

TDK-dolgozat

Papp Dávid

Reconstruction and recognition of noisy character-like symbols

Papp Dávid

Témavezető: Dr. Czúni László, egyetemi docens
Pannon Egyetem, Műszaki Informatika Kar,
Villamosmérnöki és Információs Rendszerek Tanszék

Table of Contents

1	Introduction.....	1
2	Overview of Literature.....	2
2.1	Linear convolution filters.....	2
2.2	Nonlinear filters.....	3
2.3	Anisotropic diffusion.....	3
2.4	Active contour models.....	4
3	Noise.....	6
3.1	Gaussian noise.....	6
3.2	Salt and pepper noise.....	7
3.3	Quantization (uniform) noise.....	8
3.4	Noise measurement.....	8
4	The reconstruction models.....	10
4.1	The MRF model.....	10
4.2	Markov Random Field in image reconstruction.....	11
4.3	Radon transform.....	13
4.4	Extension of the MRF model.....	14
5	Energy minimization.....	15
6	Implementation.....	18
6.1	MRF reconstruction.....	18
6.2	Noise generation.....	21
6.3	Classification.....	21
7	Results.....	22
7.1	Reconstruction of characters.....	22
7.2	Other symbols.....	26
8	Summary.....	29
9	Bibliography.....	30

Chapter 1

Introduction

In the field of image processing, the reconstruction of noisy images is a common problem. Several general methods exist for this task, for example filtering methods, patch-based reconstructions, Markov Random Field models. In several cases, some a priori information exists about the shapes, which can be used to enhance the results of the reconstruction. Furthermore, the goal of the reconstruction process is often to produce an image, which can be later used for recognition.

The goal of this paper is to propose an extension of an already existing image reconstruction algorithm, which improves the results of the reconstruction, and also provides a method for recognizing symbols.

The denoising method used in this paper is the Markov Random Field (MRF) model. As an a priori information, the shapes of the possible characters (for example the alphabet and the digits) are known. This information is incorporated into the model by the use of 2D binary discrete Radon transforms, which are easy and fast to compute, and do not need a significant amount of extra data to be stored.

After the overview of the existing literature of this field, the formulation of the MRF model and its extension with Radon transforms will be presented. It is followed by a brief description of the implementation. In the following chapter, the experiments including reconstruction and recognition performance can be seen.

Chapter 2

Overview of literature

Several methods exist for the problem of removing noise from an image, or recognizing shapes on a noisy images. In this chapter, an overview of a few of these methods will be presented.

2.1 Linear convolution filters

These methods use convolution with a filter kernel to create an approximation of the noise-free image from the noisy image [6, 28]. The mathematical definition of the convolution is the following:

$$C(x, y) = \frac{1}{k} \sum_{i=-\frac{N}{2}}^{\frac{N}{2}} \sum_{j=-\frac{M}{2}}^{\frac{M}{2}} K(i, j) f(x - i, y - j), \quad (2.1)$$

where K is a filter kernel of size $N \times M$, and f is an image. The term k is used to normalize the result to avoid amplification of the data. k is defined as:

$$\sum_{i=-\frac{N}{2}}^{\frac{N}{2}} \sum_{j=-\frac{M}{2}}^{\frac{M}{2}} K(i, j). \quad (2.2)$$

The linear convolution filters can be divided into multiple classes based on the results of their usage. For example, there are smoothing and sharpening filters.

Smoothing filters are also called low-pass filters, because the application of these filters conserves the low frequency components, and decreases the value of the high frequency components. Smoothing filters tend to blur the images. The size of the kernel affects the impact of the blurring. Because of the blurring, they are rarely used in practice solely. They are often used as the basis of nonlinear filters [7].

Examples for the smoothing filters are box filter (mean filter) and Gaussian filter. By using box filter, the value of a pixel will be the average of the pixel values in its neighborhood. This has the effect of normalizing pixels with extreme values. The volume of the smoothing is increased when the size of the kernel is increased. With larger kernel sizes, the result will be more dependent on pixels further from the current pixel. A mean filter with size 3×3 is defined as:

$$K = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.3)$$

Gaussian filter modifies the image by convolution with a Gaussian function. A two-dimensional Gaussian kernel is defined as the following:

$$K(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right), \quad (2.4)$$

where σ is the standard deviation. A Gaussian kernel can be described by its size and the its standard deviation.

To enhance edges on the image, or adjust contrast and shade characteristics, sharpening filters are used. For example, a simple high-pass filter can be defined as the following:

$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (2.5)$$

Notice that the kernel coefficient is positive in the center of the kernel, and the peripheral coefficients are negative.

A typical sharpening filter is the Laplacian of Gaussian (LoG) filter, which is the combination of a Laplacian and a Gaussian filter. This filter is determined by parameter σ (standard deviation) and the size of the kernel. The kernel is defined as:

$$K(i, j) = \frac{1}{\pi\sigma^4} \left(1 - \frac{i^2 + j^2}{2\sigma^2} e^{-(i^2 + j^2)/2\sigma^2}\right) \quad (2.6)$$

2.2 Nonlinear filters

The problem with linear smoothing filters is that they tend to blur edges and other image details. Also, they perform poorly when non-Gaussian is applied to the image. However, nonlinear filters have the possibility to preserve edges and cope well with impulsive noise [15, 7, 8]. By definition, any filter, which does not meet the linearity criteria, is called nonlinear filter.

The simplest example for nonlinear filters is the median filter. This filter has the capability to filter only the outliers, so it is a good choice in the case of salt and pepper noise (for further information on salt and pepper noise, see 3.2). In the simplest case, the value of a given pixel is equal to the median of its neighborhood. Another possible implementation is that the pixel value will only be equal to the median when the difference between original pixel value and the median is greater than a certain threshold. This threshold value depends on the local standard deviation within the examined neighborhood. There exist variations of the standard median filter. It can be made recursive by feeding back some of the last output values. Another variation is the weighted median filter. In the case of positive weights, the value is repeated, increasing the probability of its selection. Negative weights are also possible, in that case, the sign of the sample value is inverted.

Amongst the more complex nonlinear filter, there is the Wiener filter, nonlinear diffusion, the extended Kalman filter (the original Kalman filter is linear) and the unscented Kalman filter [5, 26].

2.3 Anisotropic diffusion

In image processing, the removal of a noise from an image while conserving the significant informations on the images (edges, lines, or other details) is a hard task. Anisotropic diffusion was designed to address this problem [31, 22, 10].

Diffusion is a physical process for balancing changes in concentration. In the case of an image, the intensity values can be considered similar to the concept of concentration, and the noise can be seen as concentration inhomogeneities. So the goal in image denoising is to smooth these inhomogeneities by using diffusion.

According to Fick's first law,

$$j = -D\nabla u, \quad (2.7)$$

where u is a concentration distribution, j is the diffusion flux, which measures the amount of substance which flows through a certain area during a certain time interval, and D is the diffusion tensor or the diffusion coefficient, which is a positive definite symmetric matrix.

If the diffusion tensor is constant over the whole image, the diffusion is called linear or homogeneous. If it is not constant, for example it is the function of the structure of the image, then it is called nonlinear or inhomogeneous diffusion. The isotropicity is defined by the relationship of the diffusion flux j and the concentration gradient ∇u . If the diffusion flux is parallel to the concentration gradient, then the diffusion is called isotropic. If j and ∇u are not parallel, then the diffusion is anisotropic.

Linear isotropic diffusion can be used to smooth images. However, this kind of diffusion also blurs the important image characteristics. Nonlinear isotropic diffusion has the possibility to reduce smoothing on the edges, but the diffusion on the edges can not be completely eliminated.

Nonlinear anisotropic diffusion means that its diffusion tensor is not constant (nonlinear), and the flux and the concentration gradient are not parallel. The diffusion tensor D has two eigenvectors and two eigenvalues, since it is a positive definite symmetric matrix. These eigenvectors give the direction of the diffusion, and the corresponding eigenvalues determine the strength of the diffusion in the direction determined by the eigenvectors. So the D matrix should be defined in a way that one of the eigenvectors are parallel to the gradient, and its eigenvalue prevents the diffusion. The other eigenvector should be perpendicular to the gradient, and its eigenvalue should enable the diffusion.

2.4 Active contour models

Active contour models do not represent an image denoising method, but a shape recognition one [17, 4, 3, 13]. Active contour models, or snakes are energy minimizing splines, their energies depend on their shape and their location within the image. To achieve the minima, the snakes are guided by external constraint forces and image forces that pull it toward specific image features, like edges and lines. Active contour model is a special case of matching a deformable model to an image by using energy minimization techniques.

