Clearly, in critical systems, we need detection more than just recognition.

□ **Summary:**

Object Recognition	Object Detection
Only *what* objects are present	*What* + *Where* objects are
Single label or multiple labels for entire image	Label + Bounding Box for each object
Used for basic classification tasks	Used for tracking, counting, autonomous systems

□ Patterns and Pattern Classes in Computer Vision

---

1. What are Patterns in Computer Vision?

A **Pattern** is any arrangement of visual features (shapes, colors, textures, structures) that carries useful information and can be **recognized** or **classified** by a computer.

□ In simple words:

- A pattern is **something identifiable** based on its **visual characteristics**.

□ **Examples of Patterns**:

- A human face
- The shape of a cat
- The letter "A"
- Road signs
- Tumor regions in medical images

These patterns have unique, **repeatable** properties that a system can detect and recognize.

---

2. What are Pattern Classes?

A **Pattern Class** is a group or a category of similar patterns that share common features.

 In simple words:

- **Similar patterns** are grouped into **classes**.

 Examples of Pattern Classes:

Individual Patterns	Pattern Class
Face of Person A, Face of Person B, Face of Person C	"Human Faces"
Different types of apples	"Apples"
Stop signs, yield signs	"Traffic Signs"
Dogs of different breeds	"Dogs"

Each **Pattern Class** represents **many examples** that vary slightly but belong to the same general category.

---

3. How are Patterns and Classes used in Computer Vision?

Computer Vision systems:

- **Detect** patterns in images
- **Classify** detected patterns into one of the **predefined classes**

 Steps involved:

Step	Description
1. Pattern Extraction	Capture relevant features from the image (edges, textures, colors, shapes)
2. Feature Representation	Represent patterns numerically (feature vectors)
3. Classification	Use machine learning or deep learning models to classify

Step	Description
	into pattern classes

---

4. □ Key Concepts Related to Patterns and Classes

a) Intra-class Variation

- Even within a class, patterns can look slightly different.
- Example: Different types of dogs in the "dog" class — different colors, sizes.

b) Inter-class Similarity

- Sometimes patterns from different classes look similar.
- Example: A cat and a fox might have similar fur textures.

---

5. □ How Patterns are Represented?

- **Feature Vectors**: Numerical descriptions of important attributes.
- **Templates**: Standard images or models used for matching.
- **Descriptors**: Methods like SIFT, SURF, ORB capture keypoints and describe local regions.

---

6. □ Pattern Recognition Process

Stage	Details
Sensing	Capture image/video through sensors
Preprocessing	Remove noise, enhance image
Feature Extraction	Identify important characteristics
Pattern Classification	Assign the pattern to a known class
Post-processing	Refine the results if needed

---

7. □ Real-Life Examples of Pattern Classes in Computer Vision

Application	Pattern Classes
Facial recognition	Different individuals
Traffic sign recognition	Stop, yield, speed limit signs
Medical diagnosis	Healthy tissue vs tumor
Handwritten digit recognition	Digits 0-9
Wildlife monitoring	Different animal species

---

8. □ Challenges in Pattern and Class Identification

- **Noise**: Images may be blurry or noisy.
- **Variability**: Lighting, rotation, scale differences affect pattern appearance.
- **Complex Backgrounds**: Patterns may not be isolated.
- **Partial Visibility**: Pattern may be partly occluded.

Solutions include:

- Robust feature extraction
- Data augmentation
- Deep learning (CNNs) that automatically learn important features

---

□ Summary Table

Aspect	Pattern	Pattern Class
Meaning	Specific object instance or feature	Group of similar patterns
Example	A picture of a dog	"Dogs" class
Role	Input for classification	Target for classification
Variations	Due to noise, transformations	Handled using more examples per class

□ Statistical Pattern Recognition in Computer Vision

---

1. What is Statistical Pattern Recognition?

Statistical Pattern Recognition is a method where patterns (objects, shapes, textures) are recognized **based on statistical features** extracted from data.

