Cairo University

Faculty of Engineering

Systems & Biomedical Engineering

Computer Vision

(SBE 3230)

Under Supervision:

Professor, Ahmed M, Badawi

Eng. Laila Abbas

Eng. Omar Hesham

Team Members:

1. Fady Mohsen Magdy

2. Lama Zakaria

3. Lamees Mohee

4. Rana Ibrahim

5. Camelia Marwan

Table of Contents

1. Active Contour

1.1. Introduction:

Active Contour, commonly known as snakes, is a computer vision algorithm for outlining an object within an image. It works by evolving a curve with respect to the image's content, optimizing an energy function that reflects the contour's conformity to the object's boundaries. This report outlines the detailed steps involved in implementing Active Contour, focusing on the core algorithmic components.

1.2. Step (1): Image Processing:

- **Grayscale Conversion:** Convert the input image to grayscale to simplify the analysis. This is because the active contour algorithm primarily relies on intensity variations to detect edges.

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

- **Smoothing:** Apply a Gaussian blur to the grayscale image to reduce noise and smooth out the edges, which helps in more accurate edge detection.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}}$$

```
smooth = cv2.GaussianBlur(gray, (5, 5), 0)
```

1.3. Step (2): Initial Contour:

- **Description:** The initialization of a contour is a fundamental step in the active contour or "snakes" algorithm. An appropriately chosen initial contour significantly impacts the effectiveness and efficiency of the algorithm in converging to the object's boundary. The initial contour should be placed near the target boundary to ensure that the contour can correctly evolve towards the object's edges under the influence of internal and external forces.
- **Approaches for Initializing a Contour**
    - Manual Initialization: The user manually draws the initial contour around the object of interest. This method allows precise control but is time-consuming and not practical for automated applications.
    - Automatic Initialization: Techniques such as thresholding, edge detection, or region-based methods are used to automatically generate an initial approximation of the object boundary.

1.4. Defining Energy Functions in Active Contour Models

- **Description:** The success of active contour models (snakes) in delineating object boundaries largely depends on the definition of energy functions. These functions guide the evolution of the initial contour towards the object's edges by balancing internal forces (that maintain the contour's coherence) and external forces (that attract the contour towards image features such as edges). Optionally, constraint energy terms can be introduced to incorporate specific prior knowledge about the object's shape or other attributes. This section provides an overview of these energy components.
- **Internal Energy:** The internal energy of the snake is designed to maintain the contour's smoothness and continuity, ensuring that it behaves like a physical elastic structure. It is composed of two main components:
    - Elasticity (Continuity) Energy: Prevents the snake from stretching too much, keeping the points on the contour close to each other.
    - Bending (Curvature) Energy: Discourages bending of the contour, maintaining smoothness.
    - **Let's say the internal energy will take the symbol (E_int):**
        - **Comprises two terms, elasticity ($E_{elastic}$) and rigidity (E_{bend}).**
        - $E_{int} = \alpha E_{elastic} + \beta E_{bend}$
        - $E_{elastic} = \left|v(s) - v(s+1)^2\right|$, $controllingthedistancebetweensuccessivepoints(stretching).$
        - $E_{bend} = \left|v(s-1) - 2v(s) + v(s+1)^2\right|$, **controlling the curvature.**
- **External Energy:** The external energy guides the contour towards the object's boundaries and is typically derived from the image data. It encourages the contour to align with image features such as edges, corners, or textures. Common formulations of external energy include:
    - Image Gradient: Attracts the contour to areas with high image gradients (edges).
    - Edge Function: Defined using edge detectors, such as the Sobel operator or the Canny edge detector, to create a potential field that pulls the contour towards edges.
    - *The external energy based on image gradients can be expressed as*:
    - $[E_{\text{ext}} = -|\nabla I(v)|^2]$
    - $-(E_{\text{ext}}) is the external energy.$
    - $-(\nabla I(v)) is the gradient of the image intensity at the contour position (v).$
    - $-(\,|\cdot|\,)$ ***denotes the magnitude of a vector.***
- **Total Energy:** The total energy of the snake is the sum of these components, and the goal of the snake algorithm is to minimize this energy:
    - **[E_{total} = ∫ _{0}^{1} (E_{int} + E_{ext} + E_{con}) ds]**