A snake is parametrically defined as

$$v(s) = (x(s), y(s)), \quad (2.8)$$

where $x(s)$, $y(s)$ are the x , y coordinates of the contour, and $s \in [0, 1]$. The internal energy is minimal when the snake has a shape with characteristics most similar to the sought object. This internal spline energy can be written as:

$$E_{int} = \alpha(s) \left| \frac{dv}{ds} \right|^2 + \beta(s) \left| \frac{d^2v}{ds^2} \right|^2, \quad (2.9)$$

where $\alpha(s)$ specifies the elasticity, and $\beta(s)$ specifies the stiffness of the snake. The second term, which represents the external forces, comes from the image data. For example,

if the goal is to attract the snake to lines, edges, and terminations, then the term will be a weighted combination of three different energy terms:

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term}. \quad (2.10)$$

The line based function can be defined as

$$E_{line} = f(y, x). \quad (2.11)$$

The purpose of the edge-based function is to attract the snake to contours with large image gradients, so

$$E_{edge} = -|\nabla f(x, y)|^2. \quad (2.12)$$

The term defined by line terminators and corners are the following:

$$E_{term} = \frac{\partial \psi}{\partial n_R}. \quad (2.13)$$

The snake's behaviour can be controlled by adjusting the weights.

The third term, E_{con} comes from external constraints defined by the user or some higher level process.

Based on this terms, the full energy function is the following:

$$\begin{aligned} E_{snake}^* &= \int_0^1 E_{snake}(v(s))ds \\ &= \int_0^1 (E_{int}(v(s)) + E_{image}(v(s)) - E_{con}(v(s)))ds \end{aligned} \quad (2.14)$$

Active contour models can be used for a wide variety of problems, for example detection of edges, lines and contours, image segmentation and understanding, motion tracking, stereo matching, shape recognition.

Chapter 3

Noise

Noise is basically any undesirable data, which has no meaning. It is the data that is not used in the transmission of a signal, but it is the unwanted product of some other activities affecting the transmission. In case of images, the noise is the random variation of brightness or color information, usually caused by the sensor and the circuitry of an image recording device. In the following parts, some frequently used types of noises will be discussed. The different types of noises will be demonstrated on the 8 bit grayscale image shown in Figure 3.1.



Figure 3.1: A noise-free image

3.1 Gaussian noise

Gaussian noise is frequently used for modelling thermal noise (also called Johnson-Nyquist noise). A statistical noise is Gaussian noise, if its probability density function equals to the probability density function of a normal distribution (normal distribution is also called Gaussian distribution). For defining a Gaussian noise, two parameters have to be determined; the mean, denoted by μ , and the variance, denoted by σ^2 .

The density function of an univariate Gaussian noise q is the following:

$$P_q(x) = (2\pi)^{-1/2} e^{-(x - \mu)^2 / 2\sigma^2}, \quad (3.1)$$

for $-\infty < x < \infty$. In this case, the support (the range of values of x where the probability density is not zero) is infinite in both directions (positive and negative). However, in the

case of an image, the values must have an upper and a lower bound. So in practice, the noise used here is not strictly Gaussian, it has to be truncated to have the values between the upper and lower bounds. For grayscale images, if the intensities of the pixels are represented on one byte each, and the values are unsigned, then the lower bound must be 0, and the upper bound must be 255.

An example of the Gaussian noise can be seen in figure 3.2. The noise on this image is very light, the variation of the noise is only 25.



Figure 3.2: An image with light Gaussian noise

3.2 Salt and pepper noise

In the case of salt and pepper noise, a relatively small amount of pixels are noisy, but they have very heavy noise. Salt and pepper noise appears as randomly occurring white and black pixels (white represents salt, black represents pepper; hence the name of the salt and pepper noise). It can be defined in the following way: let $f(x, y)$ be the original grayscale image, and $g(x, y)$ be the image altered by salt and pepper noise. Then,

$$P(g = f) = 1 - \alpha \quad (3.2)$$

$$P(g = MAX) = \alpha/2 \quad (3.3)$$

$$P(g = MIN) = \alpha/2, \quad (3.4)$$

where MAX and MIN are the upper and lower bound of the possible values of pixel intensities. In the case these values are represented on one byte each and the values are unsigned, then the minimum value is 0, and the maximum value is 255.

The probability of leaving a pixel unchanged is $1 - \alpha$, and the probability of altering the pixel values to the minimum or the maximum equals α ($\alpha/2$ for changing to the minimum and $\alpha/2$ for changing to the maximum).

Salt and pepper noise can be easily removed with various filters, for example contraharmonic mean filter, median filter and LUM (lower-upper-middle) filter [14]. An example image demonstrating the salt and pepper noise can be seen in figure 3.3.



Figure 3.3: An image with salt and pepper noise

3.3 Quantization (uniform) noise

Quantization noise can appear in two cases; when a continuous random variable is converted to a discrete random variable, or when a discrete random variable is converted to another discrete random variable with fewer levels. In the case of images, this kind of quantization noise usually occurs in the image acquisition process, since for further processing, the image needs to be digitalized. Quantization noise is usually modeled as uniform noise [32].

Let a be a continuous random variable, and let b be a discrete random variable represented on β bits in a way that b is the quantization of a , so $b = Q(a) = a + q$, where $-\Delta/2 \leq q \leq \Delta/2$ is the quantization noise. Δ is the difference between two adjacent quantization levels of b . When Δ is small and the number of quantization levels is large, q can be modeled as a uniform random variable between $-\Delta/2$ and $\Delta/2$, and it is independent of the continuous random variable a . The mean of q is

$$\mu = \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} s ds = 0, \quad (3.5)$$

and the variance of q is

$$\sigma^2 = \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} s^2 ds = \Delta^2/12. \quad (3.6)$$

An image representing the quantization errors can be seen in figure 3.4. This image is quantized to 4 bits, meaning that a pixel can have 16 different values, while the original image shown in figure 7.1 is quantized to 8 bits, so the pixels can have 256 different graylevel values considering that image.

3.4 Noise measurement

For the quantitative comparison of noise reduction methods, some measure of noise is needed to describe the efficiency of the algorithms. The most commonly used measures are signal-to-noise ratio (SNR), peak signal-to-noise ratio (PSNR), and mean squared error (MSE) [20].



Figure 3.4: An image demonstrating quantization errors

Signal-to-noise ratio is the ratio between the power of the signal and the noise:

$$SNR = \frac{P_{signal}}{P_{noise}}. \quad (3.7)$$

Signal-to-noise ratios are often expressed in the following logarithmic form:

$$SNR_{db} = 10 \log_{10} \frac{P_{signal}}{P_{noise}}. \quad (3.8)$$

Alternatively, SNR can be defined as:

$$SNR = \frac{\mu}{\sigma}, \quad (3.9)$$

where μ is the expected value of the signal, and σ is the standard deviation of the noise. This definition is commonly used in the field of image processing [27, 11]. In this case, the SNR is calculated as the ratio of the average value and the standard deviation of the values of the pixels in a given area.

Let f be a noise-free grayscale image, and let g be the noisy version of f . Both images share the same dimensions. The height of the images is denoted by N , and the width is denoted by M . The value of the pixel intensity in the i th row and j th column is denoted by $f(i, j)$, where $0 \leq i < N$ and $0 \leq j < M$. Using these notations, the MSE between the two images can be defined as the following:

$$MSE = \frac{1}{NM} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (f(i, j) - g(i, j))^2. \quad (3.10)$$

The peak signal-to-noise ratio is another measure of noise corruption. It is usually represented on the logarithmic scale, and defined with the use of MSE in the following form:

$$PSNR = 10 \log_{10} \frac{MAX^2}{MSE}, \quad (3.11)$$

where MAX is the maximum possible intensity value of the image.

Chapter 4

The reconstruction models

4.1 The MRF model

Markov Random Field (MRF) is a probability model based on local characteristics [9, 18]. In an MRF, the sites are related to one another via a neighborhood system.

A generalized definition of Markov Random Field can be given using graphs. Let $G = (S, E)$ be an undirected graph where $S = s_1, s_2, \dots, s_N$ is a finite set of vertices (sites) of the graph, and $E \subset S \times S$ is the set of edges of the graph, where an (s_i, s_j) pair represents an edge. By definition, two sites of the graph, s_i and s_j are neighbors if there is an edge connecting them, meaning that an (s_i, s_j) pair is the element of the E set. Given a site s , the set of points which are neighbors of s (the neighborhood of s) is denoted by V_s . By definition, $V = \{V_s | s \in S\}$ is a neighborhood system for G if

$$s \notin V_s \quad (4.1)$$

$$s \in V_r \Leftrightarrow r \in V_s, \quad (4.2)$$

meaning that a site can not be part of its own neighborhood, and if a site s is an element of the neighborhood of a site r , then the site r must be an element of the neighborhood of site s .

We assign to each site of the graph a label λ from a finite set of labels Λ . Such an assignment is called a configuration, denoted by ω . The set of all possible configurations is denoted by Ω . The configuration restricted to a subset $T \subset S$ is denoted by ω_T . The value given to a site s by the configuration ω is represented by ω_s . We assign probability measures to the set Ω of all possible configurations ω . The local characteristics of a probability measure P defined on the set Ω of all possible configurations are the conditional probabilities of the form $P(\omega_s | \omega_{S-s})$, that is, the probability that the site s is assigned the label ω_s , given the values at all other sites of the graph. By definition, a probability measure χ is a Markov Random Field with respect to a neighborhood system V if

$$\forall \omega \in \Omega : P(\chi = \omega) > 0 \quad (4.3)$$

$$\begin{aligned} \forall s \in S, \forall \omega \in \Omega : \\ P(\omega_s | \omega_{S-s}) = P(\omega_s | \omega_V) \end{aligned} \quad (4.4)$$

so that the probability of any configuration must be greater than zero, and the probability of the label ω_s being assigned to a site s given that the other sites of S are assigned certain labels must be equal to the probability of the label ω_s being assigned to the site s given that its neighbors are assigned certain labels. In other words, the local characteristics of

the probability measure depend only on the knowledge of the labels at the neighboring sites.

