Chapter 3

Pre and Post Processing Approaches in Edge Detection

Ideas developed to solve a pure mathematical problem can lead to unexpected applications years later

3.1 Introduction

Edge detection is one of the several image analysis techniques used to detect and extract information from characters. It is a process of detecting sharp changes in image brightness to capture important events and changes in the image. The result of edge detection should be linear chains (thinned edges) of accurately located pixels. An ideal edge detector should allow the following:

1. Physical edge detection, avoiding spurious edges related to noise and texture of the image,
2. Edge detection without fragmentation, and
3. Precise localization of the point sequence that best represents the contour of the object.

If an edge detector with the above characteristics exists, it is easy to assign meaning to the objects present in the image. Since a perfect edge detector does not exist, strategies that allow results as close as possible to the ideal one can be adopted [104]. Over the years, edge detection methods such as Sobel, zero crossing, Canny, Ant Colony Optimization (ACO), etc. have been proposed. It is generally accepted that Canny edge detector is one of the best edge detection algorithms of the present era, and it has also become a standard in edge detection [163].

Some parts of this chapter appear in:
Chapter 3

Canny edge detector is used to produce thinned edges of the character. This method goes through four different stages: smoothing, differentiation, nonmaxima suppression, and hysteresis. The first two stages generate an intensity surface from where the gradient magnitude is determined. On that surface, the ridges will correspond to image edges. If the surface is transformed into an image called gradient image, the edges will be visualized as lines of several pixels in width. The thinning of the ridges is accomplished by nonmaxima suppression, and the hysteresis process eliminates the spurious edges. It is important that the edge detector should give the character edges without fragmentation and displacement of edge pixels. In the actual case, it is found that a lot of spurious branches are created during edge detection due to the presence of noise in high resolution images [104]. A preprocessing activity can tackle this problem.

Problem 1: One of the common operations in edge detection is the differentiation of the image, which allows the detection of both physical and spurious edges. To minimize the undesired effect, the image is smoothed before differentiation, thus creating the duality between detection and precision of edge localization. Higher the localization precision, lower is the Signal to Noise Ratio (SNR), and consequently, the detection becomes more and more sensitive to the spurious details of the image. This duality is present in Canny edge detector. Canny edge detector cannot simultaneously enhance the insensitivity of image signal to noise and the localization precision of detected edges.

Solution: Apply Canny edge detector to a fine Gaussian scale, but only along the edge regions focused by the process of anisotropic diffusion via Partial Differential Equation (PDE). The process is used due to its notable characteristic in selectively smoothing the image. It smoothes better the homogeneous regions while preserving the physical edges, i.e., those that are actually related to objects present in the image. This approach best utilizes the advantages of both detectors. PDE detector generates results with good SNR, edge regions without displacement and with minimum fragmentation, whereas Canny detector produces high precision thinned edge at a fine scale.

Problem 2: Even though the above method helped to get accurate edge points, the final result may contain some broken edges.
**Solution:** This problem is handled in a post processing stage. An ACO based approach can be used to connect the broken edges of the character.

ACO is a population based metaheuristic method that is inspired by the shortest path searching behavior of various ant species. A metaheuristic is a set of algorithmic concepts that can be used to define heuristic methods applicable to a large set of different problems [105]. In fact, metaheuristics are now widely recognized as the most promising approaches for attacking hard combinatorial optimization problems in different fields such as economy, commerce, engineering, industry or medicine. They incorporate concepts from very different fields such as genetics, biology, artificial intelligence, mathematics, physics and neurosciences [106]. Examples of metaheuristics include simulated annealing, tabu search, iterated local search, variable neighbourhood search algorithms, greedy randomized adaptive search procedures, evolutionary algorithms and ACO [105]. ACO is motivated by the natural phenomenon that the ants deposit pheromone on the ground in order to mark some favourable path that should be followed by other members of the colony. It aims to iteratively find the optimal solution of the target problem through a guided search (i.e., the movements of a number of ants) over the solution space by constructing the pheromone information [107]. Many authors have developed different sophisticated models for ACO that were used successfully to solve a large number of complex combinatorial optimization problems.

### 3.2 Background of Diffusion Process

Most people have an intuitive impression of diffusion as a physical process that equilibrates concentration differences without creating or destroying mass. This physical observation can be expressed by Fick’s law

\[ j = -D \cdot \nabla u \]  

(3.1)

This equation states that a concentration gradient \( \nabla u \) causes a flux \( j \) which aims to compensate for this gradient. The relation between \( \nabla u \) and \( j \) is described by the diffusion tensor \( D \), a positive definite symmetric matrix. The case, where \( j \) and \( \nabla u \) are parallel, is called isotropic (here, diffusion tensor may be replaced by a positive scalar valued diffusivity \( g \)), otherwise it is anisotropic. Considering Fick’s law into the continuity equation...
Chapter 3

\[ u_t = -\text{div} \ j \]  \hspace{1cm} (3.2)

It may end up with the diffusion equation

\[ u_t = \text{div}(D \cdot \nabla u) \]  \hspace{1cm} (3.3)

This equation appears in many physical transport processes. In the context of heat transfer it is called heat equation [108].

In image processing, the concentration may be interpreted as a gray value in a certain point. The diffusion tensor does not have to be a constant; it may be a function of the local image structure. This leads to nonlinear diffusion filters. Three cases are relevant for image processing [108]:

(a) Linear isotropic diffusion filters using a constant diffusivity,

(b) Nonlinear isotropic diffusion filters with diffusivities being adapted to local image structure,

(c) Nonlinear anisotropic diffusion filters with diffusion tensors being adapted to local image structure.