1.5. Discretization of the Snake into a Set of Discrete Points:

- **Description:** The concept of a "snake" or active contour model in computer vision and image processing is fundamentally continuous. However, for practical computation, it's necessary to represent this model discretely. Discretization involves breaking down the continuous snake into a series of discrete points or nodes, which can then be manipulated using numerical methods. This section explores the process and significance of discretizing a snake for implementation in active contour models.
- **Purpose of Discretization:**
    - Feasibility: Discretization makes the mathematical manipulation of the snake feasible on digital computers, which can only handle discrete values.
    - Simplicity: Working with a finite set of points simplifies the calculation of energies and forces that guide the snake's movement.
    - Efficiency: It allows for the efficient implementation of optimization algorithms that minimize the snake's energy by adjusting the positions of discrete points.
- **Process of Discretization:**
    - Selecting the Number of Points: The first step in discretization is to decide the number of points or nodes the snake will consist of. A higher number of points can represent more complex shapes but at the cost of increased computational complexity. Conversely, too few points may not capture the necessary detail of the object's boundary.
    - Initial Placement: The points are initially placed along the preliminary contour defined in the initialization step. This placement can be uniform, with equal distances between adjacent points, or based on specific criteria relevant to the application.
    - Representation: The discretized snake is often represented as an ordered sequence of points in the 2D plane.
- Considerations
    - Adaptivity: In some implementations, the distance between points can adapt dynamically during the evolution process. Points can be added in areas of high curvature where more detail is needed or removed in flatter areas to reduce computational load.
    - Periodicity: For closed contours, the first and last points are often considered neighbors, making the discretized snake effectively a closed loop. This is important for ensuring smoothness and continuity around the entire contour.

1.6. Energy Evaluation and Optimization for Snake Points

- Once the snake is discretized into a set of discrete points, the next step involves iteratively adjusting these points to minimize the snake's overall energy. This process is essential for the snake to converge to the object's boundary in the image. Each point on the snake is individually moved to a position where the combination of internal, external, and optional constraint energies is minimized. This section delves into the method of evaluating energy at each point and the strategy for moving points to their optimal positions.
- Current Position Energy: For each discrete point on the snake, the total energy (internal, external, and constraint) is calculated based on its current position. This serves as a baseline for comparison.
- Neighboring Position Energy: The energy is also calculated for a set of neighboring positions around each point. These positions are typically defined within a small, predefined search area or radius around the point's current location. The size of this area can affect the algorithm's performance and accuracy.
- **Energy Components**
    - Internal Energy evaluates the smoothness and continuity of the snake, discouraging excessive stretching or bending.
    - External energy measures how well the point aligns with features in the image, such as edges, encouraging the snake to move towards the object boundaries.
    - Constraint energy (if used) imposes additional conditions or priors, such as shape or specific feature alignment.

1.7. Iterative Optimization and Termination Criteria in Active Contour Models

- Active contour models, or "snakes," rely on an iterative optimization process where each point on the discretized snake is adjusted to minimize the overall energy. This iterative process is crucial for the model to adapt and conform to the target object's boundary in an image. The process continues until specific termination criteria are met, ensuring that the model converges to a stable solution without unnecessary computation. This section explores the iterative optimization process and discusses common termination criteria.
- **Iterative Optimization Process**
    - Sequential Update: During each iteration, each point on the snake is evaluated and potentially moved to a new position where the total energy (internal, external, and constraint) is lower. This is done sequentially for all points on the snake.

- Global Energy Evaluation: At the end of each iteration, the total energy of the snake can be evaluated to monitor the progress of convergence. A decrease in total energy indicates that the snake is moving closer to an optimal configuration.
- Feedback Loop: Information from the previous iterations, such as changes in energy or point positions, can be used to inform adjustments in subsequent iterations. This feedback mechanism helps in fine-tuning the optimization process.

