UNIT-2

Binary shape analysis: Digital image processing makes the use of algorithms that help us to extract essential features from the images. The aim of the digital image processing is to enhance image quality so that we can extract useful information which can be used later for further detailed studies. Pixel is the smallest unit in the image that has some color value. Regions of Interest are those areas on the image on which we can perform many operations related to any digital image processing**.** Binary shape analysis is a subfield of computer vision that examines and interprets shapes in binary images. A binary image contains pixels with only two possible values, typically 0 (off) and 1 (on), representing the background and foreground, respectively. This simplified representation allows for efficient, focused analysis of an object's essential structure and properties.

To create a Binary image we use the thresholding. In thresholding, we define the threshold for the pixel value where we compare the image pixel value with the threshold value. The steps are as follows:

- If the image is colored or in RGB format change it to grayscale image
- Define a threshold value.
- For each pixel present in the image, if the pixel value lies below the threshold value set that pixel value to 0 else 1.
- The result is a binary image.

The contours segmentation or deformations of an object is a preprocessing step of shape retrieval and classification that segment the binary object shape in a shape-preserving sequence of contours segment using a coordination number shape segmentation approach. T

Fig: Binary Image

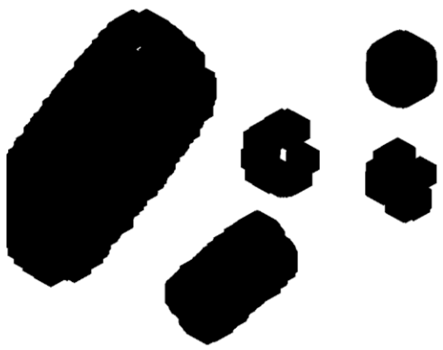
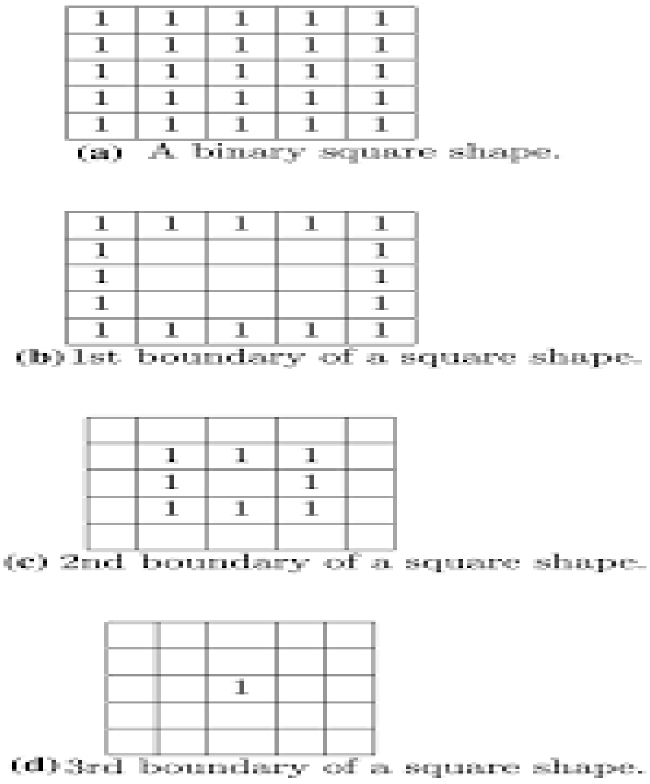


Fig: Segmented Image



1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

(a) A binary square shape.

1	1	1	1	1
1				1
1				1
1				1
1	1	1	1	1

(b) 1st boundary of a square shape.

	1	1	1	
	1		1	
	1	1	1	

(c) 2nd boundary of a square shape.

		1		

(d) 3rd boundary of a square shape.

Fig: Local Binary Patterns of Segments of a Binary Object for Shape Analysis

Core concepts and techniques

1. Connected components and labeling

- **Connected components:** These are distinct groups of connected foreground pixels that form individual objects. Connectivity is defined by whether neighboring pixels share an edge (4-connectivity) or an edge or corner (8-connectivity).
- **Object labeling:** This process assigns a unique identifier to each connected component, which allows for isolating and analyzing multiple objects in a single image.

2. Morphological operations
These mathematical operations are used to modify shapes based on the pixel configuration of an object.

- **Erosion:** Shrinks the boundaries of foreground objects and can be used to remove small noise or separate connected objects.

- **Dilation:** Expands the boundaries of foreground objects, useful for filling in small holes and bridging gaps between objects.
- **Opening:** An erosion followed by a dilation. It removes small foreground objects and smooths object boundaries.
- **Closing:** A dilation followed by an erosion. It fills in small holes and gaps within objects.

3. Shape descriptors and measurements
Shape descriptors are numerical features that quantify an object's properties for comparison and classification.

- **Area and perimeter:** The total number of pixels in a shape and the length of its boundary, respectively.
- **Moments:** A weighted average of pixel intensities that provides information on an object's position, orientation, and shape.
- **Aspect ratio:** The ratio of the object's width to its height. A value near 1 indicates a square or circular shape, while a smaller value indicates elongation.
- **Eccentricity:** Measures how elongated a shape is by calculating the ratio of the minor axis to the major axis of its equivalent ellipse.
- **Compactness:** A measure of the object's roundness, calculated as

4. Shape representation and simplification

- **Skeletons and thinning:** Thinning algorithms iteratively remove pixels from the boundary of a shape until a single-pixel-wide line or "skeleton" remains. This representation captures the object's essential structure and is invariant to rotation and scaling.
- **Contour-based representation:** Analyzing the shape by tracking and encoding its boundary pixels. This method reduces the computational load by focusing only on the perimeter.

Applications:
Binary shape analysis is a fundamental tool used in many fields:

- **Medical imaging:** Aiding diagnosis by segmenting and analyzing the shape of organs, tumors, and blood vessels in scans.
- **Optical character recognition (OCR):** Recognizing letters and characters by analyzing and comparing their simplified skeletal structures.
- **Manufacturing and quality control:** Inspecting parts to ensure they conform to a required shape and have no defects.
- **Robotics:** Assisting in robot navigation by simplifying objects and environments for pathfinding and obstacle avoidance.
- **Biological imaging:** Tracking changes in the shapes of cells or microorganisms over **time.**

Connectedness: In binary shape analysis, connectedness refers to how "on" (foreground) pixels are linked to form a single, coherent shape or region within a binary image, where pixels are either on or off. This connectivity is defined by connectivity rules (like 4-connectivity or 8-connectivity, which specify if adjacent, diagonal, or both are considered neighbors) and is used to group pixels into discrete connected components. Algorithms like connected-component labeling then assign a unique label to each distinct connected component, allowing for the extraction of meaningful shape properties and the separation of objects in the image.

How Connectedness Works

1. Binary Images:

A binary image consists of pixels with only two possible values: 0 (background) or 1 (foreground).

2. Defining Neighbors:

To determine if pixels are part of the same shape, we need rules for adjacency:

4-Connectivity: Pixels are considered connected if they are adjacent horizontally or vertically (edges touch).

8-Connectivity: Pixels are connected if they are adjacent horizontally, vertically, or diagonally (edges or corners touch).

3. Connected Components:

These are sets of foreground pixels that are connected to each other according to the chosen connectivity rule. A single connected component represents a distinct object in the binary image.

.

Object labeling and counting:

Object labeling and counting refers to labeling (identifying and categorizing) and counting (enumerating) specific items within images or video sequences. In human learning, it's a foundational math skill where children assign a number to each object. Object counting in computer vision is a task where systems automatically identify and count instances of specific objects in images or video frames. The core objective is to detect and enumerate how many objects of interest (e.g., people, vehicles, products) appear in the scene in an image or video.It automates this by using object detection and tracking to identify, locate, and then count distinct instances of objects, with applications in areas like traffic monitoring, inventory management, and security.

“Object counting in computer vision is the task of automatically identifying and enumerating specific instances of objects within digital images or video.”

Process:

1. **Image Acquisition:** Capturing images or video frames.
2. **Preprocessing:** Enhancing the image quality by reducing noise or adjusting contrast.

3. **Object Detection:** Using AI algorithms to find and locate objects of interest, often by drawing a bounding box around them.
4. **Object Tracking (for video):** Following detected objects across consecutive frames to give each a unique ID and ensure it's only counted once.
5. **Counting:** Aggregating the detected and tracked objects to produce a final count.

Methods for Object Counting

(A) Classical Computer Vision

1. **Connected Component Analysis** → Count number of labeled objects.
2. **Contour Detection (cv2.findContours in OpenCV)** → Each closed contour = 1 object.
3. **Blob Detection (Laplacian of Gaussian, DoG, MSER)** → Count blob-like objects.

(B) Deep Learning Approaches

- **Object Detection** → Detect and count objects using bounding boxes (e.g., YOLO, Faster R-CNN).
- **Instance Segmentation** → Count distinct object instances (e.g., Mask R-CNN).
- **Density Estimation** → For dense crowds, learn a density map and integrate over it (common in **crowd counting**).

Methods for Object Labeling:

(A) **Connected Component Labeling (CCL)**

- Works on binary images.
- Connectivity types:
  - **4-connectivity** → neighbors: up, down, left, right.
  - **8-connectivity** → neighbors: diagonals included.
- Algorithms:

1. **Two-pass algorithm**
   - First pass → assign temporary labels, resolve equivalences.
   - Second pass → replace temporary labels with final IDs.
2. **Union-Find (Disjoint Set)** approach.

