2 Vectorization

2.1 Introduction

2.1.1 Problem formulation

Vectorization (raster-to-vector conversion) consists of analyzing a raster image to convert its pixels representation to a vector representation The basic assumption is that such a vector representation is more suitable for further interpretation of the image; this typically holds for a scanned graphical documents (maps, schemes, drawings). The topic was in focus for the last 30 years, for more details see the books [3, 39, 130, 131, 184, 267], PhD Dissertations [123, 158, 177, 252, 271, 310], surveys [129, 128, 287, 266] and publications [2, 183, 219, 253-255].

The procedure of raster-to-vector conversion can be divided into three main stages: pre-processing, processing, and post-processing (see Fig. 2.1.1).

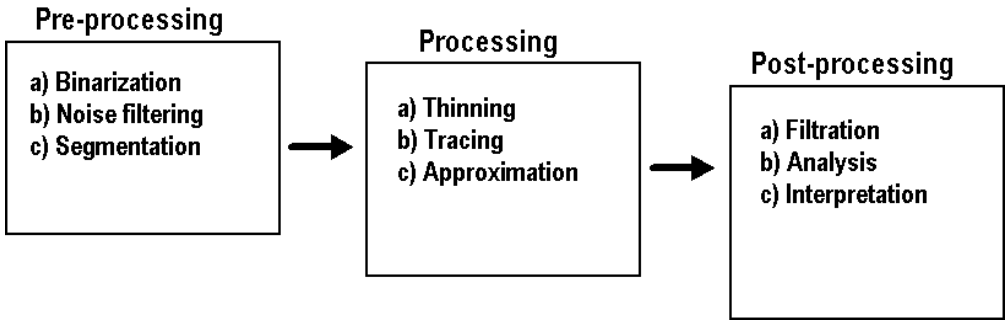


Figure 2.1.1. Three stages of raster-to-vector conversion.

2.1.2 Preprocessing

The purpose of the stage is to prepare the input raster image for processing (vectorization) at the next stage. Grayscale image should be binarized, grayscale or

binary image can be filtered for noise reduction, and color image has to be represented by monochrome layers (see Fig. 2.1.2). The type of pre-processing algorithm is defined by the type and quality of the input image.

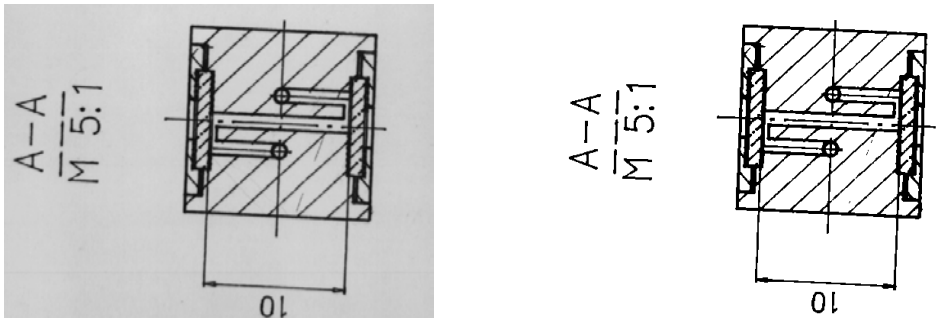


Figure 2.1.2: Illustration of raster image binarization: a) input halftone image; b) binary image after thresholding of the input image.

We developed locally adaptive thresholding algorithm for binarization of large images [**P1**]. We also studied problem of noise filtration of binary noise in application to compression [**P2**].

2.1.3 Processing (vectorization)

Conversion of the raster binary image to vector form is performed at this stage. Output data of the stage is vector presentation of the input binary image. There are six basic approaches for vectorization of binary image:

1) Thinning-based methods [123];
2) Contour-based methods [109];
3) Graph-structure based methods [164, 177];
4) Sparse pixel tracking methods [47, 66];
5) RLE-based methods [168, 191, 200];
6) Orthogonal zig-zag method [36, 65].

Every method has advantages and drawbacks. The skeleton-based methods have good results, but tend to be very sensitive to noise. Contour-based methods are more noise-tolerant but they rely on heuristic and complex matching schemes. Correct choice of method is defined by data type (maps, drawings, schemes), and practical goals. For example, for 2-dimensional object vectorization the contour-based methods are more relevant, but for elongated objects the thinning-based approach seems to be reasonable.

We use vectorization method based on thinning algorithm [**P1, P3**]. The procedure consists of three steps (see Fig. 2.1.3):

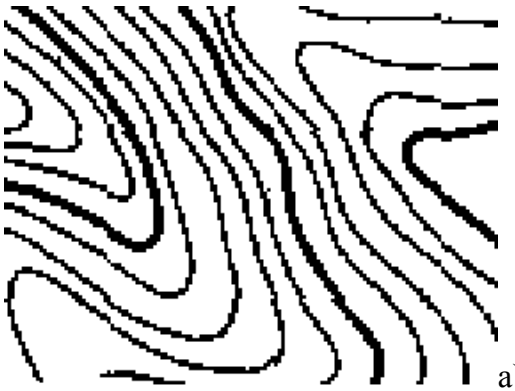
1) Binary image skeletonization by Distance Transform based thinning;
2) Tracking of skeletal branches to get chain coded digitized curves;
3) Primary vectorization of the digitized curves.