We can define a potential V as a way to assign a number $V_A(\omega)$ to every subconfiguration ω_A of a configuration ω . Given the potential V , it defines an energy $U(\omega)$ on the set Ω of all configurations ω by

$$U(\omega) = - \sum_A V_A(\omega) \quad (4.5)$$

where for a fixed ω the sum is taken over all subsets A of S .

The Gibbs measure induced by U is defined by:

$$\pi(\omega) = \frac{1}{Z} \exp(-U(\omega)), \quad (4.6)$$

where Z is the normalizing constant or partition function:

$$Z = \sum_{\omega} \exp(-U(\omega)) \quad (4.7)$$

By definition, $C \subset S$ is a clique, if every pair of points in C are neighbors. In the case when $V_A(\omega) = 0$ whenever A is not a clique, the potential V is called a nearest neighbor Gibbs potential. The Hammersley-Clifford theorem establishes the equivalence between Gibbs measures and MRFs [12]: χ is an MRF with respect to the neighborhood system V , if and only if $\pi(\omega) = P(\chi = \omega)$ is a Gibbs distribution with a nearest neighbor Gibbs potential V , that is

$$\pi(\omega) = \frac{1}{Z} \exp(- \sum_{c \in C} V_c(\omega)), \quad (4.8)$$

where C is the set of cliques. This equivalence allows us to specify potentials instead of local characteristics when defining an MRF.

4.2 Markov Random Field in image reconstruction

Markov Random Fields can be used to address many low-level image tasks, like restoration, edge detection, segmentation, motion detection [9, 21].

Let F denote the observations on a grayscale image, and f_s denote the observation belonging to the pixel s . We have to find the configuration ω which maximizes the probability $P(\omega|F)$. Bayes' theorem states that:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \quad (4.9)$$

Applying Bayes' theorem, it is possible to formulate $P(\omega|F)$ as

$$P(\omega|F) = \frac{1}{P(F)} P(F|\omega) P(\omega). \quad (4.10)$$

Since $P(F)$ is constant, the function have to be minimized is

$$P(\omega|f) \propto P(F|\omega) P(\omega). \quad (4.11)$$

The term $P(F|\omega)$ equals the product of the probabilities of the observation f_s belonging to a certain site given the label ω_s is assigned to the site:

$$P(F|\omega) = \prod_{s \in S} P(f_s|\omega_s). \quad (4.12)$$

The term $P(\omega)$ can be written according to the Hammersley-Clifford theorem as:

$$P(\omega) = \prod_{c \in C} \exp(-V_c(\omega_c)), \quad (4.13)$$

where C is the set of all possible cliques, and V is the clique potential. So we have to find the configuration which maximizes the function

$$\prod_{s \in S} P(f_s|\omega_s) \prod_{c \in C} \exp(-V_c(\omega_c)). \quad (4.14)$$

Making the assumption that $P(f_s|\omega_s)$ is Gaussian, we get the following energy functions:

$$U_1(\omega, F) = \sum_{s \in S} \left(\ln \sqrt{2\pi} \sigma_{\omega_s} + \frac{(f_s - \mu_{\omega_s})^2}{2\sigma_{\omega_s}^2} \right) \quad (4.15)$$

which stands for the observation, and

$$U_2(\omega) = \sum_{c \in C} V_2(\omega_c) \quad (4.16)$$

$$V_2(\omega_c) = V_{s,r}(\omega_s, \omega_r) = \begin{cases} -\beta & \text{if } \omega_s = \omega_r \\ +\beta & \text{if } \omega_s \neq \omega_r \end{cases}, \quad (4.17)$$

which is the sum of the clique potentials. In equation 4.17, β is a model parameter representing the homogeneity of the regions, μ_{ω_s} and σ_{ω_s} is the mean and the deviation belonging to label ω_s .

The goal is to find a configuration which minimizes the two energy functions,

$$\sum_{s \in S} V_1(\omega_s, f_s) + \sum_{c \in C} V_2(\omega_c). \quad (4.18)$$

For each label $\lambda \in \Lambda$, two parameters are assigned. One is the mean value of the corresponding label, denoted by μ_λ , and the other is the deviation, denoted by ω_λ . Based on the knowledge of the parameters assigned to the labels, two kind of segmentation can be defined. If the parameters are known, then the segmentation process is called supervised. If they are unknown, and they have to be determined during the segmentation process, then it is called an unsupervised segmentation. In this paper, the parameters are considered to be known, so the segmentation presented here is supervised. The parameters must be determined by the user and hand over to the reconstruction process. The parameters can be determined by measuring the mean and the variance in a homogeneous region which should be assigned the certain label.

4.3 Radon transform

Radon transformation is widely used in the field of tomography [25, 24, 23]. The projection of an object at a given angle θ is made up of a set of line integrals. These line integrals are the Radon transform of the object. The inverse Radon transform can be used to reconstruct an approximation of the original object. If we had an infinite number of projections of an object taken at an infinite number of angles, we could perfectly reconstruct the original object.

Let $f(x, y)$ be a two dimensional continuous function. The Radon transform is a function defined on the two dimensional space of straight lines L by the line integral along each line:

$$Rf(L) = \int_L f(x)dx \quad (4.19)$$

A straight line L can be represented in the form:

$$(\rho \cos \theta - s \sin \theta, \rho \sin \theta + s \cos \theta), \quad (4.20)$$

where ρ is the distance of L from the origin, and θ is the angle of the normal vector to L with the x axis.

The Radon transform can be expressed, using these quantities as coordinates, as:

$$Rf(\rho, \theta) = \int_{-\infty}^{\infty} f(\rho \cos \theta - s \sin \theta, \rho \sin \theta + s \cos \theta) ds \quad (4.21)$$

A generalization exists for an n-dimensional Euclidean space, but since this is not needed here, we refrain from its discussion. In this paper, we use a simple form of the discrete Radon transform, where we use only two angles (the vertical and horizontal directions), which makes it easy and fast to compute.

Let $f(i, j)$ be a binarized image, where

$$f(i, j) = \begin{cases} 1 & \text{if the pixel is black} \\ 0 & \text{if the pixel is not black} \end{cases} \quad (4.22)$$

$0 \leq i < n, 0 \leq j < m$, where n is the height of the image, and m is the width of the image.

We denote the Radon transform of the j th row of the image f as $Rr_j(f)$, and the Radon transform of the i th column of the image f as $Rc_i(f)$. In the following parts, the image as parameter will be omitted, if not stated otherwise. We define them as the sum of the values of the pixel in the j th row or the i th column:

$$Rr_j = \sum_{i=0}^{n-1} f(i, j) \quad (4.23)$$

$$Rc_i = \sum_{j=0}^{m-1} f(i, j). \quad (4.24)$$

Based on this, we can define the probability of a random pixel chosen from a row or a column being black, in respect of the chosen row or column. In the case the pixel was chosen from the j th row:

$$P(r_j(i) = 1) = \frac{Rr_j}{n} \quad (4.25)$$

In the case the pixel was chosen from the i th column:

$$P(c_i(j) = 1) = \frac{Rc_i}{m} \quad (4.26)$$

In respect to the whole image, the probability is the following:

$$\begin{aligned} P(f(i, j) = 1) = \\ P(c_i(j) = 1) \cdot P(r_j(i) = 1) = \frac{Rc_i}{n} \cdot \frac{Rr_j}{m} \end{aligned} \quad (4.27)$$

4.4 Extension of the MRF model

For the MRF reconstruction with Radon transforms (we will call it in this paper extended MRF), a hypothesis about the shape is needed since we will have a new energy term to measure the similarity to a possible shape. For simplicity we can run the extended MRF reconstruction with all of the possible symbols as hypotheses, but then we should find out the right one as the reconstruction is finished.

For extending the energy function with a property to describe the similarity between the noisy image and the hypothesis, we need to add a new energy term. The original two terms can be seen at (4.15) and (4.16) and the new term is the following:

$$\begin{aligned} U_3 = \\ \gamma \cdot \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} |P(r_j(f, i) = 1) - P(r_j(g, i) = 1)| + \\ |P(c_i(f, j) = 1) - P(c_i(g, j) = 1)| \end{aligned} \quad (4.28)$$

where γ is a constant specifying the weight of this energy term, f is the noisy image, and g is the hypothesis image. Notice that here we denoted the corresponding image of the row and column as the first parameter.

Chapter 5

Energy minimization

To obtain a correct reconstruction, the energy function presented in equation 4.18 now extended with the energy term described in equation 4.28 should be minimized. This energy function is nonconvex, so classical convex optimization methods can not be used.