(B) **Region Growing / Segmentation-based Labeling**

- Start from seed points, expand to neighboring pixels based on similarity.

(C) **Deep Learning-based Labeling**

- Instance segmentation networks (e.g., **Mask R-CNN, YOLO-seg**) automatically label each object with class + instance ID.

Applications:

- **Traffic Flow Monitoring:** Counting vehicles at intersections.
- **Crowdedness Estimation:** Estimating the number of people in a space.
- **Retail and Inventory:** Counting products on shelves or in a warehouse.
- **Security and Surveillance:** Monitoring and tracking entities for proactive threat detection.

Size filtering: Size filtering in computer vision refers to modifying images by selecting or emphasizing features based on their size, often by adjusting the size of a filter kernel or by applying filters at different resolutions to target objects of a specific scale. This technique can be used to remove noise, enhance edges, and detect objects of interest by blurring or sharpening aspects of the image. For example, a larger filter kernel can blur the image and remove noise, while smaller kernels can highlight fine details and edges.

How Size Filtering Works

1. **Filter Kernel Size**:

   The "filter" in computer vision is a small matrix (kernel) of values that is applied across the image. The size of this kernel is a key factor in determining which features the filter will affect.

2. **Spatial Filtering**:

   This traditional method applies a filter directly to the image pixels.

3. **Neighborhood Operation**:

   The filter kernel is slided across the image, and at each position, a calculation is performed on the corresponding pixels in the image to determine the output pixel's new value.

4. **Effect of Size**:
   - **Larger kernels**: are generally used for smoothing, blurring, and noise reduction. They process a larger neighborhood of pixels, which can effectively average out small variations and reduce high-frequency details.
   - **Smaller kernels**: are often used for sharpening and edge detection, as they focus on finer details and can highlight changes in pixel intensity, which correspond to edges.

Applications of Size Filtering

- **Noise Reduction**: By blurring an image with a large filter, smaller features and noise can be suppressed.
- **Edge Detection**: Filters designed to emphasize high-frequency changes, often using smaller kernels, can highlight edges and fine details.

- **Image Enhancement**: Adjusting the filter size can selectively enhance certain image features or blur others to improve contrast and visual quality.
- **Object Detection**: By processing the image at different scales (e.g., using filters of varying sizes or resizing the image), one can detect objects of specific sizes within the image.

Distance functions: In **computer vision**, distance functions (also called **distance metrics** or **similarity measures**) are mathematical tools used to quantify how similar or different two objects (e.g., feature vectors, images, shapes, or points) are. They play a critical role in tasks like **image retrieval, classification, clustering, object recognition, and tracking**.

Types of distance function:

1. **Pixel-based Distance Functions**

These work directly on pixel intensities of images (raw data).

- **Euclidean Distance** $d(x,y)= \sqrt{\sum_{i=1}^{i=n} (x_i - y_i)^2}$
  - Most common; measures straight-line distance in feature space.
  - Used in nearest neighbor classifiers, image comparison

- **Manhattan (City-block / L1) Distance**

$$d(x,y)= \Sigma|xi - yi|$$

  - More robust to small variations; useful in sparse data.

- **Chebyshev Distance**

$$d(x,y)=\max |xi-yi|$$

  - Focuses on maximum coordinate difference.

2. Feature-based Distance Functions

When images are represented by feature vectors (e.g., SIFT, HOG, deep features):

- **Cosine Similarity / Angular Distance**

$$sim(x,y)=x\cdot y \setminus \|x\|\|y\|$$

  - Measures angle between feature vectors (good for normalized data).
  - Used in face recognition and document/image retrieval.

- **Mahalanobis Distance**

$$d(x,)= \sqrt{(\vec{x}-\vec{y})^T S^{-1} (\vec{x}-\vec{y})}.$$

Considers correlation between features (matrix S is covariance).

  - Useful in classification with correlated feature sets.
- **Hamming Distance**
  - Counts the number of differing bits.
  - Used for binary descriptors (e.g., BRIEF, ORB).

3. **Histogram-based Distance Functions**

Used when images are represented as histograms (e.g., color, texture, gradient histograms).

- **Chi-Square Distance:**

$$d(x,y)= \frac{1}{2}\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{(x_i + y_i)}$$

  - Sensitive to differences in distributions.

- **Bhattacharyya Distance**

$$d(x,y)=-\ln \Sigma\sqrt{(x(i) * y(i))}$$

Measures overlap between distributions.

- **Earth Mover's Distance (EMD)**
  - Measures minimum "work" needed to transform one histogram into another.
  - Very useful in comparing color/texture distributions.

- **Kullback–Leibler (KL) Divergence**

$$D_{\mathrm{KL}}\,(p(x)\;||\;q(x)) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)}$$

  - Asymmetric; measures information loss when QQQ approximates PPP.

4. **Structural and Shape-based Distances**

For comparing **shapes, contours, or spatial arrangements**.

- **Hausdorff Distance**
  - Measures maximum distance of a point in one set to the nearest point in another set.
  - Used in shape matching and object recognition.
- **Chamfer Distance**
  - Average distance from each point in one set to the closest point in another set.
  - Often used for edge/contour matching.

5. Learning-based / Deep Metric Learning

In modern **deep learning-based vision systems**, distances are often **learned**:

- **Triplet Loss Distance** (FaceNet, etc.)
  - Encourages embeddings of the same identity to be closer than embeddings of different identities.
- **Contrastive Loss**
  - Penalizes similar pairs being far apart and dissimilar pairs being too close.
- **Siamese Networks**
  - Learn custom feature embeddings and distances for tasks like face verification.

Skeltons and Thinning: Skeletons are one-pixel-wide representations of an object's essential structure and connectivity, obtained through the process of thinning, also known as skeletonization. Thinning is a morphological operation that iteratively removes foreground pixels from the edges of a shape in a binary image, preserving its overall form and connectivity. This simplified representation has applications in areas like optical character recognition (OCR) and automated inspection, providing a structural description of complex shapes.

How Skeletonization and Thinning Work

1. **Reduction of Thickness**:

The primary goal is to reduce the width of a shape to a minimal, single-pixel representation.

2. **Preservation of Structure**:

While removing pixels, the algorithms ensure that the skeleton maintains the topological and connectivity properties of the original shape.

3. **Iterative Process**:

Thinning algorithms work by repeatedly removing specific foreground pixels (typically from the "edges") layer by layer.

4. **Maintaining Connectivity**:

A key challenge is to ensure that the remaining pixels form a single, connected component that represents the original shape's structure.

5. **Medial Axis Transformation**:
A related concept is the medial axis, which can be thought of as the set of points equidistant to the closest boundary points of the shape. The skeleton can be seen as the result of a medial axis transformation.

Applications

- **Shape Description**:

  Skeletons provide a compact and informative way to describe the shape of objects, which is useful for pattern recognition.

- **Optical Character Recognition (OCR)**:

  Thinning is crucial for OCR, as it simplifies character shapes into forms that are easier for pattern recognition techniques to analyze.

- **Automated Inspection**:
  In tasks like inspecting printed circuit boards, skeletons can help verify the structural integrity of components.
  Challenges

- **Computational Intensity**:

  Thinning can be computationally intensive, as each pixel may need to be examined multiple times to obtain the final skeleton.

- **Noise Sensitivity**:

  Small noise points on the boundary of an image can sometimes lead to large spurs in the skeleton, which may require post-processing steps like pruning.

- **Discretization**:
  Because algorithms run on a discrete grid, the resulting skeleton may not be perfectly continuous, even if the original shape was continuous.

Deformable shape analysis: It is an advanced concept in **computer vision** where we study and recognize shapes that can **deform, bend, stretch, or articulate** while still representing the *same object*.

This is important because in the real world, objects **rarely appear in rigid, fixed shapes** — think of human bodies, animals, clothes, organs in medical scans, or handwritten digits.

Types of Deformations

1. **Rigid Deformations**
   - Translation, rotation, scaling (no change in intrinsic structure).
2. **Non-rigid (Elastic) Deformations**
   - Bending, stretching, twisting, articulation (common in biological/medical shapes).
3. **Topological Changes**
   - Holes appearing/disappearing, or parts splitting/merging.

Methods for Deformable Shape Analysis

Different approaches are used depending on whether we work with **boundaries, regions, or features**:

(A) Boundary-based Approaches

- Focus on the outline/contour of the object.
- Methods:
  - **Fourier Descriptors** → Represent contour as Fourier coefficients, invariant to rotation/scale.
  - **Curvature Scale Space (CSS)** → Tracks curvature at multiple scales to identify shape features.
  - **Shape Contexts** → Describes distribution of points around contour points; useful for matching.

(B) Region-based Approaches

- Use the entire interior of the shape.
- Methods:
  - **Moments (Hu Moments, Zernike Moments)** → Capture global shape characteristics, invariant to deformation.
  - **Skeleton-based Methods** → Medial axis / skeleton that bends with the shape.

(C**) Graph-based Approaches**

- Represent shape as a **graph** of parts (nodes) and connections (edges).
- Methods:

    - **Shock Graphs** → Skeleton representation capturing shape structure.
    - **Attributed Relational Graphs** → Nodes represent shape parts, edges capture relations.