3.2.1 Scale Space

Image smoothing by parabolic PDEs is closely related to the scale space concept where one embeds the original image into a family of subsequently simpler, more global representations of it. Scale space is one parameter family of images representing a given input image on a continuum of scales [108]. A scale space image is the result of an initial image under the action of time. The zero scale limit will leave the initial image unscaled, whereas the infinite scale limit should give a complete spatial averaging of the initial image. The converged image is the desired image, and the scale space is just the way to reach it. A rescaled image is the convolution of the original image by some kernel. The following kernels are used in the respective domains [126]:

a) In Fourier domain

\[ \tilde{\mathcal{G}}(\Omega) = e^{-\frac{1}{2} \Omega^2} \]  \hspace{1cm} (3.4)
b) In dimensionful coordinates

\[ G(\omega; \sigma) = e^{-\frac{1}{2} \sigma^2 \omega^2} \]  

(3.5)

c) In the spatial domain

\[ G(x; \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2} x^2} \]  

(3.6)

Scale spaces may be classified into two: linear scale space and nonlinear scale space. Linear scale space is derived from linear heat equation. The main property of this scale space is the diffusion independence with respect to the signal. Nonlinear scale spaces relax the constraint of uncommitment in the process of visual information, keeping the main properties of a scale space. They may rely on geometry to influence the evolution of surfaces or curves, or consider morphology based on the erosion or dilation of images [109].

The scale space should have the following properties [110]:

- Causality: A scale space representation should have the property that no spurious details should be generated passing from finer to coarser scales.
- Immediate localization: At each resolution, the region boundaries should be sharp and coincide with semantically meaningful boundaries at that resolution.
- Piecewise smoothing: At all scales, intra-region smoothing should occur preferentially over inter-region smoothing.

The best investigated example of a scale space is the Gaussian scale space which is obtained via convolution with Gaussians of increasing variance, or equivalently by linear diffusion filtering [108].

**Definition 3.1:** Let \( g : \mathbb{R}^n \to \mathbb{R} \). The Gaussian scale space of \( g \) is a function \( T_g : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R} \), given by

\[ T_g(x, t) = g^*G_t(x) \]
where

\[ G_t(x) = \frac{1}{(2\pi t)^{n/2}} e^{-\frac{(x_1^2 + \ldots + x_n^2)}{2at}} \]

is the Gaussian distribution of dimension \( n \) with mean \( \mu \) and variance \( at \), where \( a \) is a positive constant. The Gaussian function given in equation (3.6) is modified by substituting the standard deviation \( \sigma = \sqrt{at} \) and \( x = (x_1, \ldots, x_n) \in \mathbb{R}^n \). The parameter \( t \) is called scale (or time) [111]. Here, \( \sigma \) is the standard deviation of the noise, which has zero mean in the Gaussian distribution. As \( \sigma \) increases, the image gets more blurred. The Gaussian scale space is a set of all blurred images. The Gaussian kernel is called isotropic if the behavior of the function in any direction is the same. When the standard deviations in different dimensions are not equal, Gaussian function is anisotropic [112].

The scale space has application in various fields such as stochastic, statistics, clustering, recognition, segmentation, and image enhancement by deblurring [113]. One of the most common problems in image processing is the removal of undesired interfaces called noises from the image surface. A widely used method for noise elimination is the Gaussian filter, in which signals are smoothed out by the convolution of the image with a Gaussian kernel [111].

### 3.2.2 Linear diffusion filtering

Let the gray scale image \( I \) be represented by a real valued mapping \( I \in L^1(\mathbb{R}^2) \). A widely used way to smooth \( I \) is by calculating the convolution

\[(G_\sigma * I)(x) := \int_{\mathbb{R}^2} G_\sigma(x-y)I(y)dy \]

where \( G_\sigma \) denotes the 2D Gaussian of width (standard deviation) \( \sigma > 0 \):

For any bounded \( I \in C(\mathbb{R}^2) \) the linear diffusion process

\[ u_t = \Delta u, \]

\[ u(x, 0) = I(x) \]

processes the unique solution
\[ u(x,t) = \begin{cases} 
I(x) & (t = 0) \\
(G_{\sqrt{t}} \ast I)(x) & (t > 0) 
\end{cases} \]  

(3.8)

From (3.8) we observe that the time \( t \) corresponds to the spatial width \( \sqrt{2t} \) of the Gaussian \cite{108}. Hence, smoothing the structures of order \( \sigma \) requires stopping the diffusion process at time

\[ T_\sigma = \frac{1}{2} \sigma^2 \]  

(3.9)

The following drawbacks are also observed in linear diffusion filtering \cite{108}:

(a) Linear diffusion filtering dislocates edges when moving from finer to coarse scales.

(b) Gaussian smoothing not only smooths the noise, but also blurs important features such as edges and makes them harder to identify.

(c) Some causality properties of Gaussian scale space do not hold for dimensions larger than one.

The first two drawbacks can be avoided by nonlinear diffusion processes, while the last one requires morphological equations.

### 3.2.3 Nonlinear diffusion filtering

According to human visual system, edges are very important primitives in natural images. They should be conserved in order to avoid wrong linking, and this paves the way to use nonlinear diffusion as a natural scale space candidate. Perona and Malik \cite{110} proposed an improvement to linear multiscale analysis with more accurate edge detection. The main idea was to introduce a part of the edge detection step in a PDE model, encouraging smoothing within homogeneous regions in preference to smoothing across the boundaries. Blurring would then take place separately in each region, letting region boundaries to remain sharp. They proposed the following nonlinear anisotropic diffusion equation to avoid blurring and localization problems of linear diffusion filtering. This nonuniform process reduces diffusivity at those locations having larger likelihood to be edges.

\[ u_t = \text{div}(g(|\nabla u|)\nabla u) \]  

(3.10)

\[ u(x, y, 0) = I(x, y) \]
Chapter 3

where $g$ is a non increasing smooth function, such that $g(0) = 1$, $g(s) \geq 0$, and $g(s) \to 0$ when $s \to \infty$. $I(x, y)$ is the image to be processed, $u(x, y, t)$ is its smoothed version in the scale $t$, $\nabla$ is the gradient operator, and $\text{div}$ denotes the divergent operator. The idea is that if $|\nabla u|$ is large, then the diffusion will be low, and therefore the exact localization of the edges will be preserved.

