Feature Extraction

Michelle Marchesini Vanegas

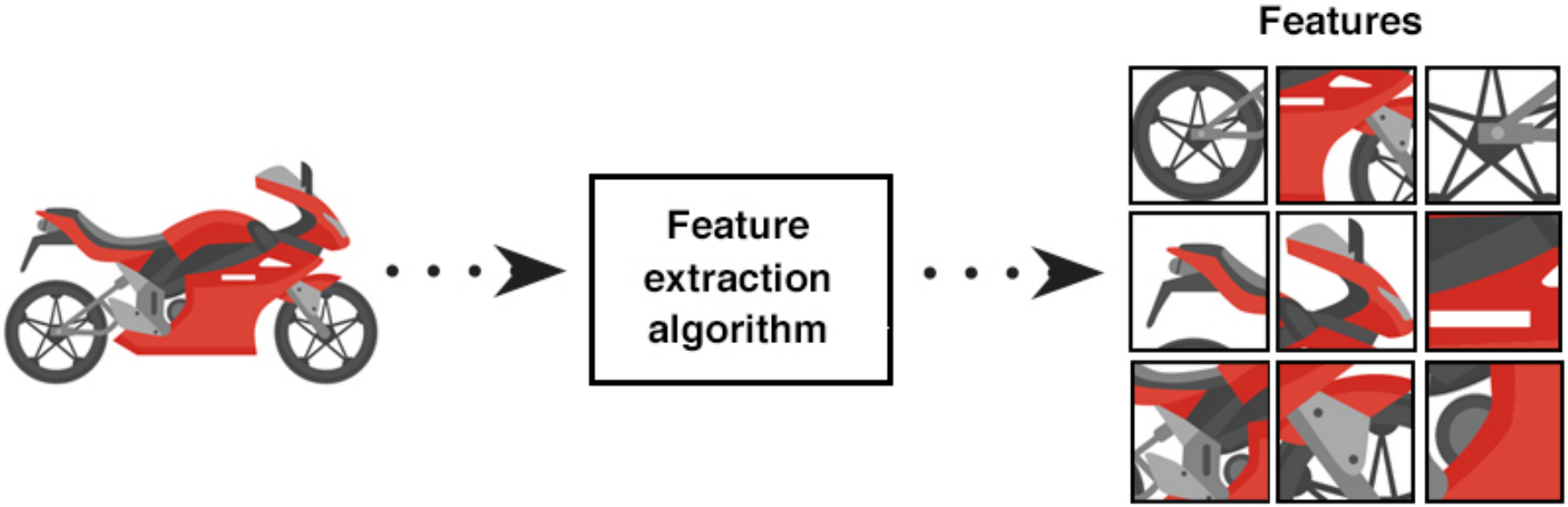
Earlham College

February 27th

Feature Extraction

Feature extraction is a process utilized by machine learning. Which is where it takes raw data and transforms it into a meaningful feature that can help a model get a better understanding of patterns and even make predictions. Feature extraction is the process of reducing the dimensionality of image data by identifying and describing important patterns.

- Converts raw pixel data into informative representations
- Enables tasks like image classification, recognition, and tracking
- Examples: edges, corners, blobs, textures



Types of Features

Features can be categorized by their level of abstraction:

- **Low-level:** Directly extracted from pixel values (e.g., edges, corners)
- **Mid-level:** Represent visual patterns and textures (e.g., Gabor features, LBP)
- **High-level:** Abstract semantic information (e.g., facial keypoints, object parts)

They can also be:

- **Global features:** Describe the entire image (e.g., color histogram)
- **Local features:** Describe small patches (e.g., SIFT keypoints)

Edge Detection

Edge detection in feature extraction is a technique used to identify points in a digital image where the image brightness changes sharply. These points typically mark object boundaries, textures, or transitions, making them highly informative features for tasks like object recognition, segmentation, and motion tracking.

- **Sobel:** Computes gradient in X and Y directions
- **Canny:** Multi-stage process with noise reduction, gradient calculation, and edge tracking
- **Prewitt:** Similar to Sobel but with different kernels.