(D) **Deformable Models**

- Explicitly model deformation mathematically.
- Examples:
    - **Active Contours (Snakes)** → Curves that deform under forces to fit shape boundaries.
    - **Level Set Methods** → Implicitly evolve contours to capture deformable boundaries.
    - **Statistical Shape Models (SSM) / Active Shape Models (ASM)** → Learn typical deformations from training data.
    - **Active Appearance Models (AAM)** → Model both shape + texture variations.

(E) **Deep Learning Approaches**

- Learn shape embeddings robust to deformation.
- Examples:
    - **Convolutional Neural Networks (CNNs)** with spatial transformer layers.
    - **Graph Neural Networks (GNNs)** for deformable meshes.
    - **PointNet/PointNet++** for 3D deformable shape recognition.

Boundary tracking procedure: In **computer vision**, **boundary tracking** (also called **contour following** or **border following**) is the process of **extracting the boundary (outline) of an object** in a binary or segmented image.

This is essential in **shape analysis, object recognition, OCR, and medical imaging**, because the **boundary carries most of the shape information**.

Boundary tracking in computer vision involves tracing the outline or edge of an object, often after initial detection through methods like thresholding or edge detection. Algorithms such as active contours (snakes) and boundary tracing or contour tracing are used, where active contours iteratively deform to fit an object's boundary by minimizing an energy function, while tracing algorithms sequentially identify and connect pixels belonging to a region's outline.

Methods for Boundary Tracking:

1. Boundary Tracing (Contour Tracing):

> **Procedure:** This is a technique for identifying the pixels that form the boundary of a binary digital region. It's essentially a segmentation technique that segments the boundary pixels from the region.

Process: After an initial edge detection or thresholding step, the algorithm starts at a boundary pixel and follows a path of connected edge pixels to form a complete contour.

Example: Math Works provides examples of boundary tracing, explaining how a starting pixel and a first step direction can influence tracing either an internal or external boundary.

2. Active Contours (Snakes):

Procedure: An initial contour, which can be hand-drawn or predicted, is iteratively deformed to match an object's boundary by minimizing an energy function.

Energy Function: This function combines internal forces (like elasticity and smoothness) with external forces derived from image gradients (e.g., edges or regions of high intensity).

Process: The contour "latchs" onto the object's boundary as forces guide its movement, ensuring the final contour is smooth and knot-free.

Active contours: Active contours, or "snakes," are deformable curves in computer vision that "seek" the boundaries of objects in images by minimizing an energy function. They start with an initial curve, which is then attracted by image features like edges and pulled by internal forces (representing the curve's elasticity and smoothness) until it locks onto the object's true outline. This process is used for image segmentation, motion tracking, and interactive tools like the magnetic lasso in image editors.

Mainly Active contour is a segmentation method that uses energy forces and constraints to separate the pixels of interest from a picture for further processing and analysis. An active contour is defined as an active model for the segmentation process. Contours are the boundaries that define the region of interest in an image. A contour is a collection of points that have been interpolated. Depending on how the curve in the image is described, the interpolation procedure might be linear, splines, or polynomial.

Why is Active Contours Needed?

Active contours are mainly used to identify uneven shapes in images. They are also used to define smooth shapes in images and construct closed contours for regions.

Active contours are used in various medical image segmentation applications. Active contour models are employed in various medical applications, particularly for separating desired regions from medical images. For example, a slice of a brain CT scan is examined for segmentation using active contour models.

How Does Active Contour Work?

Active contours are the technique of obtaining deformable models or structures in an image with constraints and forces for segmentation. Contour models define the object borders or other picture features to generate a parametric curve or contour.

The curvature of the models is determined using several contour techniques that employ external and internal forces. The energy function is always related to the image's curve. External energy is described as the sum of forces caused by the picture that is specifically used to control the location of the contour onto the image, and internal energy is used to govern deformable changes.

The contour segmentation constraints for a certain image need to be determined. The desired shape is obtained by defining the energy function. A collection of points that locate a contour describes contour deformation. This shape corresponds to the desired image contour, defined by minimizing the energy function.

Active Contour Segmentation Models

Let us now look at some active contour segmentation models.

1. Snake Model

The snake model is a technique that can solve a broad range of segmentation problems. The model's primary function is identifying and outlining the target object for segmentation. It requires prior knowledge of the target object's shape, especially for complicated things. Active snake models, often known as snakes, are generally configured by using a spline focused on minimizing energy, followed by various forces governing the image.

Advantage

The applications of the active snake model are expanding rapidly, particularly in the many imaging domains. In medical imaging, the snake model is used to segment one portion of an image with unique characteristics compared to other regions. Traditional snake model applications in medical imaging include optic disc and cup segmentation to identify glaucoma, cell image segmentation, vascular region segmentation, and several other regions segmentation for diagnosing and studying disorders or anomalies.

Disadvantage

The conventional active snake model approach has various inefficiencies, such as noise sensitivity and erroneous contour detection in high-complexity objects, addressed in advanced contour methods.

2. Gradient Vector Flow Model

The gradient vector flow model is a more developed, well-defined version of the snake or active contour models. The traditional snake model has two limitations: inadequate contour convergence for concave borders and when the snake curve flow is commenced at a great distance from the minimum. As an extension, the gradient vector flow model uses the gradient vector flow field as an energy constraint to determine the contour flow.

Advantage

The gradient vector flow model is an advanced version of the snake model used for various image processing applications, especially in medical image processing. Active contour models help segment regions with particular parameters in medical imaging. These models create a contour around the target object, separating it from the image.

Disadvantage

The main difficulty with utilizing GVF is that the smoothing term ‘μ’ causes the contour’s edges to round. Reducing the value of ‘μ’ minimizes rounding but increases the amount of smoothing.

3. Balloon Model

A snake model isn’t drawn to far-off edges. If no significant image forces apply to the snake model, its inner side will shrink. A snake larger than the minimum contour will eventually shrink into it, whereas a smaller snake will not discover the minimum contour and will instead continue to shrink. To address the constraints of the snake model, the balloon model was developed, in which an inflation factor is incorporated into the forces acting on the snake. The inflation force can overwhelm forces from weak edges, exacerbating the problem with first-guess localization.

Advantage

The balloon concept is used to segment various medical pictures. The application’s primary purpose is to propose a novel technique for segmenting 2-D images and reconstructing 3-D meshes that ensure a watertight mesh.

Disadvantage

The balloon model’s biggest problem is its slow processing, which makes it difficult to manage sharp edges and requires careful object placement. The balloon model is commonly used in analyzing picture contour extraction.

4. Geometric or Geodesic Active Contour Models

Geometric active contour (GAC) is a contour model that adjusts the smooth curve established in the Euclidean plan by moving the curve's points perpendicular. The points move proportionately to the curvature of the image's region. The curve's geometric flow and the image's recognition of items are used to characterize contours. Geometric flow encompasses internal and external geometric measures in the region of interest. A geometric replacement for snakes is utilized to detect items in an image. These contour models rely heavily on the level set functions that specify the image's unique regions for segmentation.

Advantage

Geometric active contours are mostly used in medical image computing, particularly image-based segmentation. In this case, any imaging modality's picture is examined for segmentation to research, process, and analyze the regions of interest. These regions include any aberration in the interior regions or organs of the human body, such as blood clots, traumas, lesions, cell abnormalities, metabolic interruptions, biomolecule disruptions, etc.

Disadvantage

Mostly, it has no such inefficiencies, but they are difficult to implement as they are complex.

SHAPES AND REGIONS

Shape is one of the most fundamental visual cues for recognizing and classifying objects in computer vision.

Unlike color or texture, shape is **geometry-based** and often invariant to lighting and color changes.

Applications:

Object detection and recognition

Medical imaging (organ/tumor shape)

Industrial inspection (defect detection)

Biometrics (fingerprint, face contours)

Shape Representation Models

Boundary-based Shape Models-Represent only the contour (outline) of the object. It focus on external structure rather than internal details.

Techniques:

1. Chain Codes – Represent boundary as a sequence of connected directions (e.g., Freeman chain code).

2. Fourier Descriptors – Boundary points are transformed into frequency domain; low frequencies capture global shape, high frequencies capture details.

3. Curvature Scale Space (CSS) – Analyzes curvature of the boundary at different scales (used in MPEG-7 shape descriptor).

Region-based Shape Models

Use all pixels inside the shape (not just boundary).

Better for complex or noisy boundaries.

Techniques:

Geometric Moments – Statistical measures (area, centroid, orientation).

Hu's Moment Invariants $\rightarrow$ rotation, scale, and translation invariant.

Zernike Moments – Orthogonal polynomials, robust to noise.

Shape Matrix – Represents pixel connectivity and adjacency.

Structural Shape Models

Represent shape as relationships between parts.

Example: Graph-based models, medial axis (skeletonization).

Techniques:

1. Shock Graphs – Skeleton + graph structure.
2. Attributed Relational Graphs– Nodes = parts, edges = relationships.

<u>Statistical Shape Models (SSM)</u>

Capture variation in shape across examples using statistical learning.

Popular model: Active Shape Model (ASM)

Based on Principal Component Analysis (PCA) of annotated landmarks.

Encodes shape variability (e.g., human faces, hands).