Computational experiments show that the edge detector based on this theory yields edges and boundaries that remain more stable through the scale $t$. In spite of this, the model still has several theoretical and practical difficulties. If the image is very noisy, the gradient $|\nabla u|$ will be very large. As a result, the function $g$ will be close to zero in almost every point. Consequently, most of the noise will remain in the image. From a geometrical point of view, the diffusion operator can be modified in a way that the diffusion process becomes more intense along the edges and less intense along the perpendicular direction of the edges. To achieve this, define $g$ as $g(s)=1/s$ in (3.10) and multiply the whole expression by $|\nabla u|$ to get the mean curvature equation,

$$
u_t = |\nabla u| \text{div} \left( \frac{\nabla u}{|\nabla u|} \right)$$

(3.11)

The geometric interpretation of this equation is that the level lines of the solution propagate in the normal direction with the speed proportional to the mean curvature [111].

Nordstrom [114] modified (3.10) by introducing the forcing term $(u-I)$ to it.

$$
u_t - \text{div}(g(\nabla u)\nabla u) = I - u$$

(3.12)

The new term has the property of preserving $u(x, y, t)$ close to the initial image $I(x, y)$. The addition of forcing term in (3.10) reduces the degenerative effects of the diffusion to an acceptable level. However, the model with this forcing term does not eliminate the noise satisfactorily. An improvement over the above models is given in the following section.
3.3 Edge Structure Focusing using Anisotropic Diffusion

The nonlinear anisotropic diffusion via PDE is an edge detection methodology which tries to solve the duality problem "detection versus localization". The main idea of this methodology is to carry out a selective smoothing of the image in a previous stage. The process smooths homogeneous regions of the image more intensively, removing spurious information usually related to noise. Consequently, PDE detector preserves the edges of better contrast, making it possible to detect mainly the contour of the character.

The edge detector based on anisotropic diffusion via PDE has two basic stages. The first one consists in accomplishing the anisotropic diffusion process to generate a smoothed image without displacing the prominent structures of the image. In the second stage, a simple thresholding of the gradient image is used to find the edge [104]. The ideas from the previous models are incorporated to present the following parabolic equation for image smoothing:

\[
u_t = g |\nabla u| \text{div} \left( \frac{\nabla u}{|\nabla u|} \right) - \lambda (1 - g)(u - 1), \quad t > 0,
\]

\[
u(x, y, 0) = I(x, y), \quad (x, y) \in \mathbb{R}^2
\]

where the parameter \( \lambda \) acts as a weight of the term \((1-g)\). The mathematical model given above has the purpose of selectively smoothing the image. The term \( |\nabla u| \text{div} \left( \frac{\nabla u}{|\nabla u|} \right) \) diffuses the image \( u \) along the orthogonal direction to its gradient \( \nabla u \). The goal is to allow smoothing on both sides of an edge with minimal smoothing on the edge itself. If \( \nabla u \) has a small mean in the neighbourhood of a point, it will be considered as an interior point, and the diffusion will be strong. On the other hand, if \( \nabla u \) has a large mean value, then the point will be considered an edge point.

The function \( g = \frac{1}{1 + k |G_r * \nabla u|^2} \) is used to control the diffusion speed. i.e., the selective smoothing is carried out at speeds that are lower in the surroundings of a point.
where the term $|G_{T_o} * \nabla u|$ is small. Consequently, the second term in the denominator of function $g$ will be very small. Therefore, $g \to 1$; thus the term $(1-g) \to 0$. So the term $(u-l)$, which preserves the edge does not act in the model. Consequently, the diffusion accomplished by the first term in (3.13) will be higher within the homogenous regions. If the term $|G_{T_o} * \nabla u|$ is very high, then the analyzed point would be considered an edge point. In that case, the second term in the denominator of function $g$ will be high, so $(1-g) \to 1$ when $g \to 0$. The term $(u-l)$ will act strongly in the image, keeping the original characteristics of edges. Thus the diffusion process carried out by the first term will have an insignificant effect along the edge regions. The function $g(s) \geq 0$ is a nonincreasing function, satisfying $g(0)=1$ and $g(s) \to 0$ when $s \to \infty$. Since $g(s)$ assumes considerably smaller values for large values of $s$, the diffusion will be low. The constant $k$ (I dependant constant) is chosen so as to carry out the role of function $g(s)$. The convolution $G_{T_o} * \nabla u$ defines the Gaussian scale space. It is the local estimate of $\nabla u$ used for noise elimination.

The Gaussian function $G_{T_o}$ is given by,

$$G_{T_o}(x,y) = \frac{1}{2\alpha \pi T_o} e^{-\frac{x^2 + y^2}{2\alpha^2 T_o}} \quad (x,y) \in \mathbb{R}^2$$

(3.14)

The scale parameter $\sigma = \sqrt{\alpha T_o}$, where $\alpha$ is a real constant. The optimal time needed to achieve an efficient and adequate smoothing level in a noisy image with standard deviation of the noise $\sigma$ is given by the expression [115],

$$T_o = \frac{\sigma^2}{\alpha}$$

(3.15)

$u(x_i, y_i, t_o)$ is denoted by $u_{i,j}^n$ where $t_o = n\Delta t$. The derivative of $u$ in relation to time $t$, $(u_t)$ is calculated in $(x_i, y_i, t_o)$, and it is approximated by Euler’s method, i.e., $u_t \approx \left( \left( u_{i,j}^{n+1} - u_{i,j}^n \right) / (\Delta t) \right)$ and the diffusion term.
\[
|\nabla u| \left( \text{div} \left( \frac{\nabla u}{|\nabla u|} \right) \right) = \frac{u_{x}^{2} u_{yy} - 2u_{x} u_{y} u_{xy} + u_{y}^{2} u_{xx}}{u_{x}^{2} + u_{y}^{2}}
\]