- **Termination Criteria:** The iterative optimization process needs clearly defined criteria for termination to prevent infinite loops and to ensure computational resources are used effectively. Common termination criteria include:
  - Fixed Number of Iterations: The simplest criterion is to terminate the process after a predetermined number of iterations. This method is straightforward but may not guarantee convergence to the best solution.
  - Energy Change Threshold: The process can be terminated when the change in total energy between iterations falls below a specified threshold. This indicates that the snake has reached a state of minimal energy change, suggesting convergence.
  - Position Change Threshold: Similarly, termination can occur when the maximum displacement of any point on the snake between successive iterations is below a certain threshold, indicating that the snake has stabilized.
  - Combination of Criteria: Often, a combination of the above criteria is used to ensure robustness. For example, the process might terminate when either a maximum number of iterations is reached or the changes in energy and positions fall below their respective thresholds.
- **Practical Considerations**
  - Convergence Speed: The choice of termination criteria affects the speed of convergence. Tight thresholds can lead to more precise fitting but at the cost of increased computation time.
  - Robustness to Noise: In images with noise or complex backgrounds, the snake may experience fluctuations in energy and position changes. Adaptive termination criteria or smoothing techniques may be required to achieve stable convergence.
  - Evaluation Frequency: For efficiency, the termination criteria might not need to be evaluated at every iteration. Instead, periodic evaluation can save computational resources while still effectively monitoring convergence.

1.8. Storing Iteration Frames for Video Production in Active Contour Models

In the context of active contour models, visualizing the iterative process can provide insightful information about the model's convergence behavior and effectiveness in delineating object boundaries. Storing the frames of each iteration's output allows for the creation of a video that captures the dynamic evolution of the snake from its initial position to its final, converged state. This section discusses the approach to storing these frames and producing a video from them.

2. Chain Code for Representing Object Boundaries

Chain code is a compact method for representing the boundaries of objects in digital images. It encodes the contours of objects using a sequence of directions or moves from a starting point along the object's edge. This approach is particularly useful in image processing and computer vision for tasks such as shape analysis, object recognition, and compression.

2.1. Principles of Chain Code

- Directional Encoding: The contour of an object is traced, and each move from one pixel to the next along the boundary is encoded as a direction. These directions are typically represented by numbers corresponding to the possible movements from a pixel.
- Start Point: The encoding starts at a specific point on the object's boundary, which serves as a reference for decoding the chain code later.
- Grid Orientation: The representation of movements depends on the grid used. In a 4-connected grid, movements are allowed horizontally and vertically, usually encoded as 0 (right), 1 (up), 2 (left), and 3 (down). An 8-connected grid allows diagonal movements as well, requiring 8 directions to be encoded.

2.2. Advantages of Chain Code

- **Compact Representation:** Chain code provides a more compact representation of object boundaries than storing the coordinates of all boundary pixels.
- **Facilitates Shape Analysis**: The sequence of directions can be used to analyze the shape of the object, identifying features such as corners or curves.
- **Rotation and Scale Invariance:** Modifications to the chain code can make it invariant to rotation and scale, useful for object recognition tasks.

2.3. Generating Chain Code

- **Boundary Detection:** Initially, the boundary of the object is detected, often using edge detection algorithms.
- **Selecting a Start Point:** Choose a starting point on the boundary for the encoding process.
- **Traversing the Boundary:** Move along the boundary in a predefined direction (clockwise or counterclockwise), encoding each step according to the movement direction relative to the grid.

- **Recording Directions:** Each move is recorded as a digit in the chain code sequence until the boundary is completely traversed, and the start point is reached again.

2.4. Applications of Chain Code

- **Object Recognition:** The chain code can be analyzed to recognize objects based on their shapes.
- **Contour Analysis:** It is used to study the properties of object boundaries, such as smoothness, curvature, and length.
- **Image Compression:** Chain code offers a way to compress the information required to represent a boundary, reducing the storage space.

2.5. Challenges and Considerations

- **Sensitivity to Noise:** The presence of noise in the image can lead to inaccuracies in the boundary representation.
- **Starting Point Dependency:** The chain code sequence depends on the choice of the starting point, which might affect the analysis or comparison of shapes.
- **Resolution Dependency:** The detail level of the representation can vary with the image resolution, affecting the encoded shape's fidelity.

2.6. Conclusion

Chain code is a valuable tool in digital image processing for encoding and analyzing the contours of objects. By efficiently representing boundary information as a sequence of directional moves, it facilitates various applications from object recognition to shape analysis. While chain code is powerful, careful consideration must be given to factors like noise, the selection of the start points, and resolution to ensure accurate and useful representations.

3. Hough Detection

The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure.

Before applying the Hough transform, we need to run the **grayscale** and **blur** methods, because if we have too much noise, things will get messy, and it will be difficult to calculate.