□ In simple words:

- It **analyzes numbers and probabilities** to decide **which class a new pattern belongs to**.
- It assumes **patterns can be described by statistical properties** like **mean**, **variance**, **probability distributions**, etc.

---

2. □ Where does it fit in Computer Vision?

In **Computer Vision**, when we want a system to recognize objects like faces, digits, or traffic signs:

- We first extract **features** numerically from images.
- Then, using **statistical techniques**, we **classify** the feature vectors into different **pattern classes**.

---

3. □ Key Steps in Statistical Pattern Recognition

Step	Description
1. Data Collection	Gather a large number of images or patterns
2. Feature Extraction	Extract numerical features (color, edges, shapes)
3. Classifier Design	Build a statistical model that can separate classes
4. Training	Learn the parameters (like mean, variance) from data
5. Testing	Apply the trained model to new unseen patterns

---

4. □ Important Concepts

a) Feature Vector

- An image or pattern is **converted into numbers** (vector form).
- Example: A cat image ➡️ [height, width, color histogram, edge density]

b) Probability Density Function (PDF)

- Describes the **likelihood** that a pattern belongs to a certain class.
- Example: The probability that a shape belongs to "cat" vs. "dog".

c) Bayes Decision Theory

- Fundamental theory in statistical pattern recognition.
- It says:
  "Choose the class with the highest probability given the evidence (features)."

🧠 Bayes Formula:

$$P(Class|Features) = \frac{P(Features|Class) \times P(Class)}{P(Features)}$$

Where:

- $P(Class|Features)$ → Posterior probability (what we want)
- $P(Features|Class)$ → Likelihood (how likely features match class)
- $P(Class)$ → Prior probability (general chance of class)
- $P(Features)$ → Evidence (normalizing factor)

5. Popular Statistical Classifiers

Classifier	Working Idea
k-Nearest Neighbors (k-NN)	Classify based on closest data points
Naive Bayes	Uses Bayes' theorem with strong feature independence assumption
Linear Discriminant Analysis (LDA)	Projects data in such a way that classes are well-separated
Gaussian Mixture Models	Models data as a mixture of multiple Gaussian

Classifier	Working Idea
(GMM)	distributions
Support Vector Machines (SVM)	(Though originally not fully statistical) finds best separation line between classes

---

6. □ Example: Face Recognition

Imagine building a simple face recognition system using statistical pattern recognition:

- **Collect** hundreds of face images of different people.
- **Extract features** like distances between eyes, nose width, jaw angle.
- **Compute statistics** like mean feature values for each person (class).
- **Train** a model based on these statistics.
- **Predict** the identity of a new face image based on learned probabilities.

---

7. □ Advantages

□ Works well with **limited data** (compared to deep learning).
□ **Simple and explainable** models.
□ Useful when **features are carefully chosen**.
□ Good for **real-time**, **low-computation** applications.

---

8. □ Challenges

□ Depends heavily on **good feature extraction**.
□ **Assumptions** (like normal distribution) may not hold true always.
□ May not handle **very complex or high-dimensional** data well.

Modern deep learning (like CNNs) reduces need for manual feature extraction, but **statistical methods are still fundamental** — especially in simple, interpretable, and quick systems!

□ Summary Table

Aspect	Description
Definition	Recognition based on statistical features and probabilities
Key Concepts	Feature vectors, PDFs, Bayes theory
Steps	Feature extraction → Classifier training → Prediction
Classifiers	k-NN, Naive Bayes, LDA, GMM
Strengths	Simple, interpretable, needs less data
Limitations	Struggles with complex, messy, large-scale data

□ Syntactic Pattern Recognition in Computer Vision

---

1. What is Syntactic Pattern Recognition?

Syntactic Pattern Recognition is a method where patterns are **represented and recognized based on their structural relationships** — like how words are made from letters and sentences are made from words.