(3.16)

is approximated using central differences. Using Neumann’s boundary conditions \( u_{y}^{n+1}, n=1, 2, \ldots, N \), is calculated as [111],

\[
u_{y}^{n+1} = u_{y}^{n} + \Delta t L(u_{y}^{n})
\]

(3.17)

with \( u_{y}^{0} = I(x_{o}, y_{l}) \) and \( L(u) = g |\nabla u| \text{div} \left( \frac{\nabla u}{|\nabla u|} \right) - \lambda (1 - g) (u - I) \)

All the above models present the same computational complexity as the diffusion term (the term of greater order in the differential equation) is same in all these models.

The temporal evaluation \( t \) in the model of anisotropic diffusion is directly related to \( T_{o} \)

\[
t = \frac{T_{o}}{\nabla t}
\]

(3.18)

where \( \nabla t \) represents the step size of temporal evaluation.

The mathematical model consists of an iterative process up to \( T_{o} \), which is controlled by the temporal evolution. The estimation of parameter \( \sigma \) is subjective, and the choice of a suitable threshold is through trial and error method.

On application of PDE model to the image, a smoothed image will be obtained. Then, edges can be detected from this image using the equation [104],

\[
\bar{g}(|\nabla u|) = \frac{1}{1 + k_{2} |\nabla u|^{2}}
\]

(3.19)

where \( k_{2} \) is the constant in the function \( \bar{g} \), with \( 0 \leq \bar{g} \leq 1 \). After the application of this equation to the smoothed image \( u \), the pixels whose values of \( \bar{g} \) next to the unitary value are changed to a null gray value, and the pixels with values of \( \bar{g} \) next to null value are changed to unitary gray value. The result is a binary image where the edge pixels are white and the background pixels are black.
3.3.1 Need of Canny and PDE detectors

At this stage, it is better to compare the detectors to know the need of using them together in this particular problem. The Canny detector is a complete process, and it produces thinned edges ready to be used in feature extraction. The PDE detector provides results similar to the first two stages (smoothing and differentiation) of Canny detector. These detectors differ on three characteristics: edge displacement, SNR, and localization precision of thinned edges [104].

The displacement refers to ridges on the gradient magnitude surface or to the edges on the gradient image. The edge structure obtained by the first two stages of Canny detector is just preserved at a fine scale, and in the coarse scale, edge displacement of several pixels can happen. The PDE detector is not affected by edge displacement along the diffusion process. Although displaced edges imply necessarily in localization imprecision of the corresponding thinned edges, nondisplaced edges do not necessarily imply in accurately thinned edges.

In the case of SNR, the behavior is similar for both detectors. Higher the SNR, the less sensitive is the detector to the noises and texture of the image, enabling it to detect mainly the physical edges present in the image. Thus, the application of both detectors to a coarse scale would produce a satisfactory result. But due to the low SNR, the application of two detectors to a fine scale will not give better result.

Another difference occurs with respect to localization precision of thinned edges. The ridge shape generated by the Canny detector, mainly at fine scale, possesses high curvatures along its traverse sections. As a result, the nonmaxima suppression possesses high discrimination power for detecting the maximum points along the traverse sections of ridge, allowing high precision in edge thinning. The same doesn’t happen with the edge detector based on anisotropic diffusion. This detector can generate traverse sections of ridges with low curvature or even with too flat curvature due to its capacity for homogenizing areas with similar characteristics. Consequently, nonmaxima suppression becomes an inaccurate process and it can even prove that it isn’t possible to find the localization of thinned edges.
Considering all the above factors, the solution consists in applying the Canny detector to a fine scale, but just on the edge structures focused by the edge detector of anisotropic diffusion via PDE. This procedure allows the minimization of disadvantages of each detector, because mainly high precision thinned edges are detected. Moreover, SNR of the final results drastically increases due to the previous isolation of noise and texture areas of the image. Thus, both methods work together.

3.4 Ant Colony Optimization

Swarm intelligence is a relatively new approach to problem solving that takes inspiration from the social behaviours of insects and of other animals. In particular, ants have inspired a number of methods and techniques, among which the most studied and successful is the general purpose optimization technique known as ACO. ACO takes inspiration from the foraging behavior of some ant species. The ants deposit pheromone on the ground in order to mark some favourable path that should be followed by other members of the colony. ACO exploits a similar mechanism for solving optimization problems [105].

An artificial ant is defined to be a simple computational agent, which iteratively constructs a solution to the given problem. A solution construction starts from an empty partial solution. The partial problem solutions are seen as states; each ant moves from a state to another corresponding to a more complete partial solution. In each step, ant \( t \) computes a set of feasible expansions to its current state, and moves to one of these in probability [116].

ACO is an iterative algorithm. In each iteration, a number of artificial ants are used. Each ant builds a solution by walking on the construction graph. The edge of the graph has associated two kinds of information to guide the ant movement [106]:

- **Heuristic information**, which measures the heuristic preference of moving from node \( i \) to node \( j \), i.e., of traveling the edge \( a_{ij} \). It is denoted by \( n_{ij} \). This information is not modified at runtime by the ants.
- **Artificial pheromone trail information**, which measures the “learned desirability” of the movement and mimics the real pheromone that natural ants deposit. This
information is modified at runtime depending on the solutions found by the ants. It is denoted by $\tau_i$.

After initialization, the metaheuristic iterates over three phases: ants construct a number of solutions, these solutions are then improved through a local search (optional), and finally the pheromone is updated [105].