The primary vectorization can be performed with zero or non-zero approximation error. In the first case the chain code data are converted into segments of digital line in $O(N)$ time [12]. The primary vectorization can be performed with polygonal approximation with error tolerance defined by width of the curve [**P5**].

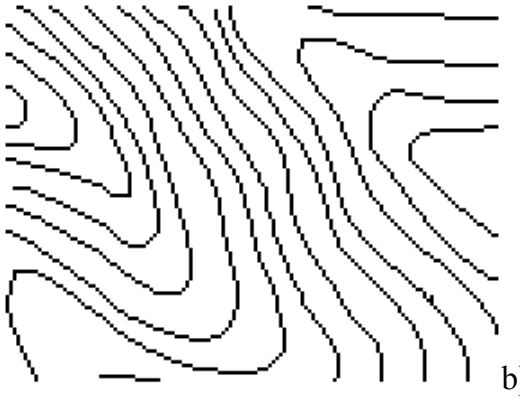
2.1.4 Postprocessing

The main goal of this stage is vector data analysis and interpretation. The purposes of this step are following: a) removing noise from the vector model; b) object recognition; c) recovering entities from vector data (object vectorization).

Usually the algorithms for vector data interpretation are based on domain knowledge [1, 271, 152, 252-255]. The main purpose of the developed system [**P1**] was to reduce the number of manual operations required for inputting the graphic data into a computer, leaving the editing of the resulting vector representation for an operator. To reduce processing time for large images the obtained vector data were presented as AVL-tree. The employed model ensures a fast and simple access to the vector data for further analysis. The analysis of vector data includes the following procedures: gaps filling, vectors classification, false branches elimination, right corners rectification [**P1**] and vector data simplification by polygonal approximation [**P5, P6**].



a)



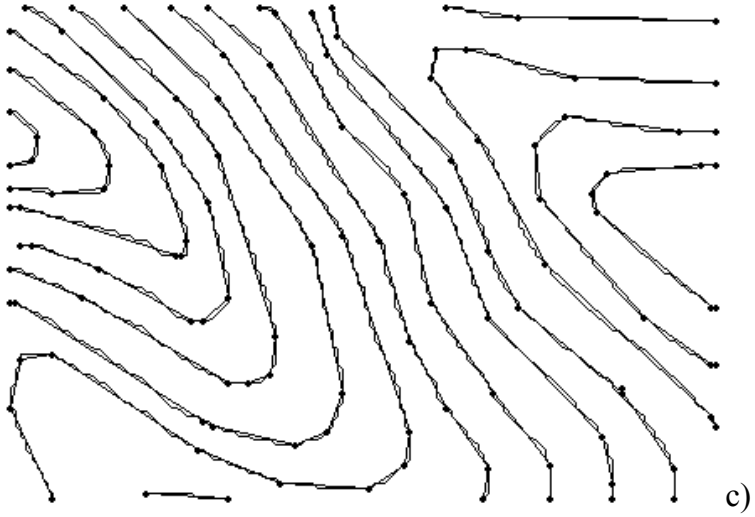


Figure 2.1.3. Illustration of primary vectorization of elevation map (fragment): a) raster image; b) skeleton of the raster image; c) primary approximation of the skeleton with error tolerance ε=0.99. Vectors are labeled with dots.

2.1.5 Proposed improvements

The purpose of the study is realization of raster-to-vector conversion system. Special attention was paid to development of efficient algorithms for processing of large size images. We have concentrated on the following low-level processing algorithms:

a) Binarization of large images [**P1**];

b) Thinning of large images [**P3]**;

c) Binary noise filtration [**P2]**;

d) Analysis of vector data [**P1**];

e) Polygonal approximation of curves [**P4-P7**].

In Sections 2.2-2.4, we consider some problems of all stages of raster-to-vector-conversion: binarization, thinning and binary noise filtering. Correspondent problems of polygonal approximation will be discussed in details in the Section 3.

2.2 Binarization of large images

2.2.1 Problem formulation

The input gray-scale image has to be binarized before the skeletonization stage. The purpose of the binarization procedure is to segment the image into background and object pixels (see Fig.2.2.1). The quality of the skeleton depends on the quality of the input binary image.

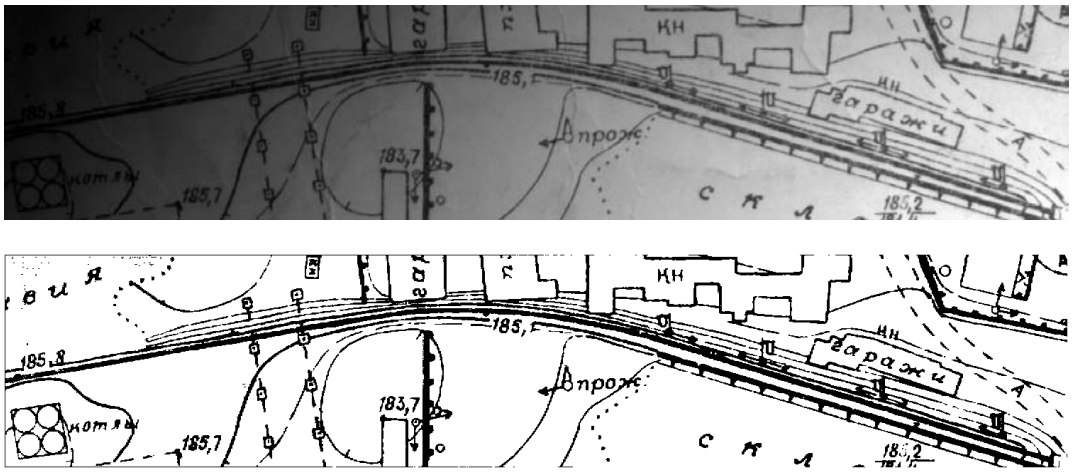


Figure 2.2.1. The input greyscale image (top); result of binarization with locally adaptive threshold (bottom).