□ In simple words:

- **Objects are seen as being built from simpler parts**, arranged according to some **rules** (like grammar rules in language).

It treats patterns **like a language**:

- **Simple elements (primitives)** = small parts like lines, circles, corners
- **Rules (grammar)** = how these parts combine to form a bigger object

---

2. □ Where does Syntactic Pattern Recognition fit in Computer Vision?

When recognizing **complex structures** in an image (like a car made of wheels, windows, and body), it's not enough to just look at color or texture —
You need to **understand how parts are arranged together**.

Syntactic pattern recognition models this **hierarchical composition**.

---

3. □ Key Concepts

Concept	Meaning
Primitives	Basic elements detected in the image (lines, edges, curves)
Grammar	Set of rules that describe how primitives combine to form patterns
Production Rules	Specific transformations (like in a language: Noun → Article + Noun)
Parse Tree	Tree-like structure showing how primitives are combined step-by-step
Recognition	Matching the observed structure to the grammar to classify the pattern

---

4. □ Simple Example: Recognizing a "House" Drawing

Suppose the drawing of a simple house is made of:

- **Triangle** (roof)
- **Square** (body)

The **grammar** might say:

House → Triangle + Square

This is a **production rule** that defines a **House**.

When the system sees a triangle over a square in an image:

- It matches it to the rule "House → Triangle + Square"
- It recognizes it as a "House".

---

5. □ How Syntactic Pattern Recognition Works

Step	Description
1. Feature Detection	Detect basic elements (primitives) like lines, arcs, circles
2. Structural Analysis	Analyze how these primitives are related (position, size, connection)
3. Grammar Definition	Define grammar that describes valid patterns
4. Parsing	Try to construct a pattern by applying grammar rules to detected primitives
5. Decision Making	If the parsing succeeds → recognize the object

---

7. □ Why Use Syntactic Pattern Recognition?

□ Good for **complex structured patterns** (multi-part objects)
□ **Hierarchical understanding** (object made of sub-objects)
□ Models **relationships between parts** (important in real-world images)

---

8. □ Challenges

□ **Grammar design** can be very hard for complicated objects
□ **Noisy images** can confuse parsing (missing parts break structure)
□ **Computational cost**: Parsing trees and matching rules can be slow

In modern computer vision, **deep learning** often implicitly learns these structures, but **syntactic methods** are still powerful for **structured, symbolic tasks** like:

- Scene understanding
- Handwritten symbol parsing
- Medical imaging (structured anatomy)

---

9. □ Summary Table

Aspect	Syntactic Pattern Recognition
Based On	Structural relationships between parts

Aspect	Syntactic Pattern Recognition
Key Units	Primitives (basic shapes), Grammar (rules)
Process	Detect parts → Apply grammar rules → Recognize object
Example	Recognizing a house made of triangle + square
Strength	Handles complex, hierarchical patterns well
Limitation	Sensitive to noise, needs careful grammar design

□ Optimization Techniques in Recognition in Computer Vision (CV)

---

1. What is Optimization in Recognition?

In Computer Vision, **Recognition** means identifying what an image contains (e.g., face, car, cat, etc.).
To **improve the recognition** — make it faster, more accurate, and more efficient — we use **Optimization Techniques**.

□ **Optimization** is the process of:

- **Minimizing errors**
- **Maximizing accuracy**
- **Improving model performance**
  by **adjusting parameters** of the recognition model.

In simple terms:
"Optimization = Find the best settings (parameters) for the best recognition results."

---

2. □ Why is Optimization Needed in Recognition?

- To find **best features** that represent objects.
- To **tune model parameters** (like weights in a classifier).
- To **reduce training error** and **improve testing accuracy**.
- To make the recognition system **fast** and **memory-efficient**.

Without optimization, a recognition system would be:

- Inaccurate (many wrong predictions)
- Slow
- Overfitted (only working well on training data)

---

3. □ Where Optimization is Used in Recognition?