<u>Deformable Shape Models</u>

Represent shapes that can change (elastic, flexible).

Examples:

1. Active Contours (Snakes) – Evolve curves under constraints to fit object boundary.
2. Level Sets – Implicit representation of evolving contours.
3. Elastic Graph Matching– Graph structure deforms to match object.

<u>Deep Learning-based Shape Models</u>

Recent approaches use CNNs and Graph Neural Networks to learn shape features.

PointNet / PointNet++ → learn directly from 3D point clouds.

Shape Autoencoders → learn compact shape embeddings.

Transformers → capture global shape context.

Shape Recognition Methods

Once shapes are represented, we need recognition techniques:

<u>Template Matching</u>

Compare input shape with stored template(s).

Works well for rigid, simple shapes.

Limitation: Sensitive to scaling/rotation.

<u>Feature-based Recognition</u>

Extract invariant features (moments, Fourier descriptors, edges).

Use distance metrics (Euclidean, Mahalanobis) to classify.

Example: Using Hu moments for recognizing handwritten digits.

Structural/Graph Matching

Match parts and their spatial relations.

Example: Shape represented as a skeleton graph → compared using graph isomorphism.

Statistical/Model-based Recognition

Learn variability of shapes.

Example: PCA-based eigen-shapes, SSM for medical organs.

Applications

Medical Imaging – Tumor detection, organ segmentation.

Biometrics – Fingerprint, face contour, iris shape.

Industrial Vision – Shape-based defect detection.

Autonomous Driving – Road sign and object shape recognition.

AR/VR & Robotics – 3D shape recognition for manipulation.

Centroidal profiles are a method to represent the shape of an object by capturing the distances of its boundary points from its centroid. This technique is used in image processing and shape analysis to extract meaningful features for object recognition, tracking, and classification, offering an invariant representation that accounts for translation and rotation.

How they work:

1. **Find the centroid:** The centroid, or center of mass, of the object is determined.
2. **Calculate distances:** A sequence of distances is measured from the centroid to various points along the object's boundary.
3. **Create the profile:** This sequence of distances forms the centroidal profile, which is a one-dimensional representation of the object's shape.

Applications:

- **Shape recognition:**

  By modeling the centroidal profile with statistical methods like circular autoregressive models, features can be extracted to classify and identify unknown shapes.
- **Object tracking:**

  The profile helps track moving objects over time by maintaining consistent features across frames.
- **Image segmentation:**

  It aids in identifying and delineating objects from the background by representing their unique shapes.
- **Quality control:**
  Centroidal profile analysis can be used to grade objects, such as grading papayas by size using their digital images.

Key benefits:

- **Invariant Representation:**

  The method can be normalized to be invariant to translation, rotation, and changes in size, making it powerful for shape comparison.

- **Feature Extraction:**

  It provides a powerful and informative representation for extracting unique features from an object's boundary.

- **Simplified Analysis:**
  It reduces a complex 2D shape into a simpler 1D representation, which can make analysis and comparison more straightforward.

Handling OCCLUSIONS

In computer vision, occlusion refers to the situation where an object of interest is partially or fully hidden by another object in the scene or by itself. Since computer vision algorithms rely on visible features such as edges, contours, and textures, occlusions significantly complicate the tasks of detection, recognition, tracking, and segmentation. For example, in an autonomous driving scenario, a pedestrian crossing behind a parked car is an occlusion case that must be handled for safe navigation. Occlusion is one of the most persistent challenges in computer vision. It disrupts the availability of complete object information, leading to errors in recognition, tracking, and segmentation. Modern solutions combine part-based recognition, temporal tracking, depth sensors, and deep learning approaches to effectively handle occluded scenarios. As computer vision moves towards real-world deployment in autonomous systems, security, healthcare, and AR/VR, robust occlusion handling becomes increasingly critical.

Types of Occlusions

1. Self-occlusion

Occurs when an object hides its own parts from the viewpoint.

Example: A human arm hiding the torso when folded across the body.

2. Inter-object occlusion

Happens when one object blocks another.

Example: A vehicle in front of another on a road.

3. Environmental occlusion

Caused by scene clutter or static obstacles.

Example: A wall hiding part of a chair in an indoor environment.

Challenges Due to Occlusions

Loss of visual features: Essential parts of an object may not be visible (e.g., missing facial landmarks in face recognition).

Ambiguity in recognition: Multiple objects with similar visible parts can confuse classifiers.

Difficulty in object tracking: An object may disappear temporarily and reappear, breaking trajectory estimation.

Errors in segmentation and reconstruction: Occluded regions lead to incomplete shape recovery.

Approaches to Handle Occlusions

Approaches to handling occlusions include using probabilistic filters (like Kalman or Particle filters) to predict object states, maintaining robust appearance models of objects over time, employing data augmentation techniques such as Random Erase and CutMix, re-initialization using object detectors, and utilizing multiple cameras or sensors to provide different viewpoints and overcome limitations of a single perspective. Other methods involve generative models that can synthesize occluded parts, spatio-temporal analysis of motion and context, and advanced sensor fusion techniques.

Here are common approaches to handle occlusions:

1. Predictive Filtering and Tracking

- **Probabilistic Filters**:

  Techniques like Kalman filters and Particle filters are used to estimate the position and motion of an occluded object based on its past observations.

- **Appearance Models**:

  Maintaining and updating appearance models for objects over time helps in tracking them even when they are partially or fully occluded.

- **Motion Estimation**:
  Algorithms can predict the future position of an occluded object by estimating the motion of its visible parts.
  2. Data-Driven Methods

- **Data Augmentation**:

  Techniques like Random Erase, Cutout, Grid Mask, and CutMix create occluded scenarios during training to make models more robust to occlusions in real-world data.

- **Generative Models**:
  These models can learn the underlying generative process of occlusion, enabling them to reconstruct or synthesize missing parts of an object and better differentiate it from the background.
  3. Object-Centric Techniques

- **Segmentation and Part Tracking**:

  An object can be segmented into multiple parts, and tracking can be performed on these parts to handle localized occlusions.

- **Spatio-Temporal Context**:
  Analyzing the object's motion history (temporal context) and its surroundings (spatial context) helps in differentiating it from other objects and handling occlusions.
  4. Sensor and Multi-View Approaches

- **Multi-Camera Systems**:

  Using multiple cameras provides different viewpoints, which can overcome limitations of a single view and reduce the likelihood of occlusion.

- **Sensor Fusion**:
  Combining data from different sensors, such as a camera and LiDAR, can provide robust object detection and tracking even when objects are occluded.
  5. Re-initialization and Detection

- **Re-initialization**: When occlusion levels become too high, a re-initialization process using an object detection scheme can be triggered to re-detect the lost object and resume tracking.
  <u>**CHAIN CODE:-**</u>

Chain code is a lossless compression technique used for representing an object in images. The co-ordinates of any continuous boundary of an object can be represented as a string of numbers where each number represents a particular direction in which the next point on the connected line is present. One point is taken as the reference/starting point and on plotting the points generated from the chain, the original figure can be re-drawn.In a rectangular grid, a point can have at most 8 surrounding points as shown below. The next point on the line has to be one of these 8 surrounding points. Each direction is assigned a code. Using this code we can find out which of the surrounding point should be plotted next. The chain codes could be generated by using conditional statements for each direction but it becomes very tedious to describe for systems having large number of directions(3-D grids can have up to 26 directions). Instead we use a hash function. The difference in X(dx) and Y(dy) co-ordinates of two successive points are calculated and hashed to generate the key for the chain code between the two points.

Chain code list: [5,6,7,4,−1,0,3,2,1]

Hash function: C(dx,dy)=3dy+dx+4

Hash table:-

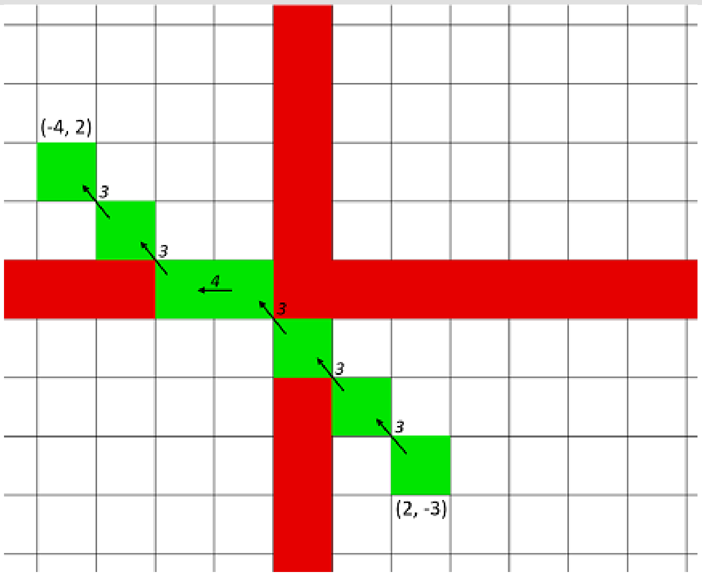
dx	***dy***	***C(dx,dy)***	***chainCode[C]***
1	0	5	0
1	1	8	1
0	1	7	2
-1	1	6	3
-1	0	3	4
-1	-1	0	5
0	-1	1	6
1	-1	2	7

The function does not generate the value 4 so a dummy value is stored there.