The basic operation mode of an ACO algorithm is as follows: the $m$ ants in a colony move concurrently and asynchronously through the adjacent states of a problem (that can be represented in the form of a weighted graph). This movement is made according to a transition rule which is based on local information available at the components (nodes). This local information comprises heuristics and memoristic (pheromone trails) information to guide the search. By moving on the construction graph, ants incrementally build solutions. Optionally, ants can release pheromone each time they cross an edge (connection) while constructing solutions (online step by step pheromone trail update). Once every ant has generated a solution, it is evaluated and deposited the pheromone depending on the quality of the ant’s solution (online delayed pheromone trail update). In the future, this information will guide the search of the other ants of the colony.

The generic operation mode of the ACO algorithm includes two additional procedures, pheromone trail evaporation and daemon actions. The pheromone evaporation is triggered by the environment and it is used as a mechanism to avoid search stagnation and to allow the ants to explore new space regions. The pheromone values on lower quality trails which are not reinforced often enough will progressively evaporate. This way, less promising paths will slowly vanish from the graph because of being visited by less number of ants. Daemon actions are optional – without a natural counterpart – to implement tasks from a global perspective. Examples are observing the quality of all the solutions generated and releasing an additional pheromone amount only on the transitions/components associated to some of the solutions, or applying a local search procedure to the solutions generated by the ants before updating the pheromone trails. In both cases, the daemon replaces the online delayed pheromone update and the process is called offline pheromone trail update. Several components are either optional or strictly dependent on the specific ACO algorithm, e.g., when and where the
pheromone is deposited. Generally, the online step by step pheromone trail update and online delayed pheromone trail update are mutually exclusive and they are not present or missing at the same time. If both are missing, the daemon updates the pheromone trails [106].

3.4.1 Steps to Solve a Problem by ACO

The following are the guidelines to attack a problem by ACO [106]:

- Represent the problem in the form of components and transitions or by means of a weighted graph that is traveled by the ants to build solutions.
- Appropriately define the meaning of the pheromone trails τ_{ij}, i.e., the type of decision they bias.
- Appropriately define the heuristic preference to each decision that an ant has to take while constructing a solution, i.e., define the heuristic information η_{ij} associated to each component or transition.
- If possible, implement an efficient local search algorithm for the problem under solution.
- Choose a specific ACO algorithm and apply it to the problem being solved.
- Tune the parameters of the ACO algorithm.

3.4.2 ACO Algorithms

The first ACO algorithm called ant system was proposed by Dorigo et al. [117] in 1991. Since then, a number of ACO algorithms have been developed, such as max-min ant system [118], the ant colony system [119], rank-based ant system [120], best-worst ant system [121], best-worst ant colony system [122], Ant-Q [123], and AntNet [124]. All ACO algorithms share the same characteristic idea. Among these algorithms, AntNet is rather application specific, and it is mainly used in network routing.

Ant System (AS) is a distributed algorithm for combinatorial optimization based on the metaphor of ant colonies. This basic idea underlying this algorithm is that of using a colony of cooperating ants to find shortest Hamiltonian tours in a weighted complete graph. AS can be
Chapter 3

interpreted as a particular kind of distributed Reinforcement Learning (RL) technique. Its main characteristic is that, the pheromone values are updated in each iteration by the ants that have built a solution [123]. The different variants of this algorithm are AS-density, AS-quantity, AS-cycle, and elitist AS, differing the way in which the pheromone trials are updated. In the first two variants, ants perform online step by step pheromone update, with the difference that the amount deposited in AS-density is constant while the one released in AS-quantity directly depends on the heuristic desirability of the transition. In AS-cycle, it follows online delayed pheromone update. This variant has the best performance and it is now referred to as AS. The extended version of this variant is known as elitist AS, which is used in the current study.

Ant Colony System (ACS) is one of the first successors of AS. It introduces three major modifications into AS:

- ACS uses a different transition rule called pseudo-random proportional rule: Let \( t \) be an ant located at a node \( i \), \( q_0 \in [0, 1] \) be a parameter, and \( q \) a random value in \([0, 1]\). The next node \( j \) is randomly chosen according to a probability based on the values of \( q \) and \( q_0 \). If \( q \leq q_0 \) it exploits the available knowledge, choosing the best option with respect to the heuristic information and the pheromone trail. Otherwise, it applies a controlled exploration as done in AS.

- Only the daemon (and not the individual ants) trigger the pheromone update, i.e., an offline pheromone trail update is done. To do so, ACS only considers one single ant, the one who generated the global best solution. The daemon can also apply a local search algorithm to improve the ants’ solutions before updating the pheromone trials.

- Ants apply an online step by step pheromone trail update that encourages the generation of different solutions to those yet found.

Max-Min Ant System (\( M:\!M:\!A\!S \)) is an extension of AS explicated in the following aspects:

- Similar to ACS, an offline pheromone trail update is applied. After all ants have constructed the solution, initially, every pheromone trail is evaporated and then it is deposited. The best ant that is allowed to add pheromone may be the iteration-best
or the global-best solution. In addition, the ants’ solutions are improved using local optimizers before the pheromone update.

- The possible values for the pheromone trails are limited to the range \([t_{\min}, t_{\max}]\). The chance of algorithm stagnation is thus decreased by giving some probability to choose each connection. As a means of further increasing the exploration of solutions, \(M:\!MAS\) also uses the occasional re-initialization of the pheromone trails.

- Instead of initializing the pheromones to a small amount, the pheromone trails are initialized to an estimate of the maximum allowed pheromone trail value.

Rank-based ant system (\(AS_{\text{rank}}\)) is another extension of \(AS\). It incorporates the idea of ranking into the pheromone update, which is again developed offline by the daemon as given below:

- The ants are ranked according to the decreasing quality of their solutions.

- The daemon deposits pheromone on the connections passed by elitist ants. The amount of pheromone deposited directly depends on the ant’s rank and on the quality of its solution.