The binarization of greyscale image can be performed by thresholding with some threshold T. If pixel value $p(x, y)$ is less than the threshold T, the pixel belongs to background, and thresholded pixel value will be '0', otherwise the pixel belongs to an object and value '1' will be assigned to the pixel.

In *global* thresholding, the same threshold value is applied to every pixel of the input image. In practice, because of non-uniformity of the background in the input image, the thresholding with global threshold provides poor result. In this case, *local* thresholding with threshold adjusted to the local properties of the image should be applied to obtain more reliable binary image (see Fig. 2.2.1). For this purpose, the input is divided into a rectangular blocks, and each of them is processed with *adaptive threshold* defined by the statistical properties of the block (see Fig. 2.2.2).

There is a lot of algorithms for threshold calculation based on different approaches [288, 185, 229, 74, 262, 154, 90, 186, 272, 273, 124, 3]. We have used Otsu's algorithm [185], which is based on clustering of gray-level histogram by maximization of between-class variance. The method has shown good results for the test images in use.

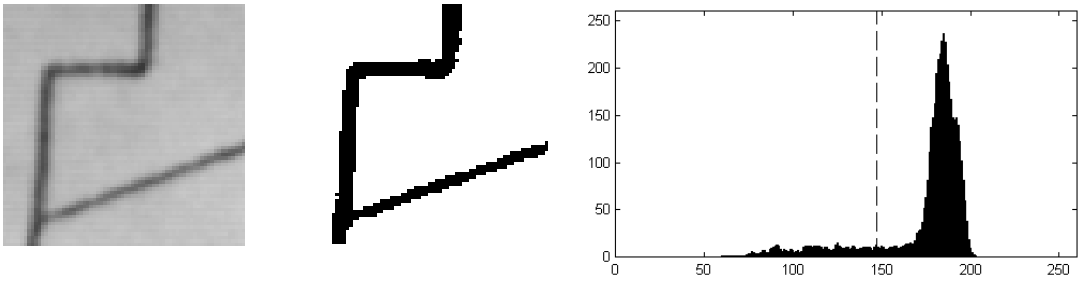


Figure 2.2.2. An example of input image thresholding: 60×60 block of greyscale image (left); result of binarization with threshold T=147 calculated by Otsu's method (center); histogram of the block (right). Threshold is shown by dashed line.

2.2.2 Locally adaptive binarization

Calculation of the locally adaptive threshold is based on information collected in a window surrounding the pixel to be processed [174, 28, 75, 239, 240].

In our study we considered the case of binarization of scanned maps and drawings under assumptions that quality is satisfactory and illumination over the image is smooth (quasi-linear linear) (see Fig. 2.2.1). The main attention was paid to development of algorithm for binarization of large size images. The term "large size image" means that the size of the image exceeds size of available computer memory.

To achieve this goal, we, first apply *locally adaptive* thresholding technique, and then perform the binarization of large image in *file-to-file* manner without keeping the whole image in memory. The input image is sequentially loaded and processed in stripe-by-stripe way so that partition of the image for loading and processing does not depend on the partition of the image for the analysis.

The developed algorithm consists of two steps: a) collecting and analysis of the histogram data, b) thresholding the image.

a) Collecting and analysis of the histogram data

The input image is processed fragment-by-fragment to collect histogram for the blocks. Histograms are collected for all blocks. If block contains objects, threshold is calculated for the block, otherwise the block is treated as empty. To improve the balance between object and background pixels in histogram we eliminate pixel from consideration when Laplacian $L(x, y)$ is *below* a certain threshold [289]. In this way, most of the background points will be excluded from histogram, while most of the object points will be kept in the histogram (see Figs. 2.2.3 and 2.2.4). Of course, mostly the noisy background points do contribution to the modified histogram. Nevertheless, usually these points belong to the background, so they can be treated as quite representative pixels for threshold calculation.

Analysis of the block histogram is performed to classify the block either as empty or non-empty. If block contains both background and object pixels, the histogram is expected to be wider than those of the blocks with background pixels only. So, if histogram width of a block is smaller than threshold W_H, the block is treated as empty without any object points. For empty blocks we get a preliminary threshold $T_0(i, j)$ from the nearest non-empty neighbours. If some non-empty block was classified as empty, it will be provided with threshold value from the nearest non-empty blocks. For non-empty fragments we calculate the preliminary threshold $T_0(i, j)$ by Otsu's method [185] from the block histogram.

We calculate threshold $T_1(i, j)$ for blocks as average of the preliminary thresholds $T_0(i, j)$ of their eight neighbours. If some empty block was erroneously classified as non-empty one with wrong preliminary threshold, this operation can correct the final threshold for the block. Taking into consideration that histogram of background pixels is relatively narrow, even a small shift down of the threshold gives correct thresholding of the block.