After that we apply **Canny Edge Detection Algorithm** before applying the Hough Transform to enhance the accuracy and efficiency of circle detection as it reduces the amount of data that needs to be processed by the Hough Transform. By focusing only on the edges, which represent significant changes in intensity

3.1. Hough Transform Line Detection:

In image processing, line detection is a fundamental task with various applications in computer vision, robotics, and medical imaging. The Hough transform is a widely used technique for detecting lines in digital images. It is a feature extraction technique that identifies shapes within an image, particularly lines. The basic idea is to represent lines in an image as points in parameter space, allowing for robust detection even in the presence of noise and occlusion.

3.1.1 Parameter Space Representation:

- Consider a Cartesian coordinate system where each point in the image space corresponds to a line in parameter space. A line in image space can be represented by the equation: $y = mx + b$. Where m is the slope of the line and b is the y-intercept. In parameter space, this equation becomes a sinusoidal curve: $b = x \,.\cos\theta + y \,.\sin\theta$

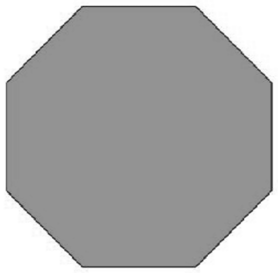
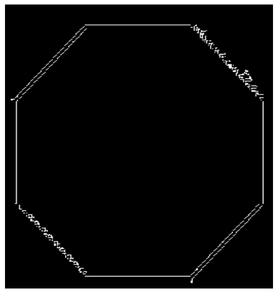
- Here theta is the angle of the normal to the line from origin, and b is the distance from the origin to the closest point on the line.

3.1.2 <u>Accumulator Array:</u>

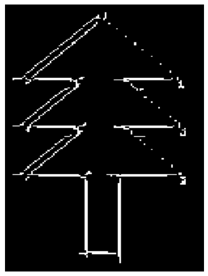
- The Hough transform constructs an accumulator array to accumulate evidence of potential lines. Each cell in the accumulator array represents a possible line in parameter space. The intersection points of curves in parameter space correspond to lines in image space.

3.1.3 <u>Results and Observations:</u>

- Upon visual inspection of the detected lines, it is evident that the Hough transform effectively captures linear features present in the images. Straight edges of objects and boundaries between regions are accurately identified.

And even more complicated structures in images that have more lines can be detected and highlighted as well.



3.2 Hough Circle Transform

The algorithm starts by initializing an accumulator array, which is a 3D array used to accumulate **votes** for potential circle centers. It then iterates over different radius, for each radius creating a **blueprint** of a circle by calculating the x and y coordinates of points on the circle using trigonometric functions.

$$x = r\cos\theta, \qquad y = r\sin\theta$$

This blueprint is then slid across the edge pixels in the image, and votes are cast for potential circle centers by incrementing the corresponding cells in the accumulator array.

After accumulating votes for circle centers, a **threshold** is applied to the accumulator array to remove low-vote centers, reducing false positives.

For each remaining center, a **region** is examined around it to find the most likely circle within that region. This region size helps refine the detected circles.

Finally, the detected circles are extracted from the accumulator array and drawn on a copy of the original image. The result is an image with the detected circles outlined in green.

Original Image

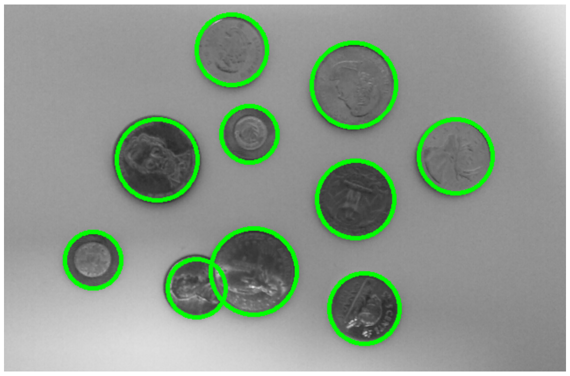


Image After Detecting Circles

3.3 <u>Ellipse Hough Transform Detection</u>

The Hough Transform is a popular technique in image processing and computer vision for detecting geometric shapes, initially developed for detecting lines but later extended to detect shapes like circles and ellipses. Ellipse detection using the Hough Transform involves identifying points that collectively form an ellipse within an image. This method is particularly useful in applications where ellipses are significant features, such as in medical imaging or industrial inspection. This section outlines the principles, challenges, and a general approach to ellipse detection using the Hough Transform.