- The connections crossed by the global-best solution receive an additional amount of pheromone which depends on the quality of that solution. This pheromone deposit is considered to be the most important, and hence it receives a weight value.

Best-Worst Ant System (\(BWAS\)) is an ACO algorithm which incorporates evolutionary computation concepts. It is also an extension of \(AS\), which uses its transition rule and pheromone evaporation mechanism. As done in \(M:\!MAS\), BWAS always considers the systematic exploitation of local optimizers to improve the ants’ solutions. At the core of BWAS, the following three daemon actions are found:

- The best-worst pheromone trail update rule reinforces the edges contained in the global best solution. In addition, the update rule penalizes every connection of the worst solution generated in the current iteration, that is not present in the global-best one through an additional evaporation of the pheromone trails.
The pheromone trail mutation is performed to introduce diversity in the search process.

As other ACO models, BWAS considers the re-initialization of pheromone trails when it gets stuck.

These three components can be incorporated into ACS to form best-worst ant colony system [106].

Ant-Q, a family of algorithms inspired by AS system, strengthens the connection between RL, in particular Q-learning, and AS. Two characteristics of these algorithms are (i) agents (ants) do not end up making the same tour, and (ii) the learned AQ-values (Ant-Q value associated with an edge) are such that they can be exploited by agents to find the short tours. Ant-Q differs with AS in the following respect [123]:

- AQ-values are updated with a reinforcement term and of the discounted evaluation of the next state.
- Only those AQ-values which correspond to edges belonging to global best/iteration best tour will receive reinforcement.

3.4.3 Applications

ACO algorithms have been applied to a large number of combinatorial optimization problems. Generally, the different applications of ACO fall into two classes. The first class comprises \( \mathcal{NP} \)-hard combinatorial optimization problems, for which the best known algorithms guarantee to identify an optimal solution have exponential time worst case complexity. The second class includes dynamic shortest path problems, where the problem instance under solution changes at algorithm run time. So far, ACO has been tested over one hundred \( \mathcal{NP} \)-hard problems. These problems fall into one of the following categories: routing problems, assignment problems, scheduling problems, and subset problems.

The scheduling problem is concerned with the allocation of scarce resources to tasks over time. In assignment problems, a set of items has to be assigned to a given number of resources subject to some constraints. The solution to a subset problem is considered to be a
selection of a subset of available items. To mention some of the problems solved by ACO metaheuristic are travelling salesman problem, quadratic assignment problem, job shop scheduling, flow shop scheduling, vehicle routing, sequential ordering, graph colouring, shortest common super sequence, generalized assignment, set covering, multiple knapsack, constraint satisfaction and packet switched network routing. It is also used in bioinformatics and in the design of learning algorithms for knowledge representation structures such as classical logic rules, fuzzy logic rules and Bayesian networks. The image processing and analysis, segmentation and edge detection can also be done using swarm-based algorithms like ACO. ACO appears to perform particularly well on problems which show high correlation between the fitness of solutions and the distance to global optima [106].

3.5 Edge Linking using Elitist Ant System

Canny edge detector is one of the best edge detection algorithms as it produces edges with one pixel wide. However, just like any other algorithms, it also suffers the problem of disjointed edges. In order to obtain a precise description and accurate analysis of an object, edge linking is necessary. In this section, the problem of linking disjointed edges is tackled by an elitist ant strategy [125]. The input to this algorithm is a binarized image of thin edges. Each pixel $x_i \in X$ represents a binary value in the image $X = \{x_i | x_i = 0 \text{ or } 1; 0 \leq i < \text{height \times width}\}$, and the pixel at $r^{th}$ row $c^{th}$ column is $x_{(r \times \text{width} + c)}$.

The first step in the algorithm is to extract the endpoints of all segments from the binarized image, where each endpoint $y_i \in Y \subseteq X$. Each endpoint is processed one by one for $N$ iterations using $m$ number of ants. There is one solution per ant. The two basic ACO procedures, the construction of solutions and the update of pheromone trails by the ants, are repeatedly applied until a termination condition is satisfied. In practice, a termination condition may be the maximum number of solutions generated, the maximum CPU time elapsed, or the maximum number of iterations without improvement in solution quality. At the beginning of iteration, the initial direction of each ant is computed based on the endpoint’s trajectory. During the iteration, the ant is continuously moved using a probabilistic approach until the ant reaches another line segment. This probability depends on two values, i) the attractiveness $\eta$ of the move (Table 3.1), as computed by some heuristic indicating a priori
desirability of that move, ii) The trail level $\tau$ of the move, indicating how proficient it has been in the past to make that particular move. Here, the objective is to find the best line segment to connect the current segment. The probability of moving from pixel $i$ to pixel $j$, where pixel $j \in N_{d}(\text{pixel } i)$, is given by:

$$
P_{ij} = \frac{\tau_{ij}^{\alpha} n_{ij}^{\beta}}{\sum_{\text{pixel } l \in N_{d}(\text{pixel } i)} \tau_{il}^{\alpha} n_{il}^{\beta}}
$$

(3.20)

where $i$ and $j$ range from 0 to $n$-1 and $n = \text{height} \times \text{width}$. $N_{d}(\text{pixel } i)$ is the set of eight neighbouring pixels surrounding the pixel $i$. $\alpha$ and $\beta$ are the two parameters set by the users depending on the problem.