Stage	Use of Optimization
Feature Selection	Find the most important features
Model Training	Learn best weights or parameters
Hyperparameter Tuning	Select best settings (learning rate, regularization)
Post-Processing	Refine results (e.g., boundary smoothing, output filtering)

---

4. □ Common Optimization Techniques

(a) Gradient Descent

- The most popular optimization method.
- Idea: Start with some guess and **move step-by-step towards minimum error** by following the slope (gradient).

□ Simple analogy:
"Imagine you are at the top of a hill (error) and you want to reach the bottom (minimum error) by taking small steps downhill."

□ Types:

- **Batch Gradient Descent**: Uses all data at once (slow but stable)
- **Stochastic Gradient Descent (SGD)**: Uses one sample at a time (faster, noisy)
- **Mini-batch Gradient Descent**: A balance between the two.

(b) Conjugate Gradient Method

- Advanced technique faster than simple gradient descent.
- Takes smarter steps considering previous directions to **avoid zig-zagging** and reach minimum faster.

(c) Newton's Method

- Uses second derivatives (curvature information) to reach the minimum very fast.
- Powerful but computationally expensive (needs calculating Hessian matrix).

(d) Evolutionary Algorithms (Genetic Algorithms)

- Inspired by natural evolution (mutation, crossover, selection).
- Useful when problem space is **very complex** or **non-differentiable**.
- Example: Finding the best combination of features for face recognition.

(e) Simulated Annealing

- Inspired by the process of metal cooling slowly.
- Explores the solution space widely at the beginning (even accepting bad moves sometimes) to **escape local minima**.
- Later gradually focuses on better solutions.

(f) Particle Swarm Optimization (PSO)

- Inspired by the behavior of bird flocks or fish schools.
- Particles (solutions) fly in the search space, sharing information, to find the best spot (optimal solution).

(g) Hyperparameter Optimization Techniques

Technique	Description
Grid Search	Try all combinations of parameters systematically
Random Search	Randomly try different parameter combinations
Bayesian Optimization	Predict which parameters to try next based on past trials
Genetic Search	Evolve a population of parameters over time

---

5. □ Practical Examples in CV

Task	Optimization Application
Face Recognition	Train model to minimize classification error
Object Detection	Optimize bounding box positions and class labels
Semantic Segmentation	Tune pixel-wise classification for minimal loss
Optical Character Recognition (OCR)	Minimize text recognition error on scanned documents
Image Matching	Optimize feature descriptor matching between images

---

6. □ Advantages of Good Optimization

□ Higher accuracy
□ Faster convergence during training
□ Better generalization to unseen images
□ Reduced overfitting
□ Efficient use of memory and computation

---

7. □ Challenges in Optimization

□ Getting stuck in **local minima** (sub-optimal solutions)
□ **Slow convergence** if learning rate is bad
□ **Overfitting** if over-optimized for training data
□ **High computation** cost for complex techniques

□ Summary Table

Aspect	Optimization in CV Recognition
Purpose	Improve recognition performance
Popular Methods	Gradient Descent, Newton's Method, Genetic Algorithms, PSO
Application Stages	Feature selection, Model training, Hyperparameter tuning
Strength	Makes models accurate, fast, efficient
Limitation	Computational cost, risk of overfitting

□ Restoration in Computer Vision (CV)

---

1. What is Restoration in CV?

Restoration in Computer Vision means:

Recovering an original, clean image from a degraded (noisy, blurred, or distorted) version.

In simple words:

- **You have a damaged photo** (because of noise, motion blur, lens problems, etc.).
- **Restoration** tries to **fix** it — making it look as close as possible to the **original, undamaged version**.

---

2. □ Why is Image Restoration Needed?