Examples:

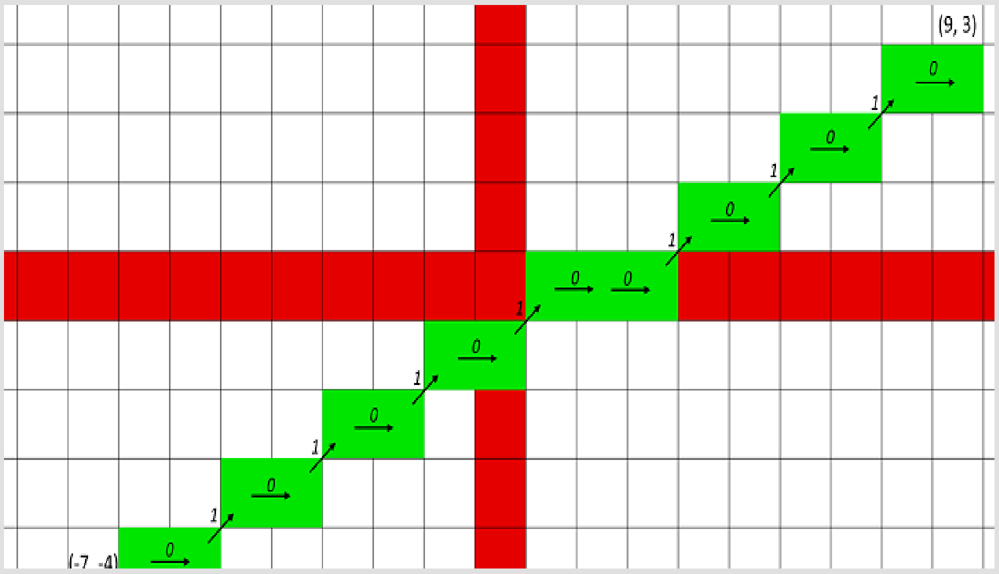
Input : (2, -3), (-4, 2)

Output : Chain code for the straight line from (2, -3) to (-4, 2) is 333433



Input : (-7, -4), (9, 3)

Output : Chain code for the straight line from (-7, -4) to (9, 3) is 0101010100101010



<u>Fourier Descriptors</u>

BASIS FUNCTIONS:

The Fourier Transform (in this case, the 2D Fourier Transform) is the series expansion of an image function (over the 2D space domain) in terms of "cosine" image (orthonormal) basis functions. The definitons of the transform (to expansion coefficients) and the inverse transform are given below:

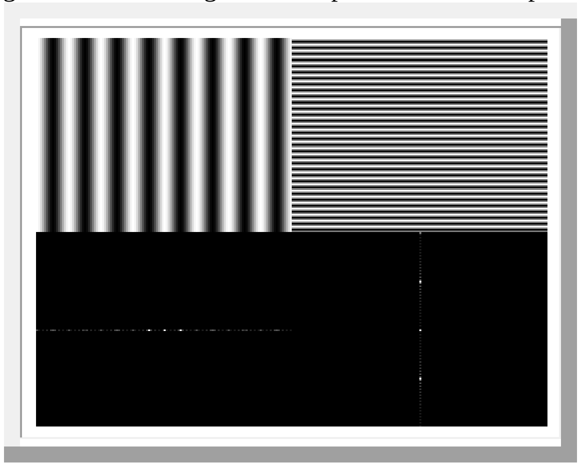
F(u,v) = SUM{ f(x,y)*exp(-j*2*pi*(u*x+v*y)/N) }

and

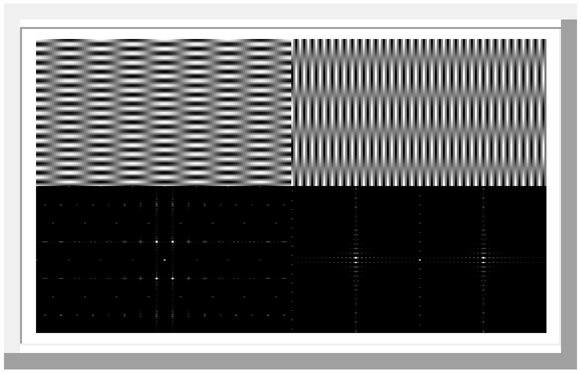
f(x,y) = SUM{ F(u,v)*exp(+j*2*pi*(u*x+v*y)/N) }

where u = 0,1,2,...,N-1 and v = 0,1,2,...,N-1
x = 0,1,2,...,N-1 and y = 0,1,2,...,N-1
j = SQRT(-1)
and SUM means double summation over proper
x,y or u,v ranges

First we will investigate the "basis" functions for the Fourier Transform (FT). The FT tries to represent all images as a summation of cosine-like images. Therefore images that are pure cosines have particularly simple FTs.



This shows 2 images with their Fourier Transforms directly underneath. The images are a pure horizontal cosine of 8 cycles and a pure vertical cosine of 32 cycles. Notice that the FT for each just has a single component, represented by 2 bright spots symmetrically placed about the center of the FT image. The center of the image is the origin of the frequency coordinate system. The u-axis runs left to right through the center and represents the horizontal component of frequency. The v-axis runs bottom to top through the center and represents the vertical component of frequency. In both cases there is a dot at the center that represents the (0,0) frequency term or average value of the image. Images usually have a large average value (like 128) and lots of low frequency information so FT images usually have a bright blob of components near the center. Notice that high frequencies in the vertical direction will cause bright dots away from the center in the vertical direction. And that high frequencies in the horizontal direction will cause bright dots away from the center in the horizontal direction.



Here are 2 images of more general Fourier components. They are images of 2D cosines with both horizontal and vertical components. The one on the left has 4 cycles horizontally and 16 cycles vertically. The one on the right has 32 cycles horizontally and 2 cycles vertically. (Note: You see a gray band when the function goes through gray = 128 which happens twice/cycle.) You may begin to notice there is a lot of symmetry. For all REAL (as opposed to IMAGINARY or COMPLEX) images, the FT is symmetrical about the origin so the 1st and 3rd quadrants are the same and the 2nd and 4th quadrants are the same. If the image is symmetrical about the x-axis (as the cosine images are) 4-fold symmetry results.

MAGNITUDE VS. PHASE:

Recall that the definition of the Fourier Transform is:

F(u,v) = SUM{ f(x,y)*exp(-j*2*pi*(u*x+v*y)/N) }

and

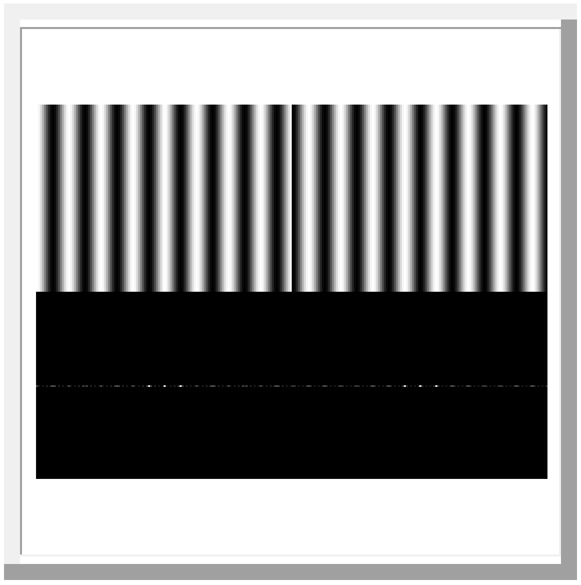
f(x,y) = SUM{ F(u,v)*exp(+j*2*pi*(u*x+v*y)/N) }

where u = 0,1,2,...,N-1 and v = 0,1,2,...,N-1
x = 0,1,2,...,N-1 and y = 0,1,2,...,N-1
and SUM means double summation over proper
x,y or u,v ranges

Note that f(x,y) is the image and is REAL, but F(u,v) (abbreviate as F) is the FT and is, in general, COMPLEX.
Generally, F is represented by its MAGNITUDE and PHASE rather that its REAL and IMAGINARY parts, where:

MAGNITUDE(F) = SQRT(REAL(F)^2+IMAGINARY(F)^2)
PHASE(F) = ATAN(IMAGINARY(F)/REAL(F))

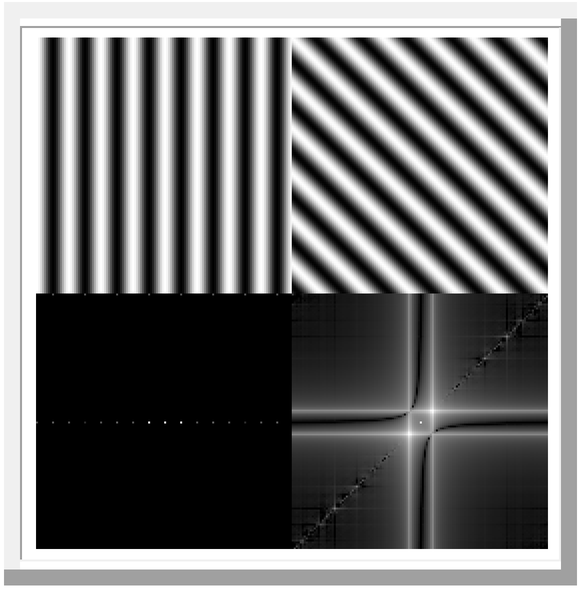
Briefly, the MAGNITUDE tells "how much" of a certain frequency component is present and the PHASE tells "where" the frequency component is in the image. To illustrate this consider the following.