**Table 3.1 Values of $n_{ij}$**

<table>
<thead>
<tr>
<th>$\frac{1}{2}$</th>
<th>1</th>
<th>$\frac{1}{2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{4}$</td>
<td>Ant</td>
<td>$\frac{1}{4}$</td>
</tr>
<tr>
<td>1/12</td>
<td>1/20</td>
<td>1/12</td>
</tr>
</tbody>
</table>

The pheromone value $\tau_{ij}$ serves as positive feedback to attract the ants to follow solutions found by other ants. Initially, it is set to be close to zero at each pixel inside the image. In order to attract ants toward endpoints, attractors (pheromone values) are placed around the endpoints with Gaussian distribution. Let $A$ be the amplitude of a 2D Gaussian function with standard deviation $\sigma$ centred at an endpoint. The pixel $s$ which is $\Delta r$ rows and $\Delta c$ columns away from that endpoint will have an initial pheromone of

$$
\tau_{s} = Ae^{-\frac{\Delta r^{2} + \Delta c^{2}}{2\sigma^{2}}}
$$

(3.21)

If the pheromone value is high, the probability to choose that particular trail will also be high.

Each ant is given a lifespan which limits its maximum tour length. If an ant fails to reach any edge segment within its lifespan, or if it returns to the original segment, then the ant will be terminated and no pheromone updates will occur. Only the ants that have successfully travelled from one segment to another segment will update the pheromone in their path due to solution found by the ants. There is a cost associated with each solution, and it is determined as given below.
The pheromone level is updated using (3.22),

$$\tau_j(new) = \tau_j(old)(1-\rho) + \sum_t \Delta \tau_j^t$$

(3.22)

where $\rho$ represents the pheromone evaporation rate. The pheromone evaporation implements a useful form of forgetting: it avoids the algorithm from covering too rapidly toward a suboptimal region, therefore favouring the exploration of new areas in the search space. The amount of pheromone added onto pixel $j$ by ant $t$ is given by:

$$\Delta \tau_j^t = \begin{cases} 
q & \text{if pixel } j \text{ selected by ant } t, \\
0 & \text{otherwise}
\end{cases}$$

(3.23)

and,

$$\text{Cost}_t = \begin{cases} 
D^t & \text{if ant } t \text{ found the pixel inside a segment}, \\
D^t + \frac{p^t}{21-\frac{1}{\eta_{ij}}} & \text{if ant } t \text{ found the endpoint of a segment}
\end{cases}$$

(3.24)

where $D^t$ is the total distance travelled by ant $t$ from one segment to another segment. $Q$ is an adjustable parameter which is set to 1. (3.24) is necessary to allow the ants to join those endpoints which are further than other segments [115].

Finally, the best solutions found at every endpoint are recorded after all the iterations have been carried out. The tours giving the least solution cost for each endpoint are the best solutions. Then, the results of these tours are drawn onto the original binarized image of thin edges, thus connecting one edge segment endpoint to another segment.

### 3.6 Experimental Results

The experimental evaluation of the two approaches presented in this chapter is given separately in the following sections.

#### 3.6.1 Edge detection

The edge detection with focused edge regions is applied on handwritten character images. To verify the effect of this approach, gray scale images of resolution 150, 300, and 600 dpi are used. As the resolution increases, the noise level is also increased. The greater the noise level, greater is the value of $t$. As the scale grows, the image gets more smoothed. The results of the experiment are presented at time $t = T_o$. The optimal time is directly related to
the noise intensity. For highly noisy images, a longer time of evolution in scale \( t \) will be necessary. On the other hand, the optimal time is related to the number of iterations. The edge detection of the character image \( \eta \) (\(/\eta/\)) with a resolution 600 dpi is shown in Fig. 3.1.

![Character Image](image)

**Figure 3.1** Edge detection with edge structure focusing: (a) character \( \eta \) (\(/\eta/\)) (b) iteration = 15, \( \Delta t = 1/15 \) (c) iteration = 45, \( \Delta t = 1/9 \) (d) iteration=65, \( \Delta t = 1/5 \)

Experiments have been conducted with different values of \( t \). When the number of iteration and \( \Delta t \) are less, spurious branches are occurred in the edge image (Fig 3.1(b)). If these parameters exceed certain limit, broken and undesired edges will appear in character edge (Fig 3.1(d)). An accurate edge is obtained for a \( \Delta t \) value of 1/9 iterated at 45 (Fig 3.1(c)).

The performance of Canny-PDE detector is also compared with that of Canny detector. Fig. 3.2 shows the edge skeletons generated using these approaches for different image resolutions.

![Character Image](image)

**Figure 3.2** Character \( \eta \) (\(/\eta/\)) (a) at resolution 150 dpi (b) Canny edge (c) Canny-PDE edge (d) at resolution 300 dpi (e) Canny edge (f) Canny-PDE edge (g) at resolution 600 dpi (h) Canny edge (i) Canny-PDE edge
The resolution of the images in the first, second and third rows are 150, 300 and 600 dpi respectively. As resolution increases, the number of iterations, Δt and computation time are also increased. The numbers of iterations required are 10, 15 and 45 respectively in Fig. 3.2(c), 3.2(f), and 3.2(i). The corresponding Δt values are 1/11, 1/11 and 1/9. The edge detection method based on edge structure focusing took the computation time of 0.19, 0.49 and 3 seconds for these images. From the above figures, it is clear that Canny detector miserably failed to produce accurate edges in high resolution edges. In case of 300 and 600 dpi images, the edge detector produced a lot of spurious branches (Fig. 3.2(e) and 3.2(h)). Traverse sections of edge regions in high resolution images are usually described by a greater number of pixels than in low resolution images. Canny detector will produce a large amount of spurious edges resulting from noise and texture of these images. But the defect could be rectified by applying Canny edge detector on focused edge regions (Fig 3.2(f) and 3.2(i)). In low resolution image, little advantage is obtained using this approach. Both edge detection methods produced edge images in binary format.

3.6.2 Edge linking

Several experiments were carried out on a set of images to test the capability of ACO algorithm. The character images used for these experiments are shown in Fig. 3.3-3.7. Even though the edge detection algorithm generated good contours, disjoint edges can be seen in some areas. These broken edges are further linked using an ACO based approach.