In real life, when capturing images, problems occur:

- Camera movement ➔ **motion blur**
- Poor lighting ➔ **noise**
- Dirty lens ➔ **blur**
- Transmission errors ➔ **missing information**

If we can **restore** the original image:

- Recognition tasks (face detection, object detection) become more accurate.
- Images look clearer and more useful for analysis.

---

3. □ Key Terms

Term	Meaning
Degradation	Process that damages the image (noise, blur, distortion)
Noise	Random variations in pixel values
Blur	Smearing or loss of sharpness in the image
Inverse Problem	Reversing the degradation to recover the original image

---

4. □ Common Causes of Degradation

Cause	Effect
Sensor Noise	Random dots, grainy appearance
Motion Blur	Smearing due to camera or object movement
Defocus Blur	Out-of-focus, soft images
Atmospheric Turbulence	Distortions in satellite or drone images
Compression Artifacts	Blockiness or fuzziness (e.g., JPEG artifacts)

---

5. □ Basic Restoration Process

Step	Description
1. **Model the Degradation**	Understand how the image got damaged (what type of noise or blur)
2. **Apply Restoration Technique**	Use mathematical tools to reverse or correct the damage
3. **Evaluate Quality**	Check if the restored image is close to the original

---

6. □ Popular Restoration Techniques

(a) Filtering Methods

- **Mean Filter**: Smooths image by averaging nearby pixels (reduces noise but blurs edges).
- **Median Filter**: Replaces each pixel with the median of its neighborhood (great for "salt-and-pepper" noise).
- **Wiener Filter**: Advanced filter that balances noise reduction and detail preservation.

(b) Inverse Filtering

- Assumes you know how the image was blurred (degradation function).
- Applies an **inverse** to "undo" the blurring.

Problem: Sensitive to noise — works well only when noise is low.

(c) Regularized Filtering (Tikhonov Regularization)

- Handles noise better than pure inverse filtering.
- Adds a "penalty" for roughness, ensuring smoother images.

(d) Blind Deconvolution

- Used when you **do not know** how the image was blurred.
- Simultaneously estimates both:
  - the original image, and
  - the blur function.

Hard but powerful!

(e) Deep Learning-Based Restoration

- **Denoising Autoencoders**: Neural networks trained to remove noise.

- **CNNs (Convolutional Neural Networks)**: Trained end-to-end to restore images.
- **GANs (Generative Adversarial Networks)**: Create very realistic restored images.

Modern restoration often uses deep learning because:

- It handles **complex degradation** easily.
- It produces **high-quality, natural-looking images**.

---

7. □ Example: Image Degradation and Restoration

Stage	Image
Original Image	Clear photo
Degraded Image	Blurred + noisy version
After Restoration	Sharper, cleaner version that looks like the original

---

8. □ Important Points

Aspect	Description
Input	Degraded image
Goal	Estimate the original clean image
Challenges	Exact degradation process often unknown
Evaluation	PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index)

---

9. □ Challenges in Restoration

□ Perfect restoration is almost impossible if too much information is lost.
□ Hard to restore when multiple degradations happen together (e.g., noise + blur).
□ Computational cost: Advanced methods can be slow.

□ Summary Table

Topic	Summary
Purpose	Recover original image from degraded one
Causes of Degradation	Noise, blur, distortions
Techniques	Filtering, inverse filtering, blind deconvolution, deep learning
Challenges	Unknown degradation, high noise, computation time
Modern Trend	Deep learning-based restoration for high-quality results

□ Image Restoration Model in Computer Vision (CV)

---

1. What is an Image Restoration Model?

In Computer Vision, an **Image Restoration Model** is a **mathematical representation** that explains:

- How an image got **degraded** (damaged),
- And how we can **reverse** that degradation to **recover the original clean image**.

In simple words:
It models the relationship between the **original image**, the **degradation process**, and the **observed (noisy or blurry) image**.

---

2. □ Why is an Image Restoration Model Needed?