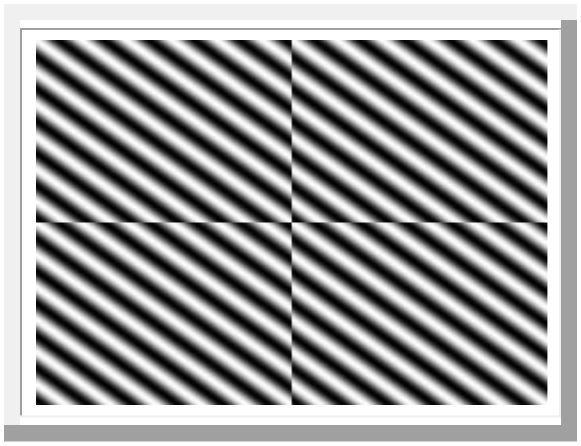
Note that the FT images we look at are just the MAGNITUDE images. The images displayed are horizontal cosines of 8 cycles, differing only by the fact that one is shifted laterally from the other by 1/2 cycle (or by PI in phase). Note that both have the same FT MAGNITUDE image. The PHASE images would be different, of course. We generally do not display PHASE images because most people who see them shortly thereafter succomb to hallucinogenics or end up in a Tibetan monastery. Nevertheless, it is wise to remember that when one looks at a common FT image and thinks about "high" frequency power and "low" frequency power, this is only the MAGNITUDE part of the FT. By the way, you may have heard of the FFT and wondered if was different from the FT. FFT stands for "Fast" Fourier Transform and is simply a fast algorithm for computing the Fourier Transform.

ROTATION AND EDGE EFFECTS:

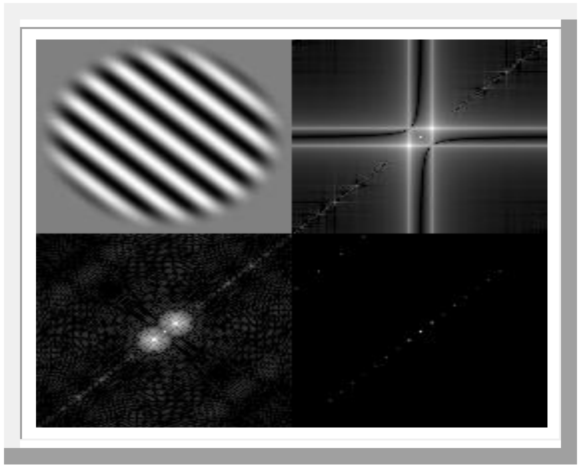
In general, rotation of the image results in equivalent rotation of its FT. To see that this is true, we will take the FT of a simple cosine and also the FT of a rotated version of the same function. The results can be seen by:



At first, the results seem rather surprising. The horizontal cosine has its normal, very simple FT. But the rotated cosine seems to have an FT that is much more complicated, with strong diagonal components, and also strong "plus sign" shaped horizontal and vertical components. The question is, where did these horizontal and vertical components come from? The answer is that the FT always treats an image as if it were part of a periodically replicated array of identical images extending horizontally and vertically to infinity. And there are strong edge effects between the neighbors of such a periodic array as can be seen by:



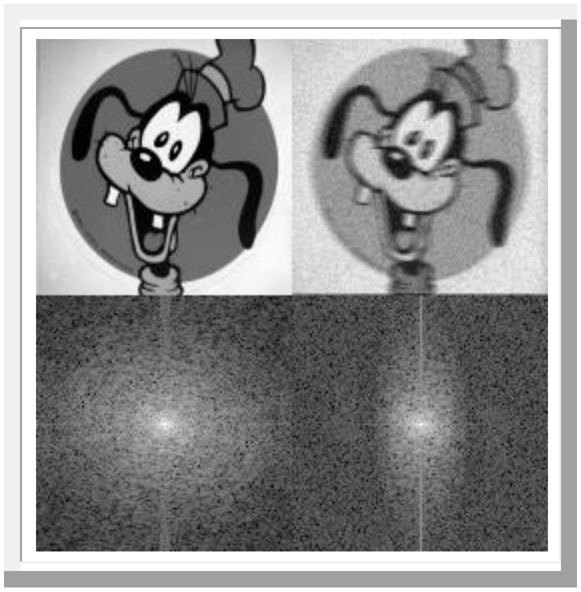
Thus, what we see as the FT in the "slant" image (lower right of the image before last) is actually the combination of the actual FT of the cosine function and that caused by the edge effects of looking at a finite part of the image. These edge effects can be significantly reduced by "windowing" the image with a function that slowly tapers off to a medium gray at the edge. The result can be seen by:



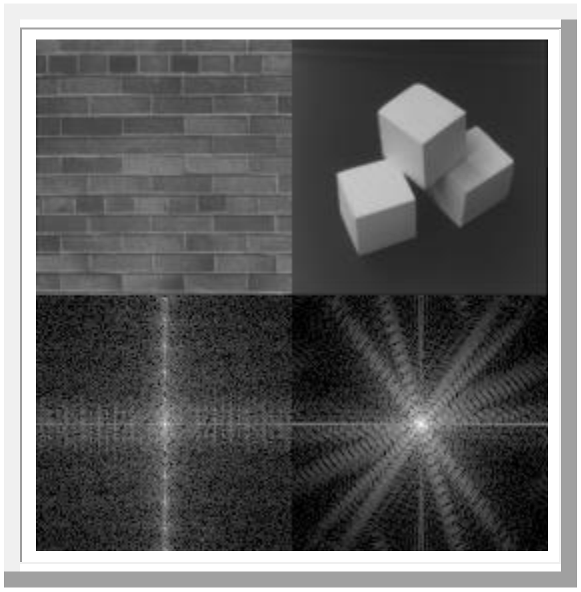
The windowed image is shown in the upper left. Its FT is shown in the lower left. The non-windowed FT is shown in the upper right and the actual, true FT of a cosine is shown in the lower right. These images are all scaled differently and the comparison is only qualitative, but it can be seen that the windowed image FT is much closer to the true FT and eliminates many of the edge effects.

SOME IMAGE TRANSFORMS:

Now, with the above introduction, the best way to become familiar with Fourier Transforms is to see lots of images and lots of their FTs. First, an interesting pair of images, one sharp and clear, and the other blurred and noisy.

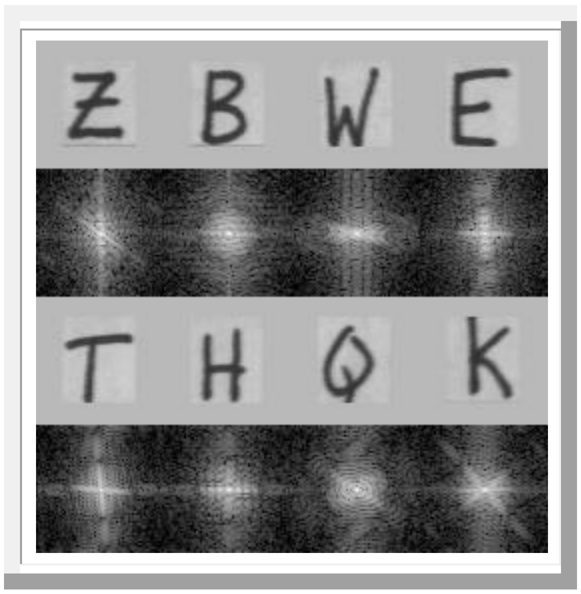


There are 2 images, goofy and the degraded goofy, with FTs below each. Notice that both suffer from edge effects as evidenced by the strong vertical line through the center. The major effect to notice is that in the transform of the degraded goofy the high frequencies in the horizontal direction have been significantly attenuated. This is due to the fact that the degraded image was formed by smoothing only in the horizontal direction. Also, if you look carefully you can see that the degraded goofy has a slightly larger background noise level at high frequencies. This is difficult to see and perhaps not even meaningful because the images are scaled differently, but if really there, it is due to the random noise added to the degraded goofy. Notice also that it is difficult to make much sense out of the low frequency information. This is typical of real life images. The next images show the effects of edges in images:



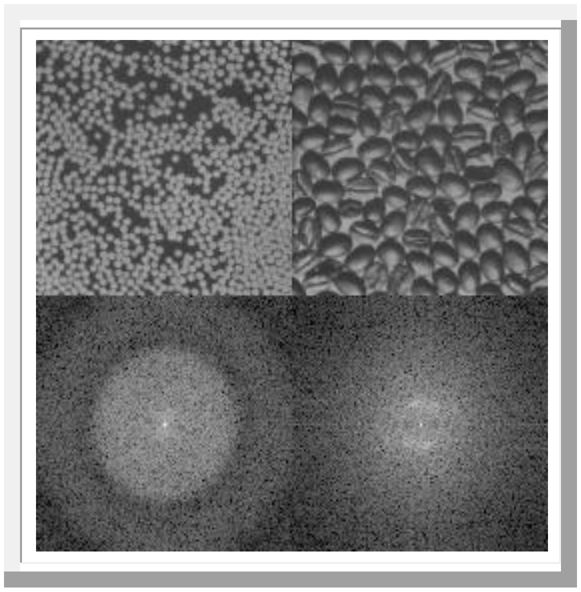
Notice the strong periodic component, especially in the vertical direction for the bricks image. Horizontal components appear closer together in the FT. In the blocks image, notice a bright line going to high frequencies perpendicular to the strong edges in the image. Anytime an image has a strong-contrast, sharp edge the gray values must change very rapidly. It takes lots of high frequency power to follow such an edge so there is usually such a line in its magnitude spectrum.