3.3.1. <u>Principles of Ellipse Detection</u>

<u>Edge Detection:</u> The first step typically involves applying an edge detection algorithm (e.g., Canny edge detector) to the image to identify potential edge points of ellipses.

<u>Parameter Space:</u> Unlike lines or circles, ellipses have a more complex parameter space because they are defined by five parameters: the center coordinates (x, y), the semi-major axis (a), the semi-minor axis (b), and the rotation angle (θ) relative to the x-axis.

<u>Accumulator Array:</u> An accumulator array is used to record votes for the presence of an ellipse with particular parameters. Due to the higher dimensionality of the ellipse parameter space, the accumulator array can become very large, making the naive Hough Transform computationally expensive.

<u>Voting Process:</u> For each edge point and for a range of possible angles, the ellipse parameters that would place an ellipse's edge at that point are computed and voted for in the accumulator array. Points in the image that belong to the same ellipse will contribute votes to the same parameters in the accumulator.

3.3.2 <u>Challenges in Ellipse Detection</u>

<u>High Computational Cost:</u> The five-dimensional parameter space of ellipses leads to a high computational cost. Efficient implementation and parameter space reduction techniques are necessary.

<u>Parameter Space Reduction:</u> Techniques such as fixing one parameter (e.g., the aspect ratio) or using gradient direction to estimate the orientation can reduce the dimensionality and complexity.

<u>Noise and Occlusion:</u> Ellipses partially occluded or in noisy images require robust detection methods that can handle incomplete or misleading information.

3.3.3 <u>General Approach to Ellipse Hough Transform Detection</u>

<u>Preprocessing:</u> Apply edge detection to the image to find potential boundary points of ellipses.

<u>Gradient Calculation:</u> Compute the gradient direction at each edge point, which can be used to limit the range of possible orientations and reduce computational load.

Accumulator Space: Initialize an accumulator space with dimensions corresponding to the ellipse parameters. Often, a reduced parameter set is used.
Voting: For each edge point, calculate possible ellipse parameters and increment the corresponding cells in the accumulator space.
Peak Detection: Identify peaks in the accumulator space. Each peak corresponds to a set of ellipse parameters with a significant number of votes.
Result Validation and Refinement: Validate detected ellipses against original image data. Refine ellipse parameters if necessary, and remove false positives.

3.3.4 Conclusion

Ellipse detection using the Hough Transform is a powerful technique for identifying elliptical shapes within images, applicable in various domains. However, due to the complexity of the ellipse's parameter space, the method faces significant computational challenges. Efficient implementations and optimizations, such as reducing the parameter space and focusing on likely ellipse orientations, are crucial for practical applications. Advances in image processing and computer vision continue to enhance the effectiveness and efficiency of ellipse detection methods, making them more accessible for real-world applications.

4. References

4.1. Active Contour Models

1. Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: Active contour models. **International Journal of Computer Vision**, 1(4), 321-331. This seminal paper introduced the concept of active contour models or "snakes" and laid the foundation for subsequent developments in the field.

2. Blake, A., & Isard, M. (1998). **Active Contours**. Springer-Verlag. This book provides a comprehensive overview of active contour models, including mathematical formulations, algorithms, and applications.

4.2. Hough Transform and Ellipse Detection

3. Duda, R. O., & Hart, P. E. (1972). Use of the Hough Transformation to Detect Lines and Curves in Pictures. **Commun. ACM**, 15(1), 11–15. This classic paper extends the Hough Transform to the detection of shapes, setting the stage for later adaptations to ellipse detection.

4. Xie, Y., & Ji, Q. (2002). A new efficient ellipse detection method. In **Proceedings of the 16th International Conference on Pattern Recognition** (Vol. 2, pp. 957-960). IEEE. This paper presents an efficient method for ellipse detection, addressing some of the computational challenges associated with the Hough Transform.

4.3. Online Resources

5. OpenCV Documentation. (n.d.). Hough Circle Transform. Retrieved from https://docs.opencv.org/master/da/d53/tutorial_py_houghcircles.html. The official OpenCV documentation provides practical examples and explanations for implementing circle detection using the Hough Transform, which is closely related to ellipse detection.

6. Scikit-image Documentation. (n.d.). Hough Transform. Retrieved from https://scikit-image.org/docs/dev/auto_examples/edges/plot_circular_ellipse.html. Scikit-image offers tools and tutorials for edge detection and shape fitting, including methods based on the Hough Transform.