- To **understand** and **mathematically describe** how an image becomes poor-quality.
- To **apply correct techniques** to restore images effectively.
- To **predict and correct distortions** systematically.
- To **design algorithms** for denoising, deblurring, and recovering lost information.

Without a proper model, restoration would just be "guesswork."
A **good model** allows for **accurate, reliable restoration**.

3. 🛠️ Basic Mathematical Model

The **general form** of the Image Restoration Model is:

$$g(x, y) = h(x, y) * f(x, y) + \eta(x, y)$$

where:

Symbol	Meaning
$g(x, y)$	Observed degraded image
$h(x, y)$	Degradation function (blur, distortion)
$f(x, y)$	Original (true) image
$*$	Convolution operation
$\eta(x, y)$	Additive noise

□ In words:

- The observed image g(x,y) is the **blurred** and **noisy** version of the original image f(x,y).
- The blur is caused by convolution with h(x,y) (the **degradation function**).
- Noise η(x,y) is **added randomly** during image acquisition or transmission.

---

4. □ Components Explained

(a) Original Image (f(x,y))

- The clean, undistorted image we want to recover.

(b) Degradation Function (h(x,y))

- Describes how the original image gets damaged.
- Examples:
  - **Motion Blur**: Camera shakes while taking the photo.
  - **Out-of-Focus Blur**: Lens not properly focused.
  - **Atmospheric Turbulence**: Distortion in satellite imaging.

It acts like a **filter** applied to the original image.

(c) Noise (η(x,y))

- Random variations.
- Sources:
  - Electronic sensor noise
  - Poor lighting
  - Transmission errors

(d) Convolution (∗)

- A mathematical way of **spreading** or **smearing** the image based on the degradation function.
- Central operation in modeling blur.

---

5. □ Visualization

Component	Meaning
f(x,y)	True Image (sharp photo)
h(x,y)	Blur Function (e.g., motion or defocus)
η(x,y)	Random Noise (salt and pepper, Gaussian noise)
g(x,y)	Observed Blurry and Noisy Image

□ Example:

Imagine:

- A clear photo of a **cat** = f(x,y)
- You move the camera while shooting = h(x,y) (motion blur)
- There is low light = η(x,y)

- The final blurry, noisy cat photo = g(x,y)

---

6. □ Restoration Goal

Given:

- g(x,y) (the bad image)
- h(x,y) (known or estimated)
- some knowledge about η(x,y)

□ **Estimate f(x,y)** as closely as possible.

In simple terms: "Find the clean image that, when blurred and added noise, would create the degraded one we see."

---

7. □ Methods for Solving the Model

Method	Description
Inverse Filtering	Directly reverse the blur assuming no noise
Wiener Filtering	Balance reversing blur and reducing noise
Blind Deconvolution	When h(x,y) is unknown, estimate both the blur and the image
Regularization Methods	Add extra conditions (like smoothness) to get better results
Deep Learning Methods	CNNs or GANs trained to learn restoration directly from examples

---

8. □ Challenges

- Often, h(x,y) and η(x,y)are **unknown or hard to estimate**.
- Noise can **amplify during inverse filtering** and create artifacts.
- Complex degradations (like **atmospheric turbulence**) are **very difficult** to model exactly.
- **Trade-off**: Remove noise but not lose important details!

□ Noise Models in Computer Vision (CV)

---

1. What is Noise in Images?

In Computer Vision, **noise** refers to **random variations** in pixel values that distort the true information in an image.

- It **degrades image quality**.
- Makes **analysis** (like object detection, recognition) **more difficult**.
- **Occurs naturally** during image capture, transmission, or processing.

In simple words:
Noise is unwanted "dirt" or "errors" sprinkled randomly in the image.

---

2. □ Why Study Noise Models?

- **Understanding** the noise type helps **choose the right denoising technique**.
- **Design better filters** and **restoration methods**.
- Some machine learning models also **simulate noise** to make systems **robust**.