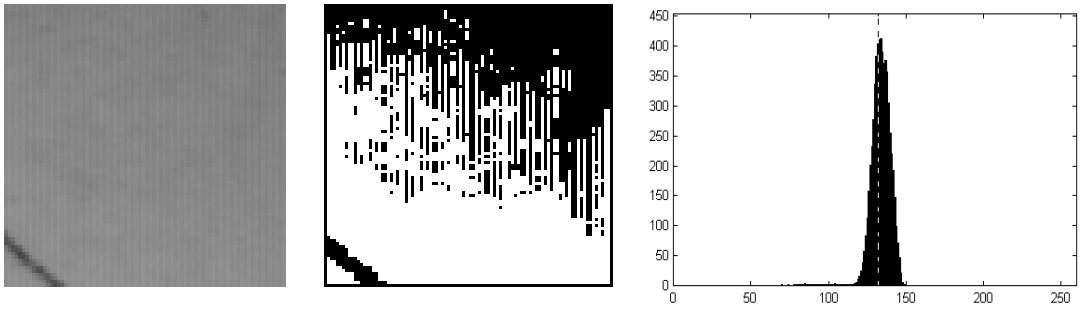


Figure 2.2.3. Thresholding of a block with a small object: block of the input greyscale image (left); result of binarization (center) with threshold T_0=132 calculated from histogram of the block (right). Threshold in histogram is labeled by dashed line.

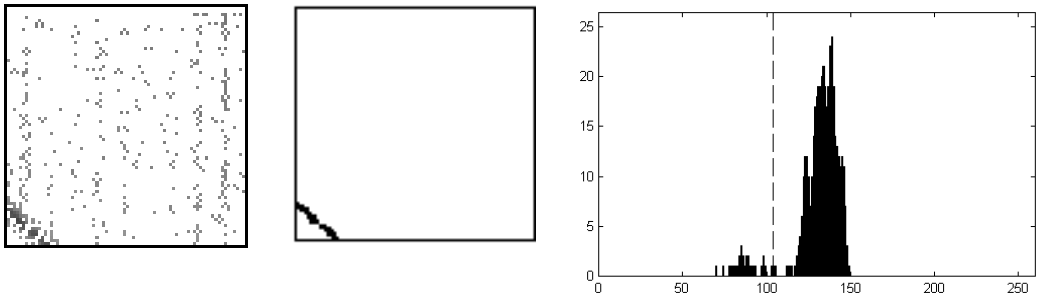


Figure 2.2.4. Thresholding of block with a small object: the masked block of the input greyscale image, defined for Laplacian threshold L_1=25 (left); result of binarization (center) with threshold T_1=104 calculated from histogram of the masked block (right). The masked pixels are shown as white. Threshold in histogram is shown by dashed line.

b) Image thresholding

The input image is thresholded using the calculated 2D array of thresholds $T_1(i,j)$ for blocks. The local threshold $T_L(x,y)$ for point (x,y) is computed as bilinear approximation of thresholds $T_1(i,j)$ for four neighbouring blocks (see Fig. 2.2.5).

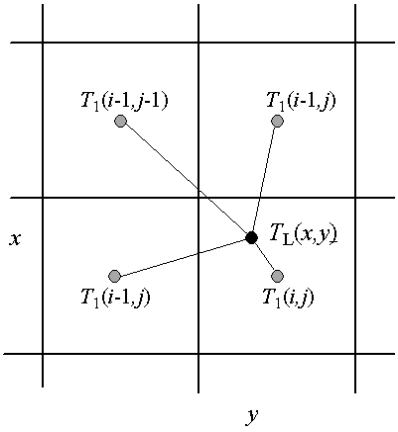


Figure 2.2.5: Scheme of four neighbouring blocks for computation of the local threshold $T_L(x, y)$ with bilinear interpolation.

2.2.3 Summary

The designed scheme for locally adaptive binarization of large grayscale images has been tested with grayscale images acquired by scanner of projective type. The provided tests have shown high efficiency of the algorithm for grayscale images with quasi-linear non-uniformity of background illumination.

2.3 Thinning algorithm for large binary images

The skeleton of a binary object is a shape descriptor, which can be regarded as a convenient alternative of the elongated object itself [195]. Thinning a binary image down to its skeleton allows one to transform the image into a line drawing, which still contains the relevant information (see Fig. 2.3.1). For many applications such transformation is very useful, because it reduces drastically the amount of data to be handled, and simplifies computation procedures required for description and classification purposes.

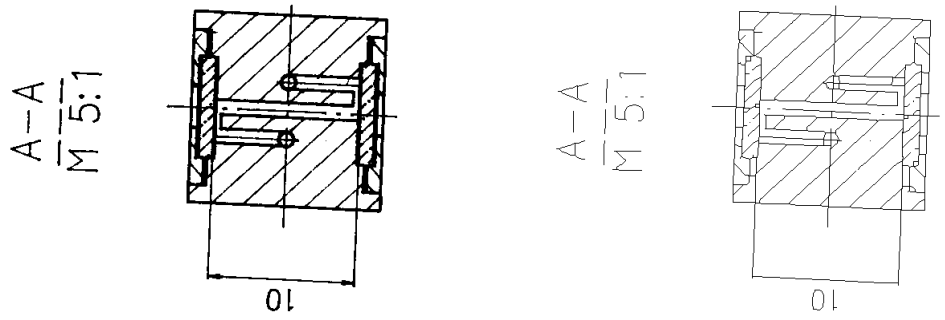


Figure 2.3.1: The input binary image (left) and the thinned image (right).