ACO algorithm accepts a binary image containing disjointed edges. Initially, segments with length 1 and 2 are removed from the edge image. An isolated pixel cannot provide the direction of ants’ movement, and the segment with length 2 will appear as it is while generating the endpoints. The performance of this algorithm mainly depends on the correct tuning of several parameters, namely: relative importance of trail (α) and attractiveness (β), trail persistence (ρ), initial trail level (τ₀), number of ants (m), lifetime of the ant, attractor’s parameters A and σ, number of iterations of the algorithm, and Q used for quantifying the quality of a solution.
Chapter 3

Figure 3.3 shows the character \( \varnothing (\hat{\eta}a/) \), its edge image and the linked images. There are four broken parts in the edge image. When the number of iteration is 10, the algorithm could correctly link three of them. The fourth one is linked with a nearby segment instead of linking with an endpoint. But, when the number of iteration is increased to 20, all the broken edges are linked as desired. As the number of iterations increase, the ants will get more chance to search for endpoints. The maximum gap to be filled between edges can be set by limiting the lifespan of the ants. Similar to Fig 3.3, Fig 3.4 also shows false edges when the lifetime of the ant is taken as 10. All the edges in the character \( \varnothing (\hat{\eta}a/) \) is linked correctly by increasing the lifetime to 11. When the lifetime is further increased, an undesired edge is generated. Even though this parameter seems to be highly sensitive, it is found that a common value will work for most characters.

The two parameters, \( \alpha, \beta \in \mathbb{R} \), control the relative importance of the pheromone versus the heuristic information. The role of these parameters is as follows: if \( \alpha = 0 \), those nodes with better heuristic preference have a higher probability of being selected, thus making the algorithm close to a classical probabilistic greedy algorithm. However, if \( \beta = 0 \), only the pheromone trails are considered to guide the constructive process, which can cause a quick stagnation. This is a situation where the pheromone trials associated to some transitions are significantly higher than the remainder, thus making the ants always build the same solutions, usually local optima. Hence, there is a need to establish a proper balance between the importance of heuristic and pheromone trail information. Figure 3.5 shows the experiments conducted with different values of \( \alpha \) and \( \beta \). A false edge is generated in Fig 3.5(c), whereas a choice of \( \alpha = 1 \) and \( \beta = 3 \) correctly linked the edges of character \( \varnothing (\hat{\eta}a/) \). Finally, the attractor’s parameters \( A \) and \( \sigma \) are tuned, and its result are shown in Fig 3.6. When both values are taken as 1, some undesired edges are generated. The edges of character \( \varnothing \hat{a} (\hat{\eta}a/) \) is linked as desired by changing the values of \( A \) and \( \sigma \) (Fig 3.6(d)).

Based on the above experiments, the following values are suggested for different parameters of this algorithm.
iteration : 30
\( \alpha \) : 1
\( \beta \) : 3
\( A \) : 10
\( \sigma \) : 10
\( \rho \) : 0.1
\( \tau_0 \) : 0.001

Figure 3.3 Edge linking on (a) Character \( \alpha \) (/ /) (b) Edge image (c) Edge linked with iteration 10 (d) Edge linked with iteration 20

Figure 3.4 Edge linking on (a) character \( \alpha \) (/ \( \beta \) /) (b) Edge image (c) Edge linked with life time 10 (d) Edge linked with lifetime 11 (e) Edge linked with lifetime 15

Figure 3.5 Edge linking on (a) Character \( \alpha \) (/ /) (b) Edge image (c) Edge linked with \( \alpha = 1, \beta = 2.5 \) (d) Edge linked with \( \alpha = 1, \beta = 3 \)

Figure 3.6 Edge linking on (a) Character \( \alpha \) (/ \( \beta \) /) (b) Edge image (c) Edge linked with \( A = 1, \sigma = 1 \) (d) Edge linked with \( A = 10, \sigma = 10 \)

Figure 3.7 Edge linking on (a) Character \( \alpha \) (/ \( \beta \) /) (b) Edge image (c) Linked image (d) Generated links

Figure 3.7 shows the edge linking using the above parameter values. This algorithm could generate the linked image with an average execution time of 0.74 second. In order to reduce the execution time, the already linked endpoints are omitted from further search. It shall be noted that, the execution time increases as the number of iterations increase. Figure
Chapter 3

3.8 shows an interesting problem of the presence of parasitic components in the edge image. These components are repeated in linked image too. They need to be handled in a further process.

![Diagram showing edge linking](image)

Figure 3.8 Edge linking on (a) Character a (/ma/) (b) Edge image (c) Linked image

3.7 Conclusion

Edge detection and edge linking are two important activities in image processing. The use of Canny detector with a priori edge focusing is usually more appropriate for applications involving automatic or semiautomatic feature extraction processes from digital images, especially high resolution ones. This methodology makes it possible to obtain high quality contour information. This work brings innovations such as the use of parameter $\sigma$ in the Gaussian function as being the standard deviation of the noise, and the optimal time estimate in the process of temporal evolution. The balanced smoothing and the use of the optimal time concept to stop the evolution of the PDE produce good results and high computational gain in noise removal and image edge detection process. Some broken edges may be observed in some segments where parts of the character are nearby. But this problem is tackled by an ACO based approach. So perfect edges are obtained for most of the characters undergoing these two processes.

The performance of ACO algorithm mainly depends on the values of pheromone trail, heuristic information, and tuned parameters. A good definition of the pheromone trails typically requires insight into the problem under solution. Also, heuristic information is crucial for the good performance if local search algorithms are not available or cannot be applied. To tune the parameters, a good starting point is to use parameter settings that were found to be good when applying ACO algorithms to similar problems or to a variety of other problems. As an alternative, automatic procedures can also be used for the same purpose.
Even though the algorithm generated the links without much computational burden, the possibility of using other ACO algorithms in edge linking has to be investigated. Furthermore, the issue of parasitic components should be handled either in the edge detection stage or in the edge linking stage. This can also be done as a separate process.