Since the cardinality of the set of all labels Λ is finite, the set of all possible configurations Ω is also finite, because $\Omega = \Lambda^{n*m}$, where n is the height of the image under reconstruction, and m is the width of this image, meaning that Ω is the $(n * m)$ -ary cartesian power of the set Λ , and the Cartesian powers of a finite set is also finite. So the most simple solution would be to calculate the energies of the possible configurations, and choose the configuration belonging to the minimum of the energies as the correct solution. However, in the case of an image with dimensions 64×64 , assuming that the image is binary (there are only two labels), the cardinality of the set of all possible configurations is $2^{64 \times 64} \sim 10^{1200}$, so the computation of the energies of all configurations are computationally infeasible.

The solution of the problem should be approximated with some relaxation methods. Several optimization algorithms exist for this, for example, iterated conditional modes (ICM), graph cuts, max-product loopy belief propagation (LBP) and tree-reweighted message passing (TRW) [29].

Iterated conditional modes are deterministic in nature. It is a greedy algorithm, for the next step it always choses the locally best option. First, an estimation of the result is set as the initial state. In each step, the method choses a new state in a way that there is a transition from the current state to the new state, and this new state has the lowest energy amongst the neighbors of the current state. This is repeated until convergence is reached. Using this method, convergence is guaranteed, and it is usually reached in a very few steps. However, the results are very sensitive to the selection of the initial state in the case of convex energy functions, where the function has many local minimum.

Belief propagation is a message passing algorithm. It calculates the marginal distribution for each unobserved node, conditional on any observed nodes. Originially, it was designed for tree graphs. It breaks the marginalization for a node, let's denote it with i , into independent subproblems. In each subproblem, the method eliminates all of the variables, except the a . The result will be the message. The method can be applied for a graph with loops, but in that case, it is not guaranteed to converge, and if it converges, it is not guaranteed that it converges to the correct marginals.

In this paper, simulated annealing is used [1, 19]. This method exploits an analogy between the annealing process of metals, the cooling and freezing into a crystalline structure with minimum energy, and the general methods of energy minimization. In the physical

annealing process, a solid substance is heated until it is in liquid state, and its particles are arranged randomly. After that, it is slowly cooled following a cooling schedule, which can be linear, but usually it is created in a way that the substance spends more time closer to the freezing point (for example, a logarithmic cooling schedule is used). The high temperatures allow a broader exploration of the space of possible states. The particles can adopt a state where the energy is higher than in the previous one. As the temperature decreases, the states with higher energies have less probability. In case the cooling schedule is too fast, the system may not reach thermal equilibrium, and it can result in a crystal with just locally optimal structures. However, with a correct cooling schedule, this method will not get stuck in a local optimum, and will provide the global optimum.

In general terms, simulated annealing is defined in the following way: let S be a finite set of possible states. Let's define a real-valued cost function C on S . For each element of the set S , $i \in S$, a set $S(i) \subset S - i$ is defined, and called as the set of neighbors of i . It is defined in a way that if $j \in S(i)$ then $i \in S(j)$, $\forall i, j \in S$. For every $i \in S$, a collection of positive coefficients, $q_{ij}, j \in S(i)$ are defined such that $\sum_{j \in S(i)} q_{ij} = 1$. The cooling schedule is defined as a nonincreasing function $T : N \rightarrow (0, \infty)$. The temperature at time t is denoted as $T(t)$. The initial state is denoted as $x(0) \in S$.

The algorithm of simulated annealing consists of a Markov chain $x(t)$. A Markov chain is a memoryless random process, meaning that a state depends only on the previous state, and not on the sequence of previous events. The evolution of this chain is the following: if the current state, $x(t)$ is i , then choose a state $j \in S(i)$ at random, in a way that the probability of selecting j equals q_{ij} . Once j is chosen, the new state $x(t + 1)$ determined in the following way:

$$\begin{aligned} &\text{If } C(j) \leq C(i), \text{ then } x(t + 1) = j, \\ &\text{If } C(j) > C(i), \text{ then } \begin{cases} x(t + 1) = j \text{ with probability } \exp(-\frac{C(j)-C(i)}{T(t)}) \\ x(t + 1) = i \text{ otherwise.} \end{cases} \end{aligned} \quad (5.1)$$

So the probability of the transition to the state j from the state i , where $j \neq i$ and $j \in S(i)$, is the following:

$$P(x(t + 1) = j | x(t) = i) = q_{ij} \exp(-\frac{1}{T(t)} \max(0, C(j) - C(i))). \quad (5.2)$$

If $j \notin S(i)$, then

$$P(x(t + 1) = j | x(t) = i) = 0. \quad (5.3)$$

Based on the general model, the definition of simulated annealing can be described in the context of an MRF reconstruction as the following. The set of all possible states S now equals to the set of all possible configurations, denoted by Ω . The cost function C is the energy function presented in equation 4.18, now it is denoted by E . The neighborhood relation is defined as the following: two configurations are neighbors, if there is exactly one label in the two configurations which is different. For each $\omega \in \Omega$, all of the $q_{\omega\eta}$ coefficients are equal, where $\eta \in \Omega$ is in the neighborhood of ω ; meaning that the transitions from a configuration to its neighbors have the same probabilities.

The algorithm is the following. First, a random configuration is chosen as the initial configuration. The temperature is set to T_0 , which is the starting temperature given as a parameter. Now a neighbor of the configuration is chosen, let's denote it with η . The difference of the energies of the two configurations is computed, $\Delta U = U(\eta) - U(\omega)$.

The new configuration is accepted in two cases; if the transition results the decreasing (or stagnating) of the energy, meaning that $\Delta U \leq 0$, or if $\xi < \exp(-\frac{\Delta U}{T})$, where ξ is a uniform random number from $[0, 1)$. Otherwise, the configuration will not change. Then the temperature is decreased in a predefined manner according to the cooling schedule, and a new configuration is chosen again. The algorithm will stop, if the percentage of refused transitions exceeds the refuse threshold, which is also a parameter.

This method is called inhomogeneous, since the temperature is decreased after each transition. If the transition would only be decreased if the system reached an equilibrium, it would be called homogeneous annealing.

An example MRF reconstruction process using simulated annealing can be seen in figure 5.2. The original noise-free image and the image with Gaussian noise can be seen in figure 5.1. Notice, that the noise here is lighter then in the following chapters, hence the better results. In the first image, the pixels has labels which are seemingly random, and it is definitely worse, than the noisy image. It is because of the high temperature, configurations with high energies can be accepted. However, as the temperature decreases, the configuration slowly converges to a minima.



Figure 5.1: A noise-free and a noisy image of the character 'd'.

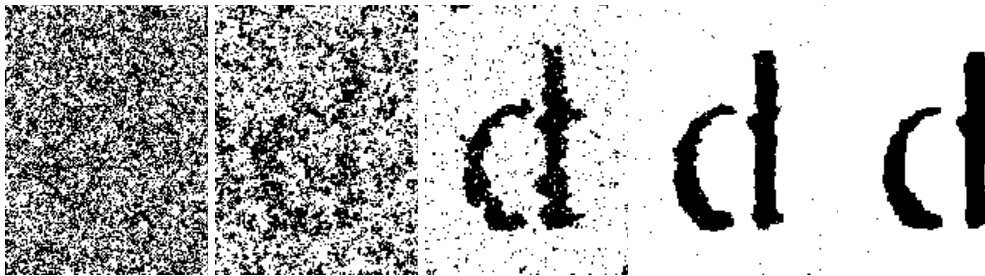


Figure 5.2: An example of the MRF reconstruction process with simulated annealing.

Chapter 6

Implementation

6.1 MRF reconstruction

The application for the MRF reconstruction algorithm and its extension was implemented in C++, using the Microsoft Visual Studio 2010 IDE (integrated development environment).

Among the main goals were to implement the algorithm in a flexible way, so that further extensions or changes can be added relatively easy. For example, if during further development of the reconstruction model, it is decided, that another descriptor should be used instead of the Radon transforms currently in use, or an additional one should be used, then it has to be simple to make the modifications to add a new energy term to the energy function. The considerations are similar in the case of the relaxation method; if a new algorithm should be used, for example iterated conditional modes, or the Gibbs sampler, then it has to be easy to modify the program to satisfy this condition.

Another main goal was to avoid unnecessary computations. Most of the image processing algorithms are computationally heavy due to the large number of data it has to work on, and the MRF reconstruction is not an exception. So attention was paid not to make redundant calculations. Also, there are differences between programming languages in term of the speed of the compiled programs, and that is one of the reason C++ was chosen, because it is thought to be relatively fast. Of course, programming languages and compilers exist that can produce even faster result. An exaggerated example is the assembly language. However, writing an assembly code for this application is a time consuming and difficult task. So C++ was chosen, because it represents a fine balance between speed of the compiled program and the difficulty of programming.

The application contains four header files (`radon.h`, `containers.h`, `energyfunction.h` and `annealing.h`) and their corresponding `.cpp` files, and the `main.cpp`. The roles of the files in brief are the following: the `main.cpp` starts the annealing using the class in the `annealing.h`, which contains the energy functions found in `energyfunction.h`. The file `radon.h` contains the class representing a Radon transform of the image, which is used in the Radon-based energy function. In `containers.h`, there are two containers, one for the labels which are assigned to the pixels during the reconstruction, and one for the values of the observation energy function, for the purpose of speeding up the application.

A simplified class diagram can be seen in figure 6.1.