The skeletonization (thinning) process can be seen as an isotropic retraction of the original object, down to its unit width subset. This subset is placed in the medial region of the object, has the same topology, and allows the evaluation of the spatial dimensions as well as orientation of the object. The skeleton can be obtained by thinning of the binary object in two ways:

a) Algorithm based on morphological thinning that preserve homotopy (iterative peeling);

b) Algorithm based on Distance Transform (DT) that preserve reversibility.

2.3.1 Iterative peeling algorithms

The skeleton of an image is built by iteratively peeling off the boundary pixels until only no erasable pixels remain in the image [13, 160, 161, 163, 201]. Number of

runs for this method is equal to half-width of the object. The iterative thinning algorithms can be further divided into two categories: *sequential* and *parallel*.

We have to distinguish difference between parallel or sequential *algorithm* and parallel or sequential *realization* of the algorithm. The terms "sequential realization" or "parallel realization" are related to practical implementation of thinning algorithm with sequential or parallel machines, whereas the terms "sequential algorithm" or "parallel algorithm" specify the main principle of raster data processing.

Sequential algorithms examine contour points for deletion by either a) raster scanning, or b) contour following algorithms. In sequential algorithm result of processing for the current pixel depends on the results for already processed points in the current point neighbourhood. Usually the thinning is performed by sequential scanning with window of 3×3 size [13], or larger [203].

In *parallel* thinning algorithms result for the current pixel is performed independently on the current states of the neigbouring points and depend only on results for the previous iteration of thinning applied to the whole image [256, 314, 127, 266, 295, 42, 315]. Parallel algorithms for preserving the connectivity of skeleton use either larger neighbourhood than 3×3 or use more than one pass over the image [314, 261, 42, 95, 96].

Sequential and parallel realization of the peeling algorithm

If image file is larger than available memory resources of the ordinary single-processor machine, the image has to be divided into overlapped fragments of smaller size and loaded for processing in fragment-by-fragment fashion. Size of the fragments is defined by size of available memory resource.

Skeletonization of large images with the iterative algorithm can be a time-consuming procedure. To reduce processing time for solving practical tasks, parallel multiprocessor systems or a special hardware are used. The parallel version of the iterative peeling algorithm for skeletonization is included into so called *Cowichan problem set* [291, 292] as benchmark task for parallel programming systems.

To be processed on distributed-memory multi-processor machines [19, 20, 50, 51, 99, 107, 163, 170, 202, 204, 276, 301-303] image is divided into overlapped rectangular fragments (blocks). The fragments are distributed among the processors. After every run the processors exchange data on the border of the fragments (one line). Number of runs is defined by maximum width of the objects in the image. It affects in big number of time-consuming data exchange operations between processors. Exist a number of massively-parallel realizations of the peeling

algorithms with O(N)–O(N^2) processing elements, here N is size of the image [167, 217, 313, 318].

2.3.2 Distance-transform based methods

The alternative method is to first calculate the Distance Transform (DT) of the binary image. Distance Transform is defined for an object point as a distance from the pixel to the nearest background point [29, 30, 195, 221, 222]. At first Distance Transform is performed then skeletal points are detected on the DT using some rules [15, 16, 39, 31, 32, 175, 195, 205, 206, 221, 222, 248, 260, 268]. Choice of the distance metrics depends on the task to be solved: Euclidean, octagonal, chessboard, city-block, weighted chamfer. In our application we use chessboard distance measure for Distance Transform, because it provides undistorted skeleton of rectangular objects (see Fig. 2.3.2).

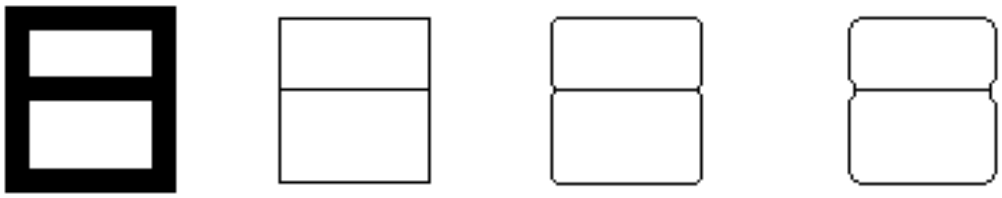


Figure 2.3.2: Illustration of skeletons for different distance metrics: a) the test rectangular shape with lines of width 9; skeleton for b) chessboard, c) Euclidean, and d) city-block distance metrics.

Distance transform can be completed in two raster scans through the whole image a 2×3 window for chessboard and city block DT or bigger one, for example, a 3×5 window for 5-7-11 distance mask [Borgefors'86]. The first scan is carried out in a top-left to bottom-right direction. During this scan for every pixel in an object the distance form the top and left object boundary is determined. The second scan in a vice versa direction determines the distance from every pixel in the object to the bottom and right boundaries in a similar way. Detection of skeletal points is performed by means of two subsequent scans through the whole image by a 3×3 window in the same fashion as for the distance transform. One additional scan is required to reduce two-pixels width skeleton to the unit-width one and to convert a distance-labeled skeleton into width-labeled one. Thus the image skeletonization can be accomplished in a sequence of five raster scans through the image. Every subsequent scan of DT and skeletal point detection is carried out in the direction opposite to the previous one and requires the complete result from the previous scan.