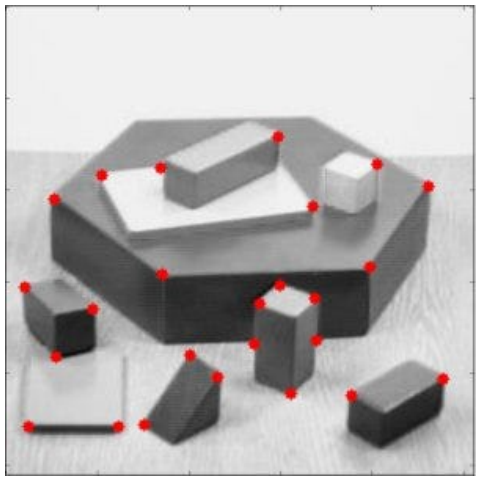
Original Image

Laplacian Edge Detection

Corner Detection

Corner detection is used to find points in an image where two or more edges meet, typically sharp changes in intensity in multiple directions. These points are often highly distinctive and reliable for tracking, matching, or recognizing objects.

- **Harris Corner Detector**: Based on gradient changes; detects corners with high intensity variation in all directions.
- **Shi-Tomasi**: Selects the best corners using the minimum eigenvalue criterion.
- **FAST (Features from Accelerated Segment Test)**: Fast and efficient corner detector for real-time applications.



Histogram Of Oriented Gradients (HOG)

The histogram of Oriented Gradients is a feature extraction in which it looks for a **specific** object detection task. For example, pedestrians.
How does it work?

1. It starts by diving the image into multiple small regions.
2. Then it calculates the edge direction of each pixel cell.
3. It creates a a histogram of these gradient direction for each cell/
4. It normalize the histogram. Which means that reduces the lighting changes
5. Finally it combines all feature extractions.

Scale-Invariant Feature Transform (SIFT)

SIFT is a feature extraction algorithm that detect and describes local features in images. What this means is that is finding the Keypoints.

- **Scale-space Extrema Detection:**
  Identify potential keypoints using Difference-of-Gaussians across multiple scales.
- **Keypoint Localization:**
  Filter out low-contrast points and poorly localized edges.
- **Orientation Assignment:**
  Assign one or more orientations based on local gradient directions.
- **Keypoint Descriptor Generation:**
  Create a 128-dimensional vector from gradients around the keypoint.

Keypoint Detection

Keypoint detection identifies distinctive, repeatable points in an image such as corners, blobs, or edges that can be reliably recognized across transformations. Its Purpose is to find stable and unique image regions for object recognition, image matching, tracking, or 3D reconstruction. Think of keypoints like landmarks on a map: distinct and easily recognizable, no matter the zoom level or orientation.