The Relaxation class is an abstract class, its Run function is pure virtual, so the class can not be instantiated. The purpose of this is that one can define multiple relaxation methods as child classes of the Relaxation class, override the Run function in each, and

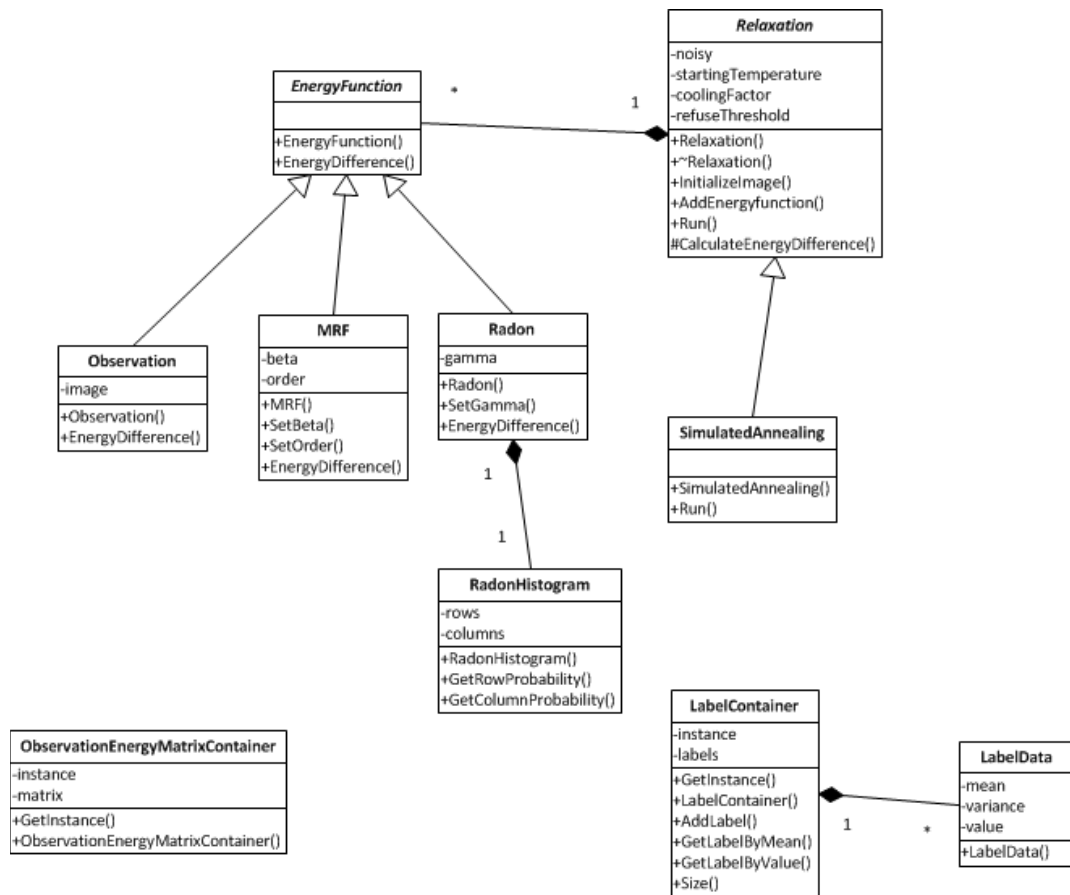


Figure 6.1: Class diagram of the reconstruction application.

then the interface of the multiple classes representing the relaxation methods will be the same. This way of implementation provides flexibility and reusability. It has a constructor that initializes the the member `noisy` from its parameter. This member represents the image under reconstruction during each step of the process. The class also contains the parameters of the relaxation process, the starting temperature, the cooling factor, and the refuse threshold. There are also functions to set these members (these are not displayed in the class diagram). The task of the `InitializeImage` function is the construct an initial configuration. Currently it assigns a label to a certain pixel based on the distance of the pixel's value and the mean of the labels. The label which has its mean the closest to the value of the pixel will be assigned to it.

The class contains a vector of pointers to the `EnergyFunction` class. A new energy function can be added to this vector through the `AddEnergyFunction` method. In each step of the reconstruction, only the energy difference between the current configuration and the possible next configuration needs to be calculated, not the whole energy of the two configurations, and computing only the difference, a lot of computational time can be saved. The function `CalculateEnergyDifference` sums up the energy differences by calling the `EnergyDifference` function of the `EnergyFunction` classes in the vector.

The `SimulatedAnnealing` class is the child class of the `Relaxation` class. The constructor's role is to call the constructor of the base class. The class also contains the mandatory override of the `Run` function from the base class. The implementation basically follows

the algorithm of the simulated annealing described in 5, but some changes were made. The temperature is set to the starting temperature. In each iteration, every pixel has the opportunity to change label. For each pixel of the image, a new label is generated, the energy difference is calculated, and if it satisfies the constraints described in the algorithm of simulated annealing, it is accepted. After iterating through each pixel, the temperature is decreased. Notice that using the original algorithm, the temperature would be decreased after examining each pixel. The algorithm will stop, if the sum of energy changes falls below the refuse threshold.

The EnergyFunction class, similarly to the Relaxation class, is also abstract. Its pure virtual function, EnergyDifference, has a float return value. It should return the difference of energies between two configuration for the corresponding energy function. The parameters are the image, on which the difference should be calculated, the x and y coordinates of the pixel subject to the label change, and the new label that is assigned to the pixel.

The first implemented child class of EnergyFunction is the Observation class. It refers to the energy based on the difference of the pixel's value and the mean belonging to the label. For exact definition, see equation 4.15. Since this energy does not depend on the current configuration, it can be computed before the reconstruction, speeding up the process. The class ObservationEnergyMatrixContainer contains these energies, which are calculated prior to the reconstruction. This class is implemented as a singleton class. It has a matrix for each label, where the element of the matrix at the position x, y corresponds to the observation energy of the pixel of the image at position x, y assuming that the actual label is assigned to it.

One of the other implemented energy function is the MRF. It has a member called beta which can be set through the function SetBeta. Also, the order of the neighborhood system can be also set. The possible values are 1 and 2. Basically, if the order is 1, then only the vertically and horizontally adjacent pixels are taken into account. If the order is 2, then the diagonally adjacent pixels are also subject to the calculation of the energy. In this class the EnergyDifference function is also overridden. This method calculates the difference in energy corresponding to equation 4.16.

The third implemented subclass of EnergyFunction is Radon. Its member, gamma can be set with the use of the SetGamma function. It calculates the energy based on equation 4.28. It also contains a RadonHistogram class, which is the representation of the 2 dimensional discrete row and column Radon transform. These class is used for calculating the energy difference. The constructor calculates the transforms of the image passed as a parameter. Then the probability of an arbitrary row or column containing a black pixel can be acquired with the GetRowProbability and the GetColumnProbability functions.

The singleton class LabelContainer contains the array of the data of the labels. A new label can be added to this array by the AddLabel function, and the size of the array can be retrieved using Size. The labels can be retrieved in two ways, GetLabelByMean uses the mean as a key, and GetLabelByValue uses the value. The LabelData class, which is contained in the LabelContainer, has three members: mean, variance and value. The first two are the parameters of the labels discussed in section 4.2. The third is used just for display; the label in the image will be denoted by this intensity value.

To run the program, a text file containing the parameters of the reconstruction should be passed as a command line argument. The structure of this file must be the following. The first three lines must contain the directories including the images. The first one is the folder with the original noise-free images, which will be used as templates for the

reconstruction (only the Radon transforms of these images will be used). The second one is the folder containing the noisy images which should be reconstructed. There must be images from a to z and 0 to 9 in both folders, and they must be named with a single character. The third folder is the one where the results should be saved. This folder must exist before running the reconstruction.

After these, the next two lines must consist of the descriptions of the two labels. The first one must be black label, and the second must be the white. The description of the label consist of two parameters, the mean and the variance belonging to that label.

In the next three lines, the parameters of the annealing must be given. The first line is the starting temperature, the second is the cooling factor, and the third is the refuse threshold.

The last two lines consist of β , the parameter for the energy term in equation 4.16, and γ , the parameter for the energy term using the Radon transforms, which can be seen in equation 4.28.

6.2 Noise generation

The application for generating noise images was implemented in C++, using the Microsoft Visual Studio 2010 IDE (integrated development environment).

For reconstruction noisy images, first noisy images should be acquired. The most simple way to generate these images systematically by a program.

The NoiseGenerator program generates images with Gaussian noise. The mean and the variance of the noise can be set. For generating normally distributed random numbers, the Box-Muller transform was used.

The program takes three parameters as command line arguments. The first is the path of the input image, the second is the path of the output image, and the third is the variance of the Gaussian noise.

6.3 Classification

Another program were used for the classification of the results. For this, three folders are needed, one for the hypotheses from a to z and 0 to 9, one for the noisy images from a to z and 0 to 9, and one for the reconstructions. The application takes a set of images belonging to a noisy observation, where each image was reconstructed with a different hypothesis. Then the images are compared to their hypotheses by an error function. The one with the smallest error compared to its hypothesis will be recognized as the correct one.