Realization of the DT-based algorithm

As we can see, the thinning procedure with DT-based algorithm can be performed in a fixed number of runs regardless the objects size, but the algorithm demands sequential processing of the image. There are a number of realizations for massively-parallel machines, but they demand special hardware with $O(N)$–$O(N^2)$ processing elements [241, 38, 125, 87, 88, 155, 46].

The main problem for realization of DT-based thinning algorithm is processing of large images. Tombre *et al*. wrote in [266]: "*However, it is difficult to compute the distance transform without storing the whole image in memory*". Although DT-based algorithm is faster than peeling procedure, the algorithm was not widely used in practical applications for processing large images in ordinary single-processor machines and in parallel multiprocessor systems.

2.3.3 Fast implementation of the thinning algorithm

The number of different skeletonization algorithms exceeds at least 300 items, for more details see the comprehensive surveys [3, 148, 149]. In our research, we have concentrated on *efficient implementation* of the existed thinning algorithms rather than on developing new methods for the thinning. The main goal of our studies was the development of efficient implementation of DT-based skeletonization algorithm, which is suitable in practice for processing of large input with ordinary single-processor or parallel multiprocessor machines. The classical DT-based thinning algorithm of Arcelli and Sanniti di Baja [15] with chessboard distance metrics has been selected for realization, but the approach can be used for any other DT-based algorithm with different distance metrics.

At first, we developed realization of the skeletonization algorithm for sequential computers as a part of raster-to-vector conversion system [**P1**, **P2**]. Later we extended the approach on the case of parallel multiprocessor systems [**P3**].

In the case of image skeletonization with ordinary single-processor machine the image file is being read and processed by overlapped blocks (see Fig. 2.3.3). The loaded current image block is processed in both directions several times according the skeletonization algorithm in use. When the total processing of the current block is completed, size δ of the overlapping is defined by the maximum value $D_{\max}$ of Distance Transform in the last row: $\delta = \lceil D_{\max}/2 \rceil$.

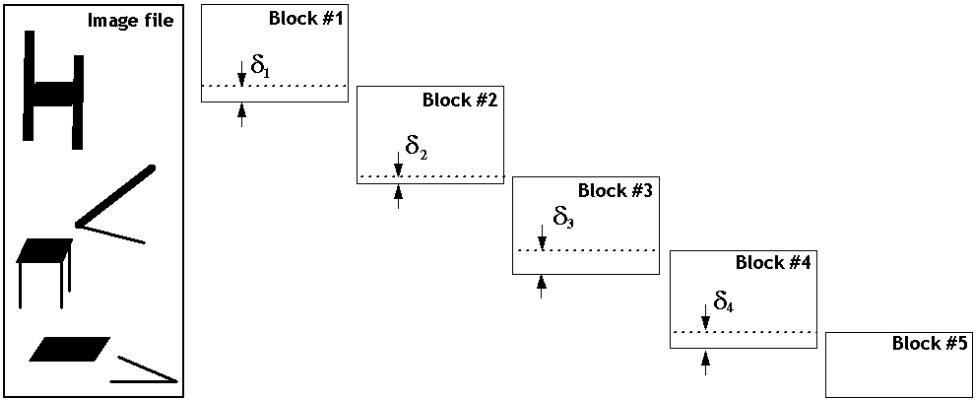


Figure 2.3.3: Scheme of overlapped blocks loading. Size δ_k of the overlapped zones depends on thickness of objects at the blocks border and calculated from the distance transform at the last row in the current block.

The main idea of the proposed approach is as follows: a) all the necessary processing operations are performed within one block, and b) size of overlapping is controlled with DT value to insure the correctness of the procedure. With this method image of any size can be processed with only one reading of the image file with minimum overlapping of blocks.

This approach can be used for processing large images on two-processor computers with shared memory. In this case processing of every block is performed simultaneously by the two processors as in [80], but the control of loading is performed as in [**P3**].

For realization of the proposed skeletonization algorithm for parallel processing system we selected distributed-memory Multiple Instructions Multiple Data (MIMD) model of parallelization [140, 142, **P3**]. The selected model is more practical than massively-parallel machine, which demands special hardware with $O(N)$–$O(N^2)$ processing elements [38, 46, 81, 87, 88, 125, 156, 241].

According the computational scheme for parallel processing the image is divided into blocks, which are distributed among the processors (see Fig.2.3.4). After the first run the processors exchange data on the border (one line). After the second run with the correspondent data exchange, an additional short run is performed. The depth of the third run δ is defined by maximum difference of values of the distance transform on the border $D_B(x)$ and in the received image row (or column) $D_R(x)$ from the neighbouring processor: $\delta=\max\{0, \lceil(D_B(x)-D_R(x)/2\rceil\}$, where $1\le x\le$size. The approach allows getting result of skeletonization after fixed number of runs regardless maximum line width with minimum number of additional operations in comparison with the case when the whole image is available.