Now lets look at a bunch of different shapes and their FTs.

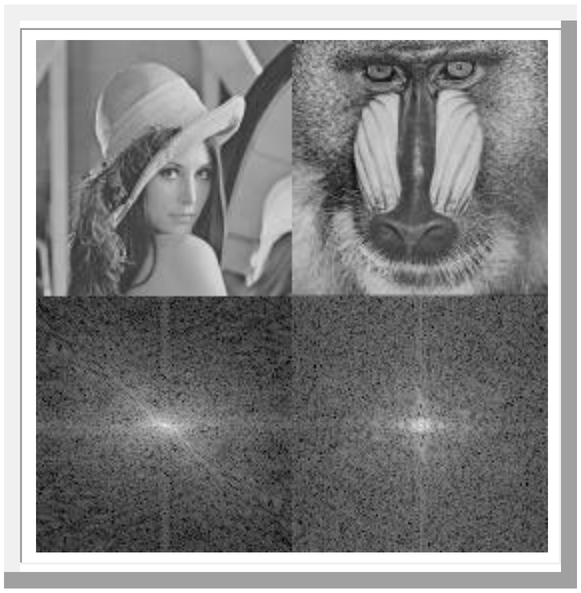


Notice that the letters have quite different FTs, especially at the lower frequencies. The FTs also tend to have bright lines that are perpendicular to lines in the original letter. If the letter has circular segments, then so does the FT.

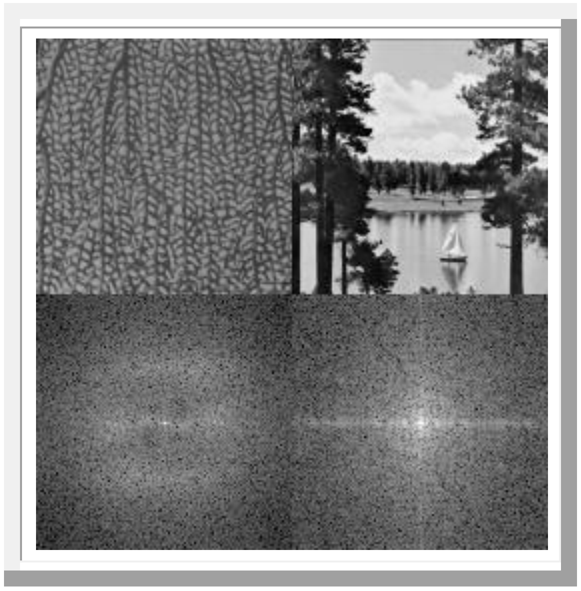
Now lets look at some collections of similar objects:



Notice the concentric ring structure in the FT of the white pellets image. It is due to each individual pellet. That is, if we took the FT of just one pellet, we would still get this pattern. Remember, we are looking only at the magnitude spectrum. The fact that there are many pellets and information about exactly where each one is is contained mostly in the phase. The coffee beans have less symmetry and are more variably colored so they do not show the same ring structure. You may be able to detect a faint "halo" in the coffee FT. What do you think this is from?

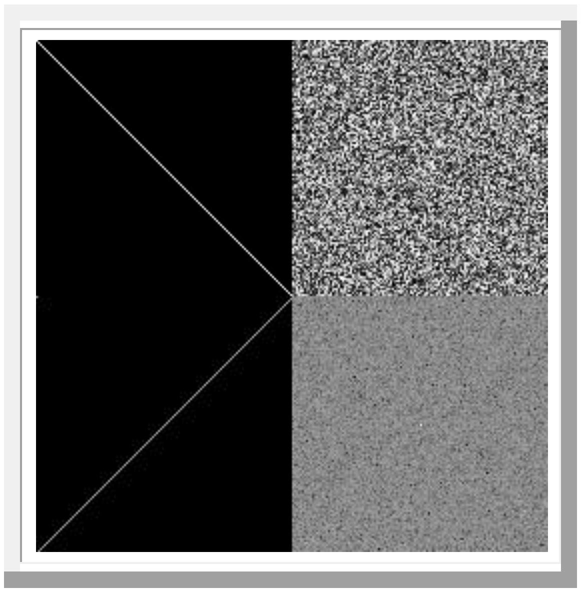


Here are our first truly general images. Notice there is very little structure. You can see a top left to bottom right slanting line in the girl image FT. It is probably due to the edge between her hat and her hair. There are also some small edge effects in both images. The mandril image appears to have more high frequency power, probably due to the hair.



The seafan image has a lot of little holes that are about the same size and somewhat randomly oriented. The size of the holes is about 2 pixels wide so that corresponds to frequency components about 1/2 way out to the maximum. The strong horizontal components in the lake image is probably due to the tree trunk edges.

Now, here is your first quiz. Consider an image that is all black except for a single pixel wide stripe from the top left to the bottom right. What is its FT? Also, consider an image that is totally random. That is, every pixel is some random value, independent of all other pixels. What is its FT?



Do you believe it? If not, you can check it yourself. By the way, notice the single bright dot in the middle of the noise FT image. Why is it there? Why does the noise FT look dark gray?

SOME FILTERS:

Now we start to illustrate the use of some filters on the girl image. The first is a lowpass filter. The upper left is the original image. The lower left is produced by:

```
fft2d 128 < girlimage > girlfft
mag2d 128 < girlfft > girlmag
```

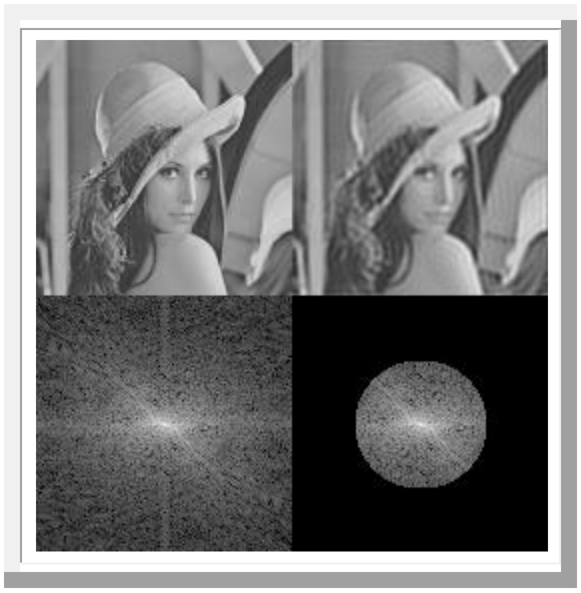
The lower right is then produced by:

```
fftfilt 128 low ideal 50 < girlfft > lpgirlfft
mag2d 128 < lpgirlfft > lpgirlmag
```

Finally, the upper right is produced by:

```
ifft2d 128 < lpgirlfft > lpgirl
```

To see the results:

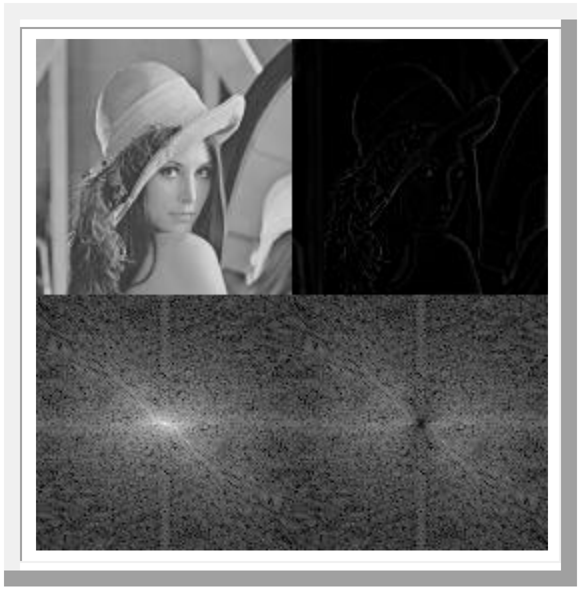


The left side of the image we have seen before. In the lower right, notice how sharply the high frequencies are cut off by the "ideal" lowpass filter. Notice also that not very much power is being thrown away beyond the circle that is cut off. In the upper right, the reconstructed image is obviously blurrier due to the loss of high frequencies. Overall contrast is still pretty good due to that fact that not too much power was thrown away. Notice also that there are obvious "ringing" artifacts in the reconstructed image. This is due to the very sharp cutoff of the "ideal" filter. A Butterworth or Exponential filter with reasonably low order would not cause these.

Now we will do a highpass filter. The following image is produced in the same way as the previous one except:

```
fftfilt 128 high butter 50 < girlfft > hpgirlfft
```

In other words, a butterworth filter of 1st order is used.