The function RadonDistance calculates the distance measure based on the the third energy term described in equation 4.28. It calculates the probabilities for a pixel being black in the rows and in the columns for both image, uses these datas as vectors, and calculates the Euclidean distances of the vectors.

Chapter 7

Results

7.1 Reconstruction of characters

The reconstruction methods were run on three different types of fonts, with 36 characters in each (the digits from 0 to 9, and the alphabet from a to z). Figure 7.1 shows a sample from the noise-free images of the different fonts.

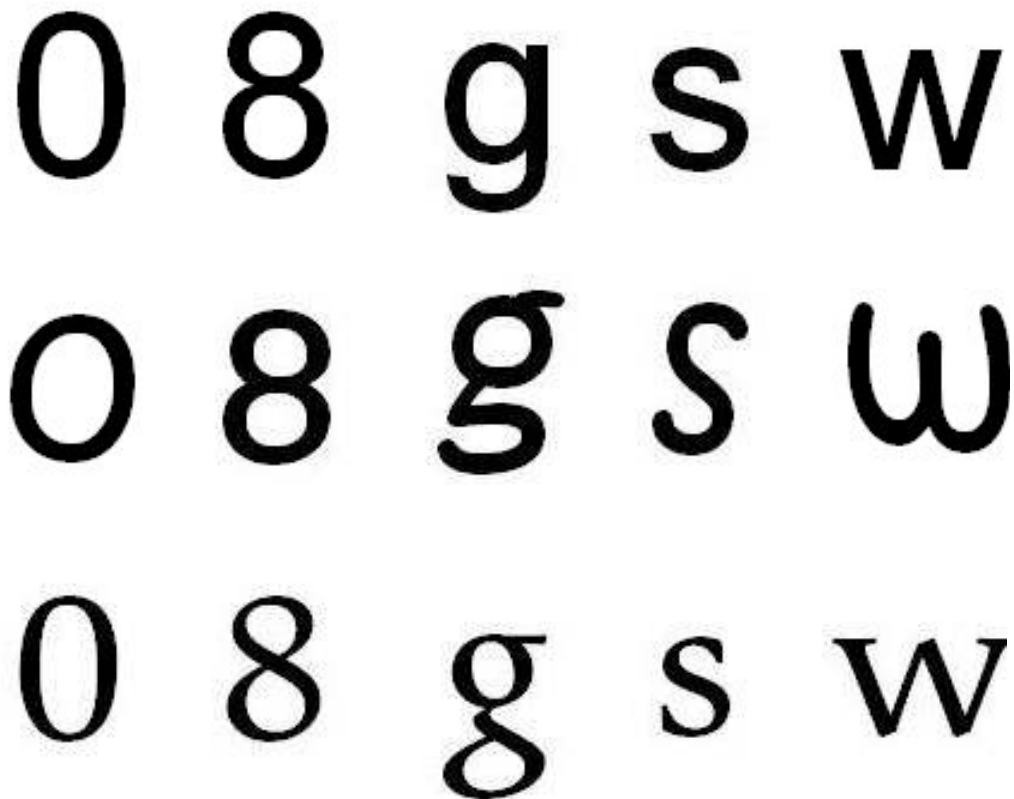


Figure 7.1: A sample of the three types of noise-free characters (arial, kristen, pescadero).

To the images heavy Gaussian noise was added to the original 8 bit grayscale images (with the program presented in section 6.1), with the mean being 0, and the variance being 400. Since the values of the pixel intensities have upper and lower bounds, the

values generated with the addition of noise have to be truncated, so the resulting noise is not strictly Gaussian. Figure 7.2 shows images from the three different fonts, now with added noise.

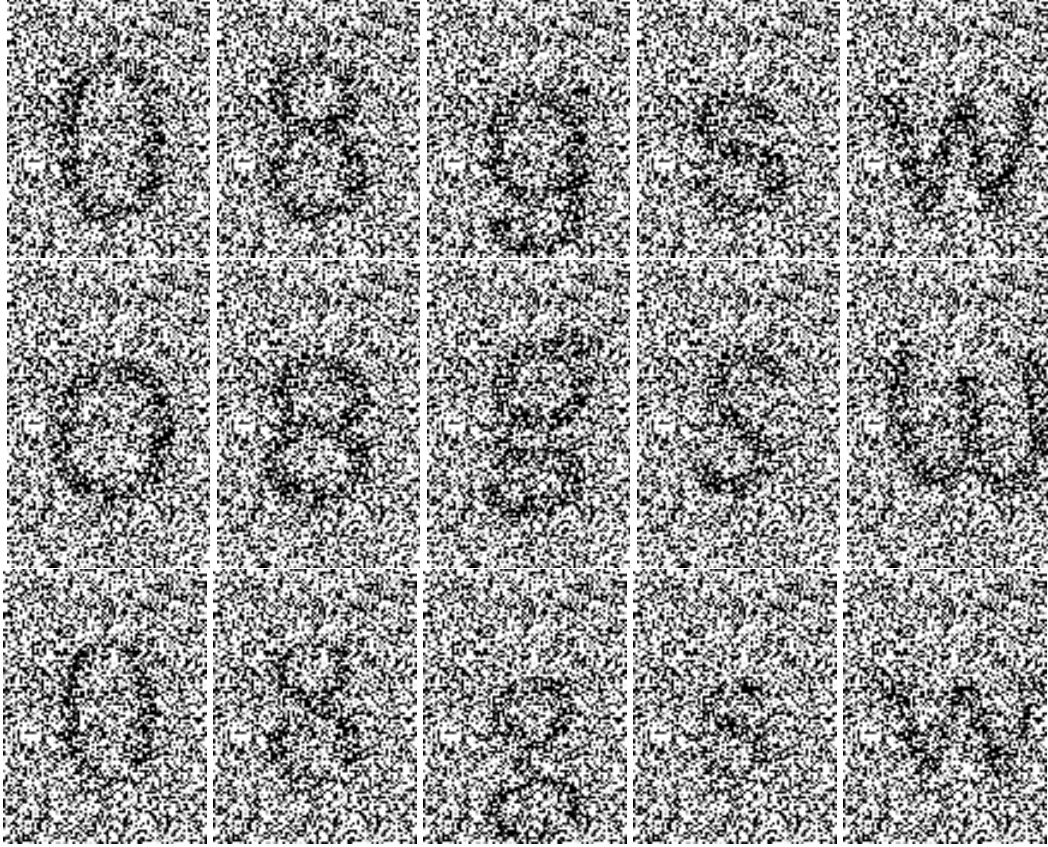


Figure 7.2: A sample of the three types of characters (arial, kristen, pescadero) with noise.

For the extended MRF reconstruction, a hypothesis is needed, and by choosing the correct hypothesis, classification of the characters can be done. Since for the correct reconstruction, a correct hypothesis is needed, because the Radon transforms of the hypothesis are used as a comparison for the third energy term, which is described in equation 4.28. So the selection of the hypothesis presents itself as an important task.

A trivial solution would be to recognize the noisy images using their Radon transforms. The row and column Radon transforms can be seen as vectors, and the Euclidean distance of these vectors and the row and column Radon transforms of the hypotheses can be used as a measure to find the closest match to the noisy image. This method is similar to the formulation of the third energy term, equation 4.28. However, due to the heavy noise, the Radon transforms can not provide a method for proper recognition in this case.

Figures 7.3-7.4. show the Radon transforms of two characters with and without noise. The noise-free versions can be easily distinguished. However, the images with the added noise do not have any unique characteristics, they are very similar to each other. The noisy images can not be recognized by simply using the distance of the Radon transforms. This is supported by test, where only 1 of the 36 character was correctly recognized.

The method used in this paper is to reconstruct a noisy image using all of the possible hypotheses (it may be possible to reduce the set of all hypotheses, but this does not

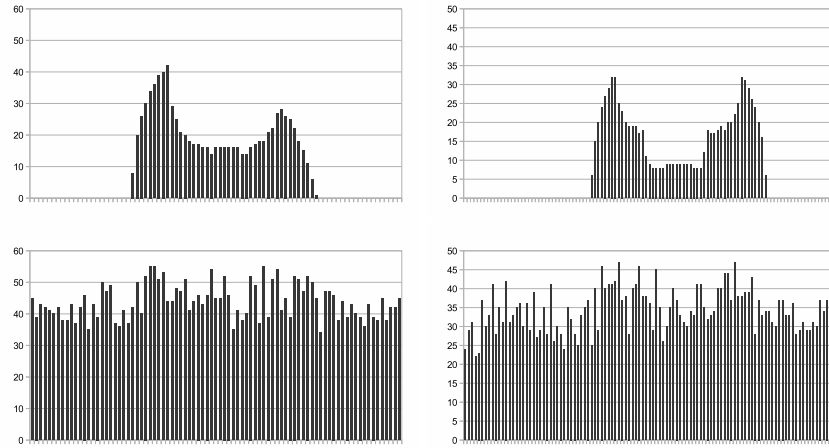


Figure 7.3: The vertical and the horizontal Radon transforms of the noise-free and noisy image of the character 'c'.