---

3. □ Types of Noise Models in Computer Vision

There are **many kinds** of noise, depending on how they arise.
Let's look at the **most common noise models**:

(A) Gaussian Noise □

- **Nature:** Random noise following a **Gaussian (normal) distribution**.
- **Common in:** Electronic sensors (thermal noise), scanning.
- **Appearance:** Slight fuzziness across the entire image

Mathematical Model:

$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)$$

Where:

- μ = mean (usually 0)
- σ = standard deviation (controls "spread" of noise)

Graph: Bell-shaped curve (normal distribution)

Example:

- Low-light photography.
- Satellite imaging.

(B) Salt-and-Pepper Noise □□

- **Nature:** Sudden appearance of random **black (0)** and **white (255)** pixels.
- **Common in:** Faulty memory chips, transmission errors.
- **Appearance:** "Salt" (white specs) and "Pepper" (black dots) scattered randomly.

Example:

- Corrupted scanned documents.
- Data dropouts in cameras.

Important:

- This noise is **impulsive** (appears sharply at few pixels).
- Best removed by **Median filters**.

(C) Poisson Noise (Shot Noise) □

- **Nature:** Noise related to the **quantum nature of light**.
- **Common in:** Photon counting devices like medical imaging, low-light photography.
- **Appearance:** Dependent on intensity — higher intensity = more noise.

Mathematical Model:

Poisson distribution:

$$p(z) = \frac{\lambda^z e^{-\lambda}}{z!}$$

Where λ is the expected number of events (like photons hitting the sensor).

Example:

- Medical X-ray imaging.
- Astronomy images.

Key Point:

- Poisson noise **increases** with brightness.

(D) Speckle Noise □

- **Nature:** Noise that **multiplies** with pixel values rather than adding.
- **Common in:** Radar, ultrasound, coherent imaging systems.
- **Appearance:** Grainy or spotty texture.

Mathematical Model:

$$g(x, y) = f(x, y) + f(x, y) \times n(x, y)$$

where:

- $f(x, y)$ = original image
- $n(x, y)$ = multiplicative noise (often Gaussian distributed)

Example:

- SAR (Synthetic Aperture Radar) images.
- Medical ultrasound imaging.

Important:

- Harder to remove because it **depends on the image content** itself.

(E) Quantization Noise □

- **Nature:** Noise from rounding pixel values when converting continuous signals into discrete form.
- **Common in:** Image compression, digitization.
- **Appearance:** Fine "banding" or loss of subtle color shades.

Example:

- JPEG compression artifacts.
- Low bit-depth images (8-bit color, etc).

Key Point:

- Occurs during **digitization and compression**, not during natural capture.

---

4. □ Summary Table

Noise Type	Nature	Appearance	Common In
Gaussian Noise	Additive, Normal distribution	Fuzzy image	Cameras, sensors
Salt-and-Pepper	Random black and white pixels	Dotted corruption	Memory faults, transmission
Poisson Noise	Intensity-dependent	Varies with brightness	X-ray, astronomy
Speckle Noise	Multiplicative	Grainy texture	Radar, ultrasound
Quantization Noise	Rounding errors	Banding artifacts	Compression, digitization

---

5. □ Real-World Examples

Scenario	Likely Noise Type
Mobile photo in low light	Gaussian noise
Satellite image transmission error	Salt-and-Pepper noise
Medical ultrasound scan	Speckle noise
Over-compressed JPEG photo	Quantization noise
Astronomical telescope image	Poisson noise

6. □ Handling Noise

Depending on the noise model, different **noise reduction techniques** are applied:

Noise Type	Popular Techniques

Noise Type	Popular Techniques
Gaussian	Gaussian filters, Wiener filter, Non-local means
Salt-and-Pepper	Median filter, Adaptive median filter
Poisson	Variance-stabilizing transform (e.g., Anscombe transform)
Speckle	Lee filter, Frost filter
Quantization	Smoothing, dithering