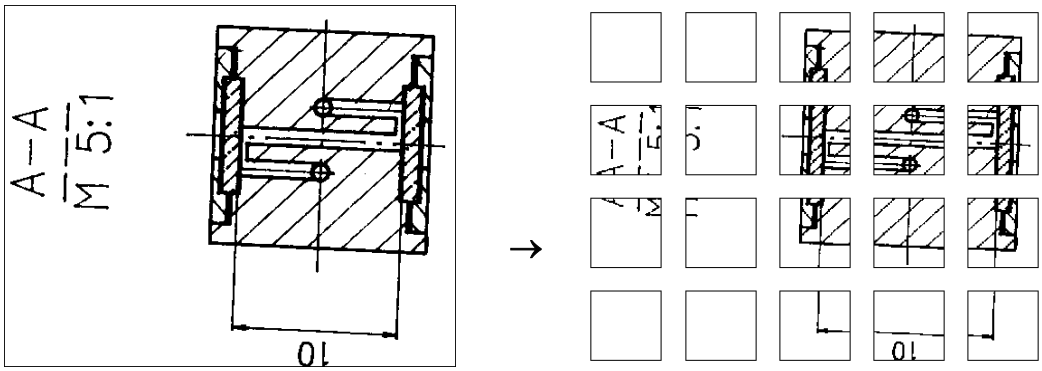


Figure 2.3.4: Scheme of input image partition into rectangular blocks among processors: input binary image (left), twenty image blocks for 4×5 processor mesh grid (right).

Vossepoel *et al*. [282] proposed the analogous approach for DT-based thinning algorithm of large images. But according to their approach depth of the third (additional) run was fixed: δ=100 pixels. In the case of processing with parallel system, which is considered in the paper [282], the difference between two approaches is not crucial, because the total processing time depends on the depth of the last run about linearly. For the ordinary single-processor machine the dependence of the processing time on the size of overlapped zone is nonlinear: $T \sim H/(H-\Sigma\delta_k)$, where H is the image height and δ_k is overlapped zone size for kth block. That is why in this case the processing time is more sensitive to the size of overlapped zones, especially for very wide input images. With the proposed method [**P3**] with control of overlapping size we can reduce time-cost of thinning algorithm to the minimum. For example, if the maximal width of lines on blocks borders does not exceed 20, the size of overlapped zone will be about 10 pixels, that it much less than 100 pixels as in the approach [282].

Another method, which has been applied to reduce processing time, was using of look-up tables for DT-based skeletonization. During scanning procedure 9-bit index I_k for the current point is computed recursively from the index I_{k-1} for the previous point using three new pixels that just appeared in the 3×3 window. To avoid redundant computation of the index for background points, using of the recursive scheme starts with beginning of object pixels run.

2.3.4 Summary

The developed efficient realization of the skeletonization algorithm for large images has allowed us to drastically reduce processing time for raster-to-vector conversion system [**P1**]. As it was mentioned in [**P2]**, where the vectorization procedure was

used as part of binary image compression scheme, processing time for vectorization takes only 10% of the total processing time.

2.4 Vectorization-based binary noise filtering

Scanned binarized images have a noise which can appear because of low quality of originals, paper defects, non-optimal threshold setting of binarization process, and non-uniform illumination. In the case of binary images, the noise appears as *content-dependent*, distorting the contours of objects, and as salt-and-pepper additive noise (randomly scattered noise pixels). The noise level may be low enough not to significantly detract from the quality, but it introduce unnecessary details that decrease the analysis or compression of the binary image, and distort the vector presentation of the binary image.

Normally the size of noise patterns is less than those of binary objects (lines, strokes, symbols). Also the objects details are more structured than the noise patterns. Therefore, noise pixels in the noise patterns are less correlated with neighbours than pixels of the objects. This correlation is used in most algorithms for binary image enhancement and noise reduction.

2.4.1 Survey of solutions

Several approaches for binary noise filtering have been considered by analyzing the local pixel neighbourhood defined by a filtering template. These filters use a set of rules to accept or reject the pixel, such as predefined masks or a quantitative description of the local neighbouring area.

Recent research in mathematical morphology has shown that morphological filtering can be used as an efficient tool for pattern restoration in an environment with a lot of additive noise [69, 70, 102, 159, 242].

Jin *et al*. [126] proposed a new class of morphological operators for binary images, it is the domain operators. The basic idea is taken from ranked-order filters, but generalized with the incorporation of the fuzzy index function in weight representation. Chinnasarn *et al.* [48] utilized mathematical morphology approach to modify *kFill* algorithm of O'Gorman [184]. With the proposed heuristic rules for image filtering in 3×3 and 4×4 window, they reduced the number of iterations to a single-pass scan over the image.

Several methods have been considered for image processing by analyzing the local pixel neighborhood defined by a filtering template. Techniques have been proposed based on the analysis of context information [199, 208, 317].

Wahl [283] introduced algorithm which operates within 5×5 window. The objective of the processing is to eliminate noise utilizing four heuristic rules. Model-based approach for binary noise filtering have also been used [242]. This approach assumes a specific probabilistic models that describe the behaviour of both signal and noise patterns, which are elementary geometrical primitives from which the signal and noise images are constructed.

The algorithm of Nikiel [176] is based on the calculation of fractal dimension of binary image in 8×8 sliding window. The filtering is performed in two steps: (1) extraction and estimation of the fractal dimension, and (2) classification and actual filtering (noise suppression and solid refinement).