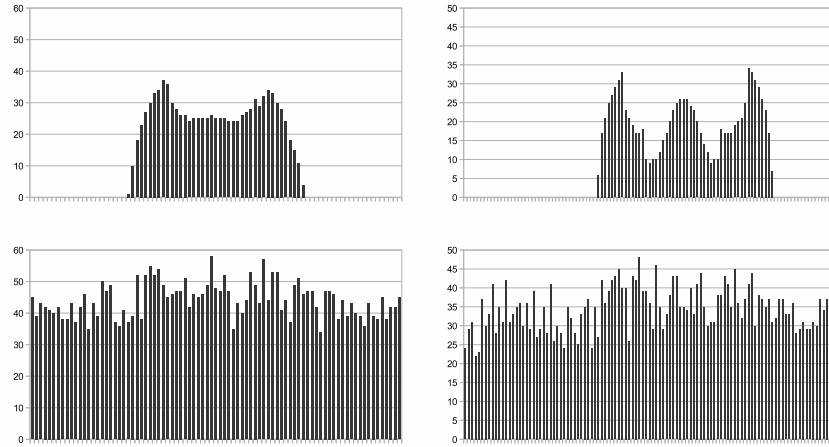


Figure 7.4: The vertical and the horizontal Radon transforms of the noise-free and noisy image of the character 's'.

change the main concept, since the one correct hypothesis still can not be chosen). After the reconstruction took place, the results need to be compared to their hypotheses, and the one with the smallest distance by some measure will be chosen as the valid reconstruction. The idea is to choose the Radon functions' similarity, which was already used during the reconstruction process, and let this energy term decide the correct hypothesis. The one with the minimal third energy term will be chosen as the right one.

Figures 7.5-7.7. show samples of the images reconstructed with MRF, and also show samples of the images reconstructed with extended MRF. The parameters of the MRF and extended MRF are the same for each font (but it can be different for different fonts).

Table 7.1. shows the numerical comparison of the results produced by the two reconstruction methods (MRF and MRF with Radon transforms). The displayed numbers are the differences of the percentage of wrong pixels after the reconstruction by MRF with Radon transforms and MRF in respect to the area of the character on the original image. As can be seen there are only 4 negative numbers that is less than 4 percent of

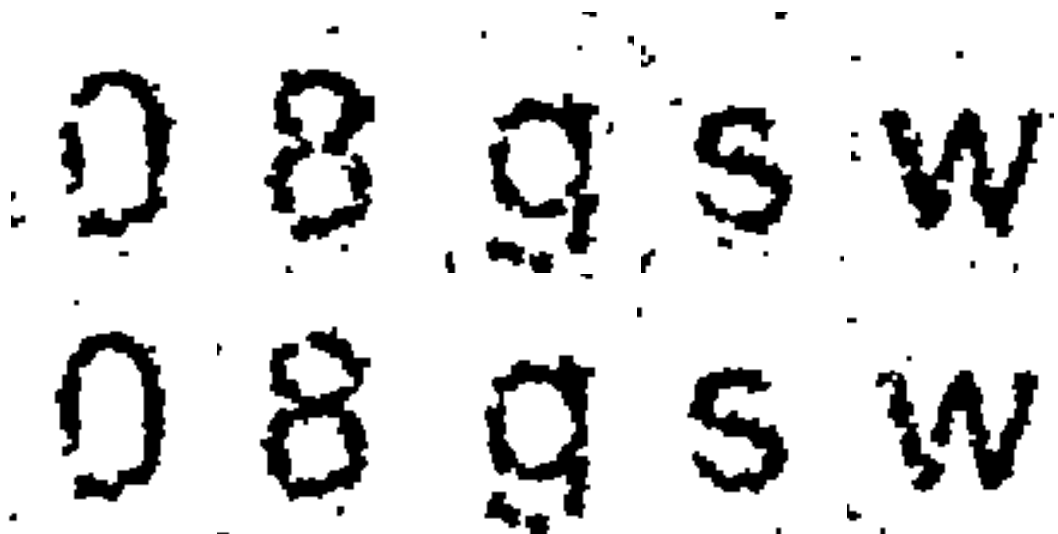


Figure 7.5: A sample of arial characters reconstructed with MRF and extended MRF.

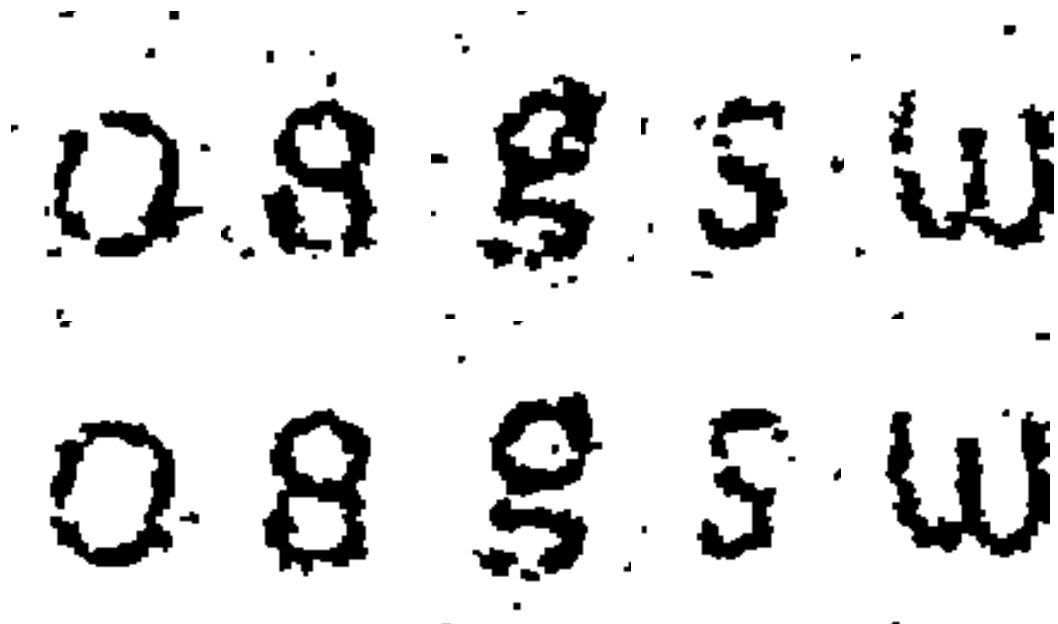


Figure 7.6: A sample of kristen characters reconstructed with MRF and extended MRF.

the characters were reconstructed worse than by the normal method. (Since in our experiments 3 characters were misclassified in three cells of the table we give the reconstruction performance also with the wrongly selected hypotheses.)

Table 7.2. shows the percentage of successful classification for each font using the proposed method. As can be seen, only 3 characters were mistaken in the case of one font.

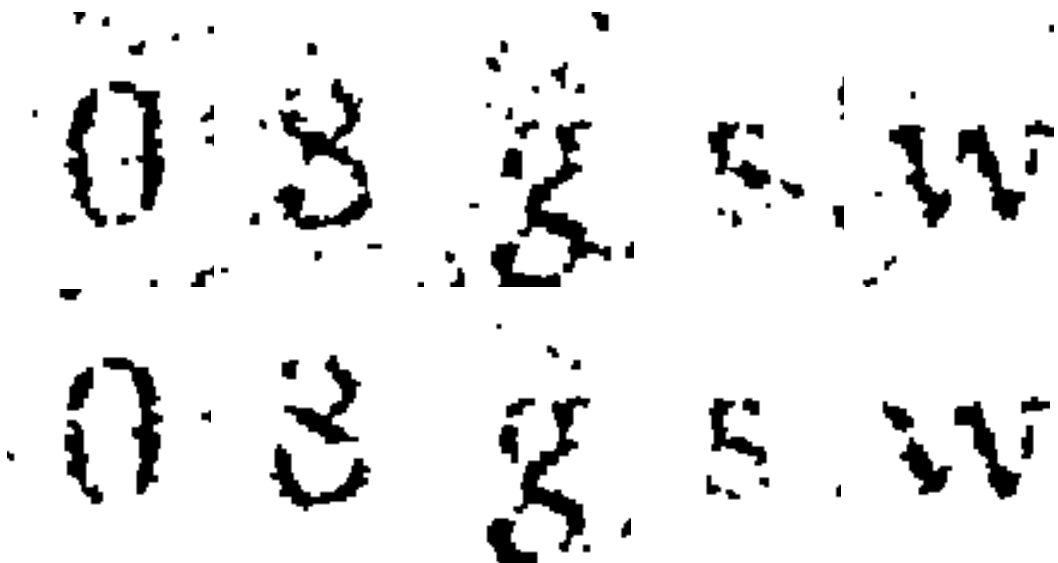


Figure 7.7: A sample of pescadero characters reconstructed with MRF and extended MRF.

7.2 Other symbols

Tests have also been ran on other character-like symbols (using the Windows's Windings character set). Figure 7.8 shows a sample of the noise-free and the noisy images. Figure 7.9. shows samples of these images reconstructed with MRF and extended MRF. The parameters of the MRF and extended MRF are the same.