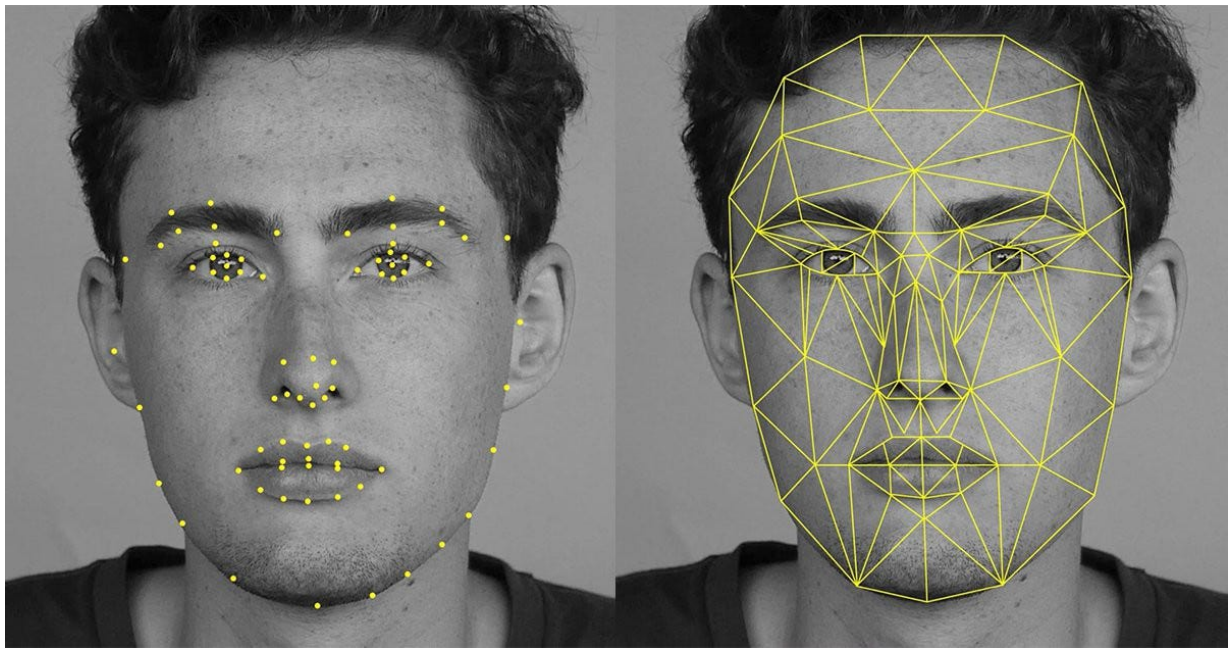


Image source: GitHub 3ba2ii

Local Binary Pattern

Local Binary Pattern is a method used to summarize local texture in an image. It works by comparing the intensity of a central pixel to its surrounding neighbors. If a neighbor's value is greater than or equal to the center pixel, assign a 1. Otherwise, assign a 0. The result is a binary number that represents the local texture pattern:

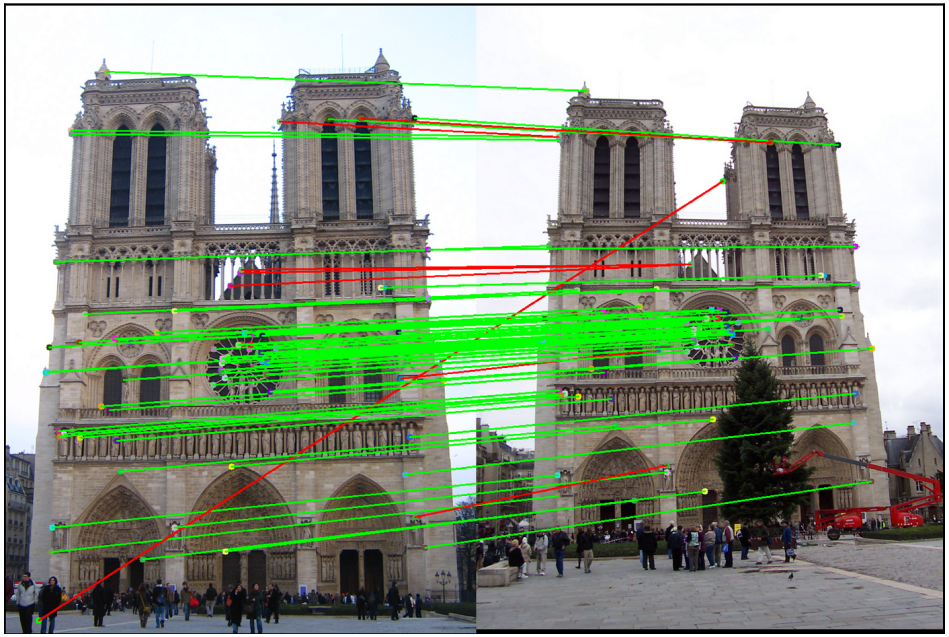
How it works:

- Take a pixel and compare it to its surrounding pixels
- Generate a binary code (e.g., 11010011)
- Convert this binary pattern to a decimal value
- Repeat for all pixels to build a LBP image

Feature Matching

Feature matching is the process of finding corresponding features (like keypoints, corners, or descriptors) between two or more images. It's a critical step in many computer vision tasks such as: image stitching, object recognition, 3D reconstruction, tracking and motion estimation

1. Detect keypoints in overlapping images.
2. Match features between images.
3. Estimate transformation (homography).
4. Align and blend images together.



Feature Extraction Summary

Feature extraction is the process of identifying meaningful visual information in images to support tasks like classification, recognition, and tracking.

Core Techniques:

- **Keypoint Detection:** Identifies distinctive and repeatable points, often used in matching and recognition.
- **Corner & Edge Detection:** Highlights sharp intensity changes (corners represent strong features, edges outline object boundaries.)
- **SIFT (Scale-Invariant Feature Transform):** Detects and describes keypoints that are invariant to scale and rotation.
- **HOG (Histogram of Oriented Gradients):** Captures local gradient orientations to describe object shape and structure.
- **Local Binary Patterns (LBP):** Encodes texture by comparing local pixel intensity patterns; effective for tasks like face recognition.
- **Feature Matching:** Compares feature descriptors across images to find correspondences for tasks like image stitching or object tracking.