Notice in the lower right that this filter does not cut off sharply at the 50% point as the lowpass did. However, the center bright spot, which accounts for most of the power in the image, is clearly gone. The image in the upper right, which looks totally black, in fact is not totally black. If you use the colormap capability of "dym" to stretch the gray values from 0-20 out over the entire range, you can see that this highpass filter has preserved the image information where there are very rapid changes in gray level. Such a process is frequently what is desired in an edge detector. However, it is not an improvement in the image. There are 2 problems. First, it is too dark. This can be fixed by rescaling or re-contrast- stretching the image after filtering. This is commonly done and is easy. Second, and harder, is the fact that too much of the low frequency tonal information is gone.

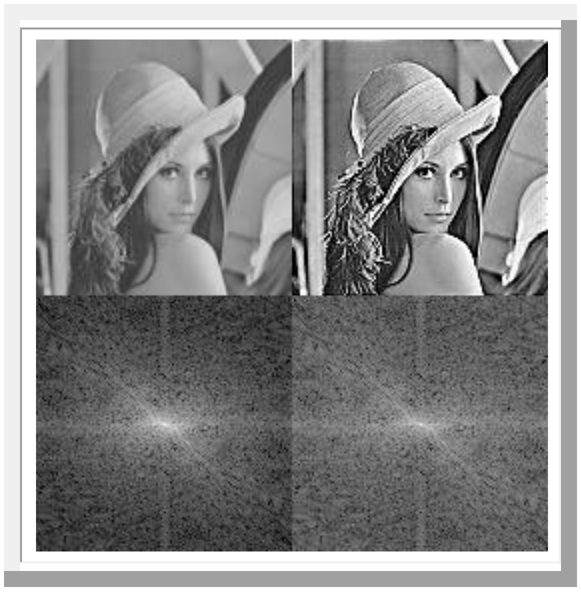
Image sharpening requires a "sharpening" filter or high frequency emphasis filter. This kind of filter preserves some of the low frequency information but relatively boosts the higher frequencies. To do such a thing, we will construct our own filter which will be piecewise-linear. The filter will be circularly symmetrical and will have coefficients as follows:

0	0.5
96	4.0
127	4.0

In other words, Fourier coefficients of frequency-distance 0 from the origin will be multiplied by 0.5. As you go away from the origin or zero frequency, out to frequency-distance 96, the multiplier will be interploated between 0.5 and 4.0. From then outward, the multiplier will be 4.0. So higher frequency coefficients are multiplied by values greater than 1.0 and lower frequency coefficients are multiplied by values less thatn 1.0. The overall net effect on the image power is that it is unchanged. The above values are in a file called "filter_coeffs". To apply the filter, the following steps are carried out:

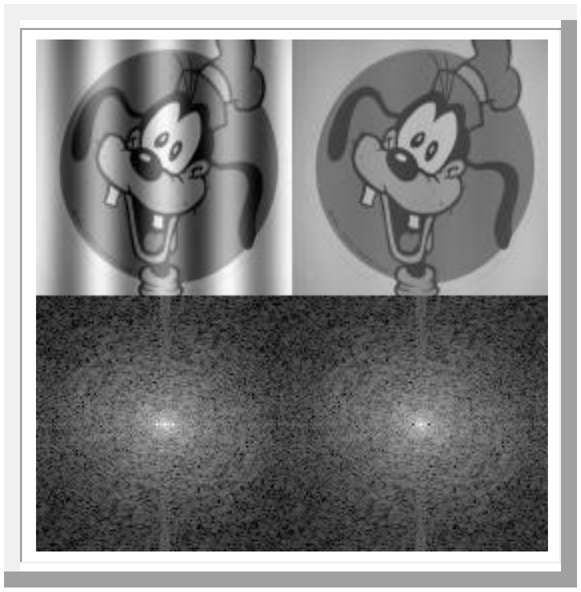
```
filttabler < filter_coeffs > filter_file
```

```
        fftfilt 128 file filterfile < girlfft > mfgirlfft
```
The rest of the image is constructed as before. To see the result:



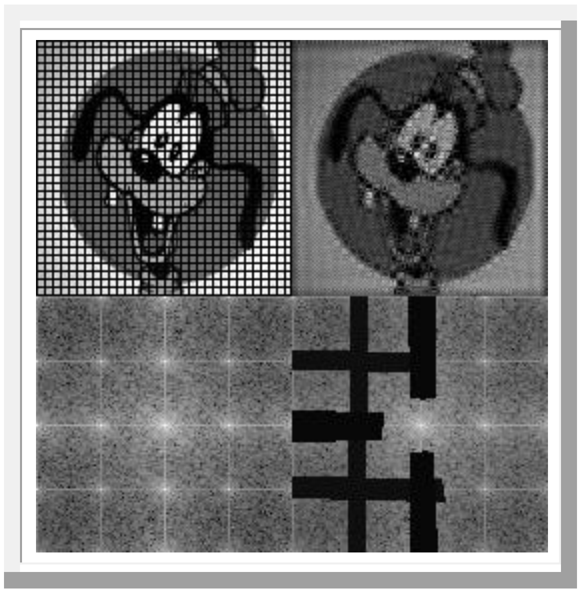
Notice the relative brightness at high frequencies in the lower right image. Which upper image is sharper? Which upper image looks better? Portraits are one of the few contradictions to the general principal that sharper is better.

Filtering can also be used to reduce noise. It is particularly effective when the noise is confined to just a few frequencies:



The image on the upper left is goofy with a superimposed cosine added to it, representing noise. In the lower left, notice the strong cosine "dots" just to the left and right of the origin. In the lower right, these "dots" have been removed (I actually did it with the "trace" capability in dym). The resulting magnitude file is then used with the "filter" command to filter the Fourier coefficients. The file of coefficients is then inverse FT'd to get the upper right image. The cosine "noise" is gone.

Life is not always this easy as is shown in the next example:



In this case, a grid has been placed over goofy. The lower left shows the resulting FT. Notice that the grid is quite sharp so it has lots of high frequencies so its impact on the frequency domain is very spread out. Dym was again used to "paint" out the grid frequencies as much as possible. The right half of the lower right image is not painted because it is the symmetric reflection of the left half and is not used by the filter.

Region Descriptors

Region descriptors in image processing are quantitative measures used to describe the characteristics of segmented regions within an image, focusing on their internal properties like shape, size, and texture. They include simple descriptors (e.g., area, perimeter, compactness), topological descriptors (e.g., number of holes or connected components), and texture descriptors that analyze smoothness, coarseness, and regularity within the region. These descriptors are essential for feature extraction, object recognition, and comparing different image regions by providing a numerical representation of their properties.

Types of Region Descriptors

- Simple Descriptors**:**

  These are basic measurements related to the size and shape of a region.

- **Area:** The number of pixels in the region.
- **Perimeter:** The length of the region's boundary.
- **Compactness:** A dimensionless measure defined as the squared perimeter divided by the area (P^2/A), which is minimal for disc-shaped regions and insensitive to uniform scaling and orientation.
- **Circularity Ratio:** The ratio of a region's area to the area of a circle with the same perimeter.
- **Mean/Median Gray Level:** The average or middle gray-level value of the pixels within the region.
- **Min/Max Gray Level:** The lowest and highest gray-level values in the region.
- Topological Descriptors**:**

  These descriptors measure the structural properties of a region and are often invariant to deformation.

- **Number of Connected Components:** Counts the distinct, separate parts of the region.

- **Number of Holes:** Quantifies the number of enclosed voids within the region.
- Texture Descriptors**:**
  These describe the surface characteristics, such as smoothness, coarseness, and regularity, within a region.

- **Statistical Approaches:** Analyze statistical properties of pixel intensities.
- **Structural Approaches:** Focus on repeating patterns or arrangements of pixels.
- **Spectral Approaches:** Use transforms like the Fourier transform to analyze frequency components.
  How They Are Used

1. 1. Segmentation:

   Image regions are first extracted using techniques like grid-based partitioning or pixel-centric methods.

2. 2. Calculation:

   Descriptors are computed for each region by applying mathematical formulas based on pixel values and their spatial relationships.

3. 3. Application:
   The computed descriptor values are then used for:

- Object Recognition: Identifying and classifying objects based on their characteristic features.
- Region Comparison: Comparing different regions to find similarities or differences.
- Image Analysis: Extracting meaningful information from images, such as analyzing land mass ratios in satellite imagery.

Moment

In computer vision and image processing, *image moments* are often used to characterize the **shape** of an object in an image. These moments capture basic information such as the **area** of the object, the **centroid** (i.e. the center *(x, y)*-coordinates of the object), the **orientation**, and other desirable properties. Hu proposed 7 moments that can be used to characterize the shape of an object in an image.

- Moment based calculations are often sensitive to the initial calculation of the centroid (since all further moment calculations are centered around the initial centroid).

- In real-world applications, this dependable and repeatable centroid is not easily obtained — and because of this centroid dependency, Hu Moments should not be used in situations where there is noise, occlusion, or a lack of (extremely) clean segmentation.

Standard image moments are implemented in OpenCV through the cv2.moments function. Hu Moments are available via the cv2.HuMoments function.

The idea is, the Hu Moments image descriptor used to quantify the shape of an object in an image.

Hu Moments are an image descriptor utilized to characterize the shape of an object in an image. The shape to be described can either be a segmented binary image or the boundary of the object (i.e. the “outline” or “contour” of the shape).

- In general, the segmented binary shape is preferred to the boundary of the shape since it is less susceptible to noise