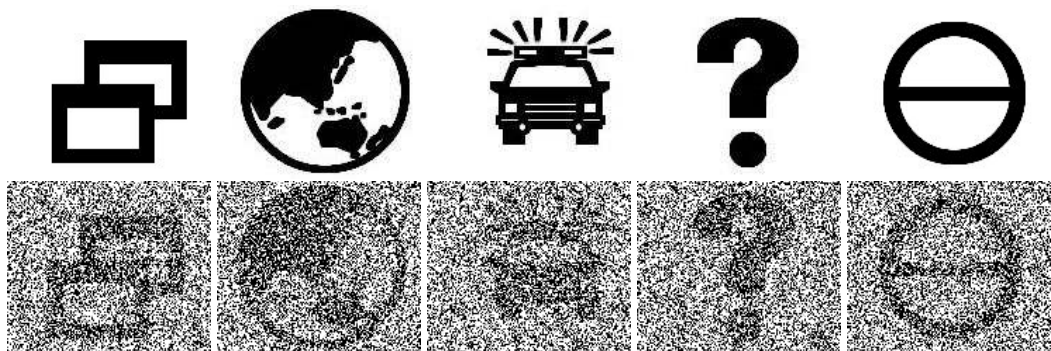


Figure 7.8: A sample of noise-free and noisy character-like symbols.

Working with 28 symbols the average improvement of the extended MRF method over the normal MRF was 10,78% while the recognition was 97,22% successful.

Table 7.1: Numerical comparison of the results of the two methods.

	arial	kristen	pescadero
0	7,17%	5,96%	7,17%
1	11,96%	17,14%	6,19%
2	10,64%	4,12%	13,57% / -0,69%
3	12,38%	9,57%	13,02%
4	2,60%	10,44%	20,67%
5	12,36%	2,97%	12,75%
6	6,34%	5,44%	22,86%
7	4,68%	14,29%	9,41%
8	-0,82%	12,06%	3,71%
9	8,96%	3,09%	7,06%
a	7,41%	7,56%	24,63%
b	8,16%	4,09%	4,12%
c	4,76%	11,71%	0,00%
d	12,67%	11,22%	19,47%
e	13,44%	16,36%	14,54%
f	21,22%	19,03%	25,55%
g	2,01%	8,36%	4,10%
h	18,27%	8,85%	21,14%
i	23,35%	7,13%	24,29%
j	5,21%	6,85%	16,07%
k	4,54%	8,23%	15,86%
l	13,04%	12,91%	43,07%
m	5,42%	4,46%	4,60%
n	8,24%	9,67%	21,85%
o	12,24%	2,55%	13,45%
p	12,67%	6,90%	13,96%
q	9,92%	12,58%	25,35%
r	18,73%	-0,88%	12,82%
s	7,21%	11,74%	11,67%
t	15,23%	13,04%	9,39%
u	9,49%	4,07%	7,64%
v	9,19%	-1,55%	10,75%
w	5,41%	10,18%	3,01%
x	7,58%	1,29%	6,64% / 6,75%
y	10,14%	3,44%	9,68% / 9,81%
z	16,98%	12,11%	27,72%
average	9,97%	8,25%	14,11% / 13,72%

Table 7.2: Percentage of successful classifications.

	arial	kristen	pescadero
E3	100,00%	100,00%	91,67%

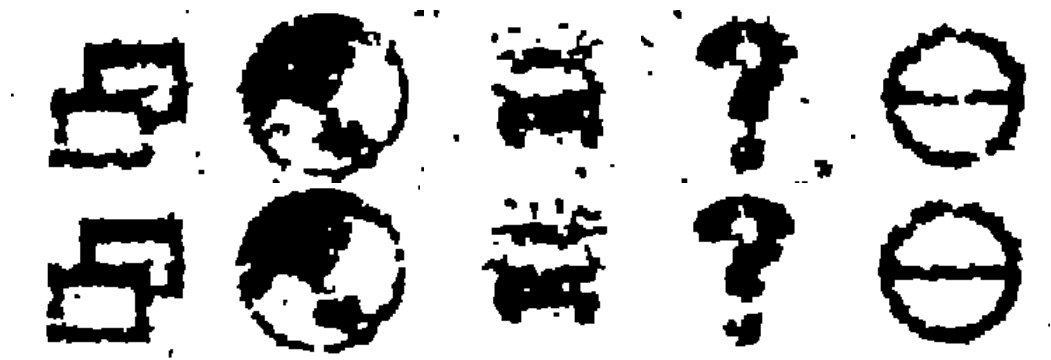


Figure 7.9: A sample of character-like symbols reconstructed with MRF and extended MRF.

Chapter 8

Summary

The goal of the paper was to present a method for extending the already existing MRF model for enhancing the reconstruction of image with a priori information and also providing recognition. For this purpose, the Radon transforms were used as a descriptor.

The results (which can be seen in chapter 7) seems to be better for eye, and they are numerically better too. The improvement between the results of the simple MRF and the extended MRF is approximately 11% on average for the tested font types. Also, the recognition of the noisy images using Radon transforms is not possible, but after this reconstruction, the recognition yields about 97% successful recognition rate. The correct template, which is necessary for the reconstruction can be chosen by the new energy term defined by the difference of Radon transforms. So a dual statistical model can be created, which is suitable for pixel based reconstruction and object based classification.

Bibliography

- [1] Dimitris Bertsimas and John Tsitsiklis. Simulated annealing. *Statistical Science*, 8(1):10–15, 1993.
- [2] Al Bovik, editor. *Handbook of Image and Video Processing*, chapter 4. Elsevier Academic Press, 2 edition, 2005.
- [3] V. Caselles. Geometric models for active contours. In *International Conference on Image Processing*, volume 3, 1995.
- [4] Tony Chan and Luminita Vese. An active contour model without edges.
- [5] Fred E. Daum. Beyond kalman filters: practical design of nonlinear filters. *SPIE*, 2561:252–262, 1995.
- [6] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*, chapter 7. Prentice Hall, 2003.
- [7] Xiang Yang Gao. *Adaptive Linear and Nonlinear Filters*. PhD thesis, University of Toronto, 1991.
- [8] Arti V. Gautam, D. M. Dakhane, and Vijay R. Tripathi. Image denoising using nonlinear filters. *International Journal of Information Technology and Knowledge Management*, 2(2):573–577, 2010.
- [9] S. Geman and C. Graffigne. Markov random field image models and their applications to computer vision. In *Proceedings of the International Congress of Mathematicians*. American Mathematical Society, 1986.
- [10] G. Gerig, O. Kubler, R. Kikinis, and F. A. Jolesz. Nonlinear anisotropic filtering of mri data. *IEEE Transactions on Medical Imaging*, pages 221–232, 1992.
- [11] Rafael C. Gonzalez and Richard Eugene Woods. *Digital image processing*, page 354. Prentice Hall, 2008.
- [12] David Griffeath. *Introduction to Markov Random Fields*, chapter 12. Springer, 2 edition, 1976.
- [13] Vedad Hadziavdic. A comparative study of active contour models for boundary detection in brain images. Master’s thesis, University of Tromso, 1999.
- [14] R. C. Hardie and C. G. Boncelet. Lum filters: A class of rank-order-based filters for smoothing and sharpening. *IEEE Transactions on Signal Processing*, 41:1061–1076, 1993.

- [15] Risto Holopainen. Nonlinear filters.
- [16] B. Jyotirmoy and C.V. Jawahar. Contextual restoration of severely degraded document images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.
- [17] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, pages 321–331, 1988.
- [18] Zoltan Kato. PhD thesis, 1994.
- [19] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
- [20] Steven Linsel. Denoiselab philosophy: A standard test set and evaluation method to compare denoising algorithms, 2007.
- [21] Stan Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer-Verlag London Limited, 3 edition, 2009.
- [22] Xinxiang Li. The anatomy of anisotropic diffusion filters. *CSEG Recorder*, 2005.
- [23] M. Miciak. Radon transformation and principal component analysis method applied in postal address recognition task. *International Journal of Computer Science and Applications*, 7(3):33–44, 2010.
- [24] M. A. Naser, A. Mahmud, T. M. Arefin, Golam Sarowar, and M. M. Naushad Ali. Comparative analysis of radon and fan-beam based feature extraction techniques for bangla character recognition. *IJCSNS International Journal of Computer Science and Network Security*, 9(9):287–289, 2009.
- [25] J. Radon. ber die bestimmung von funktionen durch ihre integralwerte lngs gewisser mannigfaltigkeiten. *Berichte ber die Verhandlungen der Schsische Akademie der Wissenschaften*, pages 262–277, 1917.
- [26] Maria Isabel Ribeiro. Kalman and extended kalman filters: Concept, derivation and properties, 2004.
- [27] John C. Russ. *The image processing handbook*. CRC Press, 2007.
- [28] Stephen J. Sangwine and Robin E. N. Horne. *The Colour Image Processing Handbook*, chapter 8. Chapman and Hall, 1998.
- [29] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Valdimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields. *ECCV 2006, Part II, LNCS 3952*, pages 16–29, 2006.
- [30] Zhou Wang and Alan C. Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, pages 98–117, 2009.
- [31] Joachim Weickert. *Anisotropic Diffusion in Image Processing*. B.G. Teubner Stuttgart, 1998.

- [32] Bernard Widrow and Istvan Kollar. *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control and Communications*. Cambridge University Press, 2008.