Ping *et al*. [199] proposed two algorithms for binary images filtering. The first algorithm called Modified Directional Morphological Filter (MDMF) is introduced with dual properties for eliminating document salt-and-pepper noise and for remedying eroded character stroke distortion. For eliminating larger noise, another algorithm called *Image Geometric Structure Filter* (IGSF) is proposed based on the geometric stroke information of characters.

Randolph and Smith [208] used a *binary angular filter banks* for directional decomposition to enhance fax documents. The filter banks provide representations that delineate the directional components in the text letters enabling edges and contours to be smoothed appropriately.

For the small window size of 3×3, there are 512 unique table entries for binary image processing, which by itself is quote manageable. With the bigger filtering window more global information is taken into account, but straightforward using of larger windows demands more memory resources: 5.0×10^{14} unique entries in analysis tables for 7×7 window, which is making the table management impractical [208]. Following a clustered mapping approach [145] based on PNN algorithm, an efficient tree structured mapping function can be constructed that allows all entries to be mapped to a set of weights.

Ageenko and Fränti [5] proposed two context-based filtering methods, namely *Simple Context Filters* and *Gain-Loss Filters* for the enhancement of document images. They used the 10- and 20-pixel causal templates from JBIG to collect statistics during the analyzing phase. Then, in the filtering phase all rare pixels in low entropy contexts are flipped.

In [317] *morphological degradation model* was proposed for binary images. According the model, the probability of a pixel flipping from foreground to background, or vice-versa, is an exponential function of its distance from the nearest boundary point. Based on the model they offered restoration algorithm, which includes two stages: a training stage to define parameters of the model, and a restoration stage. The training stage includes joint analysis of degraded and the correspondent *ideal* image by computing the conditional distribution between the noise pattern pairs.

Fränti *et al*. [84, 85] used Hough Transform (HT) for extracting vector features from binary image. A *feature image* is reconstructed from the extracted linear segments and it is utilized in the filtering phase. The filtering is based on noise removal procedure using the original and feature images. The noise-filtering algorithm was used to improve quality of context-based compression algorithm. The drawback of this approach is that the HT-based feature extraction phase dominates the processing time in the compression phase and makes it an order of magnitude slower than JBIG compression procedure. In practice it means that the feature extraction phase for 6 test binary images of total size 4 Mb takes about two hours with Pentium-200 machine. For comparison, the processing time for the compression phase is 1.5 min only.

2.4.2 Feature-based filtering

Usually, the noise filtration procedure is a part of the pre-processing stage of vectorization. We propose to use raster-to-vector conversion of input binary image for noise removal [**P2**]. We use the vector presentation to collect information about pattern structure at the neighbourhood of the pixel to be filtered. With this information we can smooth the borders of linear elements preserving details of other objects. The proposed approach is suitable for images that consist mostly of elongated linear objects (maps, drawings, schemes).

The main goal of this study is to improve the quality of binary image compression by noise filtering. The filtering reduces irregularities in the image caused by noise, and in this way, makes the image more compressible without degrading the image quality. The process of noise filtration consists of two stages: extracting of line features, and feature-based filtering.

In the first stage, global information is gathered from the image by extracting line features with vectorizing algorithm we introduced in publication **P1.** According to the vectorizing algorithm in use, the input binary image is skeletonized and then vectorized. The reference raster image is restored from the vector presentation by

simple vector-to-raster conversion. In the second stage, the original image is processed for removing noise along the borders of extracted linear elements utilizing the original and the reference raster images.

Table 2.4.1: JBIG compression results [Ageenko'00] for images by feature-based filtering using Hough transform (HT) and the raster-to-vector conversion (RVC). The compression improvement is measured in comparison with the unfiltered image.

Input Image	Original Raster image (bytes)	Without Filtering (bytes)	HT-based Filtering (bytes)	RVC-based filtering (bytes)
Bolt	317,038	12,966	10,537	10,210
Power	512,199	17,609	16,271	14,581
Plan	484,561	5,098	4,319	3,978
TOTAL:	1,313,798	35,673	30,127	28,769
Improvement:	--	0.0%	12.7%	19.4%

The filtering is applied as part of a context-based image compression procedure. The compression remains near-lossless as only isolated pixels are eliminated. Experiments with test images show that from the compression point of view, the feature-based filtering with vectorization is twice as effective as traditional median filter, or a combination of three morphological filters: *opening*, *closing* and *annular filter* [102]. Comparison to other filtering algorithm that uses Hough Transform for vector feature extraction [84, 85], shown compression improvement of 19.2% for vectorization-based algorithm, and 12.7% against in comparison the HT-based approach (see Table 2.4.1).

Now let us consider question of time performance of this approach. Raster-to-vector conversion is time-consuming process involving a lot of image processing and image analysis procedures. Processing time for large images is important issue in practical applications. In our case, due to fast implementation of algorithms we developed for the raster-to-vector conversion [**P1, P3**] the burden of the vectorization phase on the total processing time was reduced to 10%. In fact, the vectorization phase is up to 3-4 times faster than the JBIG compression procedure. In practice, the developed raster-to-vector conversion takes less 1% of the processing time for the HT-based algorithm [84, 85]. The compression ratio for the test binary images is also better for the vectorization-based method than for the HT-based approach.

2.4.3 Summary

The feature-based filtering technique removes additive noise form the original binary image and in this way, produces a better compression performance. Due to the developed of efficient methods for raster-to-vector conversion of large images the vectorization-based method outperforms Hough Transform based algorithm by quality as well by time performance.